Red Hat OpenStack Platform 17.1

使用 director 安装和管理 Red Hat OpenStack Platform

使用 director 创建和管理 Red Hat OpenStack Platform 云

Last Updated: 2024-06-21

Red Hat OpenStack Platform 17.1 使用 director 安装和管理 Red Hat OpenStack Platform

使用 director 创建和管理 Red Hat OpenStack Platform 云

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

使用 Red Hat OpenStack Platform director 在企业环境中安装 Red Hat OpenStack Platform 17。其中包括安装 director、规划您的环境以及使用 director 创建 OpenStack 环境。

目录

让开源更具包容性

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

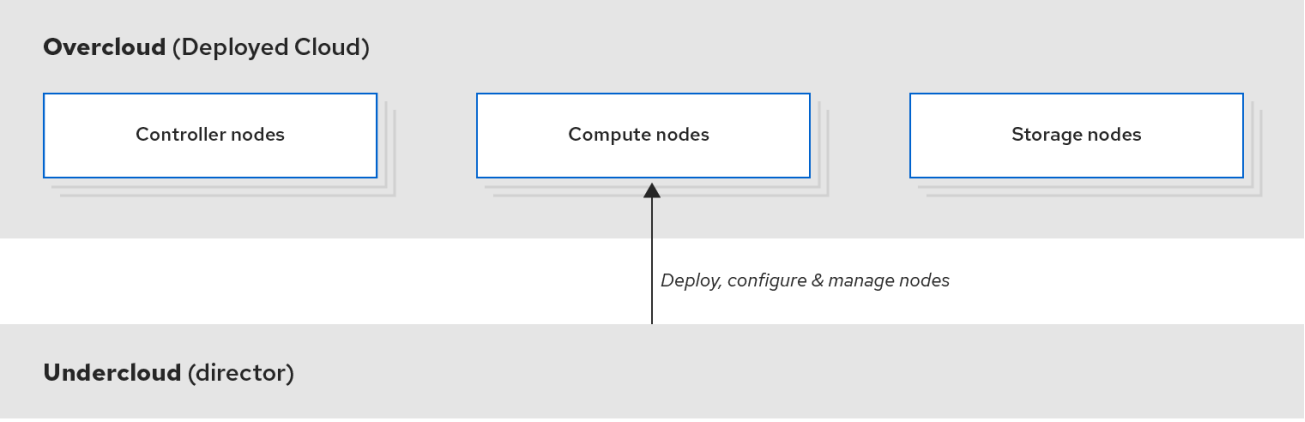
使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

第 1 章 DIRECTOR 简介

Red Hat OpenStack Platform (RHOSP) director 是安装和管理完整的 RHOSP 环境的工具组。director 主要基于 OpenStack 项目 TripleO。通过 director，您可以安装一个完全运行、精益且稳定的 RHOSP 环境，该环境可以置备和控制裸机系统以用作 RHOSP 节点。

director 使用两个主要概念：undercloud 和 overcloud。首先安装 undercloud，然后使用 undercloud 作为安装和配置 overcloud 的工具。



1.1. 了解 UNDERCLOUD

undercloud 是包含 Red Hat OpenStack Platform (RHOSP) director 工具集的主要管理节点。它是单系统 RHOSP 安装，其中包含用于置备和管理组成 RHOSP 环境的 RHOSP 节点：overcloud。组成 undercloud 的组件具有多个功能：

RHOSP 服务

undercloud 使用 RHOSP 服务组件作为其基本工具集。每个服务都在 undercloud 上的独立容器内运行：

- Identity service (keystone)：提供 director 服务的身份验证和授权。
- 裸机调配服务(ironic)和计算服务(nova)：管理裸机节点。
- 网络服务(neutron)和 Open vSwitch：控制裸机节点的网络。
- 编排服务(heat)：在 director 将 overcloud 镜像写入到磁盘后提供节点的编配。

环境规划

undercloud 包括用户可用于创建和分配某些节点角色的规划功能。undercloud 包括一组可分配给特定节点的默认节点角色：计算 (Compute)、控制器 (Controller) 和各种存储角色。您也可以设计自定义角色。另外，您还可以选择每个节点角色中包含的 RHOSP 服务，它提供了一种对新节点类型建模或隔离自己主机上的特定组件的方法。

裸机系统控制

undercloud 使用每个节点的带外管理接口（通常是智能平台管理接口(IPMI)）进行电源管理控制，以及基于 PXE 的服务来发现硬件属性并在每个节点上安装 RHOSP。您可以使用此功能将裸机系统置备为 RHOSP 节点。有关电源管理驱动程序的完整列表，请参阅电源管理驱动程序。

编配

undercloud 包含一组 YAML 模板，这些模板代表您环境中的一系列计划。undercloud 导入这些计划，并按照其说明创建生成的 RHOSP 环境。这些计划还可以包括 hook。通过使用 hook，可以在环境创建过程中的特定点上融入您自己的自定义设置。

1.2. 了解 OVERCLOUD

overcloud 是 undercloud 创建的 Red Hat OpenStack Platform (RHOSP) 环境。overcloud 由多个具有不同角色的节点组成，它们根据您要创建的 RHOSP 环境定义。undercloud 包括一组默认的 overcloud 节点角色：

Controller

Controller 节点为 RHOSP 环境提供管理、网络和高可用性。推荐的 RHOSP 环境在高可用性集群中包含三个 Controller 节点。
一个默认 Controller（控制器）节点角色支持以下组件。不是所有这些服务都默认启用。其中一些组件需要自定义或者预打包环境文件才能启用：

- dashboard 服务(horizon)
- Identity 服务 (keystone)
- 计算服务(nova)
- Networking 服务 (neutron)
- 镜像服务(glance)
- Block Storage 服务 (cinder)
- Object Storage 服务 (swift)
- 编配服务(heat)
- 共享文件系统服务(manila)
- 裸机置备服务(ironic)
- Load Balancing-as-a-Service (octavia)
- 密钥管理器服务(barbican)
- MariaDB
- Open vSwitch
- 高可用性服务的 Pacemaker 和 Galera。

计算

Compute 节点为 RHOSP 环境提供计算资源。随着时间的推移，可以通过添加更多节点来扩展您的环境。一个默认 Compute （计算）节点包括以下组件：

- 计算服务(nova)
- KVM/QEMU
- Open vSwitch

存储

存储节点为 RHOSP 环境提供存储。以下列表包含有关 RHOSP 中存储节点的各种类型的信息：

- Ceph Storage 节点 - 用来组成存储集群。每个节点包含一个 Ceph Object Storage Daemon (OSD)。此外，当您部署 Ceph Storage 节点作为环境一部分时，director 将 Ceph Monitor 安装到 Controller 节点上。
- Block Storage (cinder)- 用作高可用性 Controller 节点的外部块存储。这类节点包括以下组件：
  - Block Storage (cinder)卷
  - Telemetry 代理
  - Open vSwitch.
- Object Storage (swift)- 这些节点为 RHOSP 对象存储提供外部存储层。Controller 节点通过 Swift 代理访问对象存储节点。对象存储节点包含以下组件：
  - Object Storage (swift)存储
  - Telemetry 代理
  - Open vSwitch.

1.3. 在 RHOSP 中使用 RED HAT CEPH STORAGE

对于使用 Red Hat OpenStack Platform (RHOSP) 为成千上万的客户端提供服务的大型机构来说，这很常见。每个 OpenStack 客户端在消耗块存储资源时可能会有自己的独特需求。在单个节点上部署镜像服务(glance)、块存储服务(cinder)和计算服务(nova)可能无法管理具有数千个客户端的大型部署中。在外部扩展 RHOSP 可解决这个问题。

但是，在实际的环境中，仍然需要一个存储层的虚拟化解决方案（如 Red Hat Ceph Storage）来扩展 RHOSP 的存储层，使它可以支持 terabyte、petabyte 甚至 exabyte 数量级的存储要求。Red Hat Ceph Storage 提供了这样一个存储虚拟化层，在商用硬件上运行时具有高可用性和高性能。虽然虚拟化可能看起来像是性能损失，但 Red Hat Ceph Storage 会将块设备镜像作为对象分布在集群中的对象，这意味着大型 Ceph 块设备镜像的性能比独立磁盘的性能更高。另外，Cept Block 设备还支持缓存、copy-on-write cloning 和 copy-on-read cloning 功能来提高性能。

有关 Red Hat Ceph Storage 的更多信息，请参阅 Red Hat Ceph Storage 。

第 2 章 规划您的 UNDERCLOUD

在 undercloud 上配置并安装 director 之前，您必须规划 undercloud 主机以确保它满足某些要求。

2.1. 准备 UNDERCLOUD 网络

undercloud 至少需要 2 个 1 Gbps 网络接口卡(NIC)，每个主要网络对应一个：

- **Provisioning 或 Control Plane 网络** ： director 用来置备节点并在执行 Ansible 配置时通过 SSH 访问这些节点。此网络还支持从 undercloud 到 overcloud 节点的 SSH 访问。undercloud 包含在此网络上内省和置备其他节点的 DHCP 服务，这意味着此网络上不应存在其他 DHCP 服务。director 为此网络配置接口。

- **外部网络** ：启用对 Red Hat OpenStack Platform (RHOSP)存储库、容器镜像源和其他服务器（如 DNS 服务器或 NTP 服务器）的访问。使用此网络从您的工作站对 undercloud 进行标准访问。您必须在 undercloud 上手动配置一个接口以访问外部网络。

规划网络时，请查看以下准则：

- 红帽建议将一个网络用于置备，并将 control plane 和另一个网络用于数据平面。

- 置备和 control plane 网络可以在 Linux 绑定或单独的接口上配置。如果您使用 Linux 绑定，请将其配置为 active-backup 绑定类型。

  - 在非控制器节点上，在置备和 control plane 网络中流量数量相对较低，它们不需要高带宽或负载均衡。

  - 在 Controller 上，置备和 control plane 网络需要额外的带宽。带宽增加的原因是，控制器为其他角色中的多个节点提供服务。对环境进行频繁更改时，还需要更多的带宽。
    管理超过 50 个 Compute 节点的控制器，或者同时置备超过四个裸机节点，在非控制器节点上应具有 4-10 倍的接口带宽。

- 在调配超过 50 个 overcloud 节点时，undercloud 应具有与 provisioning 网络的高带宽连接。

- 不要使用与您从工作站访问 director 机器相同的 Provisioning 或 Control Plane NIC。director 安装会使用 Provisioning NIC 创建一个网桥，它会忽略所有远程连接。使用 External NIC 来远程连接 director 系统。

- Provisioning 网络需要一个与您的环境大小相匹配的 IP 范围。使用以下原则来决定包括在这个范围内的 IP 地址总数量：

  - 至少包括一个临时 IP 地址，用于内省期间连接到 Provisioning 网络的每个节点。

  - 至少包括一个永久 IP 地址，用于部署期间连接到 Provisioning 网络的每个节点。

  - 包括一个额外的 IP 地址，用于 Provisioning 网络上 overcloud 高可用性集群的虚拟 IP。

  - 包括此范围内的额外 IP 地址，以用于扩展环境。

- 为防止 Controller 节点网卡或网络交换机故障破坏 overcloud 服务可用性，请确保 keystone 管理端点位于使用绑定网卡或网络硬件冗余的网络中。如果将 Keystone 端点移到不同的网络，如 **internal_api**，请确保 undercloud 可以访问 VLAN 或子网。有关更多信息，请参阅红帽知识库解决方案如何将 Keystone 管理端点迁移到 internal_api 网络 。

2.2. 确定环境规模

在安装 undercloud 前，请确定环境的规模。规划环境时应考虑到以下因素：

想要在 overcloud 中部署多少个节点？

undercloud 管理 overcloud 中的每个节点。置备 overcloud 节点会消耗 undercloud 上的资源。您必须为 undercloud 提供足够的资源用来置备和控制所有 overcloud 节点。

您希望 undercloud 同步执行多少个操作？

undercloud 上的大多数 Red Hat OpenStack Platform (RHOSP)服务都使用一组 worker。每个 worker 执行特定于该服务的一个操作。多个 worker 提供同步操作。undercloud 上 worker 的默认数量是 undercloud 上 CPU 线程总数的一半在本实例中，线程数是指 CPU 内核数乘以超线程值。例如，如果 undercloud 的 CPU 具有 16 个线程，则 director 服务默认生成 8 个 worker。director 还默认使用一组最小值和最大值限制

服务	最小值	最大值
编配服务(heat)	4	24
所有其他服务	2	12

undercloud 具有以下最小 CPU 和内存要求：

- 支持 Intel 64 或 AMD64 CPU 扩展的 8 线程 64 位 x86 处理器。这可为每个 undercloud 服务提供 4 个 worker。
- 最少 24 GB RAM。

要使用更多 worker，使用以下建议增加 undercloud 的 vCPU 和内存：

- **最小值：**每个线程使用 1.5 GB 内存。例如，一台有 48 个线程的机器需要 72 GB RAM 来为 24 个 heat worker 和 12 个其他服务的 worker 提供最小的覆盖范围。
- **建议值：**每个线程使用 3 GB 内存。例如，一台有 48 个线程的机器需要 144 GB RAM 来为 24 个 heat worker 和 12 个其他服务的 worker 提供建议的覆盖范围。

2.3. UNDERCLOUD 磁盘大小调整

建议的最小 undercloud 磁盘大小为在根磁盘上有 **100 GB** 可用磁盘空间：

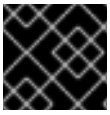
- 20 GB 用于容器镜像
- 10 GB 在节点部署过程中用于 QCOW2 镜像转换和缓存
- 70 GB 或更大空间供常规使用、保存日志和指标以及满足增长需要

2.4. 虚拟化 UNDERCLOUD 节点支持

红帽仅支持以下平台上的虚拟化 undercloud：

平台	备注

平台	备注
基于内核的虚拟机 (KVM)	由 Red Hat Enterprise Linux 托管，如 Red Hat OpenStack Platform、OpenShift Virtualization 和带有 KVM 的 Red Hat Enterprise Linux 中认证的客户机操作系统中列出的
Microsoft Hyper-V	由各种版本的 Hyper-V 托管，如红帽客户门户认证目录上所列。
VMware ESX 和 ESXi	由各种版本的 ESX 和 ESXi 托管，如红帽客户门户认证目录上所列。



重要

确定您的管理程序支持 Red Hat Enterprise Linux 9.2 客户机。

虚拟机要求

虚拟 undercloud 的资源要求与裸机 undercloud 的资源要求类似。在置备时，请考虑各种调优选项，如网络模型、客户机 CPU 功能、存储后端、存储格式和缓存模式。

网络注意事项

电源管理

undercloud 虚拟机(VM)需要访问 overcloud 节点的电源管理设备。这是注册节点时为 **pm_addr** 参数设置的 IP 地址。

Provisioning 网络

用于 provisioning 网络 **ctlplane** 的 NIC 需要能够广播和为 overcloud 的裸机节点的 NIC 提供 DHCP 请求。创建一个网桥，将虚拟机的 NIC 连接到与裸机 NIC 相同的网络。

允许来自未知地址的流量

您必须配置虚拟 undercloud hypervisor (VMware ESX 或 ESXi)，以防止虚拟机监控程序阻止 undercloud 从未知地址传输流量：

- 在 IPv4 **ctlplane** 网络上：允许伪传输。
- 在 IPv6 **ctlplane** 网络上：允许伪传输、MAC 地址更改和混杂模式操作。
  有关如何配置 VMware ESX 或 ESXi 的更多信息，请参阅 VMware 文档网站上 保护 vSphere 标准切换。

应用这些设置后，必须关闭再打开 director 虚拟机。仅重启虚拟机是不够的。

2.5. UNDERCLOUD 软件仓库

您可以在 Red Hat Enterprise Linux (RHEL) 9.2 上运行 Red Hat OpenStack Platform (RHOSP) 17.1。

注意

如果您将软件仓库与 Red Hat Satellite 同步，您可以启用 Red Hat Enterprise Linux 软件仓库的特定版本。但是，无论选择哪种版本，软件仓库均保持不变。例如，您可以启用 BaseOS 存储库的 9.2 版本，但存储库名称仍然是 **rhel-9-for-x86_64-baseos-eus-rpms**，尽管您选择的特定版本。

警告

除此处指定的软件仓库外，不支持在此指定的软件仓库。除非建议，否则请不要启用除下表中列出的其他产品或软件仓库之外，否则可能会遇到软件包依赖关系问题。请勿启用 Extra Packages for Enterprise Linux (EPEL)。

核心软件仓库

下表列出了用于安装 undercloud 的核心软件仓库。

名称	软件仓库	要求说明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-baseos-eus-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-eus-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat Enterprise Linux 9 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux 的高可用性工具。用于 Controller 节点的高可用性功能。
Red Hat OpenStack Platform for RHEL 9 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	Red Hat OpenStack Platform 核心软件仓库，包含 Red Hat OpenStack Platform director 的软件包。
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	为 OpenStack Platform 提供 Open vSwitch (OVS) 软件包。

第 3 章 DIRECTOR 安装准备

要安装和配置 director，您必须完成一些准备任务，以确保已将 undercloud 注册到红帽客户门户网站或 Red Hat Satellite 服务器，您已安装了 director 软件包，并且您已为 director 配置了容器镜像源，以便在安装过程中拉取容器镜像。

3.1. 准备 UNDERCLOUD

在安装 director 前，您必须在主机上完成一些基本配置。

步骤

1. 以 **root** 用户身份登录 undercloud。
2. 创建 **stack** 用户：

```
[root@director ~]# useradd stack
```

3. 为该用户设置密码：

```
[root@director ~]# passwd stack
```

4. 进行以下操作，以使用户在使用 **sudo** 时无需输入密码：

```
[root@director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
[root@director ~]# chmod 0440 /etc/sudoers.d/stack
```

5. 切换到新的 **stack** 用户：

```
[root@director ~]# su - stack
[stack@director ~]$
```

6. 为系统镜像和 heat 模板创建目录：

```
[stack@director ~]$ mkdir ~/images
[stack@director ~]$ mkdir ~/templates
```

   director 使用系统镜像和 heat 模板来创建 overcloud 环境。红帽建议创建这些目录来帮助您组织本地文件系统。

7. 检查 undercloud 的基础和完整主机名：

```
[stack@director ~]$ hostname
[stack@director ~]$ hostname -f
```

   如果上述命令没有显示正确的完全限定主机名或报告错误，则使用 **hostnamectl** 设置主机名：

```
[stack@director ~]$ sudo hostnamectl set-hostname undercloud.example.com
```

8. 如果您所使用的 DNS 服务器无法解析 undercloud 主机完全限定域名 (FQDN)，请编辑 **/etc/hosts** 并为系统主机名包含一个条目。**/etc/hosts** 中的 IP 地址必须与您计划用于 undercloud 公共 API 的地址匹配。例如，如果系统使用 **undercloud.example.com** 作为 FQDN，使用

10.0.0.1 作为 IP 地址，则将以下行添加到 **/etc/hosts**：

```
10.0.0.1  undercloud.example.com undercloud
```

9. 如果您计划让 Red Hat OpenStack Platform director 位于 overcloud 或其身份提供程序之外的独立域，则必须将额外的域添加到 /etc/resolv.conf：

```
search overcloud.com idp.overcloud.com
```

重要

您必须为 DNS 启用端口扩展(**dns_domain_ports**)的 DNS 域，以便内部解析 RHOSP 环境中端口的名称。使用 **NeutronDnsDomain** 默认值 **openstacklocal** 意味着网络服务不会内部解析 DNS 的端口名称。如需更多信息，*请参阅配置 Red Hat OpenStack Platform 网络* 中的 指定 DNS 分配给端口的名称。

3.2. 注册 UNDERCLOUD 并附加订阅

在安装 director 前，您必须运行 **subscription-manager** 来注册 undercloud 并附加有效的 Red Hat OpenStack Platform 订阅。

步骤

1. 以 **stack** 用户身份登录 undercloud。
2. 在红帽 Content Delivery Network 或 Red Hat Satellite 注册您的系统。例如，运行以下命令在 Content Delivery Network 中注册系统。根据提示输入您的客户门户网站用户名和密码：

```
[stack@director ~]$ sudo subscription-manager register
```

3. 查找 Red Hat OpenStack Platform (RHOSP) director 的权利池 ID：

```
[stack@director ~]$ sudo subscription-manager list --available --all --matches="Red Hat
OpenStack"
Subscription Name:   Name of SKU
Provides:            Red Hat Single Sign-On
                     Red Hat Enterprise Linux Workstation
                     Red Hat CloudForms
                     Red Hat OpenStack
                     Red Hat Software Collections (for RHEL Workstation)
SKU:                 SKU-Number
Contract:            Contract-Number
Pool ID:             Valid-Pool-Number-123456
Provides Management: Yes
Available:           1
Suggested:           1
Service Level:       Support-level
Service Type:        Service-Type
Subscription Type:   Sub-type
Ends:                End-date
System Type:         Physical
```

4. 找到 **池 ID** 值并附加 Red Hat OpenStack Platform 17.1 权利：

```
[stack@director ~]$ sudo subscription-manager attach --pool=Valid-Pool-Number-123456
```

5. 将 undercloud 锁定到 Red Hat Enterprise Linux 9.2:

```
$ sudo subscription-manager release --set=9.2
```

3.3. 为 UNDERCLOUD 启用软件仓库

启用 undercloud 所需的软件仓库，并将系统软件包更新至最新版本。

步骤

1. 以 **stack** 用户身份登录 undercloud。
2. 禁用所有默认软件仓库，并启用所需的 Red Hat Enterprise Linux (RHEL)软件仓库：

```
[stack@director ~]$ sudo subscription-manager repos --disable=*
[stack@director ~]$ sudo subscription-manager repos \
--enable=rhel-9-for-x86_64-baseos-eus-rpms          \
--enable=rhel-9-for-x86_64-appstream-eus-rpms       \
--enable=rhel-9-for-x86_64-highavailability-eus-rpms \
--enable=openstack-17.1-for-rhel-9-x86_64-rpms \
--enable=fast-datapath-for-rhel-9-x86_64-rpms
```

   这些仓库包括了安装 director 所需的软件包。

3. 在系统上执行更新，确保您有最新的基本系统软件包：

```
[stack@director ~]$ sudo dnf update -y
[stack@director ~]$ sudo reboot
```

4. 安装用于安装和配置 director 的命令行工具：

```
[stack@director ~]$ sudo dnf install -y python3-tripleoclient
```

3.4. 准备容器镜像

undercloud 安装需要一个环境文件来确定从何处获取容器镜像以及如何存储它们。生成并自定义可用于准备容器镜像的环境文件。

注意

如果需要为您的 undercloud 配置特定的容器镜像版本，必须将镜像固定到特定的版本。有关更多信息，请参阅为 undercloud 固定容器镜像。

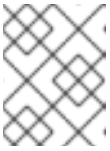
流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 生成默认的容器镜像准备文件：

```
$ openstack tripleo container image prepare default \
  --local-push-destination \
  --output-env-file containers-prepare-parameter.yaml
```

此命令包括以下附加选项：

- **--local-push-destination**，在 undercloud 上设置 registry 作为存储容器镜像的位置。这意味着 director 从 Red Hat Container Catalog 拉取必要的镜像并将其推送到 undercloud 上的 registry 中。director 将该 registry 用作容器镜像源。如果直接从 Red Hat Container Catalog 拉取镜像，请忽略这个选项。
- **--output-env-file** 是环境文件名称。此文件的内容包括用于准备您的容器镜像的参数。在本例中，文件的名称是 **containers-prepare-parameter.yaml**。



注意

您可以使用相同的 **containers-prepare-parameter.yaml** 文件为 undercloud 和 overcloud 定义容器镜像源。

3. 修改 **containers-prepare-parameter.yaml** 以符合您的需求。有关容器镜像参数的更多信息，请参阅 容器镜像准备参数。

3.5. 从私有 REGISTRY 获取容器镜像

registry.redhat.io registry 需要身份验证才能访问和拉取镜像。要通过 **registry.redhat.io** 和其他私有 registry 进行身份验证，请在 **containers-prepare-parameter.yaml** 文件中包括 **ContainerImageRegistryCredentials** 和 **ContainerImageRegistryLogin** 参数。

ContainerImageRegistryCredentials

有些容器镜像 registry 需要进行身份验证才能访问镜像。在这种情况下，请使用您的 **containers-prepare-parameter.yaml** 环境文件中的 **ContainerImageRegistryCredentials** 参数。**ContainerImageRegistryCredentials** 参数使用一组基于私有 registry URL 的键。每个私有 registry URL 使用其自己的键和值对定义用户名（键）和密码（值）。这提供了一种为多个私有 registry 指定凭据的方法。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      namespace: registry.redhat.io/...
      ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      my_username: my_password
```

在示例中，用身份验证凭据替换 **my_username** 和 **my_password**。红帽建议创建一个 registry 服务帐户并使用这些凭据访问 **registry.redhat.io** 内容，而不使用您的个人用户凭据。

要指定多个 registry 的身份验证详情，请在 **ContainerImageRegistryCredentials** 中为每个 registry 设置多个键对值：

```
parameter_defaults:
  ContainerImagePrepare:
```

```
  - push_destination: true
    set:
      namespace: registry.redhat.io/...
      ...
  - push_destination: true
    set:
      namespace: registry.internalsite.com/...
      ...
  ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      myuser: 'p@55w0rd!'
    registry.internalsite.com:
      myuser2: '0th3rp@55w0rd!'
    '192.0.2.1:8787':
      myuser3: '@n0th3rp@55w0rd!'
```

重要

默认 **ContainerImagePrepare** 参数从需要进行身份验证的 **registry.redhat.io** 拉取容器镜像。

如需更多信息，请参阅 Red Hat Container Registry 身份验证 。

ContainerImageRegistryLogin

ContainerImageRegistryLogin 参数用于控制 overcloud 节点系统是否需要登录到远程 registry 来获取容器镜像。当您想让 overcloud 节点直接拉取镜像，而不是使用 undercloud 托管镜像时，会出现这种情况。

如果 **push_destination** 设置为 false 或未用于给定策略，则必须将 **ContainerImageRegistryLogin** 设置为 **true**。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: false
    set:
      namespace: registry.redhat.io/...
      ...
  ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      myuser: 'p@55w0rd!'
  ContainerImageRegistryLogin: true
```

但是，如果 overcloud 节点没有与 **ContainerImageRegistryCredentials** 中定义的 registry 主机的网络连接，并将此 **ContainerImageRegistryLogin** 设置为 **true**，则尝试进行登录时部署可能会失败。如果 overcloud 节点没有与 **ContainerImageRegistryCredentials** 中定义的 registry 主机的网络连接，请将 **push_destination** 设置为 **true**，将 **ContainerImageRegistryLogin** 设置为 **false**，以便 overcloud 节点从 undercloud 拉取镜像。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
```

```
    namespace: registry.redhat.io/...
    ...
...
ContainerImageRegistryCredentials:
  registry.redhat.io:
    myuser: 'p@55w0rd!'
ContainerImageRegistryLogin: false
```

第 4 章 在 UNDERCLOUD 上安装 DIRECTOR

要配置并安装 director，请在 **undercloud.conf** 文件中设置适当的参数，并运行 undercloud 安装命令。安装 director 后，导入 director 将在节点置备过程中写入裸机节点的 overcloud 镜像。

4.1. 配置 UNDERCLOUD

您必须先配置 undercloud，然后才能安装 director。您可以在 **undercloud.conf** 文件中配置 undercloud，director 从 **stack** 用户的主目录中读取该文件。

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. 将 **undercloud.conf** 文件示例复制到 **stack** 用户的主目录：

```
[stack@director ~]$ cp \
  /usr/share/python-tripleoclient/undercloud.conf.sample \
  ~/undercloud.conf
```

3. 修改 **undercloud.conf** 文件中用于部署的参数值。有关可用于配置 undercloud 的参数的详情，请参考 Undercloud 配置参数。

注意

如果省略或注释掉参数，director 将使用默认值。

4. 保存 **undercloud.conf** 文件。

4.2. UNDERCLOUD 配置参数

以下列表包含用于配置 **undercloud.conf** 文件的参数的相关信息。将所有参数保留在相关部分内以避免出错。

重要

您至少必须将 **container_images_file** 参数设置为包含容器镜像配置的环境文件。如果没有将此参数正确设置为适当的文件，则 director 无法从 **ContainerImagePrepare** 参数获取容器镜像规则集，也无法从 **ContainerImageRegistryCredentials** 参数获取容器 registry 身份验证详情。

默认值

以下参数会在 **undercloud.conf** 文件的 **[DEFAULT]** 部分中进行定义：

additional_architectures

overcloud 支持的附加（内核）架构的列表。目前，overcloud 仅支持 **x86_64** 架构。

certificate_generation_ca

为所请求证书签名的 CA 的 **certmonger** 别名。仅在设置了 **generate_service_certificate** 参数的情况下使用此选项。如果您选择 **local** CA，certmonger 会将本地 CA 证书提取到 **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** 并将该证书添加到信任链中。

clean_nodes

确定是否在部署之间和内省之后擦除硬盘。

cleanup

删除临时文件。把它设置为 **False** 以保留部署期间使用的临时文件。如果出现错误，临时文件可帮助您调试部署。

container_cli

用于容器管理的 CLI 工具。将此参数设置为 **podman**。Red Hat Enterprise Linux 9.2 只支持 **podman**。

container_healthcheck_disabled

禁用容器化服务运行状况检查。红帽建议您启用运行状况检查，并将此选项设置为 **false**。

container_images_file

含有容器镜像信息的 Heat 环境文件。此文件可能包含以下条目：

- 所有需要的容器镜像的参数
- **ContainerImagePrepare** 参数（用于推动必要的镜像准备）。通常，含有此参数的文件被命名为 **containers-prepare-parameter.yaml**。

container_insecure_registries

供 **podman** 使用的不安全 registry 列表。如果您想从其他来源（如私有容器 registry）拉取镜像，则使用此参数。在大多数情况下，如果在 Satellite 中注册了 undercloud，**podman** 就有从 Red Hat Container Catalog 或 Satellite Server 拉取容器镜像的证书。

container_registry_mirror

配置的 **podman** 使用的可选 **registry-mirror**。

custom_env_files

要添加到 undercloud 安装中的其他环境文件。

deployment_user

安装 undercloud 的用户。如果此参数保留为不设置，则使用当前的默认用户 **stack**。

discovery_default_driver

为自动注册的节点设置默认驱动程序。需要启用 **enable_node_discovery** 参数，且必须在 **enabled_drivers** 列表中包含驱动程序。

enable_ironic; enable_ironic_inspector; enable_tempest; enable_validations

定义要为 director 启用的核心服务。保留这些参数设为 **true**。

enable_node_discovery

自动注册通过 PXE 引导内省虚拟内存盘 (ramdisk) 的所有未知节点。新节点使用 **fake** 作为默认驱动程序，但您可以设置 **discovery_default_driver** 覆盖它。您也可以使用内省规则为新注册的节点指定驱动程序信息。

enable_routed_networks

定义是否支持路由的 control plane 网络。

enabled_hardware_types

要为 undercloud 启用的硬件类型的列表。

generate_service_certificate

定义 undercloud 安装期间是否生成 SSL/TLS 证书，此证书用于 **undercloud_service_certificate** 参数。undercloud 安装会保存生成的证书 **/etc/pki/tls/certs/undercloud-[undercloud_public_vip].pem**。**certificate_generation_ca** 参数中定义的 CA 将为此证书签名。

heat_container_image

要使用的 heat 容器镜像的 URL。请保留不设置。

heat_native

使用 **heat-all** 运行基于主机的 undercloud 配置。请保留为 **true**。

hieradata_override

在 director 上配置 Puppet hieradata 的 **hieradata** 覆盖文件的路径，为 **undercloud.conf** 参数外的服务提供自定义配置。如果设置此参数，undercloud 安装会将此文件复制到 **/etc/puppet/hieradata** 目录并将其设为层次结构中的第一个文件。有关使用此功能的更多信息，请参阅在 undercloud 上配置 hieradata。

inspection_extras

指定在内省的过程中是否启用额外的硬件集合。此参数在内省镜像上需要 **python-hardware** 或 **python-hardware-detect** 软件包。

inspection_interface

该 director 用来进行节点内省的网桥。这是 director 配置创建的自定义网桥。**LOCAL_INTERFACE** 会附加到这个网桥。请保留使用默认的值（**br-ctlplane**）。

inspection_runbench

在节点内省过程中运行一组基准测试。将此参数设为 **true** 以启用基准测试。如果您需要在检查注册节点的硬件时执行基准数据分析操作，则需要使用这个参数。

ipv6_address_mode

undercloud 置备网络的 IPv6 地址配置模式。以下列表包含这个参数的可能值：

- dhcpv6-stateless - 使用路由器公告 (RA) 的地址配置以及使用 DHCPv6 的可选信息。
- DHCPv6-stateful - 地址配置和使用 DHCPv6 的可选信息。

ipxe_enabled

定义使用 iPXE 还是标准的 PXE。默认为 **true**，其启用 iPXE。将此参数设置为 **false** 以使用标准 PXE。对于 PowerPC 部署，或混合了 PowerPC 和 x86 的部署，请将此值设置为 **false**。

local_interface

指定 director Provisioning NIC 的接口。这也是该 director 用于 DHCP 和 PXE 引导服务的设备。把这个项的值改为您选择的设备。使用 **ip addr** 命令可以查看连接了哪些设备。以下是一个 **ip addr** 命令的结果输出示例：

```
2: em0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 52:54:00:75:24:09 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.178/24 brd 192.168.122.255 scope global dynamic em0
       valid_lft 3462sec preferred_lft 3462sec
    inet6 fe80::5054:ff:fe75:2409/64 scope link
       valid_lft forever preferred_lft forever
3: em1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noop state DOWN
    link/ether 42:0b:c2:a5:c1:26 brd ff:ff:ff:ff:ff:ff
```

在这个例子中，External NIC 使用 **em0**，Provisioning NIC 使用 **em1**（当前没有被配置）。在这种情况下，将 **local_interface** 设置为 **em1**。配置脚本会把这个接口附加到一个自定义的网桥（由 **inspection_interface** 参数定义）上。

local_ip

为 director Provisioning NIC 定义的 IP 地址。这也是 director 用于 DHCP 和 PXE 引导服务的 IP 地址。除非 Provisioning 网络需要使用其他子网（如该 IP 地址与环境中的现有 IP 地址或子网冲突）保留默认值 **192.168.24.1/24**。

对于 IPv6，本地 IP 地址前缀长度必须是 **/64**，才能支持有状态和无状态连接。

local_mtu

要用于 **local_interface** 的最大传输单元 (MTU)。对于 undercloud 不要超过 1500。

local_subnet

要用于 PXE 引导和 DHCP 接口的本地子网。**local_ip** 地址应该属于这个子网。默认值为 **ctlplane-subnet**。

net_config_override

网络配置覆盖模板的路径。如果设置此参数，undercloud 将使用 JSON 或 YAML 格式的模板以使用 **os-net-config** 配置网络，并忽略 **undercloud.conf** 中设置的网络参数。当您要配置绑定或向接口添加一个选项时，请使用此参数。有关自定义 undercloud 网络接口的更多信息，请参阅配置 undercloud 网络接口。

networks_file

覆盖用于 **heat** 的网络文件。

output_dir

输出状态目录、处理的 heat 模板和 Ansible 部署文件。

overcloud_domain_name

要在部署 overcloud 时使用的 DNS 域名。

注意

配置 overcloud 时，必须将 **CloudDomain** 参数设置为匹配的值。配置 overcloud 时，在环境文件中设置此参数。

roles_file

要用来覆盖用于 undercloud 安装的默认角色文件的角色文件。强烈建议您将此参数保留为不设置，以便 director 安装使用默认的角色文件。

scheduler_max_attempts

调度程序尝试部署实例的次数上限。此值必须大于或等于您期望一次部署的裸机节点数，以避免调度时的潜在争用情形。

service_principal

使用该证书的服务的 Kerberos 主体。仅在您的 CA 需要 Kerberos 主体（如在 FreeIPA 中）时使用此参数。

subnets

用于置备和内省的路由网络子网的列表。默认值仅包括 **ctlplane-subnet** 子网。如需更多信息，请参阅 子网。

templates

要覆盖的 heat 模板文件。

undercloud_admin_host

通过 SSL/TLS 为 director 管理 API 端点定义的 IP 地址或主机名。director 配置将 IP 地址作为路由的 IP 地址附加到 director 软件网桥，其使用 **/32** 子网掩码。

如果 **undercloud_admin_host** 不在与 **local_ip** 相同的 IP 网络中，您必须配置您希望 undercloud 上的 admin API 侦听的接口。默认情况下，admin API 侦听 **br-ctlplane** 接口。有关如何配置 undercloud 网络接口的详情，请参考 配置 undercloud 网络接口。

undercloud_debug

把 undercloud 服务的日志级别设置为 **DEBUG**。将此值设置为 **true** 以启用 **DEBUG** 日志级别。

undercloud_enable_selinux

在部署期间启用或禁用 SELinux。除非调试问题，否则强烈建议保留此值设为 **true**。

undercloud_hostname

定义 undercloud 的完全限定主机名。如果设置，undercloud 安装将配置所有系统主机名设置。如果保留未设置，undercloud 将使用当前的主机名，但您必须相应地配置所有主机名设置。

undercloud_log_file

用于存储 undercloud 安装和升级日志的日志文件的路径。默认情况下，日志文件是主目录中的 **install-undercloud.log**。例如，**/home/stack/install-undercloud.log**。

undercloud_nameservers

用于 undercloud 主机名解析的 DNS 名称服务器列表。

undercloud_ntp_servers

帮助同步 undercloud 日期和时间的网络时间协议服务器列表。

undercloud_public_host

通过 SSL/TLS 为 director 公共 API 端点定义的 IP 地址或主机名。director 配置将 IP 地址作为路由的 IP 地址附加到 director 软件网桥，其使用 **/32** 子网掩码。
如果 **undercloud_public_host** 不在与 **local_ip** 相同的 IP 网络中，您必须将 **PublicVirtualInterface** 参数设置为您希望 undercloud 上公共 API 侦听的接口。默认情况下，公共 API 侦听 **br-ctlplane** 接口。在自定义环境文件中设置 **PublicVirtualInterface** 参数，并通过配置 **custom_env_files** 参数在 **undercloud.conf** 文件中包含自定义环境文件。

有关自定义 undercloud 网络接口的详情，请参考 配置 undercloud 网络接口。

undercloud_service_certificate

用于 OpenStack SSL/TLS 通信的证书的位置和文件名。理想的情况是从一个信任的证书认证机构获得这个证书。否则，生成自己的自签名证书。

undercloud_timezone

undercloud 的主机时区。如果未指定时区，director 将使用现有时区配置。

undercloud_update_packages

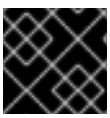
定义是否在安装 undercloud 期间更新软件包。

子网

每个置备子网在 **undercloud.conf** 文件中都有一个对应的同名部分。例如，要创建称为 **ctlplane-subnet** 的子网，在 **undercloud.conf** 文件中使用以下示例：

```
[ctlplane-subnet]
cidr = 192.168.24.0/24
dhcp_start = 192.168.24.5
dhcp_end = 192.168.24.24
inspection_iprange = 192.168.24.100,192.168.24.120
gateway = 192.168.24.1
masquerade = true
```

您可以根据自身环境所需来指定相应数量的置备网络。



重要

director 在创建子网后无法更改子网的 IP 地址。

cidr

director 用来管理 overcloud 实例的网络。这是 undercloud **neutron** 服务管理的 Provisioning 网络。保留其默认值 **192.168.24.0/24**，除非您需要 Provisioning 网络使用其他子网。

masquerade

定义是否伪装 **cidr** 中定义的用于外部访问的网络。这为 Provisioning 网络提供了网络地址转换 (NAT)，使 Provisioning 网络能够通过 director 进行外部访问。

注意

director 配置还使用相关 **sysctl** 内核参数自动启用 IP 转发。

dhcp_start; dhcp_end

overcloud 节点 DHCP 分配范围的开始值和终止值。确保此范围包含分配给节点的足够 IP 地址。如果没有为子网指定，director 通过从子网的完整的 IP 范围内删除为 **local_ip**, **gateway**, **undercloud_admin_host**, **undercloud_public_host**, and **inspection_iprange** 参数设置的值来决定。
您可以通过指定开始和结束地址对列表，为 undercloud control plane 子网配置非连续分配池。另外，您可以使用 **dhcp_exclude** 选项排除 IP 地址范围中的 IP 地址。例如，以下配置都创建分配池 **172.20.0.100-172.20.0.150** 和 **172.20.0.200-172.20.0.250** ：

选项 1

```
dhcp_start = 172.20.0.100,172.20.0.200
dhcp_end = 172.20.0.150,172.20.0.250
```

选项 2

```
dhcp_start = 172.20.0.100
dhcp_end = 172.20.0.250
dhcp_exclude = 172.20.0.151-172.20.0.199
```

dhcp_exclude

DHCP 分配范围中排除的 IP 地址。例如，以下配置排除 IP 地址 **172.20.0.105** 和 IP 地址范围 **172.20.0.210-172.20.0.219** ：

```
dhcp_exclude = 172.20.0.105,172.20.0.210-172.20.0.219
```

dns_nameservers

特定于子网的 DNS 名称服务器。如果没有为子网定义名称服务器，子网将使用 **undercloud_nameservers** 参数中定义的名称服务器。

gateway

overcloud 实例的网关。它是 undercloud 主机，会把网络流量转发到外部网络。保留其默认值 **192.168.24.1**，除非您需要 director 使用其他 IP 地址，或想直接使用外部网关。

host_routes

此网络上 overcloud 实例的 Neutron 管理的子网的主机路由。这也为 undercloud 上的 **local_subnet** 配置主机路由。

inspection_iprange

在检查过程中，此网络上的节点要使用的临时 IP 范围。这个范围不得与 **dhcp_start** 和 **dhcp_ end** 定义的范围重叠，但必须位于同一个 IP 子网中。

4.3. 使用环境文件配置 UNDERCLOUD

您通过 **undercloud.conf** 文件配置 undercloud 的主要参数。您还可以使用包含 heat 参数的环境文件来执行额外的 undercloud 配置。

步骤

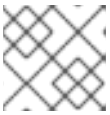
1. 创建名为 **/home/stack/templates/custom-undercloud-params.yaml** 的环境文件。
2. 编辑此文件并包括您的 heat 参数。例如，要为特定的 OpenStack Platform 服务启用调试功能，请在 **custom-undercloud-params.yaml** 文件中包括以下代码段：

```
parameter_defaults:
  Debug: True
```

   有关您可以为 undercloud 配置的 heat 参数的信息，请参阅 undercloud 配置的通用 heat 参数。

3. 保存环境文件。
4. 编辑 **undercloud.conf** 文件，找到 **custom_env_files** 参数。编辑该参数以指向 **custom-undercloud-params.yaml** 环境文件：

```
custom_env_files = /home/stack/templates/custom-undercloud-params.yaml
```



注意

您可以使用逗号分隔列表指定多个环境文件。

director 安装在下次安装或升级 undercloud 的操作过程中包括此环境文件。

4.4. 用于 UNDERCLOUD 配置的常见 HEAT 参数

下表包含您可能在自定义环境文件中为 undercloud 设置的一些常见 heat 参数。

参数	描述
AdminPassword	设置 undercloud **admin** 用户密码。
AdminEmail	设置 undercloud **admin** 用户电子邮件地址。
Debug	启用调试模式。

在自定义环境文件的 **parameter_defaults** 部分下设置这些参数：

```
parameter_defaults:
  Debug: True
  AdminPassword: "myp@ssw0rd!"
  AdminEmail: "admin@example.com"
```

-

4.5. 在 UNDERCLOUD 上配置 HIERADATA

您可以通过在 director 上配置 Puppet hieradata，为可用 **undercloud.conf** 参数之外的服务提供自定义配置。

步骤

1. 创建一个 hieradata 覆盖文件，例如 **/home/stack/hieradata.yaml**。
2. 将自定义的 hieradata 添加到该文件。例如，添加以下代码段，将 Compute (nova) 服务参数 **force_raw_images** 从默认值 **True** 改为 **False**：

```
nova::compute::force_raw_images: False
```

如果没有为您要设置的参数实施 Puppet，则使用以下方法配置该参数：

```
nova::config::nova_config:
  DEFAULT/<parameter_name>:
    value: <parameter_value>
```

例如：

```
nova::config::nova_config:
  DEFAULT/network_allocate_retries:
    value: 20
  ironic/serial_console_state_timeout:
    value: 15
```

3. 将 **undercloud.conf** 文件中的 **hieradata_override** 参数设置为新 **/home/stack/hieradata.yaml** 文件的路径：

```
hieradata_override = /home/stack/hieradata.yaml
```

4.6. 安装 DIRECTOR

完成以下步骤以安装 director 并执行一些基本安装后任务。

步骤

1. 运行以下命令，以在 undercloud 上安装 director：

```
[stack@director ~]$ openstack undercloud install
```

此命令会启动 director 配置脚本。director 安装附加软件包，根据 **undercloud.conf** 中的配置配置其服务，并启动所有 RHOSP 服务容器。这个脚本会需要一些时间来完成。

此脚本会生成两个文件：

- **/home/stack/tripleo-deploy/undercloud/tripleo-undercloud-passwords.yaml** - director 服务的所有密码列表。
- **/home/stack/stackrc** - 一组初始化变量，可帮助您访问 director 命令行工具。

2. 确认 RHOSP 服务容器正在运行：

```
[stack@director ~]$ sudo podman ps -a --format "{{.Names}} {{.Status}}"
```

以下命令输出显示 RHOSP 服务容器正在运行(**Up**)：

```
memcached Up 3 hours (healthy)
haproxy Up 3 hours
rabbitmq Up 3 hours (healthy)
mysql Up 3 hours (healthy)
iscsid Up 3 hours (healthy)
keystone Up 3 hours (healthy)
keystone_cron Up 3 hours (healthy)
neutron_api Up 3 hours (healthy)
logrotate_crond Up 3 hours (healthy)
neutron_dhcp Up 3 hours (healthy)
neutron_l3_agent Up 3 hours (healthy)
neutron_ovs_agent Up 3 hours (healthy)
ironic_api Up 3 hours (healthy)
ironic_conductor Up 3 hours (healthy)
ironic_neutron_agent Up 3 hours (healthy)
ironic_pxe_tftp Up 3 hours (healthy)
ironic_pxe_http Up 3 hours (unhealthy)
ironic_inspector Up 3 hours (healthy)
ironic_inspector_dnsmasq Up 3 hours (healthy)
neutron-dnsmasq-qdhcp-30d628e6-45e6-499d-8003-28c0bc066487 Up 3 hours
...
```

3. 运行以下命令初始化 **stack** 用户来使用命令行工具：

```
[stack@director ~]$ source ~/stackrc
```

提示现在指示 OpenStack 命令对 undercloud 进行验证并执行；

```
(undercloud) [stack@director ~]$
```

director 的安装已完成。您现在可以使用 director 命令行工具了。

4.7. 为 OVERCLOUD 节点获取镜像

director 需要几个磁盘镜像用于置备 overcloud 节点：

- 一个内省内核和 ramdisk 用于通过 PXE 引导进行裸机系统内省。
- 一个部署内核和 ramdisk 用于系统置备和部署。
- overcloud 内核、ramdisk 和完整镜像组成了 director 写入节点的硬盘的基本 overcloud 系统。

您可以获取并安装您需要的镜像。当您不想运行任何其他 Red Hat OpenStack Platform (RHOSP)服务或消耗其中一个订阅权利时，您还可以获取并安装基本镜像来置备裸机操作系统。

注意

如果您的 RHOSP 部署使用 IPv6，您必须修改 overcloud 镜像以禁用 **cloud-init** 网络配置。有关修改镜像的更多信息，请参阅红帽知识库解决方案 使用 virt-customize 修改 Red Hat Linux OpenStack Platform Overcloud 镜像。

4.7.1. 安装 overcloud 镜像

您的 Red Hat OpenStack Platform (RHOSP)安装包括了为您提供 director 的 **overcloud-hardened-uefi-full.qcow2** overcloud 镜像的软件包。该镜像是部署具有默认 CPU 架构 x86-64 的 overcloud 所必需的。将此镜像导入到 director 也会在 director PXE 服务器上安装内省镜像。

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. Source **stackrc** 文件：

```
[stack@director ~]$ source ~/stackrc
```

3. 安装 **rhosp-director-images-uefi-x86_64** 和 **rhosp-director-images-ipa-x86_64** 软件包：

```
(undercloud) [stack@director ~]$ sudo dnf install rhosp-director-images-uefi-x86_64 rhosp-director-images-ipa-x86_64
```

4. 在 **stack** 用户的主目录中创建 **images** 目录 **/home/stack/images** ：

```
(undercloud) [stack@director ~]$ mkdir /home/stack/images
```

如果目录已存在，请跳过这一步。

5. 将镜像存档提取到 **images** 目录中：

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ for i in /usr/share/rhosp-director-images/ironic-python-agent-latest.tar /usr/share/rhosp-director-images/overcloud-hardened-uefi-full-latest.tar; do tar -xvf $i; done
```

6. 将镜像导入 director：

```
(undercloud) [stack@director images]$ openstack overcloud image upload --image-path /home/stack/images/
```

此命令将镜像格式从 QCOW 转换为 RAW，并提供镜像上传进度状态的详细更新。

7. 验证 overcloud 镜像是否已复制到 **/var/lib/ironic/images/** 中：

```
(undercloud) [stack@director images]$ ls -l /var/lib/ironic/images/
total 1955660
-rw-r--r--. 1 root 42422 40442450944 Jan 29 11:59 overcloud-hardened-uefi-full.raw
```

8. 验证 director 是否已将内省 PXE 镜像复制到 **/var/lib/ironic/httpboot** ：

```
(undercloud) [stack@director images]$ ls -l /var/lib/ironic/httpboot
total 417296
-rwxr-xr-x. 1 root  root    6639920 Jan 29 14:48 agent.kernel
-rw-r--r--. 1 root  root  420656424 Jan 29 14:48 agent.ramdisk
-rw-r--r--. 1 42422 42422       758 Jan 29 14:29 boot.ipxe
-rw-r--r--. 1 42422 42422       488 Jan 29 14:16 inspector.ipxe
```

4.7.2. 最小 overcloud 镜像

您可以使用 **overcloud-minimal** 镜像置备裸机操作系统，您可以在其中运行任何其他 Red Hat OpenStack Platform (RHOSP)服务，或消耗其中一个订阅权利。

您的 RHOSP 安装中包含 **overcloud-minimal** 软件包，为您提供 director 的以下 overcloud 镜像：

- **overcloud-minimal**
- **overcloud-minimal-initrd**
- **overcloud-minimal-vmlinuz**

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. Source **stackrc** 文件：

```
[stack@director ~]$ source ~/stackrc
```

3. 安装 **overcloud-minimal** 软件包：

```
(undercloud) [stack@director ~]$ sudo dnf install rhosp-director-images-minimal
```

4. 将镜像存档解包到 **stack** 用户主目录 (**/home/stack/images**) 中的 **images** 目录中：

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ tar xf /usr/share/rhosp-director-images/overcloud-minimal-latest-17.1.tar
```

5. 将镜像导入 director：

```
(undercloud) [stack@director images]$ openstack overcloud image upload --image-path /home/stack/images/ --image-type os --os-image-name overcloud-minimal.qcow2
```

该命令提供镜像上传进度状态的更新：

```
Image "file:///var/lib/ironic/images/overcloud-minimal.vmlinuz" was copied.
+------------------------------------------------------+------------------+----------+
|                         Path                         |       Name       |   Size   |
+------------------------------------------------------+------------------+----------+
| file:///var/lib/ironic/images/overcloud-minimal.vmlinuz | overcloud-minimal | 11172880 |
+------------------------------------------------------+------------------+----------+
Image "file:///var/lib/ironic/images/overcloud-minimal.initrd" was copied.
+------------------------------------------------------+------------------+----------+
```

```
|                    Path                    |       Name       |   Size   |
+--------------------------------------------+------------------+----------+
| file:///var/lib/ironic/images/overcloud-minimal.initrd | overcloud-minimal | 63575845 |
+--------------------------------------------+------------------+----------+
Image "file:///var/lib/ironic/images/overcloud-minimal.raw" was copied.
+-------------------------------------------+------------------+------------+
|                    Path                   |       Name       |    Size    |
+-------------------------------------------+------------------+------------+
| file:///var/lib/ironic/images/overcloud-minimal.raw | overcloud-minimal | 2912878592 |
+-------------------------------------------+------------------+------------+
```

4.8. 更新 UNDERCLOUD 配置

如果需要更改 undercloud 配置以适应新要求，则可在安装后更改 undercloud 配置，请编辑相关配置文件并重新运行 **openstack undercloud install** 命令。

步骤

1. 修改 undercloud 配置文件。例如，编辑 **undercloud.conf** 文件并将 **idrac** 硬件类型添加到已启用硬件类型列表中：

```
enabled_hardware_types = ipmi,redfish,idrac
```

2. 运行 **openstack undercloud install** 命令以使用新更改刷新 undercloud：

```
[stack@director ~]$ openstack undercloud install
```

等待命令运行完成。

3. 初始化 **stack** 用户以使用命令行工具：

```
[stack@director ~]$ source ~/stackrc
```

提示现在指示 OpenStack 命令对 undercloud 进行验证并执行：

```
(undercloud) [stack@director ~]$
```

4. 确认 director 已应用新配置。在此示例中，检查已启用硬件类型列表：

```
(undercloud) [stack@director ~]$ openstack baremetal driver list
+---------------------+---------------------+
| Supported driver(s) | Active host(s)      |
+---------------------+---------------------+
| idrac               | director.example.com |
| ipmi                | director.example.com |
| redfish             | director.example.com |
+---------------------+---------------------+
```

undercloud 重新配置完成。

4.9. UNDERCLOUD 容器 REGISTRY

Red Hat Enterprise Linux 9.2 不再包含 **docker-distribution** 软件包，该软件包安装了 Docker Registry v2。为了保持兼容性和相同的功能级别，director 安装使用称为 **image-serve** 的 vhost 创建 Apache Web 服务器以提供 registry。该 registry 也使用禁用了 SSL 的端口 8787/TCP。基于 Apache 的 registry 未容器化，这意味着您必需运行以下命令以重启 registry：

```
$ sudo systemctl restart httpd
```

您可以在以下位置找到容器 registry 日志：

- /var/log/httpd/image_serve_access.log
- /var/log/httpd/image_serve_error.log。

镜像内容来自 **/var/lib/image-serve**。此位置使用特定目录布局和 **apache** 配置来实施 registry REST API 的拉取功能。

基于 Apache 的 registry 不支持 **podman push** 或 **buildah push** 命令，这意味着您无法使用传统方法推送容器镜像。要在部署过程中修改镜像，请使用容器准备工作流，如 **ContainerImagePrepare** 参数。要管理容器镜像，请使用容器管理命令：

OpenStack tripleo container image list

列出 registry 上存储的所有镜像。

OpenStack tripleo container image show

显示 registry 上特定镜像的元数据。

OpenStack tripleo container image push

将镜像从远程 registry 推送到 undercloud registry。

OpenStack tripleo container image delete

从 registry 中删除镜像。

4.10. 默认 UNDERCLOUD 目录的内容

在 RHOSP 17 中，您可以在单个目录中找到所有配置文件。目录的名称是所用 openstack 命令和堆栈名称的组合。目录具有默认位置，但您可以使用 **--working-dir** 选项更改默认位置。您可以将这个选项与读取或创建用于部署的文件的任何 **tripleoclient** 命令一起使用。

默认位置	命令
$HOME/tripleo-deploy/undercloud	**Undercloud 安装**，它基于 **tripleo 部署**
$HOME/tripleo-deploy/<stack>	**tripleo deploy**, <stack> 默认为 **standalone**
$HOME/overcloud-deploy/<stack>	**overcloud deploy**, <stack> 默认为 **overcloud**

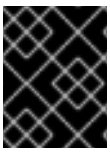
下表详细介绍了 **~/tripleo-deploy/undercloud** 目录中所含的文件和目录。

表 4.1. undercloud 目录内容

目录	描述
heat-launcher	临时 Heat 工作目录包含临时 Heat 配置和数据库备份。
install-undercloud.log	undercloud 安装和升级日志。
tripleo-ansible-inventory.yaml	overcloud 的 Ansible 清单。
tripleo-config-generated-env-files	包含生成的环境文件。
tripleo-undercloud-outputs.yaml	包含保存的 undercloud 输出。
tripleo-undercloud-passwords.yaml	包含 undercloud 密码。
undercloud-install-*.tar.bzip2	工作目录的 tarball，例如 **undercloud-install-20220823210316.tar.bzip2**。

第 5 章 规划您的 OVERCLOUD

以下部分包含在规划 Red Hat OpenStack Platform（RHOSP）环境的各个方面的指导信息。这包括定义节点角色、规划您的网络拓扑结构和存储。



重要

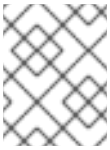
部署 overcloud 节点后，请勿重命名这些节点。在部署后重命名节点会导致实例管理问题。

5.1. 节点角色

director 包含以下默认节点类型用于构建 overcloud：

Controller

提供用于控制环境的关键服务。它包括仪表板服务 (horizon)、认证服务 (keystone)、镜像存储服务 (glance)、联网服务 (neutron)、编配服务 (heat) 以及高可用性服务。Red Hat OpenStack Platform（RHOSP）环境需要三个 Controller 节点以实现高可用生产级环境。



注意

将只有一个 Controller 节点的环境用于测试目的，不应该用于生产环境。不支持由两个 Controller 节点或由三个以上 Controller 节点组成的环境。

计算

用作虚拟机监控程序并包含在环境中运行虚拟机所需的处理能力的物理服务器。基本 RHOSP 环境需要至少一个 Compute 节点。

Ceph Storage

提供 Red Hat Ceph Storage 的一个主机。额外的 Ceph Storage 主机可以在一个集群中扩展。这个部署角色是可选的。

Swift Storage

为 OpenStack Object Storage (swift) 服务提供外部对象存储的主机。这个部署角色是可选的。

下表包含一些不同 overcloud 的示例并为每个场景定义节点类型。

表 5.1. 节点部署示例

	Controller	计算	Ceph Storage	Swift Storage	总计
小型 overcloud	3	1	-	-	4
中型 overcloud	3	3	-	-	6
带有额外对象存储的中型 overcloud	3	3	-	3	9

	Controller	计算	Ceph Storage	Swift Storage	总计
带有 Ceph Storage 集群的中型 overcloud	3	3	3	-	9

此外，还需思考是否要将各个服务划分成不同的自定义角色。有关可组合角色架构的更多信息，请参阅自定义 *Red Hat OpenStack Platform 部署指南中的可组合服务和自定义角色*。https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/customizing_your_red_hat_openstack_platform_deployment/asservices-and-custom-roles

表 5.2. 用于概念验证部署的节点部署角色

	undercloud	Controller	计算	Ceph Storage	总计
概念验证	1	1	1	1	4

警告

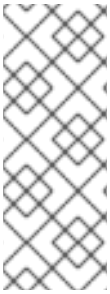
Red Hat OpenStack Platform 在第 2 天运维中维护一个可正常运行的 Ceph Storage 集群。因此，在少于三个 MON 或三个存储节点的部署中，无法进行某些第 2 天运维，如 Ceph Storage 集群的升级或次要更新。如果使用单个 Controller 节点或单个 Ceph Storage 节点，则第 2 天运维将失败。

5.2. OVERCLOUD 网络

规划环境的网络拓扑和子网非常重要，它可以确保映射角色和服务，以使其可以正确地相互通信。Red Hat OpenStack Platform (RHOSP) 使用 Openstack Networking (neutron) 服务，此服务可自主运行，并可管理基于软件的网络、静态和浮动 IP 地址以及 DHCP。

默认情况下，director 配置节点以使用 **Provisioning/Control Plane** 获得连接。不过，可以将网络流量隔离到一系列的可组合网络，供您自定义和分配服务。

在典型的 RHOSP 安装中，网络类型的数量通常会超过物理网络链路的数量。为了可以把所有网络都连接到正确的主机，overcloud 使用 VLAN 标记（VLAN tagging）来为每个接口提供多个网络。大多数网络都是隔离的子网，但有些网络需要第 3 层网关来提供路由用于互联网访问或基础架构网络连接。如果使用 VLAN 来隔离网络流量类型，则必需使用支持 802.1Q 标准的交换机来提供 tagged VLAN。



注意

您可以使用 VLAN 创建项目（租户）网络。您可以为特殊使用网络创建 Geneve 隧道，而无需消耗项目 VLAN。红帽建议您使用 Geneve 部署项目网络隧道，即使您打算在禁用隧道的 neutron VLAN 模式中部署 overcloud。如果您使用 Geneve 部署项目网络隧道，您仍然可以更新您的环境，以使用隧道网络作为实用程序网络或虚拟化网络。可以为带有项目 VLAN 的部署添加 Geneve 功能，但无法将项目 VLAN 添加到现有的 overcloud 中，而不会造成中断。

director 还包括一组模板，可用于使用隔离的可组合网络配置 NIC。以下配置为默认配置：

- 单 NIC 配置 - 一个 NIC 在原生 VLAN 中用于 Provisioning 网络，并用于 tagged VLAN（使用子网处理不同的 overcloud 网络类型）。
- 绑定 NIC 配置 - 一个 NIC 在原生 VLAN 中用于 Provisioning 网络，tagged VLAN 绑定中的两个 NIC 用于不同的 overcloud 网络类型。
- 多 NIC 配置 - 每个 NIC 都使用一个子网来分别处理 overcloud 中不同的网络类型。

您也可以创建自己的模板来映射特定的 NIC 配置。

在考虑网络配置时，以下详细信息也很重要：

- 在 overcloud 创建过程中，在所有 overcloud 机器间使用同一个名称来代表 NIC。理想情况下，您应该在每个 overcloud 节点上对每个相关的网络都使用相同的 NIC 来避免混淆。例如，Provisioning 网络使用主（primary）NIC，OpenStack 服务使用从（secondary）NIC。
- 设置所有 overcloud 系统为从 Provisioning NIC 进行 PXE 引导，并禁用通过外部 NIC 和系统上的任何其他 NIC 进行 PXE 引导。另外，还需要确保 Provisioning NIC 在 PXE 引导设置中位于引导顺序的最上面（在硬盘和 CD/DVD 驱动器之前）。
- 所有 overcloud 裸机系统都需要一个受支持的电源管理接口，如智能平台管理接口 (IPMI)，以便 director 能够控制每个节点的电源管理。
- 请记录下每个 overcloud 系统的以下信息：Provisioning NIC 的 MAC 地址、IPMI NIC 的 IP 地址、IPMI 用户名和 IPMI 密码。稍后配置 overcloud 节点时，这些信息很有用。
- 如果一个实例必须可以被外部互联网访问，则需要从公共网络中分配一个浮动 IP 地址，并把浮动 IP 和这个实例相关联。这个实例仍然会保留它的私人 IP，但网络流量可以通过 NAT 到达浮动 IP 地址。请注意，一个浮动 IP 地址只能分配给一个接口，而不能分配给多个私人 IP 地址。但是，浮动 IP 地址只保留给一个租户使用，这意味着租户可以根据需要将浮动 IP 地址与特定实例相关联或取消关联。此配置会使您的基础架构暴露于外部互联网，您必须使用了适当的安全保护措施。
- 为了减少 Open vSwitch 中出现网络环路的风险，只能有一个接口或一个绑定作为给定网桥的成员。如果需要多个绑定或接口，可以配置多个网桥。
- 红帽建议使用 DNS 主机名解析，以便您的 overcloud 节点能够连接到外部服务，如 Red Hat Content Delivery Network 和网络时间服务器。
- 红帽建议将置备接口、外部接口和任何浮动 IP 接口保留在 1500 的默认 MTU 。否则可能会发生连接问题。这是因为路由器通常无法跨第 3 层边界转发巨型帧。

5.3. OVERCLOUD 存储

您可以使用 Red Hat Ceph Storage 节点作为 overcloud 环境的后端存储。您可以将 overcloud 配置为使用 Ceph 节点进行以下存储类型：

镜像

Image 服务(glance)管理用于创建虚拟机实例的镜像。镜像是不可变的二进制 Blob。您可以使用镜像服务将镜像存储在 Ceph 块设备中。有关支持的镜像格式的详情，请参考 创建和管理镜像 中的镜像服务(glance)。

卷

块存储服务(cinder)为实例管理持久存储卷。块存储服务卷是块设备。您可以使用卷来引导实例，您可以将卷附加到正在运行的实例。您可以使用 Block Storage 服务通过镜像的 copy-on-write clone 来引导虚拟机。

对象（object）

当 overcloud 存储后端是 Red Hat Ceph Storage 时，Ceph 对象网关(RGW)在 Ceph 集群上提供默认的 overcloud 对象存储。如果您的 overcloud 没有 Red Hat Ceph Storage，则 overcloud 将使用 Object Storage 服务(swift)来提供对象存储。您可以将 overcloud 节点专用于对象存储服务。当您需要扩展或替换 overcloud 环境中的 Controller 节点，同时需要在一个高可用性集群外保留对象存储时，这将非常有用。

文件系统

共享文件系统服务(manila)管理共享文件系统。您可以使用共享文件系统服务来管理由 CephFS 文件系统（数据在 Ceph Storage 节点上）支持的共享。

实例磁盘

当您启动实例时，实例磁盘作为文件存储在虚拟机监控程序的实例目录中。默认文件位置为 ***/var/lib/nova/instances****。*

有关 Ceph Storage 的更多信息，请参阅 Red Hat Ceph Storage 架构指南 。

5.3.1. overcloud 存储的配置注意事项

在规划存储配置时请考虑以下问题：

实例安全性和性能

在使用后端块存储卷的实例上使用 LVM 会导致性能、卷可见性和可用性以及数据崩溃问题。使用 LVM 过滤器来缓解问题。如需更多信息，请参阅 配置持久性存储 中的 在 overcloud 节点上启用 LVM2 过滤，使用 cinder 卷上的 LVM 的红帽知识库解决方案会将数据公开给计算主机 。

本地磁盘分区

考虑部署的存储和保留要求，以确定以下默认磁盘分区是否满足您的要求：

分区	默认大小
/	8GB
/tmp	1GB
/var/log	10GB
/var/log/audit	2GB
/home	1GB

分区	默认大小
/var	节点角色依赖： • Object Storage nodes：剩余的磁盘大小的 10%。 • Controller 节点：剩余的磁盘大小的 90% • 非对象存储节点：在分配所有其他分区后，将剩余的磁盘大小存活。
/srv	在 Object Storage 节点上：分配磁盘剩余大小后，所有其他分区被分配。

*要更改为分区分配的磁盘大小，请在 **overcloud-baremetal-deploy.yaml** 节点定义中更新 **ansible_playbooks** 定义中的 **role_growvols_args** 额外 Ansible 变量。如需更多信息，请参阅为对象存储服务配置整个磁盘分区。*

5.4. OVERCLOUD 安全性

OpenStack 平台环境的安全性在一定程度上取决于网络的安全性。遵循网络环境中的良好安全原则，确保正确控制网络访问：

- *使用网络分段缓解网络移动并隔离敏感数据。扁平网络的安全性要低得多。*
- *限制对服务和端口的访问。*
- *强制执行正确的防火墙规则并使用密码。*
- *确保启用 SELinux。*

有关保护系统安全的更多信息，请参阅以下红帽指南：

- *Red Hat Enterprise Linux 9 的 安全强化*
- *为 Red Hat Enterprise Linux 9 使用 SELinux*

5.5. OVERCLOUD 高可用性

要部署高度可用的 overcloud，director 将配置多个 Controller、Compute 和 Storage 节点，并以单一集群的形式协同工作。出现节点故障时，根据故障的节点类型来触发自动隔离和重新生成流程。有关 overcloud 高可用性架构和服务的更多信息，请参阅管理高可用性服务。

注意

不支持在不使用 STONITH 的情况下部署高可用性 overcloud。您必须在高可用性 overcloud 中为作为 Pacemaker 集群一部分的每个节点配置 STONITH 设备。有关 STONITH 和 Pacemaker 的信息，请参阅红帽高可用性集群中的隔离和 RHEL 高可用性集群的支持策略。

您也可以通过 director 为 Compute 实例配置高可用性 (Instance HA)。使用此高可用性机制，可在节点出现故障时在 Compute 节点上自动清空并重新生成实例。对 Instance HA 的要求与一般的 overcloud 要求相同，但必须执行一些附加步骤以准备好环境进行部署。有关 Instance HA 和安装说明的更多信息，请参阅

实例配置高可用性指南。

5.6. CONTROLLER 节点要求

Controller 节点在 Red Hat OpenStack Platform 环境中托管核心服务，如 Dashboard (horizon)、后端数据库服务器、Identity 服务 (keystone) 和高可用性服务。

处理器

支持 Intel 64 或 AMD64 CPU 扩展的 64 位 x86 处理器。

内存

最小内存为 32 GB。不过，建议根据 vCPU 数量（CPU 内核数乘以超线程值）来决定内存大小。使用以下计算确定 RAM 要求：

- ***控制器 RAM 最小值计算：***
  - *每个 vCPU 使用 1.5 GB 内存。例如，拥有 48 个 vCPU 的计算机应当具有 72 GB RAM。*
- ***控制器 RAM 建议值计算：***
  - *每个 vCPU 使用 3 GB 内存。例如，拥有 48 个 vCPU 的计算机应当具有 144 GB RAM。*

有关衡量内存要求的更多信息，请参阅红帽客户门户网站上的"高可用性控制器的 Red Hat OpenStack Platform 硬件要求"。

磁盘存储和布局

如果 Object Storage 服务 (swift) 不在 Controller 节点上运行，则需要最小 50 GB 的存储。但是，Telemetry 和 Object Storage 服务都安装在 Controller 上，且二者均配置为使用根磁盘。这些默认值适合部署在商用硬件上构建的小型 overcloud。这些环境通常用于概念验证和测试环境。您可以使用这些默认布局，只需最少的规划即可部署 overcloud，但它们只能提供很低的工作负载容量和性能。然而在企业环境中，默认布局可能造成很大的瓶颈。这是因为 Telemetry 会不断地访问存储资源，导致磁盘 I/O 使用率很高，从而严重影响所有其他 Controller 服务的性能。在这种环境中，必须规划 overcloud 并进行相应的配置。

网络接口卡

最少两个 1 Gbps 网络接口卡。对绑定的接口使用额外的网络接口卡，或代理标记的 VLAN 流量。

电源管理

每个 Controller 节点在服务器的主板上都要有一个受支持的电源管理接口，如智能平台管理接口 (IPMI) 功能。

5.6.1. 使用 NUMA 时的限制

*Compute 服务(nova)为具有非统一内存访问(NUMA)拓扑的所有虚拟机(VM)强制严格的内存关联性。这意味着 NUMA 虚拟机的内存关联至与其 CPU 相同的主机 NUMA 节点。不要在同一主机上运行 NUMA 和非 NUMA 虚拟机。如果非 NUMA 虚拟机已在运行主机，且 NUMA 虚拟机在该主机上启动，这可能会导致内存不足(OOM)事件，因为 NUMA 虚拟机无法访问主机内存，并仅限于其 NUMA 节点。要避免 OOM 事件，请确保在所有 NUMA 效率的实例上启用 NUMA 感知内存跟踪。为此，请配置 **hw:mem_page_size** 类别额外规格。*

5.7. COMPUTE 节点要求

Compute 节点负责在启动虚拟机实例后运行虚拟机实例。Compute 节点需要支持硬件虚拟化的裸机系统。Compute 节点还必须有足够的内存和磁盘空间来支持其托管的虚拟机实例的要求。

处理器

支持 Intel 64 或 AMD64 CPU 扩展并启用了 AMD-V 或 Intel VT 硬件虚拟扩展的 64 位 x86 处理器。我们推荐所使用的处理器最少有 4 个内核。

内存

主机操作系统至少有 6 GB RAM，以及满足以下注意事项的额外内存：

- 添加您要提供给虚拟机实例的额外内存。
- 添加附加内存以在主机上运行特殊功能或其他资源，如附加内核模块、虚拟交换机、监控解决方案和其他额外的后台任务。
- 如果要使用非统一内存访问 (NUMA)，红帽建议每个 CPU 插槽节点使用 8GB，或如果您有 256 GB 的物理 RAM，则建议每插槽节点使用 16GB。
- 至少配置 4 GB 交换空间。

磁盘空间

最少具有 50GB 可用磁盘空间。

网络接口卡

最少一个 1 Gbps 网络接口卡。对绑定的接口使用额外的网络接口卡，或代理标记的 VLAN 流量。

电源管理

每个 Compute 节点在服务器的主板上都要有一个受支持的电源管理接口，如智能平台管理接口 (IPMI) 功能。

5.8. RED HAT CEPH STORAGE 节点要求

使用 director 创建 Ceph Storage 集群需要额外的节点要求：

- Red Hat Ceph Storage 硬件指南 提供了处理器、内存和网络接口卡选择和磁盘布局等硬件要求 。
- 每个 Ceph Storage 节点在服务器的主板上都需要有一个受支持的电源管理接口，如智能平台管理接口(IPMI)功能。
- 每个 Ceph Storage 节点必须至少有两个磁盘。RHOSP director 使用 **cephadm** 来部署 Ceph Storage 集群。cephadm 功能不支持在节点的根磁盘上安装 Ceph OSD。

5.8.1. Red Hat Ceph Storage 节点和 RHEL 兼容性

RHOSP 17.1 在 RHEL 9.2 上被支持。但是，映射到 Red Hat Ceph Storage 角色的主机会更新到最新的主 RHEL 版本。在升级前，请参阅红帽知识库文章 Red Hat Ceph Storage: 支持的配置 。

5.9. OBJECT STORAGE 节点要求

Object Storage 节点提供 overcloud 的对象存储层。Object Storage 代理安装在 Controller 节点上。存储层需要每个节点上有多个磁盘的裸机节点。

处理器

支持 Intel 64 或 AMD64 CPU 扩展的 64 位 x86 处理器。

内存

内存要求取决于存储空间大小。每 1TB 硬盘空间需要至少 1GB 内存。要获得最佳性能，建议每 1 TB 硬盘空间使用 2 GB，特别是用于文件小于 100GB 的工作负载。

磁盘空间

存储要求取决于工作负载需要的容量。建议使用 SSD 驱动器来存储帐户和容器数据。帐户和容器数据与对象的容量比约为 1%。例如，对于每 100TB 硬盘容量，请提供 1TB 容量来存储帐户和容器数据。不过，这还取决于所存储数据的类型。如果主要是要存储小对象，则提供更多 SSD 空间。而对于大对象（视频和备份等），可提供较少的 SSD 空间。

磁盘配置

建议的节点配置需要类似以下示例的磁盘布局：

- **/dev/sda** - 根磁盘。director 把主 overcloud 镜像复制到该磁盘。
- **/dev/sdb** - 用于帐户数据。
- **/dev/sdc** - 用于容器数据。
- **/dev/sdd 及后续** - 对象服务器磁盘。可以根据您的存储需要使用多个磁盘。

网络接口卡

最少两个 1 Gbps 网络接口卡。对绑定的接口使用额外的网络接口卡，或代理标记的 VLAN 流量。

电源管理

每个 Controller 节点在服务器的主板上都要有一个受支持的电源管理接口，如智能平台管理接口 (IPMI) 功能。

5.10. OVERCLOUD 软件仓库

您可以在 Red Hat Enterprise Linux (RHEL) 9.2 上运行 Red Hat OpenStack Platform (RHOSP) 17.1。

注意

如果您将软件仓库与 Red Hat Satellite 同步，您可以启用 Red Hat Enterprise Linux 软件仓库的特定版本。但是，无论选择哪种版本，软件仓库均保持不变。例如，您可以启用 BaseOS 存储库的 9.2 版本，但存储库名称仍然是 **rhel-9-for-x86_64-baseos-eus-rpms**，尽管您选择的特定版本。

警告

除此处指定的软件仓库外，不支持在此指定的软件仓库。除非建议，否则请不要启用除下表中列出的其他产品或软件仓库之外，否则可能会遇到软件包依赖关系问题。请勿启用 Extra Packages for Enterprise Linux (EPEL)。

Controller 节点软件仓库

下表列出了用于 overcloud 中 Controller 节点的核心软件仓库。

名称	软件仓库	要求说明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-baseos-eus-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-eus-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat Enterprise Linux 9 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux 的高可用性工具。
Red Hat OpenStack Platform for RHEL 9 x86_64 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	Red Hat OpenStack Platform 核心软件仓库。
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	为 OpenStack Platform 提供 Open vSwitch (OVS) 软件包。
Red Hat Ceph Storage Tools 6 for RHEL 9 x86_64 (RPMs)	**rhceph-6-tools-for-rhel-9-x86_64-rpms**	Red Hat Ceph Storage 6 for Red Hat Enterprise Linux 9 的工具。

Compute 和 ComputeHCI 节点软件仓库

下表列出了用于 overcloud 中 Compute 和 ComputeHCI 节点的核心软件仓库。

名称	软件仓库	要求说明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-baseos-eus-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-eus-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat Enterprise Linux 9 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux 的高可用性工具。
Red Hat OpenStack Platform for RHEL 9 x86_64 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	Red Hat OpenStack Platform 核心软件仓库。
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	为 OpenStack Platform 提供 Open vSwitch (OVS) 软件包。
Red Hat Ceph Storage Tools 6 for RHEL 9 x86_64 (RPMs)	**rhceph-6-tools-for-rhel-9-x86_64-rpms**	Red Hat Ceph Storage 6 for Red Hat Enterprise Linux 9 的工具。

Ceph Storage 节点软件仓库

下表列出了用于 overcloud 的与 Ceph Storage 有关的软件仓库。

名称	软件仓库	要求说明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs)	**rhel-9-for-x86_64-baseos-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat OpenStack Platform Deployment Tools for RHEL 9 x86_64 (RPMs)	**openstack-17.1-deployment-tools-for-rhel-9-x86_64-rpms**	帮助 director 配置 Ceph Storage 节点的软件包。此软件仓库包含在单机 Ceph Storage 订阅中。如果您使用 OpenStack Platform 和 Ceph Storage 组合订阅，请使用 **openstack-17.1-for-rhel-9-x86_64-rpms** 存储库。
Red Hat OpenStack Platform for RHEL 9 x86_64 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	帮助 director 配置 Ceph Storage 节点的软件包。此软件仓库包含在 Red Hat OpenStack Platform 和 Red Hat Ceph Storage 组合订阅中。如果您使用独立的 Red Hat Ceph Storage 订阅，请使用 **openstack-17.1-deployment-tools-for-rhel-9-x86_64-rpms** 存储库。
Red Hat Ceph Storage Tools 6 for RHEL 9 x86_64 (RPMs)	**rhceph-6-tools-for-rhel-9-x86_64-rpms**	提供节点与 Ceph Storage 集群进行通信的工具。
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	为 OpenStack Platform 提供 Open vSwitch (OVS) 软件包。如果您在 Ceph Storage 节点上使用 OVS，请将此存储库添加到网络接口配置(NIC)模板中。

5.11. 节点置备和配置

您可以使用 OpenStack Bare Metal (ironic)服务或外部工具为 Red Hat OpenStack Platform (RHOSP)环境置备 overcloud 节点。置备节点时，您可以使用 director 配置它们。

使用 OpenStack Bare Metal (ironic) 服务置备

使用裸机服务置备 overcloud 节点是标准置备方法。如需更多信息，请参阅置备裸机 overcloud 节点。

使用外部工具置备

*您可以使用外部工具（如 Red Hat Satellite）来置备 overcloud 节点。如果您要创建没有电源管理控制的 overcloud，请使用具有 DHCP/PXE 引导限制的网络，或者使用具有自定义分区布局的节点，这不依赖于 **overcloud-hardened-uefi-full.qcow2** 镜像。这种置备方法不使用 OpenStack Bare Metal 服务(ironic)来管理节点。如需更多信息，请参阅使用预置备节点配置基本 overcloud。*

第 6 章 配置 OVERCLOUD 网络

要为 overcloud 配置物理网络，请创建一个网络配置文件 **network_data.yaml**，该文件遵循网络数据架构中定义的结构。

6.1. 定义 OVERCLOUD 网络

Red Hat OpenStack Platform (RHOSP)提供默认的 overcloud 网络，您可以使用它来隔离托管特定类型的网络流量。如果没有配置隔离网络，RHOSP 将 provisioning 网络用于所有服务。

您可以定义以下隔离的 overcloud 网络：

IPMI

用于节点电源管理的网络。此网络是在安装 undercloud 前预定义的网络。

置备

director 使用这个网络进行部署和管理。provisioning 网络通常配置在专用的接口上。初始部署使用带有 PXE 的 DHCP，然后网络转换为静态 IP。默认情况下，必须在原生 VLAN 上进行 PXE 引导，但有些系统控制器允许从 VLAN 启动。默认情况下，计算和存储节点使用调配接口作为 DNS、NTP 和系统维护的默认网关。

undercloud 可用作默认网关。但是，所有流量都位于 IP 伪装 NAT （网络地址转换），无法从其余 RHOSP 网络访问。undercloud 也是 overcloud 默认路由的单点故障。如果在 provisioning 网络上的路由器设备上配置了外部网关，undercloud neutron DHCP 服务器可以改为提供该服务。

内部 API

内部 API 网络用于使用 API 通信、RPC 消息和数据库通信 RHOSP 服务之间的通信。

租户

网络服务(neutron)使用以下方法之一为每个租户（项目）提供自己的网络：

- VLAN 分隔，其中每个租户网络都是网络 VLAN。
- 通过 VXLAN 或 GRE 进行隧道连接。

每个租户网络中的网络流量被隔离。每个租户网络关联了一个 IP 子网，网络命名空间意味着多个租户网络可以使用相同的地址范围，而不会导致冲突。

存储

用于块存储、NFS、iSCSI 等的网络。理想情况下，由于性能原因，这被隔离为完全独立的交换机结构。

存储管理

对象存储服务(swift)使用此网络在参与副本节点之间同步数据对象。代理服务充当用户请求和底层存储层之间的中间接口。代理接收传入的请求，并找到所需的副本来检索请求的数据。使用 Red Hat Ceph Storage 后端的服务通过存储管理网络连接，因为它们不直接与 Red Hat Ceph Storage 交互，而是使用前端服务。RBD 驱动程序是一个例外，因为此流量直接连接到 Red Hat Ceph Storage。

外部

托管用于图形系统管理的 Dashboard 服务(horizon)、RHOSP 服务的公共 API，并为目的地用于实例的传入流量执行 SNAT。

浮动 IP

允许传入流量使用浮动 IP 地址之间的 1 到-1 IP 地址映射来访问实例，并且实际分配给租户网络中的实例的 IP 地址。如果在独立于外部的 VLAN 上托管浮动 IP，您可以将浮动 IP VLAN 中继到 Controller 节点，并在创建 overcloud 后通过网络服务(neutron)添加 VLAN。这提供了创建附加到多个网桥的多个

浮动 IP 网络的方法。VLAN 是中继的，但没有配置为接口。相反，网络服务(neutron)为每个浮动 IP 网络在所选网桥上创建一个带有 VLAN 分段 ID 的 OVS 端口。

注意

provisioning 网络必须是原生 VLAN，其他网络可以中继。

您必须为每个角色定义特定的隔离网络：

角色	Network
Controller	置备、内部 API、存储、存储管理、租户、外部
计算	置备、内部 API、存储、租户
Ceph Storage	置备, 内部 API, 存储, 存储管理
Cinder 存储	置备, 内部 API, 存储, 存储管理
Swift Storage	置备, 内部 API, 存储, 存储管理

6.2. 创建网络定义文件

创建网络定义文件来配置隔离的 overcloud 网络。

步骤

1. 将您需要的示例网络定义模板从 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 复制到环境文件目录中：

```
$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/<sample_network_definition_file> /home/stack/templates/<networks_definition_file>
```

   - 将 **<sample_network_definition_file>** 替换为您要复制的示例网络定义文件的名称，如 **default-network-isolation-ipv6.yaml**。
   - 将 **<networks_definition_file>** 替换为网络定义文件的名称，如 **network_data.yaml**。

2. 更新本地网络定义文件中的网络和网络属性，以匹配 overcloud 网络环境的要求。有关您可以在网络定义文件中配置网络属性的属性的详情，请参考 网络定义文件配置选项。有关节点定义文件的示例，请参阅 网络定义文件 示例。

3. 可选：要更改默认网络的用户可见名称，请将 **name_lower** 字段改为网络的新名称，并使用新名称更新 **ServiceNetMap**：

```
- name: InternalApi
  name_lower: <custom_name>
  service_net_map_replace: <default_name>
```

   - 不要修改 **name** 字段。

- 将 **<custom_name>** 替换为您要分配给网络的新名称，如 **MyCustomInternalApi**。
- 将 **<default_name>** 替换为 **name_lower** 参数的默认值，如 **internal_api**。

4. 可选：在网络定义文件中添加自定义网络。以下示例为存储备份添加网络：

```
- name: StorageBackup
  vip: false
  name_lower: storage_backup
  subnets:
    storage_backup_subnet:
      ip_subnet: 172.16.6.0/24
      allocation_pools:
        - start: 172.16.6.4
        - end: 172.16.6.250
      gateway_ip: 172.16.6.1
```

6.3. 创建网络 VIP 定义文件

创建网络虚拟 IP (VIP) 定义文件，为您的 overcloud 配置网络 VIP。

步骤

1. 将您需要的示例网络 VIP 定义模板从 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 复制到环境文件目录中：

```
$ cp /usr/share/openstack-tripleo-heat-templates/network-data-
samples/<sample_network_VIP_definition_file>
/home/stack/templates/<network_VIP_definition_file>
```

- 将 **<sample_network_VIP_definition_file>** 替换为您要复制的示例网络 VIP 定义文件的名称，如 **vip-data-default-network-isolation.yaml**。
- 将 **<network_VIP_definition_file>** 替换为网络 VIP 定义文件的名称，如 **vip_data.yaml**。

2. 可选：为您的环境配置网络 VIP 定义文件：

```
- network: <network>
  dns_name: <dns_name>
  name: <vip_name>
  ip_address: <vip_address>
  subnet: <subnet>
```

- 将 **<network>** 替换为分配 IP 地址的网络的名称，如 **internal_api** 或 **storage**。
- 可选：将 **<dns_name>** 替换为 FQDN （完全限定域名），如 **overcloud**。
- 可选：将 & **lt;vip_name&** gt; 替换为 VIP 名称。
- 可选：将 **<vip_address>** 替换为网络的虚拟 IP 地址。
- 可选：将 **<**subnet> 替换为创建虚拟 IP 端口时要使用的子网名称。路由网络需要。例如，以下配置定义了外部网络和 control plane VIP：

```
- network: external
  dns_name: overcloud
  ip_address: '1.2.3.4'
- network: ctlplane
  dns_name: overcloud
```

有关可用于配置网络 VIP 的属性的详情，请参考 网络 VIP 属性。

6.4. 网络定义文件配置选项

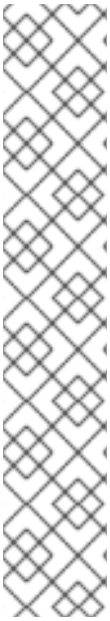
*您可以使用下表中的属性在 **network_data.yaml** 文件中配置网络属性。*

表 6.1. 网络定义属性

属性	值
name	网络的名称。
name_lower	网络名称的小写版本。director 将此名称映射到分配给 **roles_data.yaml** 文件中的角色的对应网络。默认值为 **name.lower （）**。
dns_domain	网络的 DNS 域名。仅在 undercloud 节点部署和管理多个 overcloud 时设置 **dns_domain**。 要确定 **dns_domain** 值，请完成以下步骤： • 将网络名称转换为小写。 • 将网络名称中的所有下划线("_")替换为连字符("-")。 • 使用点(".")和 **CloudDomain** 的名称后缀网络名称。 Example: • **network.name** = **InternalApi** • **CloudDomain** = **cloud.example.com.** • **dns_domain** = **internalapi.cloud.example.com.**  **注意** 您不能将相同的 **dns_domain** 用于不同的网络。
mtu	最大传输单元(MTU)。默认值为 **1500**。
ipv6	对于 IPv6，设置为 **true**。默认值为 **false**。
vip	设置为 **true**，以在网络上创建 VIP。默认值为 **false**。
subnets	包含网络的子网定义。

表 6.2. 子网定义属性

属性	值
subnet_name	子网的名称。
ip_subnet	CIDR 块表示法中的 IPv4 子网。默认值为 **192.0.5.0/24**。
ipv6_subnet	CIDR 块表示法中的 IPv6 子网。默认值为 **2001:db6:fd00:1000::/64**
gateway_ip	IPv4 网络的网关地址。默认值为 **192.0.5.1**。
gateway_ipv6	IPv6 网络的网关。
allocation_pools	IPv4 子网的 IP 范围。默认值： start: 192.0.5.100 end: 192.0.5.150
ipv6_allocation_pools	IPv6 子网的 IP 范围。默认值： start: 2001:db6:fd00:1000:100::1 end: 2001:db6:fd00:1000:150::1
Routes	需要通过网络网关路由的 IPv4 网络列表。
routes_ipv6	需要通过网络网关路由的 IPv6 网络列表。
vlan	网络的 VLAN ID。



注意

***routes** 和 **routes_ipv6** 选项包含一个路由列表。每个路由都是一个带有 **destination** 和 **nexthop** 键的字典条目。这两个选项都是字符串类型。*

```
routes:
  - destination: 198.51.100.0/24
    nexthop: 192.0.5.1
  - destination: 203.0.113.0/24
    nexthost: 192.0.5.1
```

```
routes:
  - destination: 2001:db6:fd00:2000::/64
    nexthop: 2001:db6:fd00:1000:100::1
  - destination: 2001:db6:fd00:3000::/64
    nexthost: 2001:db6:fd00:1000:100::1
```

6.5. 网络 VIP 属性属性

*您可以使用以下属性在 **network_data.yaml** 文件中配置网络 VIP 属性。*

表 6.3. 网络 VIP 属性属性

属性	值
name	虚拟 IP 名称。默认值为 **$network_name_virtual_ip**。
network	（必需）网络名称。
ip_address	VIP 的固定 IP 地址。
子网	子网名称。指定虚拟 IP 端口的子网。使用路由网络的部署需要。
dns_name	FQDN （完全限定域名）。默认值为 **overcloud**。

6.6. 网络定义文件示例

以下示例网络定义文件配置存储、存储管理、内部 API、租户和外部网络。

```
- name: Storage
  name_lower: storage
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    storage_subnet:
      ipv6_subnet: fd00:fd00:fd00:3000::/64
      ipv6_allocation_pools:
        - start: fd00:fd00:fd00:3000::10
          end: fd00:fd00:fd00:3000:ffff:ffff:ffff:fffe
      vlan: 30
- name: StorageMgmt
  name_lower: storage_mgmt
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    storage_mgmt_subnet:
      ipv6_subnet: fd00:fd00:fd00:4000::/64
      ipv6_allocation_pools:
        - start: fd00:fd00:fd00:4000::10
          end: fd00:fd00:fd00:4000:ffff:ffff:ffff:fffe
      vlan: 40
- name: InternalApi
  name_lower: internal_api
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    internal_api_subnet:
      ipv6_subnet: fd00:fd00:fd00:2000::/64
      ipv6_allocation_pools:
```

```
      - start: fd00:fd00:fd00:2000::10
        end: fd00:fd00:fd00:2000:ffff:ffff:ffff:fffe
    vlan: 20
- name: Tenant
  name_lower: tenant
  vip: false # Tenant networks do not use VIPs
  ipv6: true
  mtu: 1500
  subnets:
    tenant_subnet:
      ipv6_subnet: fd00:fd00:fd00:5000::/64
      ipv6_allocation_pools:
        - start: fd00:fd00:fd00:5000::10
          end: fd00:fd00:fd00:5000:ffff:ffff:ffff:fffe
      vlan: 50
- name: External
  name_lower: external
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    external_subnet:
      ipv6_subnet: 2001:db8:fd00:1000::/64
      ipv6_allocation_pools:
        - start: 2001:db8:fd00:1000::10
          end: 2001:db8:fd00:1000:ffff:ffff:ffff:fffe
      gateway_ipv6: 2001:db8:fd00:1000::1
      vlan: 10
```

第 7 章 置备和部署 OVERCLOUD

要创建 overcloud，您必须执行以下任务：

1. 为您的物理网络置备网络资源：
   a. 如果要部署网络隔离或自定义可组合网络，则以 YAML 格式创建一个网络定义文件。
   b. 运行网络调配命令，包括网络定义文件。
   c. 以 YAML 格式创建网络虚拟 IP (VIP) 定义文件。
   d. 运行 network VIP 置备命令，包括网络 VIP 定义文件。
2. 置备裸机节点：
   a. 以 YAML 格式创建节点定义文件。
   b. 运行裸机节点置备命令，包括节点定义文件。
3. 部署 overcloud。
   a. 运行部署命令，包括置备命令生成的 heat 环境文件。

7.1. 置备 OVERCLOUD 网络

要为 Red Hat OpenStack Platform (RHOSP) 物理网络环境配置网络资源，您必须执行以下任务：

1. 为您的 overcloud 配置并调配网络资源。
2. 为您的 overcloud 配置并调配网络虚拟 IP。

7.1.1. 配置和置备 overcloud 网络定义

您可以使用 YAML 格式的网络定义文件为 overcloud 配置物理网络。置备过程会从您的网络定义文件创建一个 heat 环境文件，其中包含您的网络规格。部署 overcloud 时，请将此 heat 环境文件包含在部署命令中。

先决条件

- 已安装 undercloud。如需更多信息，请参阅安装 director。

流程

1. 查找 **stackrc** undercloud 凭据文件：

```
$ source ~/stackrc
```

2. 将您需要的示例网络定义模板从 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 复制到环境文件目录中：

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/default-network-isolation.yaml /home/stack/templates/network_data.yaml
```

3. 为您的网络环境配置网络定义文件。例如，您可以更新外部网络定义：

```
- name: External
  name_lower: external
  vip: true
  mtu: 1500
  subnets:
    external_subnet:
      ip_subnet: 10.0.0.0/24
      allocation_pools:
        - start: 10.0.0.4
          end: 10.0.0.250
      gateway_ip: 10.0.0.1
      vlan: 10
```

4. 为您的环境配置任何其他网络和网络属性。有关您可以在网络定义文件中配置网络属性的属性的更多信息，请参阅配置 overcloud 网络。

5. 置备 overcloud 网络：

```
(undercloud)$ openstack overcloud network provision \
[--templates <templates_directory> \]
--output  <deployment_file> \
/home/stack/templates/<networks_definition_file>
```

- 可选：包含 **--templates** 选项以使用您自己的模板，而不是位于 **/usr/share/openstack-tripleo-heat-templates** 中的默认模板。将 **<templates_directory>** 替换为包含模板的目录的路径。
- 将 **<deployment_file>** 替换为用于部署命令生成的 heat 环境文件的名称，如 **/home/stack/templates/overcloud-networks-deployed.yaml**。
- 将 **<networks_definition_file>** 替换为网络定义文件的名称，如 **network_data.yaml**。

6. 网络置备完成后，您可以使用以下命令检查创建的网络和子网：

```
(undercloud)$ openstack network list
(undercloud)$ openstack subnet list
(undercloud)$ openstack network show <network>
(undercloud)$ openstack subnet show <subnet>
```

- 将 **<network>** 替换为您要检查的网络的名称或 UUID。
- 将 **<subnet>** 替换为您要检查的子网的名称或 UUID。

后续步骤

- 为 overcloud 配置和置备网络 VIP

7.1.2. 为 overcloud 配置和置备网络 VIP

您可以使用 YAML 格式的网络 VIP 定义文件为您的 overcloud 配置网络虚拟 IP (VIP)。置备过程从 VIP 定义文件创建一个 heat 环境文件，其中包含您的 VIP 规格。部署 overcloud 时，请将此 heat 环境文件包含在部署命令中。

先决条件

- 已安装 undercloud。如需更多信息，请参阅安装 director。
- 您的 overcloud 网络已调配。如需更多信息，请参阅配置和置备 overcloud 网络定义。

流程

1. 查找 **stackrc** undercloud 凭据文件：

```
$ source ~/stackrc
```

2. 将您需要的示例网络 VIP 定义模板从 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 复制到环境文件目录中：

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/vip-data-default-network-isolation.yaml /home/stack/templates/vip_data.yaml
```

3. 可选：为您的环境配置 VIP 定义文件。例如，以下命令定义了外部网络和 control plane VIP：

```
- name: external_vip
  network: external
  ip_address: 10.0.0.0
  subnet: external_vip_subnet
  dns_name: overcloud
- name: ctlplane_vip
  network: ctlplane
  ip_address: 192.168.122.0
  subnet: ctlplane_vip_subnet
  dns_name: overcloud
```

有关您可以在 VIP 定义文件中配置网络 VIP 属性的属性的更多信息，请参阅 Network VIP 属性属性。

4. 置备网络 VIP：

```
(undercloud)$ openstack overcloud network vip provision \
[--templates <templates_directory> \]
 --stack <stack> \
 --output <deployment_file> \
 /home/stack/templates/<vip_definition_file>
```

- 可选：包含 **--templates** 选项以使用您自己的模板，而不是位于 **/usr/share/openstack-tripleo-heat-templates** 中的默认模板。将 **<templates_directory>** 替换为包含模板的目录的路径。
- 将 **<**stack> 替换为置备网络 VIP 的堆栈的名称，如 **overcloud**。
- 将 **<deployment_file** > 替换为用于部署命令生成的 heat 环境文件的名称，如 **/home/stack/templates/overcloud-vip-deployed.yaml**。
- 将 **<vip_definition_file** > 替换为 VIP 定义文件的名称，如 **vip_data.yaml**。

5. 当网络 VIP 置备完成后，您可以使用以下命令检查创建的 VIP：

```
(undercloud)$ openstack port list
(undercloud)$ openstack port show <port>
```

- 将 **<port>** 替换为您要检查的端口的名称或 UUID。

后续步骤

- 置备裸机 overcloud 节点

7.2. 置备裸机 OVERCLOUD 节点

要配置 Red Hat OpenStack Platform (RHOSP)环境，您必须执行以下任务：

1. 为 overcloud 注册裸机节点。
2. 为 director 提供裸机节点硬件清单。
3. 在节点定义文件中配置裸机节点的数量、属性和网络布局。
4. 为每个裸机节点分配一个与节点匹配的资源类，并将其指定角色分配。

您还可以执行额外的可选任务，如匹配配置集来指定 overcloud 节点。

7.2.1. 为 overcloud 注册节点

director 需要节点定义模板来指定节点的硬件和电源管理详情。您可以使用 JSON 格式、**nodes.json** 或 YAML 格式 **nodes.yaml** 创建此模板。

流程

1. 创建名为 **nodes.json** 或 **nodes.yaml** 的模板，它将列出您的节点。使用以下 JSON 和 YAML 模板示例了解如何创建节点定义模板的结构：

示例 JSON 模板

```
{
  "nodes": [{
    "name": "node01",
    "ports": [{
      "address": "aa:aa:aa:aa:aa:aa",
      "physical_network": "ctlplane",
      "local_link_connection": {
        "switch_id": "52:54:00:00:00:00",
        "port_id": "p0"
      }
    }],
    "cpu": "4",
    "memory": "6144",
    "disk": "40",
    "arch": "x86_64",
    "pm_type": "ipmi",
    "pm_user": "admin",
    "pm_password": "p@55w0rd!",
    "pm_addr": "192.168.24.205"
  },
  {
    "name": "node02",
```

```
    "ports": [{
      "address": "bb:bb:bb:bb:bb:bb",
      "physical_network": "ctlplane",
      "local_link_connection": {
        "switch_id": "52:54:00:00:00:00",
        "port_id": "p0"
      }
    }],
    "cpu": "4",
    "memory": "6144",
    "disk": "40",
    "arch": "x86_64",
    "pm_type": "ipmi",
    "pm_user": "admin",
    "pm_password": "p@55w0rd!",
    "pm_addr": "192.168.24.206"
  }]
}
```

示例 YAML 模板

```
nodes:
  - name: "node01"
    ports:
      - address: "aa:aa:aa:aa:aa:aa"
        physical_network: ctlplane
        local_link_connection:
          switch_id: 52:54:00:00:00:00
          port_id: p0
    cpu: 4
    memory: 6144
    disk: 40
    arch: "x86_64"
    pm_type: "ipmi"
    pm_user: "admin"
    pm_password: "p@55w0rd!"
    pm_addr: "192.168.24.205"
  - name: "node02"
    ports:
      - address: "bb:bb:bb:bb:bb:bb"
        physical_network: ctlplane
        local_link_connection:
          switch_id: 52:54:00:00:00:00
          port_id: p0
    cpu: 4
    memory: 6144
    disk: 40
    arch: "x86_64"
    pm_type: "ipmi"
    pm_user: "admin"
    pm_password: "p@55w0rd!"
    pm_addr: "192.168.24.206"
```

此模板包含以下属性：

name

节点的逻辑名称。

ports

访问特定 IPMI 设备的端口。您可以定义以下可选端口属性：

- **地址**：节点上网络接口的 MAC 地址。对于每个系统的 Provisioning NIC，只使用 MAC 地址。
- **physical_network**：连接到 Provisioning NIC 的物理网络。
- **local_link_connection**：如果在内省期间使用 IPv6 置备并且 LLDP 未正确填充本地链接连接，则必须在 **local_link_connection** 参数中包含带有 **switch_id** 和 **port_id** 字段的虚拟数据。有关如何包含虚拟数据的更多信息，请参阅使用 director 内省来收集裸机节点硬件信息。

cpu

节点上的 CPU 数量。（可选）

memory

以 MB 为单位的内存大小。（可选）

disk

以 GB 为单位的硬盘的大小。（可选）

arch

系统架构。（可选）

pm_type

要使用的电源管理驱动程序。此示例使用 IPMI 驱动程序 (**ipmi**)。

注意

IPMI 是首选的受支持电源管理驱动程序。有关支持的电源管理类型及其选项的更多信息，请参阅 电源管理驱动程序。如果这些电源管理驱动程序不能正常工作，请将 IPMI 用于电源管理。

pm_user; pm_password

IPMI 的用户名和密码。

pm_addr

IPMI 设备的 IP 地址。

2. 验证模板格式化和语法：

```
$ source ~/stackrc
(undercloud)$ openstack overcloud node import --validate-only ~/nodes.json
```

3. 将模板文件保存到 **stack** 用户的主目录(**/home/stack/nodes.json**)。

4. 将模板导入到 director，将每个节点从模板注册到 director：

```
(undercloud)$ openstack overcloud node import ~/nodes.json
```

5. 等待节点完成注册和配置。完成后，确认 director 已成功注册节点：

```
(undercloud)$ openstack baremetal node list
```

7.2.2. 创建裸机节点硬件的清单

director 需要 Red Hat OpenStack Platform (RHOSP) 部署中节点的硬件清单，用于配置集标记、基准测试和手动根磁盘分配。

您可以使用以下方法之一向 director 提供硬件清单：

- 自动：您可以使用 director 的内省过程，从每个节点收集硬件信息。这个过程在每个节点上引导内省代理。内省代理从节点收集硬件数据并将数据送回 director。director 将硬件数据存储在 OpenStack 内部数据库中。
- 手动：您可以为每个裸机手动配置基本硬件清单。此清单存储在裸机置备服务(ironic)中，用于管理和部署裸机。

与设置裸机置备服务端口的手动方法相比，director 自动内省过程具有以下优点：

- 内省记录在硬件信息中的所有连接端口，包括用于在 **nodes.yaml** 中尚未配置用于 PXE 引导的端口。
- 如果属性可以使用 LLDP 发现，则内省会为每个端口设置 **local_link_connection** 属性。使用手动方法时，您必须在注册节点时为每个端口配置 **local_link_connection**。
- 在部署 spine-and-leaf 或 DCN 架构时，内省为裸机置备服务端口设置 **physical_network** 属性。

7.2.2.1. 使用 director 内省来收集裸机节点硬件信息

在将物理机注册为裸机节点后，您可以使用 director 内省自动添加其硬件详情并为每个以太网 MAC 地址创建端口。

提示

作为自动内省的替代选择，您可以手动为 director 提供裸机节点的硬件信息。如需更多信息，请参阅 *手动配置裸机节点硬件信息*。

先决条件

- 您已为 overcloud 注册了裸机节点。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 运行 pre-introspection 验证组来检查内省要求：

```
(undercloud)$ validation run --group pre-introspection \
 --inventory <inventory_file>
```

- 将 **<inventory_file>** 替换为 Ansible 清单文件的名称和位置，如 **~/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml**。

注意

当您运行验证时，输出中的 **Reasons** 列仅限于 79 个字符。要查看验证结果已满，请查看验证日志文件。

4. 查看验证报告的结果。

5. 可选：查看特定验证的详细输出：

```
(undercloud)$ validation history get --full <UUID>
```

- 将 **<**UUID> 替换为您要查看的报告中特定验证的 UUID。

重要

FAILED 验证不会阻止您部署或运行 Red Hat OpenStack Platform。但是，**FAILED** 验证可能会显示生产环境中潜在的问题。

6. 检查每个节点的硬件属性。您可以检查所有节点或特定节点的硬件属性：

- 检查所有节点的硬件属性：

```
(undercloud)$ openstack overcloud node introspect --all-manageable --provide
```

  - 使用 **--all-manageable** 选项仅内省处于受管理状态的节点。在此示例中，所有节点都处于受管理状态。
  - 使用 **--provide** 选项在内省后将所有节点重置为 **available** 状态。

- 检查特定节点的硬件属性：

```
(undercloud)$ openstack overcloud node introspect --provide <node1> [node2] [noden]
```

  - 使用 **--provide** 选项，在内省后将所有指定的节点重置为 **available** 状态。
  - 将 **<node1>**, **[node2]**, 及所有节点（直到 **[noden]**）替换为您要内省的每个节点的 UUID。

7. 在一个单独的终端窗口中监控内省进度日志：

```
(undercloud)$ sudo tail -f /var/log/containers/ironic-inspector/ironic-inspector.log
```

重要

确保内省进程已运行完。内省通常需要 15 分钟用于裸机节点。但是，不正确的内省网络可能会导致其需要更长的时间，这可能会导致内省失败。

8. 可选：如果您为通过 IPv6 的裸机置备配置了 undercloud，那么您还需要检查 LLDP 是否已为 Bare Metal Provisioning 服务(ironic)端口设置 **local_link_connection** ：

```
$ openstack baremetal port list --long -c UUID -c "Node UUID" -c "Local Link Connection"
```

- 如果裸机节点上的端口的 Local Link Connection 字段为空，则必须使用虚拟数据手动填充 **local_link_connection** 值。以下示例将假的交换机 ID 设置为 **52:54:00:00:00:00**，并将虚拟端口 ID 设置为 **p0** ：

```
$ openstack baremetal port set <port_uuid> \
--local-link-connection switch_id=52:54:00:00:00:00 \
--local-link-connection port_id=p0
```

- 验证 "Local Link Connection" 字段是否包含虚拟数据：

```
$ openstack baremetal port list --long -c UUID -c "Node UUID" -c "Local Link Connection"
```

内省完成后，所有节点都会变为 **available** 状态。

7.2.2.2. 手动配置裸机节点硬件信息

将物理机注册为裸机节点后，您可以手动添加其硬件详情，并为每个以太网 MAC 地址创建裸机端口。在部署 overcloud 之前，您必须至少创建一个裸机端口。

提示

作为手动内省的替代方法，您可以使用自动 director 内省流程来收集裸机节点的硬件信息。如需更多信息，请参阅使用 director 内省来收集裸机节点硬件信息。

先决条件

- 您已为 overcloud 注册了裸机节点。
- 您已为 **nodes.json** 中注册的每个端口配置了 **local_link_connection**。有关更多信息，请参阅为 overcloud 注册节点。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 指定部署内核并为节点驱动程序部署 ramdisk：

```
(undercloud)$ openstack baremetal node set <node> \
  --driver-info deploy_kernel=<kernel_file> \
  --driver-info deploy_ramdisk=<initramfs_file>
```

- 将 **<node>** 替换为裸机节点的 ID。
- 将 **<kernel_file>** 替换为 **.kernel** 镜像的路径，例如 **file:///var/lib/ironic/httpboot/agent.kernel**。

- 将 **<initramfs_file>** 替换为 **.initramfs** 镜像的路径，例如 **file:///var/lib/ironic/httpboot/agent.ramdisk**。

4. 更新节点属性以匹配节点上的硬件规格：

```
(undercloud)$ openstack baremetal node set <node> \
 --property cpus=<cpu> \
 --property memory_mb=<ram> \
 --property local_gb=<disk> \
 --property cpu_arch=<arch>
```

- 将 **<node>** 替换为裸机节点的 ID。
- 将 **<cpu>** 替换为 CPU 数量。
- 将 **<ram>** 替换为 RAM （以 MB 为单位）。
- 将 **<disk>** 替换为磁盘大小（以 GB 为单位）。
- 将 **<arch>** 替换为构架类型。

5. 可选：为每个节点指定 IPMI 密码套件：

```
(undercloud)$ openstack baremetal node set <node> \
 --driver-info ipmi_cipher_suite=<version>
```

- 将 **<node>** 替换为裸机节点的 ID。
- 将 **<version>** 替换为节点上要使用的密码套件版本。设置为以下有效值之一：
  - **3** - 节点使用带有 SHA1 密码套件的 AES-128。
  - **17** - 节点使用带有 SHA256 密码套件的 AES-128。

6. 可选：如果您有多个磁盘，请设置 root 设备提示，以通知部署 ramdisk 用于部署的磁盘：

```
(undercloud)$ openstack baremetal node set <node> \
 --property root_device='{"<property>": "<value>"}'
```

- 将 **<node>** 替换为裸机节点的 ID。
- 将 **<property> 和 <value>** 替换为您要用于部署的磁盘的详情，如 **root_device='{"size": "128"}'**
  RHOSP 支持以下属性：
  - **model**（字符串）：设备识别码。
  - **vendor**（字符串）：设备厂商。
  - **serial**（字符串）：磁盘序列号。
  - **hctl**（字符串）：SCSI 的 Host:Channel:Target:Lun。
  - **size**（整数）：设备的大小（以 GB 为单位）。
  - **wwn**（字符串）：唯一的存储 ID。

- **wwn_with_extension**（字符串）：唯一存储 ID 附加厂商扩展名。
- **wwn_vendor_extension**（字符串）：唯一厂商存储标识符。
- **rotational**（布尔值）：旋转磁盘设备为 true (HDD)，否则为 false (SSD)。
- **name**（字符串）：设备名称，例如：/dev/sdb1 仅对具有持久名称的设备使用此属性。

注意

如果您指定多个属性，该设备必须与所有这些属性匹配。

7. 通过在 provisioning 网络中创建带有 NIC 的 MAC 地址的端口来告知节点网卡的裸机置备服务：

```
(undercloud)$ openstack baremetal port create --node <node_uuid> <mac_address>
```

- 将 **<node_uuid>** 替换为裸机节点的唯一 ID。
- 将 **<mac_address>** 替换为用于 PXE 引导的 NIC 的 MAC 地址。

8. 验证节点的配置：

```
(undercloud)$ openstack baremetal node validate <node>
+------------+--------+----------------------------------------------+
| Interface  | Result | Reason                                       |
+------------+--------+----------------------------------------------+
| bios       | True   |                                              |
| boot       | True   |                                              |
| console    | True   |                                              |
| deploy     | False  | Node 229f0c3d-354a-4dab-9a88-ebd318249ad6 |
|            |        | failed to validate deploy image info.     |
|            |        | Some parameters were missing. Missing are:|
|            |        | [instance_info.image_source]              |
| inspect    | True   |                                              |
| management | True   |                                              |
| network    | True   |                                              |
| power      | True   |                                              |
| raid       | True   |                                              |
| rescue     | True   |                                              |
| storage    | True   |                                              |
+------------+--------+----------------------------------------------+
```

验证输出结果 **Result** 表示以下内容：

- **false**：接口验证失败。如果提供的原因缺少 **instance_info.image_source** 参数，这可能是因为在置备过程中填充，因此此时还没有设置它。如果您使用整个磁盘镜像，则可能需要设置 **image_source** 来传递验证。
- **true**：接口已通过验证。
- **None**: 接口不支持您的驱动。

7.2.3. 为 overcloud 置备裸机节点

要置备裸机节点，您可以定义您要以 YAML 格式的节点定义文件部署的裸机节点的数量和属性，并为这些节点分配 overcloud 角色。您还定义节点的网络布局。

置备过程会从节点定义文件创建一个 heat 环境文件。此 heat 环境文件包含您在节点定义文件中配置的节点规格，包括节点数、预测节点放置、自定义镜像和自定义 NIC。当您部署 overcloud 时，请将此文件包括在部署命令中。置备过程还会为每个节点定义的所有网络置备端口资源，或者在节点定义文件中角色。

先决条件

- *已安装 undercloud。如需更多信息，请参阅安装 director。*
- *裸机节点已注册、内省并可用于调配和部署。如需更多信息，请参阅为 overcloud 注册节点 和 创建裸机节点硬件清单。*

流程

1. *查找 **stackrc** undercloud 凭据文件：*

```
$ source ~/stackrc
```

2. *创建 **overcloud-baremetal-deploy.yaml** 节点定义文件，并为您要置备的每个角色定义节点数。例如，要置备三个 Controller 节点和三个 Compute 节点，请在 **overcloud-baremetal-deploy.yaml** 文件中添加以下配置：*

```
- name: Controller
  count: 3
- name: Compute
  count: 3
```

3. *可选：配置预测节点放置。例如，使用以下配置在节点 **node00**、**node01** 和 **node02** 上置备三个 Controller 节点，以及 **node04**、**node05** 和 **node06** 上的三个 Compute 节点：*

```
- name: Controller
  count: 3
  instances:
  - hostname: overcloud-controller-0
    name: node00
  - hostname: overcloud-controller-1
    name: node01
  - hostname: overcloud-controller-2
    name: node02
- name: Compute
  count: 3
  instances:
  - hostname: overcloud-novacompute-0
    name: node04
  - hostname: overcloud-novacompute-1
    name: node05
  - hostname: overcloud-novacompute-2
    name: node06
```

4. *可选：默认情况下，置备过程使用 **overcloud-hardened-uefi-full.qcow2** 镜像。您可以通过指定镜像的本地或远程 URL 来更改特定节点上使用的镜像，或用于角色的所有节点的镜像。以下示例将镜像改为本地 QCOW2 镜像：*

特定节点

```
- name: Controller
  count: 3
  instances:
  - hostname: overcloud-controller-0
    name: node00
    image:
      href: file:///var/lib/ironic/images/overcloud-custom.qcow2
  - hostname: overcloud-controller-1
    name: node01
    image:
      href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
  - hostname: overcloud-controller-2
    name: node02
    image:
      href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
```

角色的所有节点

```
- name: Controller
  count: 3
  defaults:
    image:
      href: file:///var/lib/ironic/images/overcloud-custom.qcow2
  instances:
  - hostname: overcloud-controller-0
    name: node00
  - hostname: overcloud-controller-1
    name: node01
  - hostname: overcloud-controller-2
    name: node02
```

5. *为角色的所有节点定义网络布局，或为特定节点定义网络布局：*

特定节点

以下示例为特定 Controller 节点置备网络，并为内部 API 网络的节点分配可预测的 IP：

```
- name: Controller
  count: 3
  defaults:
    network_config:
      template: /home/stack/templates/nic-config/myController.j2
      default_route_network:
      - external
    instances:
      - hostname: overcloud-controller-0
        name: node00
        networks:
          - network: ctlplane
            vif: true
          - network: external
            subnet: external_subnet
          - network: internal_api
```

```
      subnet: internal_api_subnet01
      fixed_ip: 172.21.11.100
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
```

角色的所有节点

以下示例为 Controller 和 Compute 角色置备网络：

```
- name: Controller
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
      template: /home/stack/templates/nic-config/myController.j2 (1)
      default_route_network:
      - external
- name: Compute
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: internal_api
      subnet: internal_api_subnet02
    - network: tenant
      subnet: tenant_subnet02
    - network: storage
      subnet: storage_subnet02
    network_config:
      template: /home/stack/templates/nic-config/myCompute.j2
```

(1) 您可以使用位于 **/usr/share/ansible/roles/tripleo_network_config/templates** 中的示例 NIC 模板，在本地环境文件目录中创建自己的 NIC 模板。

6. 可选：如果默认磁盘分区的大小不满足您的要求，请配置磁盘分区大小。例如：**/var/log** 分区的默认分区大小为 10 GB。考虑您的日志存储和保留要求，以确定 10 GB 是否满足您的要求。如果您需要为日志存储增加分配的磁盘大小，请在节点定义文件中添加以下配置来覆盖默认值：

```
ansible_playbooks:
  - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
    extra_vars:
      role_growvols_args:
        default:
          /=8GB
          /tmp=1GB
          /var/log=<log_size>GB
          /var/log/audit=2GB
          /home=1GB
          /var=100%
```

- 将 **<log_size>** 替换为要分配给日志文件的磁盘大小。

7. 如果您使用 Object Storage 服务(swift)和整个磁盘 overcloud 镜像，**overcloud-hardened-uefi-full**，您需要根据磁盘大小以及 **/var** 和 **/srv** 的存储要求来配置 **/srv** 分区的大小。如需更多信息，请参阅为对象存储服务配置整个磁盘分区。

8. 可选：使用自定义资源类或配置集功能为特定的角色指定 overcloud 节点。如需更多信息，请参阅通过匹配资源类并为角色指定 overcloud 节点和通过匹配配置集为角色设计 overcloud 节点。

9. 定义您要分配给节点的任何其他属性。有关您可以在节点定义文件中配置节点属性的属性的更多信息，请参阅 裸机节点置备属性。有关节点定义文件的示例，请参阅 节点定义文件 示例。

10. 置备 overcloud 节点：

```
(undercloud)$ openstack overcloud node provision \
[--templates <templates_directory> \]
--stack <stack> \
--network-config \
--output <deployment_file> \
/home/stack/templates/<node_definition_file>
```

- 可选：包含 **--templates** 选项以使用您自己的模板，而不是位于 **/usr/share/openstack-tripleo-heat-templates** 中的默认模板。将 **<templates_directory>** 替换为包含模板的目录的路径。
- 将 **&** lt;stack> 替换为置备裸机节点的堆栈名称。如果未指定，则默认为 **overcloud**。
- 包含 **--network-config** 可选参数，为 **cli-overcloud-node-network-config.yaml** Ansible playbook 提供网络定义。**cli-overcloud-node-network-config.yaml** playbook 使用 **os-net-config** 工具在部署的节点上应用网络配置。如果不使用 **--network-config** 来提供网络定义，则必须在 **network-environment.yaml** 文件中配置 **{{role.name}}NetworkConfigTemplate** 参数，否则会使用默认网络定义。
- 将 **<deployment_file>** 替换为用于部署命令生成的 heat 环境文件的名称，如 **/home/stack/templates/overcloud-baremetal-deployed.yaml**。
- 将 **<node_definition_file>** 替换为节点定义文件的名称，如 **overcloud-baremetal-deploy.yaml**。

11. 在一个单独的终端中监控置备进度：

```
(undercloud)$ watch openstack baremetal node list
```

- 当置备成功后，节点状态将从 **available** 变为 **active**。

- 如果因为节点硬件或网络配置失败，节点置备失败，您可以在再次运行置备步骤前删除失败的节点。如需更多信息，请参阅从节点定义文件中删除失败的裸机节点。

12. 使用 **metalsmith** 工具获取节点的统一视图，包括分配和端口：

```
(undercloud)$ metalsmith list
```

13. 验证节点与主机名的关联：

```
(undercloud)$ openstack baremetal allocation list
```

后续步骤

- 配置和部署 overcloud

7.2.4. 裸机节点置备属性

使用下表了解配置节点属性的可用属性，以及使用 **openstack baremetal node provision** 命令置备裸机节点时要使用的值。

- 角色属性：使用角色属性来定义各个角色。
- 每个角色的默认和实例属性：使用 default 或 instance 属性指定从可用节点池中分配节点的选择条件，并在裸机节点上设置属性和网络配置属性。

有关创建 裸机定义文件的详情，请参考为 overcloud 置备裸机节点。

表 7.1. 角色属性

属性	值
name	（必需）角色名称。
æ•°é‡�	要为这个角色置备的节点数量。默认值为 **1**。
默认值	**instances**条目属性的默认值字典。**instances**条目属性覆盖您在 **defaults** 参数中指定的任何默认值。
实例	用于为特定节点指定属性的值的字典。有关 **instances** 参数中支持的属性的更多信息，请参阅 **默认值** 和实例属性。定义的节点数量不能大于 **count** 参数的值。
hostname_format	覆盖这个角色的默认主机名格式。默认生成的主机名来自 overcloud 堆栈名称、角色和递增索引，都小写。例如，Controller 角色的默认格式为 **%stackname%-controller-%index%**。只有 Compute 角色不会遵循角色名称规则。Compute 默认格式为 **%stackname%-novacompute-%index%**。
ansible_playbooks	Ansible playbook 和 Ansible 变量的值字典。在节点网络配置之前，playbook 会根据节点调配后针对角色实例运行。有关指定 Ansible playbook 的更多信息，请参阅 **ansible_playbooks** 属性。

表 7.2. defaults 和 instances 属性

属性	值
hostname	（实例 仅指定 实例 属性应用到的节点的主机名。主机名派生自 **hostname_format** 属性。您可以使用自定义主机名。
name	(实例 仅)您要置备的节点的名称。
image	要在节点上置备的镜像的详情。有关支持的 **镜像** 属性的详情，请参考 **镜像属性**。
功能	选择与节点功能匹配的条件。
config_drive	将 data 和 first-boot 命令添加到传递给节点的 config-drive 中。如需更多信息，请参阅 **config_drive 属性**。 **注意** 仅将 **config_drive** 用于必须在第一次引导时执行的配置。对于所有其他自定义配置，创建一个 Ansible playbook，并使用 **ansible_playbooks** 属性在节点置备后对角色实例执行 playbook。
受管	设置为 **true** （默认）以使用 metalsmith 置备实例。设置为 **false** 以处理实例预置备。
网络	代表实例网络的字典列表。有关配置网络属性的更多信息，请参阅 **网络属性**。
network_config	链接到角色或实例的网络配置文件。有关配置到网络配置文件的链接的更多信息，请参阅 **network_config 属性**。
配置集	针对配置文件匹配的选择条件。如需更多信息，请参阅 根据匹配的配置集为角色指定 overcloud 节点。
已置备	设置为 **true** （默认）以置备节点。设置为 **false** 以取消置备节点。如需更多信息，请参阅 缩减裸机节点。
resource_class	与节点的资源类匹配的选择条件。默认值为 **baremetal**。如需更多信息，请参阅 根据匹配的资源类 为角色指定 overcloud 节点。
root_size_gb	GiB 中根分区的大小。默认值为 **49**。
swap_size_mb	MiB 中 swap 分区的大小。
遍历	作为与节点遍历匹配的选择条件的遍历列表。

表 7.3. 镜像 属性

属性	值
href	指定您要在节点上置备的根分区或整个磁盘镜像的 URL。支持的 URL 方案：**file://**、**http://** 和 **https://**。  **注意** 如果您使用 **file://** URL 方案为镜像指定本地 URL，则镜像路径必须指向 **/var/lib/ironic/images/** 目录，因为 **/var/lib/ironic/images** 从 undercloud 明确挂载到 **ironic-conductor** 容器中，以便服务镜像。
checksum	指定根分区或完整磁盘镜像的 MD5 校验和。当 **href** 是 URL 时是必需的。
kernel	指定内核镜像的镜像引用或 URL。仅在分区镜像中使用此属性。
ramdisk	指定 ramdisk 镜像的镜像引用或 URL。仅在分区镜像中使用此属性。

表 7.4. 网络 属性

属性	值
fixed_ip	要用于此网络的特定 IP 地址。
network	要创建网络端口的网络。
子网	要创建网络端口的子网。
端口	使用现有端口而不是创建新端口。
vif	在 provisioning 网络(**ctlplane**)上设为 **true**，以将网络作为虚拟接口(VIF)连接。设置为 **false**，以在没有 VIF 附加的情况下创建网络服务 API 资源。

表 7.5. network_config 属性

属性	值
模板	指定应用节点网络配置时要使用的 Ansible J2 NIC 配置模板。有关配置 NIC 模板的详情，请参考配置 overcloud 网络。
physical_bridge_name	用于访问外部网络的 OVS 网桥的名称。默认网桥名称为 **br-ex**。
public_interface_name	指定要添加到公共网桥的接口的名称。默认接口为 **nic1**。
network_config_update	设置为 **true**，以在更新时应用网络配置更改。默认禁用此选项。
net_config_data_lookup	为每个节点或节点组指定 NIC 映射配置 **os-net-config**。

属性	值
default_route_network	用于默认路由的网络。默认路由网络是 **ctlplane**
networks_skip_config	配置节点网络时要跳过的网络列表。
dns_search_domains	要添加到 **resolv.conf** 中的 DNS 搜索域列表，按优先级顺序排列。
bond_interface_ovs_options	用于绑定接口的 OVS 选项或绑定选项，如 **lacp=active** 和 **bond_mode=balance-slb** 用于 OVS 绑定，以及 Linux 绑定的 **mode=4**。
num_dpdk_interface_rx_queues	指定 DPDK 绑定或 DPDK 端口所需的 RX 队列数量。

表 7.6. config_drive 属性

属性	值

cloud_config	**cloud-init** 云配置数据的字典，用于在节点引导时运行任务。例如，要在第一次引导时将自定义名称服务器写入 **resolve.conf** 文件，请将以下 **cloud_config** 添加到 **config_drive** 属性中：
	
	config_drive:
	  cloud_config:
	  manage_resolv_conf: true
	  resolv_conf:
	  nameservers:
	  - 8.8.8.8
	  - 8.8.4.4
	  searchdomains:
	  - abc.example.com
	  - xyz.example.com
	  domain: example.com
	  sortlist:
	  - 10.0.0.1/255
	  - 10.0.0.2
	  options:
	  rotate: true
	  timeout: 1

meta_data	要使用 config-drive **cloud-init** 元数据包含的额外元数据。元数据添加到角色名称上生成的元数据集中：**public_keys**、**uuid**、**name**、**hostname** 和 **instance-type**。**cloud-init** 使此元数据作为实例数据可用。

表 7.7. ansible_playbooks 属性

属性	值
playbook	相对于角色定义 YAML 文件的 Ansible playbook 路径。
extra_vars	在运行 playbook 时设置的额外 Ansible 变量。使用以下语法指定额外变量： ansible_playbooks: - playbook: a_playbook.yaml extra_vars: param1: value1 param2: value2 例如，要增大使用整个磁盘 overcloud 镜像 **overcloud-hardened-uefi-full.qcow2** 部署的节点的 LVM 卷，请将以下额外变量添加到您的 playbook 属性中： ansible_playbooks: - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml extra_vars: role_growvols_args: default: /=8GB /tmp=1GB /var/log=10GB /var/log/audit=2GB /home=1GB /var=100% Controller: /=8GB /tmp=1GB /var/log=10GB /var/log/audit=2GB /home=1GB /srv=50GB /var=100%

7.2.5. 从节点定义文件中删除失败的裸机节点

如果因为节点硬件或网络配置失败，节点置备失败，您可以在再次运行置备步骤前删除失败的节点。要删除在置备过程中失败的裸机节点，请在节点定义文件中标记您要从堆栈中删除的节点，并在置备正常工作的裸机节点前取消置备节点。

先决条件

- *已安装 undercloud。如需更多信息，请参阅安装 director。*
- *由于节点硬件故障，裸机节点置备会失败。*

流程

1. *查找 **stackrc** undercloud 凭据文件：*

```
-
```

```
$ source ~/stackrc
```

2. 打开 **overcloud-baremetal-deploy.yaml** 节点定义文件。

3. 缩减节点分配给的角色的 **count** 参数。例如，以下配置更新 **ObjectStorage** 角色的 count 参数，以反映专用于 **ObjectStorage** 的节点数量被减少为 3：

```
- name: ObjectStorage
  count: 3
```

4. 定义您要从堆栈中删除的节点的**主机名**和**名称**（如果尚未在角色的 **instances** 属性中定义）。

5. 将 attribute **provisioned: false** 添加到您要删除的节点。例如，要从堆栈中删除节点 **overcloud-objectstorage-1**，请在 **overcloud-baremetal-deploy.yaml** 文件中包含以下代码片段：

```
- name: ObjectStorage
  count: 3
  instances:
  - hostname: overcloud-objectstorage-0
    name: node00
  - hostname: overcloud-objectstorage-1
    name: node01
    # Removed from cluster due to disk failure
    provisioned: false
  - hostname: overcloud-objectstorage-2
    name: node02
  - hostname: overcloud-objectstorage-3
    name: node03
```

6. 取消置备裸机节点：

```
(undercloud)$ openstack overcloud node unprovision \
 --stack <stack> \
 --network-ports \
 /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 将 **&** lt;stack> 替换为置备裸机节点的堆栈名称。如果未指定，则默认为 **overcloud**。

7. 置备 overcloud 节点以生成更新的 heat 环境文件，以包括在部署命令中：

```
(undercloud)$ openstack overcloud node provision \
--stack <stack> \
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 将 **<deployment_file>** 替换为用于部署命令生成的 heat 环境文件的名称，如 **/home/stack/templates/overcloud-baremetal-deployed.yaml**。

7.2.6. 通过匹配资源类为角色设计 overcloud 节点

您可以使用自定义资源类为特定角色指定 overcloud 节点。资源类与节点匹配以部署角色。默认情况下，所有节点都被分配 **baremetal** 的资源类。

注意

要在置备节点后更改分配给节点的资源类，您必须使用缩减流程取消置备节点，然后使用扩展步骤使用新资源类分配重新置备节点。有关更多信息，请参阅 扩展 overcloud 节点。

先决条件

- 为 overcloud 执行裸机节点的初始置备。

流程

1. 使用自定义资源类分配您要为角色指定的每个裸机节点：

```
(undercloud)$ openstack baremetal node set \
 --resource-class <resource_class> <node>
```

   - 将 **<resource_class>** 替换为资源类的名称，例如 **baremetal.CPU-PINNING**。
   - 将 **<node>** 替换为裸机节点的 ID。

2. 将角色添加到 **overcloud-baremetal-deploy.yaml** 文件中（如果尚未定义）。
3. 指定您要分配给角色节点的资源类：

```
- name: <role>
  count: 1
  defaults:
    resource_class: <resource_class>
```

   - 将 **<role>** 替换为角色的名称。
   - 将 **<resource_class>** 替换为在第 1 步中为资源类指定的名称。

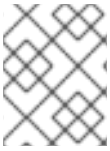
4. 返回到 为 overcloud 置备裸机节点 以完成置备过程。

7.2.7. 根据匹配的配置集为角色设计 overcloud 节点

您可以使用配置集功能为特定角色指定 overcloud 节点。配置集与节点功能与部署角色匹配。

提示

您还可以使用内省规则执行自动配置集分配。如需更多信息，请参阅配置自动配置集标记。



注意

要在置备节点后更改分配给节点的配置集，您必须使用缩减流程取消置备节点，然后使用扩展步骤使用新配置集分配重新置备节点。有关更多信息，请参阅 扩展 overcloud 节点。

先决条件

- 为 overcloud 执行裸机节点的初始置备。

流程

1. 每次添加新节点功能时，现有节点功能都会覆盖。因此，您必须检索每个注册节点的现有功能，以便再次设置它们：

```
$ openstack baremetal node show <node> \
 -f json -c properties | jq -r .properties.capabilities
```

2. 通过将 profile:<profile> 添加到节点的现有功能，给您要与角色 **配置集匹配** 的每个裸机节点分配配置集功能：

```
(undercloud)$ openstack baremetal node set  <node> \
 --property capabilities="profile:<profile>,<capability_1>,...,<capability_n>"
```

   - 将 **<node** > 替换为裸机节点的名称或 ID。
   - 将 **<profile** > 替换为与角色配置集匹配的配置集名称。
   - 将 **<capability_1&** gt; 以及所有功能（直到 **<capability_n** >）替换为在第 1 步中获得的每个功能。

3. 将角色添加到 **overcloud-baremetal-deploy.yaml** 文件中（如果尚未定义）。

4. 定义您要分配给角色节点的配置集：

```
- name: <role>
  count: 1
  defaults:
    profile: <profile>
```

   - 将 **<role** > 替换为角色的名称。
   - 将 **<profile** > 替换为与节点功能匹配的配置集名称。

5. 返回到 *为 overcloud 置备裸机节点* 以完成置备过程。

7.2.8. 为对象存储服务配置完整磁盘分区

整个磁盘镜像 **overcloud-hardened-uefi-full** 会被分区到单独的卷中。默认情况下，使用整个磁盘 overcloud 镜像部署的节点的 **/var** 分区会自动增加，直到磁盘被完全分配为止。如果您使用 Object Storage 服务(swift)，请根据您的磁盘大小以及 **/var** 和 **/srv** 的存储要求来配置 **/srv** 分区的大小。

先决条件

- 为 overcloud 执行裸机节点的初始置备。

流程

1. 使用 **role_growvols_args** 作为 **overcloud-baremetal-deploy.yaml** 节点定义文件中的 Ansible playbook 定义中的额外 Ansible 变量配置 **/srv** 和 **/var** 分区。将 **/srv** 或 **/var** 设置为一个绝对大小（以 GB 为单位），并将其他大小设置为 100% 以使用剩余的磁盘空间。

   - 以下示例配置将 **/srv** 设置为在 Controller 节点上部署的对象存储服务的绝对路径，将 **/var** 设置为 100% 以使用剩余的磁盘空间：

```
ansible_playbooks:
  - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
```

```
    extra_vars:
      role_growvols_args:
        default:
          /=8GB
          /tmp=1GB
          /var/log=10GB
          /var/log/audit=2GB
          /home=1GB
          /var=100%
        Controller:
          /=8GB
          /tmp=1GB
          /var/log=10GB
          /var/log/audit=2GB
          /home=1GB
          /srv=50GB
          /var=100%
```

- 以下示例配置将 **/var** 设置为绝对大小，将 **/srv** 设置为 100%，以使用 Object Storage 服务的 Object Storage 节点的剩余磁盘空间：

```
ansible_playbooks:
  - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
    extra_vars:
      role_growvols_args:
        default:
          /=8GB
          /tmp=1GB
          /var/log=10GB
          /var/log/audit=2GB
          /home=1GB
          /var=100%
        ObjectStorage:
          /=8GB
          /tmp=1GB
          /var/log=10GB
          /var/log/audit=2GB
          /home=1GB
          /var=10GB
          /srv=100%
```

2. 返回到 为 overcloud 置备裸机节点 以完成置备过程。

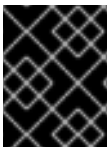
7.2.9. 节点定义文件示例

以下示例节点定义文件定义了三个 Controller 节点和三个 Compute 节点及其使用的默认网络的预测节点放置。这个示例还演示了如何定义具有根据资源类或节点功能配置集指定节点的自定义角色。

```
- name: Controller
  count: 3
  defaults:
    image:
      href: file:///var/lib/ironic/images/overcloud-custom.qcow2
    networks:
    - network: ctlplane
```

```
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storagemgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
        template: /home/stack/templates/nic-config/myController.j2
        default_route_network:
        - external
    profile: nodeCapability
  instances:
  - hostname: overcloud-controller-0
    name: node00
  - hostname: overcloud-controller-1
    name: node01
  - hostname: overcloud-controller-2
    name: node02
- name: Compute
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: internal_api
      subnet: internal_api_subnet02
    - network: tenant
      subnet: tenant_subnet02
    - network: storage
      subnet: storage_subnet02
    network_config:
      template: /home/stack/templates/nic-config/myCompute.j2
    resource_class: baremetal.COMPUTE
  instances:
  - hostname: overcloud-novacompute-0
    name: node04
  - hostname: overcloud-novacompute-1
    name: node05
  - hostname: overcloud-novacompute-2
    name: node06
```

7.2.10. 启用虚拟介质引导



重要

该功能在此发行版本中作为技术预览提供，因此不享有红帽的全面支持。它只应用于测试，不应部署在生产环境中。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

您可以使用 Redfish 虚拟介质引导，向节点的 Baseboard Management Controller (BMC) 提供引导镜像，以便 BMC 可将镜像插入到其中一个虚拟驱动器中。然后，节点可以从虚拟驱动器引导到镜像中存在的操作系统。

Redfish 硬件类型支持通过虚拟介质引导部署、救援和用户镜像。裸机置备服务(ironic)使用与节点关联的内核和 ramdisk 镜像，在节点部署时为 UEFI 或 BIOS 引导模式构建可引导的 ISO 镜像。虚拟介质引导的主要优点是可以消除 PXE 的 TFTP 镜像传输阶段，并使用 HTTP GET 或其他方法。

*要通过虚拟介质使用 **redfish** 硬件类型引导节点，请将引导接口设置为 **redfish-virtual-media**，并定义 EFI 系统分区(ESP)镜像。然后将注册节点配置为使用 Redfish 虚拟介质引导。*

前提条件

- *在 **undercloud.conf** 文件的 **enabled_hardware_types** 参数中启用 redfish 驱动程序。*
- *注册并注册的裸机节点。*
- *Image Service (glance) 中的 IPA 和实例镜像。*
- *对于 UEFI 节点，还必须在 Image Service (glance) 中有一个 EFI 系统分区镜像 (ESP)。*
- *用于清理和置备的网络。*

流程

1. *以 **stack** 用户身份登录 undercloud 主机。*
2. *查找 **stackrc** undercloud 凭证文件：*

```
$ source ~/stackrc
```

3. *将 Bare Metal Provisioning 服务引导接口设置为 **redfish-virtual-media** ：*

```
(undercloud)$ openstack baremetal node set --boot-interface redfish-virtual-media <node>
```

   - *将 **<node>** 替换为节点的名称。*

4. *定义 ESP 镜像：*

```
(undercloud)$ openstack baremetal node set --driver-info bootloader=<esp> <node>
```

   - *将 **<esp>** 替换为镜像服务(glance)镜像 UUID 或 ESP 镜像的 URL。*
   - *将 **<node>** 替换为节点的名称。*

5. *在裸机节点上创建一个端口，并将端口与裸机节点上 NIC 的 MAC 地址关联：*

```
(undercloud)$ openstack baremetal port create --pxe-enabled True \
 --node <node_uuid> <mac_address>
```

   - *将 **<node_uuid>** 替换为裸机节点的 UUID。*
   - *将 **<mac_address>** 替换为裸机节点中 NIC 的 MAC 地址。*

7.2.11. 为多磁盘 Ceph 集群定义根磁盘

Ceph Storage 节点通常使用多个磁盘。director 必须在多个磁盘配置中识别根磁盘。在置备过程中，overcloud 镜像被写入根磁盘。

硬件属性用于识别根磁盘。有关可以用来识别根磁盘的属性的更多信息，请参阅 标识根磁盘的属性。

流程

1. 从每个节点的硬件内省验证磁盘信息：

```
(undercloud)$ openstack baremetal introspection data save <node_uuid> --file
<output_file_name>
```

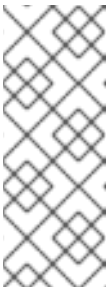
   - 将 **<node_uuid>** 替换为节点的 UUID。
   - 将 **<output_file_name>** 替换为包含节点内省输出的文件的名称。
     例如，一个节点的数据可能会显示 3 个磁盘：


```
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdb",
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f380d001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380d00",
    "serial": "61866da04f380d001ea4e13c12e36ad6"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdc",
    "wwn_vendor_extension": "0x1ea4e31e121cfb45",
    "wwn_with_extension": "0x61866da04f37fc001ea4e31e121cfb45",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f37fc00",
    "serial": "61866da04f37fc001ea4e31e121cfb45"
  }
]
```


2. 使用唯一硬件属性为节点设置根磁盘：

```
(undercloud)$ openstack baremetal node set --property root_device='{<property_value>}'
<node-uuid>
```

- 将 **<property_value>** 替换为来自内省数据的唯一硬件属性值，以设置根磁盘。
- 将 **<node_uuid>** 替换为节点的 UUID。



注意

唯一的硬件属性是硬件内省步骤中唯一标识磁盘的任何属性。例如，以下命令使用磁盘序列号来设置根磁盘：

```
(undercloud)$ openstack baremetal node set --property
root_device='{"serial": "61866da04f380d001ea4e13c12e36ad6"}'
1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
```

3. 将每个节点的 BIOS 配置为从网络首次引导，然后是根磁盘。

director 识别特定磁盘以用作根磁盘。运行 **openstack overcloud node provision** 命令时，director 置备 overcloud 镜像并将其写入根磁盘。

7.2.12. 标识根磁盘的属性

您可以定义多个属性以帮助 director 识别根磁盘：

- **model**（字符串）：设备识别码。
- **vendor**（字符串）：设备厂商。
- **serial**（字符串）：磁盘序列号。
- **hctl**（字符串）：SCSI 的 Host:Channel:Target:Lun。
- **size**（整数）：设备的大小（以 GB 为单位）。
- **wwn**（字符串）：唯一的存储 ID。
- **wwn_with_extension**（字符串）：唯一存储 ID 附加厂商扩展名。
- **wwn_vendor_extension**（字符串）：唯一厂商存储标识符。
- **rotational**（布尔值）：旋转磁盘设备为 true (HDD)，否则为 false (SSD)。
- **name**（字符串）：设备名称，例如：/dev/sdb1。

重要

将 **name** 属性用于具有持久名称的设备。不要使用 **name** 属性为没有持久名称的设备设置根磁盘，因为该值可在节点引导时更改。

7.2.13. 使用 overcloud-minimal 镜像来避免使用红帽订阅授权

Red Hat OpenStack Platform (RHOSP) 部署的默认镜像是 **overcloud-hardened-uefi-full.qcow2**。**overcloud-hardened-uefi-full.qcow2** 镜像使用有效的 Red Hat OpenStack Platform (RHOSP) 订阅。当您不希望使用您的订阅权利时，您可以使用 **overcloud-minimal** 镜像，以避免达到您

付费红帽订阅的限制。例如，当您要只置备带有 Ceph 守护进程的节点，或者想要调配您不想运行任何其他 OpenStack 服务的裸机操作系统(OS)时，这非常有用。有关如何获取 **overcloud-minimal** 镜像的详情，请参考 *获取 overcloud 节点的镜像*。

注意

overcloud-minimal 镜像仅支持标准 Linux 网桥。**overcloud-minimal** 镜像不支持 Open vSwitch (OVS)，因为 OVS 是一个需要 Red Hat OpenStack Platform 订阅权利的 OpenStack 服务。部署 Ceph Storage 节点不需要 OVS。使用 **linux_bond** 定义绑定，而不使用 **ovs_bond**。有关 **linux_bond** 的更多信息，请参阅创建 Linux 绑定。

流程

1. 打开 **overcloud-baremetal-deploy.yaml** 文件。
2. 为您要使用 **overcloud-minimal** 镜像的节点添加或更新 **image** 属性。您可以将特定节点上的镜像设置为 **overcloud-minimal**，或将角色的所有节点设置为 overcloud-minimal：

   特定节点

   ```
   - name: Ceph
     count: 3
     instances:
     - hostname: overcloud-ceph-0
       name: node00
       image:
         href: file:///var/lib/ironic/images/overcloud-minimal.qcow2
     - hostname: overcloud-ceph-1
       name: node01
       image:
         href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
     - hostname: overcloud-ceph-2
       name: node02
       image:
         href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
   ```

   角色的所有节点

   ```
   - name: Ceph
     count: 3
     defaults:
       image:
         href: file:///var/lib/ironic/images/overcloud-minimal.qcow2
     instances:
     - hostname: overcloud-ceph-0
       name: node00
     - hostname: overcloud-ceph-1
       name: node01
     - hostname: overcloud-ceph-2
       name: node02
   ```

3. 在 **roles_data.yaml** 角色定义文件中，将 **rhsm_enforce** 参数设置为 **False**。

   ```
   rhsm_enforce: False
   ```

4. 运行 provisioning 命令：

```
(undercloud)$ openstack overcloud node provision \
--stack stack \
--output /home/stack/templates/overcloud-baremetal-deployed.yaml \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

5. 将 **overcloud-baremetal-deployed.yaml** 环境文件传递给 **openstack overcloud deploy** 命令。

7.3. 配置和部署 OVERCLOUD

为 overcloud 置备网络资源和裸机节点后，您可以使用随 director 安装提供的未编辑 heat 模板文件以及您创建的任何自定义环境文件来配置 overcloud。完成 overcloud 的配置后，您可以部署 overcloud 环境。

重要

一个基本的 overcloud 会使用本地 LVM 存储作为块存储，这种配置不受支持。红帽建议您使用外部存储解决方案（如 Red Hat Ceph Storage）进行块存储。

7.3.1. 先决条件

- 您已置备了 overcloud 所需的网络资源和裸机节点。

7.3.2. 使用环境文件配置 overcloud

undercloud 包括一组构成您的 overcloud 创建计划的 heat 模板。您可以使用环境文件来自定义 overcloud 的各个方面，这些文件是 YAML 格式的文件，其内容可覆盖核心 heat 模板集合中的参数和资源。您可以根据需要纳入多个环境文件。环境文件扩展名必须为 **.yaml** 或 **.template**。

红帽建议将自定义环境文件组织在一个单独目录中，比如 **templates** 目录。

使用 **-e** 选项，将环境文件包含在 overcloud 部署中。使用 **-e** 选项添加到 overcloud 的所有环境文件都会成为 overcloud 堆栈定义的一部分。环境文件的顺序非常重要，因为后续环境文件中定义的参数和资源具有优先权。

要在初始部署后修改 overcloud 配置，请执行以下操作：

1. 修改定制环境文件和 heat 模板中的参数。
2. 使用相同的环境文件再次运行 **openstack overcloud deploy** 命令

不要直接编辑 overcloud 配置，因为 director 在更新 overcloud 栈时会覆盖任何手动配置。

注意

Open Virtual Networking (OVN) 是 Red Hat OpenStack Platform 17.1 中的默认网络机制驱动程序。如果要将 OVN 与分布式虚拟路由 (DVR) 搭配使用，则必须在 **openstack overcloud deploy** 命令中包含 **environments/services/neutron-ovn-dvr-ha.yaml** 文件。如果要在没有 DVR 的情况下使用 OVN，则必须在 **openstack overcloud deploy** 命令中包含 **environment/services/neutron-ovn-ha.yaml** 文件。

7.3.3. 创建 undercloud CA 信任的环境文件

如果您的 undercloud 使用 TLS，而证书颁发机构 (CA) 未获得公开信任，可将此 CA 用于 undercloud 操作的 SSL 端点加密。为确保其余部署可以访问 undercloud 端点，请将您的 overcloud 节点配置成信任 undercloud CA。

注意

要使这种方法奏效，overcloud 节点必须有一个指向 undercloud 公共端点的网络路由。依赖于脊叶型网络的部署可能必须应用这种配置。

有两种自定义证书可用于 undercloud：

- **用户提供的证书** - 当您自行提供证书时，会应用此定义。证书可能来自于您自己的 CA，也可能是自签名的。这通过使用 **undercloud_service_certificate** 选项来传递。在这种情况下，您必须信任自签名证书，或 CA（具体取决于您的部署）。
- **自动生成的证书** - 当您通过 **certmonger** 生成使用自己的本地 CA 的证书时，会应用此定义。使用 **undercloud.conf** 文件中的 **generate_service_certificate** 选项启用自动生成的证书。在这种情况下，director 在 **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** 生成 CA 证书，并配置 undercloud 的 HAProxy 实例以使用服务器证书。将 CA 证书添加到 **inject-trust-anchor-hiera.yaml** 文件中以将其呈现给 OpenStack Platform。

本例中使用了位于 **/home/stack/ca.crt.pem** 的一个自签名证书。如果您使用自动生成的证书，请改为使用 **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem**。

步骤

1. 打开证书文件，仅复制证书部分。不要包括其密钥：

```
$ vi /home/stack/ca.crt.pem
```

您需要的证书部分与下方简写的示例类似：

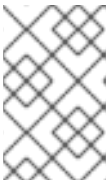
```
-----BEGIN CERTIFICATE-----
MIIDITCCAn2gAwIBAgIJAOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV
BAYTAlVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECgwH
UmVkIEhhdDELMAkGA1UECwwCUUUxFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
-----END CERTIFICATE-----
```

2. 创建一个名为 **/home/stack/inject-trust-anchor-hiera.yaml** 的 YAML 文件，其包含下列内容以及您从 PEM 文件复制而来的证书：

```
parameter_defaults:
  CAMap:
    undercloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDITCCAn2gAwIBAgIJAOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV

BAYTAlVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECgwH

UmVkIEhhdDELMAkGA1UECwwCUUUxFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
        -----END CERTIFICATE-----
```



注意

- 证书字符串必须采用 PEM 格式。
- **CAMap** 参数可能包含其他与 SSL/TLS 配置相关的证书。

3. 将 **/home/stack/inject-trust-anchor-hiera.yaml** 文件添加到部署命令中。在部署 overcloud 的过程中，director 将 CA 证书复制到每个 overcloud 节点。因此，每个节点都会信任 undercloud 的 SSL 端点提供的加密。

7.3.4. 在新部署中禁用 TSX

从 Red Hat Enterprise Linux 8.3 开始，内核默认禁用对 Intel 事务同步扩展 (TSX) 功能的支持。

您必须为新的 overcloud 显式禁用 TSX，除非您特别需要将其用于工作负载或第三方供应商。

在环境文件中设置 **KernelArgs** heat 参数。

```
parameter_defaults:
  ComputeParameters:
    KernelArgs: "tsx=off"
```

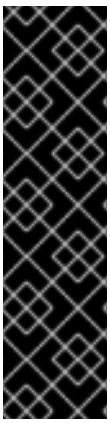
在运行 **openstack overcloud deploy** 命令时包含该环境文件：

其他资源

- "Intel TSX 对 OpenStack 虚拟客户机的影响的指南（适用于 RHEL 8.3 及更高版本）"

7.3.5. 验证 overcloud 配置

在部署 overcloud 之前，请验证 heat 模板和环境文件。



重要

- 由于对 17.0 中的 API 的更改，以下验证目前不稳定：
  - switch-vlans
  - network-environment
  - dhcp-provisioning
- **FAILED** 验证不会阻止您部署或运行 Red Hat OpenStack Platform。但是，**FAILED** 验证可能会显示生产环境中潜在的问题。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 使用部署所需的所有环境文件更新 overcloud 堆栈：

```
$ openstack overcloud deploy --templates \
  -e environment-file1.yaml \
  -e environment-file2.yaml \
  ...
  --stack-only
```

4. **可选** ：创建一个 YAML 文件，其中包含您要从验证运行中计算的任何验证：

```
<validation_name>:
  hosts: <targeted_hostname>
```

- 将< **validation_name** > 替换为您要从验证运行中排除的验证名称。
- 将 **<targeted_hostname** > 替换为包含验证的主机的名称。

5. 验证您的 overcloud 堆栈：

```
$ validation run \
  --group pre-deployment \
  --inventory <inventory_file>
  [--skiplist <validation_skips>]
```

- 将 **<inventory_file** > 替换为 Ansible 清单文件的名称和位置，如 **~/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml**。
- 将< **validation_skips** > 替换为包含验证运行中排除的 YAML 文件的名称和位置。
  注：
- 验证 **run --group pre-deployment** 命令包括 **node-disks** 验证。由于一个已知问题，这个验证目前会失败。要避免这个问题，请将包含 **node-disks** 验证的 **yaml** 文件添加到验证 **run --group pre-deployment** 命令中。
- **当您运行验证时，输出中的 Reasons 列仅限于 79 个字符。要查看验证结果已满，请查看验证日志文件。**

6. 查看验证报告的结果：

```
$ validation history get [--full] [--validation-log-dir <log_dir>] <uuid>
```

- **可选： 使用 --full 选项查看验证运行的详细输出。**
- **可选： 使用 --validation-log-dir 选项将验证运行的输出写入验证日志。**
- **将 <uuid > 替换为验证运行的 UUID。**

7.3.6. 创建 overcloud

创建 Red Hat OpenStack Platform (RHOSP) overcloud 环境的最后一个阶段是运行 openstack overcloud deploy 命令以创建 overcloud。有关 openstack overcloud deploy 命令可用选项的信息，请参阅 部署命令选项。

流程

1. 更正 overcloud 环境所需的环境文件和配置文件，以及 director 安装提供的未编辑 heat 模板文件，以及您创建的自定义环境文件。这应该包括以下文件：

   - overcloud-baremetal-deployed.yaml 节点定义文件。
   - overcloud-networks-deployed.yaml 网络定义文件。
   - overcloud-vip-deployed.yaml 网络 VIP 定义文件。
   - 容器化 OpenStack 服务的容器镜像位置。
   - 任何用于红帽 CDN 或 Satellite 注册的环境文件。
   - 任何其它自定义环境文件。

2. 按优先级顺序组织环境文件和配置文件，首先列出未编辑的 heat 模板文件，后跟包含自定义配置的环境文件，如 overrides to the default 属性。

3. 构建 openstack overcloud deploy 命令，按所需顺序指定配置文件和模板，例如：

```
(undercloud) $ openstack overcloud deploy --templates \
 [-n /home/stack/templates/network_data.yaml \ ]
  -e /home/stack/templates/overcloud-baremetal-deployed.yaml\
  -e /home/stack/templates/overcloud-networks-deployed.yaml\
  -e /home/stack/templates/overcloud-vip-deployed.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /home/stack/inject-trust-anchor-hiera.yaml \
 [-r /home/stack/templates/roles_data.yaml ]
```

-n /home/stack/templates/network_data.yaml

指定自定义网络配置。如果您使用网络隔离或自定义可组合网络，则需要此项。有关配置 overcloud 网络的详情，请参考 配置 overcloud 网络。

-e /home/stack/containers-prepare-parameter.yaml

添加容器镜像准备环境文件。您在安装 undercloud 的过程中生成了此文件，可使用此文件创建 overcloud。

-e /home/stack/inject-trust-anchor-hiera.yaml

添加用于在 undercloud 中安装自定义证书的环境文件。

-r /home/stack/templates/roles_data.yaml

如果您使用自定义角色或希望启用多架构云，则生成的角色数据。

4. overcloud 创建完毕后，director 会提供为配置 overcloud 而执行的 Ansible play 的总结：

```
PLAY RECAP *********************************************************************
overcloud-compute-0     : ok=160  changed=67   unreachable=0    failed=0
overcloud-controller-0  : ok=210  changed=93   unreachable=0    failed=0
undercloud              : ok=10   changed=7    unreachable=0    failed=0

Tuesday 15 October 2018  18:30:57 +1000 (0:00:00.107) 1:06:37.514 ******
===============================================================================
```

5. 在 overcloud 创建完成后，director 会提供访问 overcloud 的详细信息：

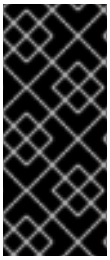
```
Ansible passed.
Overcloud configuration completed.
Overcloud Endpoint: http://192.168.24.113:5000
Overcloud Horizon Dashboard URL: http://192.168.24.113:80/dashboard
Overcloud rc file: /home/stack/overcloudrc
Overcloud Deployed
```

提示

您可以将部署命令保存在每次使用新的环境文件更新您的配置时添加到的文件中。

7.3.7. 部署命令选项

下表列出 openstack overcloud deploy 命令的其他参数。



重要

一些选项在此发行版本中作为技术预览提供，因此不享有红帽的全面支持。它们只应用于测试，不应在生产环境中使用。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

表 7.8. 部署命令选项

参数	描述
--templates [TEMPLATES]	包含您要部署的 heat 模板的目录。如果为空，部署命令会使用位于 **/usr/share/openstack-tripleo-heat-templates/** 的默认模板。
--stack STACK	要创建或更新的堆栈的名称
-t [TIMEOUT]，--timeout [TIMEOUT]	以分钟为单位的部署超时持续时间
--libvirt-type [LIBVIRT_TYPE]	要用于虚拟机监控程序的虚拟化类型
--ntp-server [NTP_SERVER]	要用于同步时间的网络时间协议 (NTP) 服务器。您也可以在以逗号分隔的列表中指定多个 NTP 服务器，例如：**--ntp-server 0.centos.pool.org，1.centos.pool.org**。对于高可用性集群部署，重要的是各个 Controller 节点始终引用同一时间源。但请注意，通常的环境可能已经指定了符合公认规范的 NTP 时间源。
--no-proxy [NO_PROXY]	为环境变量 **no_proxy** 指定自定义值。 这个环境变量被用来在代理通信中排除特定的主机名。
--overcloud-ssh-user OVERCLOUD_SSH_USER	定义访问 overcloud 节点的 SSH 用户。SSH 访问通常通过 **tripleo-admin** 用户进行。
--overcloud-ssh-key OVERCLOUD_SSH_KEY	定义用于 SSH 访问 overcloud 节点的密钥路径。
--overcloud-ssh-network OVERCLOUD_SSH_NETWORK	定义要用于 SSH 访问 overcloud 节点的网络名称。
-e [EXTRA HEAT TEMPLATE]，--environment-file [ENVIRONMENT FILE]	要传递给 overcloud 部署的额外环境文件。您可以多次指定此选项。请注意，传递到 **openstack overcloud deploy** 命令的环境文件顺序是非常重要的。例如，如果一个参数在多个环境文件中出现，则后续环境文件中的参数将覆盖前面文件中的同一参数。

参数	描述
--environment-directory	包含要在部署中包括的环境文件的目录。部署命令以数字顺序，然后以字母顺序处理这些环境文件。
-r ROLES_FILE	定义角色文件并覆盖 **--templates** 目录里的默认 **roles_data.yaml** 。文件位置可以是绝对路径或者相对于 **--templates** 的路径。
-n NETWORKS_FILE	定义网络文件并覆盖 **--templates** 目录里的默认 network_data.yaml。文件位置可以是绝对路径或者相对于 **--templates** 的路径。
-p PLAN_ENVIRONMENT_FILE	定义计划环境文件，并覆盖 **--templates** 目录里的默认 **plan-environment.yaml** 。文件位置可以是绝对路径或者相对于 **--templates** 的路径。
--no-cleanup	如果您不希望在部署后删除临时文件，并记录其位置，则使用此选项。
--update-plan-only	如果您要在不执行实际部署的情况下更新计划，则使用此选项。
--validation-errors-nonfatal	overcloud 在创建过程中会执行一组部署前检查。如果部署前检查出现任何非严重错误，则此选项会退出创建。我们推荐使用此选项，因为任何错误都有可能造成部署失败。
--validation-warnings-fatal	overcloud 在创建过程中会执行一组部署前检查。如果部署前检查出现任何非关键警告，则此选项会退出创建。openstack-tripleo-validations
--dry-run	如果您要在不创建 overcloud 的情况下对 overcloud 执行验证检查，则使用此选项。
--run-validations	使用此选项从 **openstack-tripleo-validations** 软件包运行外部验证。
--skip-postconfig	使用此选项跳过 overcloud 部署后配置。
--force-postconfig	使用此选项强制进行 overcloud 部署后配置。
--skip-deploy-identifier	如果您不希望部署命令为 **DeployIdentifier** 参数生成唯一标识符，则使用此选项。软件配置部署步骤仅当配置发生实际更改时才会触发。使用此选项要非常谨慎，仅当您确信不需要运行软件配置（如扩展某些角色）时方可使用。
--answers-file ANSWERS_FILE	带有选项和参数的 YAML 文件的路径。

参数	描述
--disable-password-generation	如果要禁用 overcloud 服务的密码生成，则使用此选项。
--deployed-server	如果要部署预置备 overcloud 节点，则使用此选项。与 **--disable-validations** 结合使用。
--no-config-download， --stack-only	如果您要禁用 **config-download** 工作流，仅创建堆栈和相关 OpenStack 资源，则使用此选项。此命令不会对 overcloud 应用软件配置。
--config-download-only	如果您要禁用 overcloud 栈创建，并仅运行 **config-download** 工作流以应用软件配置，则使用此选项。
--output-dir OUTPUT_DIR	要用于保存 **config-download** 输出的目录。该目录必须可由 mistral 用户写入。如果没有指定，director 将使用默认值，即 **/var/lib/mistral/overcloud**。
--override-ansible-cfg OVERRIDE_ANSIBLE_CFG	Ansible 配置文件的路径。该文件的配置会覆盖 **config-download** 默认生成的所有配置。
--config-download-timeout CONFIG_DOWNLOAD_TIMEOUT	您要用于 **config-download** 步骤的超时持续时间（以分钟为单位）。如果未设置，director 会在堆栈部署操作后将默认值设置为从 **--timeout** 参数中保留的时间。
--limit NODE1，NODE2	**（技术预览）**将此选项与以逗号分隔的节点列表一起使用，将 config-download playbook 的执行限制在特定节点或一组节点。例如，当想要仅在新节点上运行 config-download 时，**--limit** 选项对扩展操作很有用。此参数可能会导致主机之间实时迁移实例失败，请参阅使用 ansible-playbook-command.sh 脚本运行 config-download
--tags TAG1，TAG2	**（技术预览）**将此选项与 config-download playbook 中的以逗号分隔的标签列表一起使用，通过一组特定 config-download 任务来运行部署。
--skip-tags TAG1，TAG2	**（技术预览）**将此选项与您要从 config-download playbook 中跳过的以逗号分隔的标签列表一起使用。

运行以下命令查看完整选项列表：

```
(undercloud) $ openstack help overcloud deploy
```

某些命令行参数已过时或已弃用，它们的功能可以通过环境文件的 parameter_defaults 部分中所包含

的 heat 模板参数实现。下表将已弃用的参数与 heat 模板中的等效参数对应了起来。

表 7.9. 将被弃用的 CLI 参数映射到 heat 模板参数

参数	描述	Heat 模板参数
--validation-errors-fatal	overcloud 在创建过程中会执行一组部署前检查。在使用这个选项时，如果部署前检查出现任何严重错误，则会退出创建。我们推荐使用此选项，因为任何错误都有可能造成部署失败。	未进行参数映射
--disable-validations	完全禁用部署前验证。这些验证是内置部署前验证，已由 **openstack-tripleo-validations** 软件包中的外部验证替代。	未进行参数映射
--config-download	使用 **config-download** 机制运行部署。现在这是默认选项，以后可以删除该 CLI 选项。	未进行参数映射
--rhel-reg	使用此选项把 overcloud 节点注册到客户门户或 Satellite 6。	**RhsmVars**
--reg-method	使用此选项定义您要用于 overcloud 节点的注册方法。**satellite** 代表 Red Hat Satellite 6 或 Red Hat Satellite 5，**portal** 代表客户门户（Customer Portal）。	**RhsmVars**
--reg-org [REG_ORG]	要用于注册的组织。	**RhsmVars**
--reg-force	使用此选项注册系统（即使已经注册过）。	**RhsmVars**

参数	描述	Heat 模板参数
--reg-sat-url [REG_SAT_URL]	注册 overcloud 节点的 Satellite 服务器的基本 URL。此参数需要使用 Satellite 的 HTTP URL 而不是 HTTPS URL。例如，使用 http://satellite.example.com 而不是 https://satellite.example.com。overcloud 的创建过程会使用此 URL 来确定服务器是 Red Hat Satellite 5 还是 Red Hat Satellite 6 服务器。如果服务器是 Red Hat Satellite 6 服务器，则 overcloud 获取 **katello-ca-consumer-latest.noarch.rpm** 文件，使用 **subscription-manager** 注册，并安装 **katello-agent**。如果服务器是 Red Hat Satellite 5 服务器，则 overcloud 获取 **RHN-ORG-TRUSTED-SSL-CERT** 文件并使用 **rhnreg_ks** 注册。	**RhsmVars**
--reg-activation-key [REG_ACTIVATION_KEY]	使用此选项定义您要用于注册的激活码。	**RhsmVars**

这些参数计划从未来的 Red Hat OpenStack Platform 版本中移除。

7.3.8. 默认 overcloud 目录的内容

在 RHOSP 17 中，您可以在单个目录中找到所有配置文件。目录的名称是所用 openstack 命令和堆栈名称的组合。目录具有默认位置，但您可以使用 --working-dir 选项更改默认位置。您可以将这个选项与读取或创建用于部署的文件的任何 tripleoclient 命令一起使用。

默认位置	命令
$HOME/tripleo-deploy/undercloud	**Undercloud 安装**，它基于 **tripleo 部署**
$HOME/tripleo-deploy/<stack>	**tripleo deploy**, <stack> 默认为 **standalone**
$HOME/overcloud-deploy/<stack>	**overcloud deploy**, <stack> 默认为 **overcloud**

*下表详细介绍了 ~/**overcloud-deploy/overcloud** 目录中所含的文件和目录。*

表 7.10. Overcloud 目录内容

目录	描述
CLIjpeg'	**ansible-runner** 用于 CLI ansible-runner 的目录： cli-config-download cli-enable-ssh-admin cli-grant-local-access cli-undercloud-get-horizon-url
config-download	**config-download** 目录。在之前的 Red Hat OpenStack Platform (RHOSP)中，此目录名为 **~/config-download** 或 **/var/lib/mistral/<stack>**。
环境	包含使用 **openstack stack environment show <stack> 命令生成的保存的堆栈** 环境。
heat-launcher	临时 Heat 工作目录包含临时 Heat 配置和数据库备份。
输出	包含使用 **openstack stack output show <stack> <output> 命令生成的已保存的堆栈输出**。
<stack>-deployment_status.yaml	包含保存的堆栈状态。其中 **overcloud** 是堆栈的名称，此文件名为 **overcloud-deployment_status.yaml**。
<stack>-export.yaml	包含堆栈导出信息，使用 **openstack overcloud export --stack <stack>** 命令生成。其中 **overcloud** 是堆栈的名称，该文件名为 **overcloud-export.yaml**。
<stack>-install-*.tar.bzip2	工作目录的 tarball，例如 **overcloud-install-20220823213643.tar.bzip2**。
<stack>-passwords.yaml	包含堆栈密码。其中 **overcloud** 是堆栈的名称，此文件名为 **overcloud-passwords.yaml**。
<stack>rc	使用 overcloud API 所需的凭据文件，如 **overcloudrc**。
tempest-deployer-input.conf	包含 Tempest 配置。
tripleo-ansible-inventory.yaml	overcloud 的 Ansible 清单。
tripleo-heat-templates	包含 rendered jinja2 模板的副本。您可以在 **/usr/share/openstack-tripleo-heat-templates** 中找到源模板，也可以使用 CLI 上的 **--templates** 选项指定模板。

目录	描述
tripleo-overcloud-baremetal-deployment.yaml	用于置备 overcloud 节点的裸机部署输入。
tripleo-overcloud-network-data.yaml	用于置备 overcloud 网络的网络部署输入。
tripleo-overcloud-roles-data.yaml	使用 CLI 上的 **-r** 选项指定的角色数据。
tripleo-overcloud-virtual-ips.yaml	用于置备 overcloud 网络 VIP 的 VIP 部署输入。

7.3.9. 验证 overcloud 部署

验证您的部署的 overcloud。

先决条件

- *您已部署了 overcloud。*

流程

1. *查找 stackrc 凭证文件：*

```
$ source ~/stackrc
```

2. *验证您的 overcloud 部署：*

```
$ validation run \
  --group post-deployment \
  [--inventory <inventory_file>]
```

   - *将 <inventory_file > 替换为 ansible 清单文件的名称。默认情况下，动态清单名为 tripleo-ansible-inventory。*

注意

当您运行验证时，输出中的 **Reasons** 列仅限于 **79** 个字符。要查看验证结果已满，请查看验证日志文件。

3. 查看验证报告的结果：

```
$ validation history get --full <UUID>
```

- 将 **<UUID >** 替换为验证运行的 **UUID**。
- 可选： 使用 **--full** 选项查看验证运行的详细输出。

重要

FAILED 验证不会阻止您部署或运行 **Red Hat OpenStack Platform**。但是， **FAILED** 验证可能会显示生产环境中潜在的问题。

7.3.10. 访问 overcloud

director 生成一个凭据文件，其中包含从 undercloud 操作 overcloud 所需的凭据。director 将此文件保存为 stack 用户主目录的 overcloudrc 文件。

流程

1. 获取 overcloudrc 文件：

```
(undercloud)$ source ~/overcloudrc
```

命令提示符的变化表示您正在访问 overcloud：

```
(overcloud)$
```

2. 要返回与 undercloud 交互，请 source stackrc 文件：

```
(overcloud)$ source ~/stackrc
(undercloud)$
```

命令提示符的变化表示您要访问 **undercloud**：

```
(undercloud)$
```

7.4. 使用预置备节点配置基本 OVERCLOUD

本章包含基本配置流程，可用于使用预置备节点配置 **Red Hat OpenStack Platform (RHOSP)** 环境。这种情境与标准 **overcloud** 创建情境有所不同，具体体现在以下几个方面：

- 您可以使用外部工具置备节点，让 **director** 仅控制 **overcloud** 的配置。
- 您无需依赖 **director** 的置备方法，可直接使用节点。如果您想创建没有电源管理控制的 **overcloud** 或使用具有 **DHCP/PXE** 引导限制的网络，该方法很有用。
- **director** 不使用 **OpenStack Compute (nova)**、**OpenStack Bare Metal (ironic)** 或者 **OpenStack Image (glance)** 来管理节点。
- 预置备的节点可以使用不依赖于 **QCOW2 overcloud-full** 镜像的自定义分区布局。

这种情境仅包括没有自定义功能的基本配置。

重要

您无法将预置备节点与 **director** 置备的节点合并。

7.4.1. 预置备节点要求

在开始使用预置备节点部署 **overcloud** 前，请确保环境中存在以下配置：

- 您在 第 4 章 在 undercloud 上安装 director 中创建的 **director** 节点。

- 节点的一组裸机。所需的节点数量由您需要创建的 overcloud 类型所决定。这些机器必须符合为每个节点类型设定的要求。这些节点需要 Red Hat Enterprise Linux 9.2 作为主机操作系统安装。红帽建议使用最新的可用版本。

- 用于管理预置备节点的一个网络连接。本情境需要具备对节点的不间断 SSH 访问权限以进行编配代理配置。

- Control Plane 网络的一个网络连接。这种网络具备两种主要情境：

  - 将 Provisioning 网络用作 Control Plane，这种是默认情境。这个网络通常是从预置备节点到 director 的第 3 层 (L3) 可路由网络连接。本情境示例使用以下 IP 地址分配：

    表 7.11. Provisioning 网络 IP 分配信息

节点名	IP 地址
Director	192.168.24.1
控制器 0	192.168.24.2
计算 0	192.168.24.3

  - 使用单独的网络。如果 director 的 Provisioning 网络是私有非路由网络，则可从任何子网定义节点的 IP 地址，通过公共 API 端点与 director 通信。有关此情境要求的详情，请参考 第 7.4.6 节 “为预置备节点使用单独网络”。

- 本示例中的所有其他网络类型使用 Control Plane 网络来提供 OpenStack 服务。不过，您可以为其他网络流量类型创建额外的网络。

- 如果有任何节点使用 Pacemaker 资源，服务用户 hacluster 和服务组 haclient 必须具有 189 的 UID/GID。这是因为 CVE-2018-16877。如果与操作系统一起安装了 Pacemaker，则安装会自动创建这些 ID。如果错误设置 ID 值，请按照文章 OpenStack 小幅更新/快进升级可能在 controller 节点上在 pacemaker 步骤失败，显示消息“Could not evaluate: backup_cib” 中的步骤来更改 ID 值。

- 要防止某些服务绑定到不正确的 IP 地址并导致部署失败，请确保 /etc/hosts 文件不包含 node-name=127.0.0.1 映射。

7.4.2. 在预置备节点上创建用户

使用预置备节点配置 overcloud 时，director 需要对 overcloud 节点进行 SSH 访问。在预置备的节点上，您必须创建一个使用 SSH 密钥身份验证的用户，并为该用户配置免密码 sudo 访问。在预置备的节点上创建用户后，可以结合使用 --overcloud-ssh-user 和 --overcloud-ssh-key 选项及 openstack overcloud deploy 命令，创建具有预置备节点的 overcloud。

默认情况下，overcloud SSH 用户和 overcloud SSH 密钥的值是 stack 用户和 ~/.ssh/id_rsa。要创建 stack 用户，请完成以下步骤。

步骤

1. 在每个 overcloud 节点上，创建 stack 用户并设定密码。例如，在 Controller 节点上运行以下命令：

```
[root@controller-0 ~]# useradd stack
[root@controller-0 ~]# passwd stack  # specify a password
```

2. 进行以下操作以使这个用户使用 sudo 时不需要输入密码：

```
[root@controller-0 ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
/etc/sudoers.d/stack
[root@controller-0 ~]# chmod 0440 /etc/sudoers.d/stack
```

3. 在所有预置备节点上创建并配置 stack 用户后，将 stack 用户的公共 SSH 密钥从 director 节点复制到每个 overcloud 节点。例如，要将 director 的公共 SSH 密钥复制到 Controller 节点，请运行以下命令：

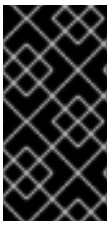
```
[stack@director ~]$ ssh-copy-id stack@192.168.24.2
```

重要

要复制 SSH 密钥，您可能必须在每个 overcloud 节点的 SSH 配置中临时设置 PasswordAuthentication Yes。复制 SSH 密钥后，设置 PasswordAuthentication No，并在以后使用 SSH 密钥进行身份验证。

7.4.3. 为预置备节点注册操作系统

每个节点需要具备访问红帽订阅的权限。在每个节点上完成以下步骤，将节点注册到 Red Hat Content Delivery Network 中。以 root 用户身份或具有 sudo 特权的用户身份执行命令。



重要

仅启用列出的软件仓库。其他软件仓库可能导致软件包和软件冲突。请勿启用任何其他软件仓库。

步骤

1. 运行注册命令，按照提示输入您的客户门户用户名和密码：

```
[root@controller-0 ~]# sudo subscription-manager register
```

2. 查找 Red Hat OpenStack Platform (RHOSP) 17.1 的权利池：

```
[root@controller-0 ~]# sudo subscription-manager list --available --all --matches="Red Hat OpenStack"
```

3. 使用上一步中的池 ID 来附加 RHOSP 17.1 权利：

```
[root@controller-0 ~]# sudo subscription-manager attach --pool=pool_id
```

4. 禁用所有默认软件仓库：

```
[root@controller-0 ~]# sudo subscription-manager repos --disable=*
```

5. 启用所需的 Red Hat Enterprise Linux (RHEL)软件仓库：

```
[root@controller-0 ~]# sudo subscription-manager repos \
 --enable=rhel-9-for-x86_64-baseos-eus-rpms \
 --enable=rhel-9-for-x86_64-appstream-eus-rpms \
 --enable=rhel-9-for-x86_64-highavailability-eus-rpms \
 --enable=openstack-17.1-for-rhel-9-x86_64-rpms \
 --enable=fast-datapath-for-rhel-9-x86_64-rpms
```

6. 如果 overcloud 使用 Red Hat Ceph Storage 节点，请启用相关的 Red Hat Ceph Storage 存储库：

```
[root@cephstorage-0 ~]# sudo subscription-manager repos \
 --enable=rhel-9-for-x86_64-baseos-rpms \
 --enable=rhel-9-for-x86_64-appstream-rpms \
 --enable=openstack-deployment-tools-for-rhel-9-x86_64-rpms
```

7. 锁定除 Red Hat Ceph Storage 节点以外的所有 overcloud 节点上的 RHEL 版本：

```
[root@controller-0 ~]# subscription-manager release --set=9.2
```

8. 更新您的系统，确保您具备最新的基础系统软件包：

```
[root@controller-0 ~]# sudo dnf update -y
[root@controller-0 ~]# sudo reboot
```

节点现在可供您的 overcloud 使用。

7.4.4. 配置对 director 的 SSL/TLS 访问权限

如果 director 使用 SSL/TLS，那么预置备节点需要用于签署 director 的 SSL/TLS 证书的证书授权机构文件。如果使用自己的证书颁发机构（CA），请在每个 overcloud 节点上执行以下操作。

步骤

1. 将证书授权机构文件复制到各预置备节点上的 /etc/pki/ca-trust/source/anchors/ 目录中。

2. 在每个 overcloud 节点上运行以下命令：

```
[root@controller-0 ~]#  sudo update-ca-trust extract
```

这些步骤将确保 overcloud 节点可以通过 SSL/TLS 访问 director 的公共 API。

7.4.5. 配置 control plane 网络

预置备 overcloud 节点使用标准 HTTP 请求从 director 获取元数据。这表示所有 overcloud 节点都需要 L3 权限来访问以下之一：

- director 的 Control Plane 网络，这是在 undercloud.conf 文件中使用 network_cidr 参数定义的子网。overcloud 节点需要对该子网的直接访问权限或对该子网的可路由访问权限。

-

director 的公共 API 端点，使用 undercloud.conf 文件中的 undercloud_public_host 参数指定。如果没有指向 Control Plane 的 L3 路由或希望使用 SSL/TLS 通信，则此选项可用。有关将 overcloud 节点配置为使用公共 API 端点的更多信息，请参阅 第 7.4.6 节 “为预置备节点使用单独网络”。

director 使用 Control Plane 网络管理和配置标准 overcloud。对于具有预置备节点的 overcloud，可能需要对您的网络配置进行修改以适应 director 和预置备节点之间的通信。

使用网络隔离

您可以使用网络隔离分组服务以使用特定网络，包括 Control Plane。您可以为 Control Plane 上的节点定义特定 IP 地址。

注意

如果使用网络隔离，请确保您的 NIC 模板不包括用于 undercloud 访问的 NIC。这些模板可能会重新配置 NIC，这会导致在部署过程中出现连接和配置问题。

分配 IP 地址

如果不使用网络隔离，则可使用一个 Control Plane 网络管理所有服务。这需要手动配置每个节点的 Control Plane NIC，以便使用 Control Plane 网络范围内的 IP 地址。如果将 director 的 Provisioning 网络用作 Control Plane，请确保所选的 overcloud IP 地址不在置备（dhcp_start 和 dhcp_end）和内省 (inspection_iprange) DHCP 范围之内。

在创建标准 overcloud 期间，director 将创建 OpenStack Networking (neutron) 端口并为 Provisioning/Control Plane 网络上的 overcloud 节点自动分配 IP 地址。但是，这可能导致 director 分配的地址与您为每个节点手动配置的 IP 地址不同。在这种情况下，使用可预测的 IP 地址策略来强制 director 使用 Control Plane 上的预置备 IP 分配。

如果使用网络隔离，请创建一个自定义环境文件 deployed-ports.yaml 来实现可预测的 IP 策略。以下示例自定义环境文件 deployed-ports.yaml 将资源 registry 映射和参数传递给 director，并定义预置备节点的 IP 分配。NodePortMap、ControlPlaneVipData 和 VipPortMap 参数定义与每个 overcloud 节点对应的 IP 地址和子网 CIDR。

```
resource_registry:
  # Deployed Virtual IP port resources
  OS::TripleO::Network::Ports::ExternalVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_external.yaml
  OS::TripleO::Network::Ports::InternalApiVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_internal_api.yaml
  OS::TripleO::Network::Ports::StorageVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_storage.yaml
```

```
  OS::TripleO::Network::Ports::StorageMgmtVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_storage_mgmt.yaml

  # Deployed ControlPlane port resource
  OS::TripleO::DeployedServer::ControlPlanePort: /usr/share/openstack-tripleo-heat-
templates/deployed-server/deployed-neutron-port.yaml

  # Controller role port resources
  OS::TripleO::Controller::Ports::ExternalPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_external.yaml
  OS::TripleO::Controller::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_internal_api.yaml
  OS::TripleO::Controller::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage_mgmt.yaml
  OS::TripleO::Controller::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage.yaml
  OS::TripleO::Controller::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_tenant.yaml

  # Compute role port resources
  OS::TripleO::Compute::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_internal_api.yaml
  OS::TripleO::Compute::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage.yaml
  OS::TripleO::Compute::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_tenant.yaml

  # CephStorage role port resources
  OS::TripleO::CephStorage::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage_mgmt.yaml
  OS::TripleO::CephStorage::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage.yaml

parameter_defaults:
  NodePortMap: 1
    # Controller node parameters
    controller-00-rack01: 2
      ctlplane: 3
        ip_address: 192.168.24.201
        ip_address_uri: 192.168.24.201
        ip_subnet: 192.168.24.0/24
      external:
        ip_address: 10.0.0.201
        ip_address_uri: 10.0.0.201
        ip_subnet: 10.0.0.10/24
      internal_api:
        ip_address: 172.16.2.201
        ip_address_uri: 172.16.2.201
        ip_subnet: 172.16.2.10/24
      management:
        ip_address: 192.168.1.201
        ip_address_uri: 192.168.1.201
        ip_subnet: 192.168.1.10/24
      storage:
        ip_address: 172.16.1.201
        ip_address_uri: 172.16.1.201
```

```
    ip_subnet: 172.16.1.10/24
  storage_mgmt:
    ip_address: 172.16.3.201
    ip_address_uri: 172.16.3.201
    ip_subnet: 172.16.3.10/24
  tenant:
    ip_address: 172.16.0.201
    ip_address_uri: 172.16.0.201
    ip_subnet: 172.16.0.10/24
...

# Compute node parameters
compute-00-rack01:
  ctlplane:
    ip_address: 192.168.24.11
    ip_address_uri: 192.168.24.11
    ip_subnet: 192.168.24.0/24
  internal_api:
    ip_address: 172.16.2.11
    ip_address_uri: 172.16.2.11
    ip_subnet: 172.16.2.10/24
  storage:
    ip_address: 172.16.1.11
    ip_address_uri: 172.16.1.11
    ip_subnet: 172.16.1.10/24
  tenant:
    ip_address: 172.16.0.11
    ip_address_uri: 172.16.0.11
    ip_subnet: 172.16.0.10/24
...

# Ceph node parameters
ceph-00-rack01:
  ctlplane:
    ip_address: 192.168.24.101
    ip_address_uri: 192.168.24.101
    ip_subnet: 192.168.24.0/24
  storage:
    ip_address: 172.16.1.101
    ip_address_uri: 172.16.1.101
    ip_subnet: 172.16.1.10/24
  storage_mgmt:
    ip_address: 172.16.3.101
    ip_address_uri: 172.16.3.101
    ip_subnet: 172.16.3.10/24
...

# Virtual IP address parameters
ControlPlaneVipData:
  fixed_ips:
  - ip_address: 192.168.24.5
  name: control_virtual_ip
  network:
    tags: [192.168.24.0/24]
    subnets:
    - ip_version: 4
```

```
VipPortMap
 external:
   ip_address: 10.0.0.100
   ip_address_uri: 10.0.0.100
   ip_subnet: 10.0.0.100/24
 internal_api:
   ip_address: 172.16.2.100
   ip_address_uri: 172.16.2.100
   ip_subnet: 172.16.2.100/24
 storage:
   ip_address: 172.16.1.100
   ip_address_uri: 172.16.1.100
   ip_subnet: 172.16.1.100/24
 storage_mgmt:
   ip_address: 172.16.3.100
   ip_address_uri: 172.16.3.100
   ip_subnet: 172.16.3.100/24

RedisVirtualFixedIPs:
 - ip_address: 192.168.24.6
     use_neutron: false
```

1. **NodePortMap 映射定义节点分配的名称。**

2. **节点的短主机名，其格式为 < node_hostname>。**

3. **节点的网络定义和 IP 分配。网络包括 ctlplane、external、internal_api、管理、storage、storage_mgmt 和 租户。IP 分配包括 ip_address、ip_address_uri 和 ip_subnet：**

   - **IPv4: ip_address 和 ip_address_uri 应设置为相同的值。**
   - **IPv6 ：**
     - **ip_address ：设置为没有括号的 IPv6 地址。**
     - **ip_address_uri: 设置为方括号中的 IPv6 地址，例如 [2001:0db8:85a3:0000:0000:8a2e:0370:7334]。**

7.4.6. 为预置备节点使用单独网络

默认情况下，director 使用 Provisioning 网络作为 overcloud Control Plane。但是，如果此网络被隔离且不可路由，则节点在配置期间不能与 director 的内部 API 通信。在这种情况下，您可能需要为节点指定一个单独的网络，并进行配置，以便通过公共 API 与 director 通信。

对于此情境，有几个需要满足的要求：

- overcloud 节点必须容纳来自 第 7.4.5 节 “配置 control plane 网络” 的基本网络配置。
- 您必须在 director 上启用 SSL/TLS，以便使用公共 API 端点。如需更多信息，请参阅在 overcloud 公共端点中启用 SSL/TLS。
- 您必须为 director 指定一个可访问的完全限定域名（FQDN）。此 FQDN 必须解析为 director 的可路由 IP 地址。使用 undercloud.conf 文件中的 undercloud_public_host 参数来设置这个 FQDN。

本节中的示例使用了与主要情境不同的 IP 地址分配：

表 7.12. Provisioning 网络 IP 分配信息

节点名	IP 地址或 FQDN
Director（内部 API）	192.168.24.1 （Provisioning 网络和 Control Plane）
Director（公共 API）	10.1.1.1 / director.example.com
overcloud 虚拟 IP	192.168.100.1
控制器 0	192.168.100.2
计算 0	192.168.100.3

以下章节为需要单独的 overcloud 节点网络的情境提供额外配置。

IP 地址分配

IP 分配的方法类似于 第 7.4.5 节 “配置 control plane 网络”。但是，由于 Control Plane 可能无法从部署的服务器路由，因此您可以使用 NodePortMap、ControlPlaneVipData 和 VipPortMap 参数从您选

择的 overcloud 节点子网分配 IP 地址，包括访问 Control Plane 的虚拟 IP 地址。以下示例是来自 第 7.4.5 节 "配置 control plane 网络" 的 deployed-ports.yaml 自定义环境文件的修改版本，它适用于这个网络架构：

```
parameter_defaults:
  NodePortMap:
    controller-00-rack01
      ctlplane
        ip_address: 192.168.100.2
        ip_address_uri: 192.168.100.2
        ip_subnet: 192.168.100.0/24
...
    compute-00-rack01:
      ctlplane
        ip_address: 192.168.100.3
        ip_address_uri: 192.168.100.3
        ip_subnet: 192.168.100.0/24
...
  ControlPlaneVipData:
    fixed_ips:
    - ip_address: 192.168.100.1
    name: control_virtual_ip
    network:
      tags: [192.168.100.0/24]
    subnets:
    - ip_version: 4
  VipPortMap:
    external:
      ip_address: 10.0.0.100
      ip_address_uri: 10.0.0.100
      ip_subnet: 10.0.0.100/24
....
  RedisVirtualFixedIPs: 1
    - ip_address: 192.168.100.10
      use_neutron: false
```

1

RedisVipPort 资源被映射至 network/ports/noop.yaml。由于默认 Redis VIP 地址来自 Control Plane，因此该映射是必要的。在这种情况下，使用 noop 来禁用这种 Control Plane 映射。

7.4.7. 映射预置备的节点主机名

配置预置备节点时，必须将基于 heat 的主机名映射到实际主机名，以便 ansible-playbook 能够到达可解析主机。请使用 HostnameMap 来映射这些值。

步骤

1.

创建环境文件，例如 **hostname-map.yaml**，并包括 **HostnameMap** 参数和主机名映射。请遵循以下语法：

```
parameter_defaults:
  HostnameMap:
    [HEAT HOSTNAME]: [ACTUAL HOSTNAME]
    [HEAT HOSTNAME]: [ACTUAL HOSTNAME]
```

[HEAT HOSTNAME] 通常符合以下惯例：**[STACK NAME]-[ROLE]-[INDEX]**：

```
parameter_defaults:
  HostnameMap:
    overcloud-controller-0: controller-00-rack01
    overcloud-controller-1: controller-01-rack02
    overcloud-controller-2: controller-02-rack03
    overcloud-novacompute-0: compute-00-rack01
    overcloud-novacompute-1: compute-01-rack01
    overcloud-novacompute-2: compute-02-rack01
```

2. 保存 **hostname-map.yaml** 文件。

7.4.8. 为预置备节点配置 Ceph Storage

在 **undercloud** 主机上完成以下步骤，为已部署的节点配置 **Ceph**。

流程

1. 在 **undercloud** 主机上，创建环境变量 **OVERCLOUD_HOSTS**，并将该变量设置为您要用作 **Ceph** 客户端的 **overcloud** 主机的 **IP** 地址列表（以空格分隔）：

```
export OVERCLOUD_HOSTS="192.168.1.8 192.168.1.42"
```

2. 默认的 **overcloud** 计划名称是 **overcloud**。如果使用其他名称，请创建一个环境变量 **OVERCLOUD_PLAN** 来存储您的自定义名称：

```
export OVERCLOUD_PLAN="<custom-stack-name>"
```

- 将 **<custom-stack-name>** 替换为堆栈的名称。

3. 运行 **enable-ssh-admin.sh** 脚本，以在 **Ansible** 可用于配置 **Ceph** 客户端的 **overcloud** 节

点上配置用户：

```
bash /usr/share/openstack-tripleo-heat-templates/deployed-server/scripts/enable-ssh-admin.sh
```

运行 openstack overcloud deploy 命令时，Ansible 配置您在 OVERCLOUD_HOSTS 变量中定义为 Ceph 客户端的主机。

7.4.9. 利用预置备节点创建 overcloud

overcloud 部署使用标准 CLI 方法。对于预置备的节点，该部署命令需要使用来自核心 heat 模板集的一些额外选项和环境文件：

- --disable-validations - 使用此选项禁止对未用于预置备基础架构的服务执行基本的 CLI 验证功能。如果不禁止这些验证，部署将失败。
- environments/deployed-server-environment.yaml - 包含此环境文件以创建和配置预置备基础架构。这种环境文件用 OS::Heat::DeployedServer 资源代替 OS::Nova::Server 资源。

以下命令是示例 overcloud 部署命令，且环境文件特定于预置备的架构：

```
$ source ~/stackrc
(undercloud)$ openstack overcloud deploy \
  --disable-validations \
  -e /usr/share/openstack-tripleo-heat-templates/environments/deployed-server-environment.yaml \
  -e /home/stack/templates/deployed-ports.yaml \
  -e /home/stack/templates/hostname-map.yaml \
  --overcloud-ssh-user stack \
  --overcloud-ssh-key ~/.ssh/id_rsa \
  <OTHER OPTIONS>
```

--overcloud-ssh-user 和 --overcloud-ssh-key 选项用于在配置阶段通过 SSH 连接每个 overcloud 节点，创建初始 tripleo-admin 用户，以及将 SSH 密钥注入 /home/tripleo-admin/.ssh/authorized_keys。要注入 SSH 密钥，请为初始 SSH 连接指定包含 --overcloud-ssh-user 和 --overcloud-ssh-key（默认为 ~/.ssh/id_rsa）的凭证。要限制暴露使用 --overcloud-ssh-key 选项指定的私钥，director 不会将该密钥传递给任何 API 服务，而只有 director openstack overcloud deploy 命令使用此密钥来为 tripleo-admin 用户启用访问权限。

7.4.10. 访问 overcloud

director 生成一个凭据文件，其中包含从 undercloud 操作 overcloud 所需的凭据。director 将此文件保存为 stack 用户主目录的 overcloudrc 文件。

流程

1. 获取 overcloudrc 文件：

```
(undercloud)$ source ~/overcloudrc
```

命令提示符的变化表示您正在访问 overcloud：

```
(overcloud)$
```

2. 要返回与 undercloud 交互，请 source stackrc 文件：

```
(overcloud)$ source ~/stackrc
(undercloud)$
```

命令提示符的变化表示您要访问 undercloud：

```
(undercloud)$
```

7.4.11. 扩展预置备节点

扩展预置备节点的流程与扩展 overcloud 节点 中的标准扩展流程类似。但是，添加新预置备节点的流程有所不同，因为预置备节点不使用裸机置备服务(ironic)和计算服务(nova)中的标准注册和管理流程。

7.4.11.1. 扩展预置备节点

使用预置备节点扩展 overcloud 时，必须在每个节点上配置编配代理以对应 director 的节点计数。

流程

1. 准备新的预置备节点。如需更多信息，请参阅 预置备节点要求。

2. 扩展节点。有关更多信息，请参阅 扩展 overcloud 节点。

7.4.11.2. 缩减预置备节点

在缩减具有预置备节点的 overcloud 时，请遵循 扩展 overcloud 节点 中的缩减说明。

在缩减操作中，您可以对置备或预置备节点的 Red Hat OpenStack Platform (RHOSP)使用主机名。您还可以将 UUID 用于 RHOSP 置备的节点。但是，pre-provisoned 节点没有 UUID，因此您总是使用主机名。

流程

1. 检索您要删除的节点的名称：

```
$ openstack stack resource list overcloud -n5 --filter
type=OS::TripleO::ComputeDeployedServerServer
```

2. 删除节点：

```
$ openstack overcloud node delete --stack <overcloud> <node> [... <node>]
```

- 将 <overcloud> 替换为 overcloud 堆栈的名称或 UUID。
- 将 & lt;node > 替换为您要删除的节点的主机名，从第 1 步中返回的 stack_name 列中获取。

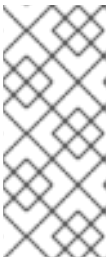
3. 确保节点已被删除：

```
$ openstack stack list
```

当删除操作完成后，overcloud 堆栈的状态会显示 UPDATE_COMPLETE。

4. 关闭已删除的节点。在标准部署中，director 上的裸机服务关闭移除的节点。使用预置备节点时，您必须手动关闭删除的节点，或者为每个物理系统使用电源管理控制。从堆栈中移除节点之后，如果您不关闭它们，它们可能保持运行，并作为 overcloud 环境的组成部分重新连接。

5. 将移除的节点重新置备为基础操作系统配置，以便它们不会在以后意外加入 overcloud。

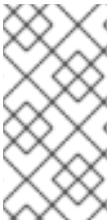


注意

在将之前已经从 overcloud 移除的节点重新部署到新的基础操作系统之前，不要尝试再次使用它们。缩减流程只从 overcloud 堆栈移除节点，不会卸载任何软件包。

7.4.12. 移除预置备 overcloud

要移除使用预置备节点的整个 overcloud，请参阅 第 9.7 节 “删除 overcloud 堆栈” 获取标准 overcloud 移除流程。移除 overcloud 后，关闭所有节点并将其重新置备为基础操作系统配置。



注意

在将之前已经从 overcloud 移除的节点重新部署到新的基础操作系统之前，不要尝试再次使用它们。移除流程只删除 overcloud 堆栈，不会卸载任何软件包。

第 8 章 执行 OVERCLOUD 安装后任务

本章介绍创建 overcloud 后要立即执行的任务。这些任务可确保您的 overcloud 随时可用。

8.1. 检查 OVERCLOUD 部署状态

要检查 overcloud 的部署状态，可使用 openstack overcloud status 命令。此命令返回所有部署步骤的结果。

步骤

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 运行部署状态命令：

```
$ openstack overcloud status
```

此命令的输出显示 overcloud 的状态：

```
+-----------+---------------------+---------------------+-------------------+
| Plan Name |       Created       |       Updated       | Deployment Status |
+-----------+---------------------+---------------------+-------------------+
| overcloud | 2018-05-03 21:24:50 | 2018-05-03 21:27:59 |   DEPLOY_SUCCESS  |
+-----------+---------------------+---------------------+-------------------+
```

如果您的 overcloud 使用其他名称，请使用 --stack 参数来选择具有不同名称的 overcloud：

```
$ openstack overcloud status --stack <overcloud_name>
```

- 将 <overcloud_name > 替换为 overcloud 的名称。

8.2. 创建基本 OVERCLOUD 类别

本指南中的步骤假定您的安装中包含有类型 (flavor)。如果您尚未创建任何 flavor，请完成以下步骤，

创建一组具有多种存储和处理功能的基本默认 flavor：

步骤

1. 获取 overcloudrc 文件：

```
$ source ~/overcloudrc
```

2. 运行 openstack flavor create 命令以创建类别。使用以下选项指定每种类别的硬件要求：

--disk

为虚拟机卷定义硬盘空间。

--ram

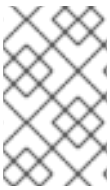
定义虚拟机所需的 RAM。

--vcpus

定义虚拟机的虚拟 CPU 的数量。

3. 以下示例创建默认 overcloud 类别：

```
$ openstack flavor create m1.tiny --ram 512 --disk 0 --vcpus 1
$ openstack flavor create m1.smaller --ram 1024 --disk 0 --vcpus 1
$ openstack flavor create m1.small --ram 2048 --disk 10 --vcpus 1
$ openstack flavor create m1.medium --ram 3072 --disk 10 --vcpus 2
$ openstack flavor create m1.large --ram 8192 --disk 10 --vcpus 4
$ openstack flavor create m1.xlarge --ram 8192 --disk 10 --vcpus 8
```



注意

使用 $ openstack flavor create --help 来进一步了解 openstack flavor create 命令。

8.3. 创建默认租户网络

overcloud 需要默认租户网络，以便虚拟机在内部通信。

步骤

1. 获取 overcloudrc 文件：

```
$ source ~/overcloudrc
```

2. 创建默认租户网络：

```
(overcloud) $ openstack network create default
```

3. 在网络中创建一个子网：

```
(overcloud) $ openstack subnet create default --network default --gateway 172.20.1.1 --
subnet-range 172.20.0.0/16
```

4. 确认所创建的网络：

```
(overcloud) $ openstack network list
+----------------------+-------------+--------------------------------------+
| id                   | name        | subnets                              |
+----------------------+-------------+--------------------------------------+
| 95fadaa1-5dda-4777... | default     | 7e060813-35c5-462c-a56a-1c6f8f4f332f |
+----------------------+-------------+--------------------------------------+
```

这些命令创建名为 **default** 的基本 **Networking** 服务 **(neutron)** 网络。**overcloud** 自动使用内部 **DHCP** 机制将此网络中的 **IP** 地址分配给虚拟机。

8.4. 创建默认浮动 IP 网络

要从 **overcloud** 以外访问您的虚拟机，您必须配置一个外部网络，为虚拟机提供浮动 **IP** 地址。

此流程包含两个示例。使用最适合您环境的示例：

- 原生 **VLAN**（扁平网络）
- 非原生 **VLAN**（**VLAN** 网络）

这两个示例都涉及使用名称 public 创建网络。overcloud 需要将这一特定名称用于默认的浮动 IP 池。这个名称对于 第 8.7 节 “验证 overcloud” 中的验证测试也很重要。

默认情况下，Openstack Networking (neutron) 将名为 datacentre 的物理网络名称映射到主机节点上的 br-ex 网桥。您将 public overcloud 网络连接到物理 datacentre，这将通过 br-ex 网桥提供网关。

先决条件

- 专用接口或原生 VLAN 用于浮动 IP 网络。

步骤

1. 获取 overcloudrc 文件：

```
$ source ~/overcloudrc
```

2. 创建公共网络：

- 创建扁平网络获取原生 VLAN 连接：

```
(overcloud) $ openstack network create public --external --provider-network-type flat --provider-physical-network datacentre
```

- 创建 vlan 网络获取非原生 VLAN 连接：

```
(overcloud) $ openstack network create public --external --provider-network-type vlan --provider-physical-network datacentre --provider-segment 201
```

使用 --provider-segment 选项定义您要使用的 VLAN。在本例中，VLAN 是 201。

3. 使用浮动 IP 地址分配池创建子网。在本示例中，IP 范围为 10.1.1.51 到 10.1.1.250：

```
(overcloud) $ openstack subnet create public --network public --dhcp --allocation-pool start=10.1.1.51,end=10.1.1.250 --gateway 10.1.1.1 --subnet-range 10.1.1.0/24
```

确保此范围与外部网络中的其他 IP 地址不冲突。

8.5. 创建默认提供商网络

提供商网络是另一类型的外部网络连接，将流量从私有租户网络路由到外部基础架构网络。该提供商网络类似于浮动 IP 网络，但提供商网络使用逻辑路由器将私有网络连接到提供商网络。

此流程包含两个示例。使用最适合您环境的示例：

- 原生 VLAN（扁平网络）
- 非原生 VLAN（VLAN 网络）

默认情况下，Openstack Networking (neutron) 将名为 datacentre 的物理网络名称映射到主机节点上的 br-ex 网桥。您将 public overcloud 网络连接到物理 datacentre，这将通过 br-ex 网桥提供网关。

步骤

1. 获取 overcloudrc 文件：

```
$ source ~/overcloudrc
```

2. 创建提供商网络。
   - 创建扁平网络获取原生 VLAN 连接：

```
(overcloud) $ openstack network create provider --external --provider-network-type flat --provider-physical-network datacentre --share
```

   - 创建 vlan 网络获取非原生 VLAN 连接：

```
(overcloud) $ openstack network create provider --external --provider-network-type vlan --provider-physical-network datacentre --provider-segment 201 --share
```

使用 **--provider-segment** 选项定义您要使用的 **VLAN**。在本例中，**VLAN** 是 **201**。

- 使用 **--share** 选项创建共享网络。或者，指定租户而不是指定 **--share**，以便只有租户才能访问新网络。
- 使用 **--external** 选项将提供商网络标记为外部网络。只有 **Operator** 才能在外部网络上创建端口。

3. 向提供商网络添加一个子网以提供 **DHCP** 服务：

```
(overcloud) $ openstack subnet create provider-subnet --network  provider --dhcp --allocation-pool start=10.9.101.50,end=10.9.101.100 --gateway 10.9.101.254 --subnet-range 10.9.101.0/24
```

4. 创建一个路由器，使其他网络能够通过提供商网络路由流量：

```
(overcloud) $ openstack router create external
```

5. 将路由器的外部网关设置为提供商网络：

```
(overcloud) $ openstack router set --external-gateway provider external
```

6. 将其他网络添加到该路由器。例如，运行以下命令以将子网 **subnet1** 附加到路由器上：

```
(overcloud) $ openstack router add subnet external subnet1
```

此命令将 **subnet1** 添加到路由表中，并允许使用 **subnet1** 的虚拟机中的流量路由到提供商网络。

8.6. 创建其他网桥映射

只要部署期间映射其他网桥，浮动 **IP** 网络可以使用任何网桥，而不只是 **br-ex**。

流程

1.

要将名为 br-floating 的新网桥映射到浮动物理网络，请在环境文件中包含 NeutronBridgeMappings 参数：

```
parameter_defaults:
  NeutronBridgeMappings: "datacentre:br-ex,floating:br-floating"
```

2. 使用此方法，您可以在创建 overcloud 后创建单独的外部网络。例如，要创建映射到 floating 物理网络的浮动 IP 网络，请运行以下命令：

```
$ source ~/overcloudrc
(overcloud) $ openstack network create public --external --provider-physical-network floating
--provider-network-type vlan --provider-segment 105
(overcloud) $ openstack subnet create public --network public --dhcp --allocation-pool
start=10.1.2.51,end=10.1.2.250 --gateway 10.1.2.1 --subnet-range 10.1.2.0/24
```

8.7. 验证 OVERCLOUD

Overcloud 会使用 OpenStack Integration Test Suite (tempest) 工具集来执行一系列集成测试。本节介绍运行集成测试的准备工作。有关如何使用 OpenStack Integration Test Suite 的完整说明，请参阅使用 Red Hat OpenStack Platform Integration Test Suite 验证您的云。

Integration Test Suite 需要执行一些安装后步骤以确保成功测试。

步骤

1. 如果在 undercloud 上运行这个测试，请确保 undercloud 主机能够访问 overcloud 的内部 API 网络。例如，在 undercloud 主机上添加一个临时的 VLAN，用于使用 172.16.0.201/24 地址访问内部 API 网络 (ID: 201)：

```
$ source ~/stackrc
(undercloud) $ sudo ovs-vsctl add-port br-ctlplane vlan201 tag=201 -- set interface vlan201
type=internal
(undercloud) $ sudo ip l set dev vlan201 up; sudo ip addr add 172.16.0.201/24 dev vlan201
```

2. 运行集成测试，如 使用 Red Hat OpenStack Platform Integration Test Suite 验证云 中所述。

3. 在验证完成后，删除所有到 overcloud 的内部 API 的临时连接。在这个示例中，使用以下命令删除以前在 undercloud 中创建的 VLAN：

```
$ source ~/stackrc
(undercloud) $ sudo ovs-vsctl del-port vlan201
```

8.8. 防止 OVERCLOUD 被移除

您可以为编排服务(heat)设置自定义策略，以防止 overcloud 被删除。

要重新启用堆栈删除，请从 custom_env_files 参数中删除 prevent-stack-delete.yaml 文件，再运行 openstack undercloud install 命令。

步骤

1. 创建名为 prevent-stack-delete.yaml 的环境文件。

2. 设置 HeatApiPolicies 参数：

```
parameter_defaults:
  HeatApiPolicies:
    heat-deny-action:
      key: 'actions:action'
      value: 'rule:deny_everybody'
    heat-protect-overcloud:
      key: 'stacks:delete'
      value: 'rule:deny_everybody'
```

- heat-deny-action 是您必须在 undercloud 安装中包含的默认策略。

- 将 heat-protect-overcloud 策略设置为 rule:deny_everybody，以防止任何人删除 overcloud 中的任何堆栈。

注意

将 overcloud 保护设置为 rule:deny_everybody 表示您无法执行以下任一功能：

- 删除 overcloud。
- 删除单独的 Compute 或 Storage 节点。
- 替换 Controller 节点。

3. 将 prevent-stack-delete.yaml 环境文件添加到 undercloud.conf 文件中的 custom_env_files 参数：

```
custom_env_files = prevent-stack-delete.yaml
```

4. 运行 undercloud 安装命令以刷新配置：

```
$ openstack undercloud install
```

第 9 章 执行基本 OVERCLOUD 管理任务

本章介绍在 overcloud 生命周期过程中可能需要执行的基本任务。

9.1. 通过 SSH 访问 OVERCLOUD 节点

您可以通过 SSH 协议访问每个 overcloud 节点。

- 每个 overcloud 节点都包含一个 tripleo-admin 用户，以前称为 heat-admin 用户。
- undercloud 上的 stack 用户具有对各个 overcloud 节点上的 tripleo-admin 用户进行基于密钥的 SSH 访问权限。
- 所有 overcloud 节点都有一个短主机名，undercloud 将其解析为 control plane 网络上的 IP 地址。每个短主机名都使用 .ctlplane 后缀。例如，overcloud-controller-0 的短名称为 overcloud-controller-0.ctlplane

先决条件

- 具有可正常工作的 control plane 网络的已部署 overcloud。

步骤

1. 以 stack 用户身份登录 undercloud。
2. 查找要访问的节点的名称：

```
(undercloud)$ metalsmith list
```

3. 以 tripleo-admin 用户身份连接到节点：

```
(undercloud)$ ssh tripleo-admin@overcloud-controller-0
```

9.2. 管理容器化服务

Red Hat OpenStack (RHOSP) 平台在 undercloud 和 overcloud 节点上的容器中运行服务。在某些情况下，您可能需要控制主机上的单个服务。本节介绍了可在节点上运行的用于管理容器化服务的一些常见命令。

列出容器和镜像

要列出运行中的容器，请运行以下命令：

```
$ sudo podman ps
```

要在命令输出中包括停止的或失败的容器，将 --all 选项添加到命令中：

```
$ sudo podman ps --all
```

要列出容器镜像，请运行以下命令：

```
$ sudo podman images
```

检查容器属性

要查看容器或容器镜像的属性，请使用 podman inspect 命令。例如，要检查 keystone 容器，请运行以下命令：

```
$ sudo podman inspect keystone
```

使用 Systemd 服务管理容器

早期版本的 OpenStack Platform 使用 Docker 及其守护进程管理容器。现在，Systemd 服务接口管理容器的生命周期。每个容器都是一个服务，您运行 Systemd 命令，为每个容器执行特定操作。

注意

不建议使用 Podman CLI 停止、启动和重启容器，因为 Systemd 会应用重启策略。请使用 Systemd 服务命令。

要检查容器状态，请运行 systemctl status 命令：

```
$ sudo systemctl status tripleo_keystone
```

```
● tripleo_keystone.service - keystone container
   Loaded: loaded (/etc/systemd/system/tripleo_keystone.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-02-15 23:53:18 UTC; 2 days ago
 Main PID: 29012 (podman)
   CGroup: /system.slice/tripleo_keystone.service
           └─29012 /usr/bin/podman start -a keystone
```

要停止容器，请运行 systemctl stop 命令：

```
$ sudo systemctl stop tripleo_keystone
```

要启动容器，请运行 systemctl start 命令：

```
$ sudo systemctl start tripleo_keystone
```

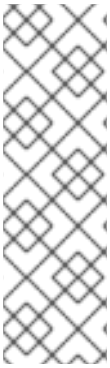
要重启容器，请运行 systemctl restart 命令：

```
$ sudo systemctl restart tripleo_keystone
```

由于没有守护进程监控容器状态，Systemd 在以下情况下自动重启大多数容器：

- **清除退出代码或信号，如运行 podman stop 命令。**
- **取消清除退出代码，如启动后的 podman 容器崩溃。**
- **取消清除信号。**
- **如果容器启动时间超过 1 分 30 秒，则超时。**

有关 Systemd 服务的更多信息，请参阅 systemd.service 文档。



注意

在重启容器后，针对其中的服务配置文件所做的所有更改都会恢复。这是因为容器基于 **/var/lib/config-data/puppet-generated/** 中节点的本地文件系统上的文件重新生成服务配置。例如，如果您编辑了 **keystone** 容器中的 **/etc/keystone/keystone.conf**，并重启了该容器，则该容器会使用节点的本地文件系统上的 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 来重新生成配置，以覆盖重启之前在该容器中所做的所有更改。

使用 Systemd 计时器监控 podman 容器

Systemd 计时器接口管理容器运行健康检查。每个容器都有一个计时器，它会运行一个服务单员来执行健康检查脚本。

要列出所有 **OpenStack Platform** 容器计时器，请运行 **systemctl list-timers** 命令并将输出限制为包含 **tripleo** 的行：

```
$ sudo systemctl list-timers | grep tripleo
Mon 2019-02-18 20:18:30 UTC  1s left       Mon 2019-02-18 20:17:26 UTC  1min 2s ago
tripleo_nova_metadata_healthcheck.timer           tripleo_nova_metadata_healthcheck.service
Mon 2019-02-18 20:18:34 UTC  5s left       Mon 2019-02-18 20:17:23 UTC  1min 5s ago
tripleo_keystone_healthcheck.timer                tripleo_keystone_healthcheck.service
Mon 2019-02-18 20:18:35 UTC  6s left       Mon 2019-02-18 20:17:13 UTC  1min 15s ago
tripleo_memcached_healthcheck.timer               tripleo_memcached_healthcheck.service
(...)
```

要检查特定容器计时器的状态，请对运行状况检查服务运行 **systemctl status** 命令：

```
$ sudo systemctl status tripleo_keystone_healthcheck.service
● tripleo_keystone_healthcheck.service - keystone healthcheck
   Loaded: loaded (/etc/systemd/system/tripleo_keystone_healthcheck.service; disabled; vendor
preset: disabled)
   Active: inactive (dead) since Mon 2019-02-18 20:22:46 UTC; 22s ago
  Process: 115581 ExecStart=/usr/bin/podman exec keystone /openstack/healthcheck (code=exited,
status=0/SUCCESS)
 Main PID: 115581 (code=exited, status=0/SUCCESS)

Feb 18 20:22:46 undercloud.localdomain systemd[1]: Starting keystone healthcheck...
Feb 18 20:22:46 undercloud.localdomain podman[115581]: {"versions": {"values": [{"status": "stable",
"updated": "2019-01-22T00:00:00Z", "..."}]}}
Feb 18 20:22:46 undercloud.localdomain podman[115581]: 300 192.168.24.1:35357 0.012 seconds
Feb 18 20:22:46 undercloud.localdomain systemd[1]: Started keystone healthcheck.
```

要停止、启动、重启和显示容器计时器的状态，请根据 **.timer Systemd** 资源运行相关 **systemctl** 命令。例如，要检查 **tripleo_keystone_healthcheck.timer** 资源的状态，可运行以下命令：

```
$ sudo systemctl status tripleo_keystone_healthcheck.timer
● tripleo_keystone_healthcheck.timer - keystone container healthcheck
   Loaded: loaded (/etc/systemd/system/tripleo_keystone_healthcheck.timer; enabled; vendor preset:
disabled)
   Active: active (waiting) since Fri 2019-02-15 23:53:18 UTC; 2 days ago
```

如果健康状况检查服务被禁用，但该服务的计时器存在并启用，则意味着检查当前超时，但将根据计时器运行。您还可以手动启动检查。

注意

podman ps 命令不显示容器运行状态。

检查容器日志

Red Hat OpenStack Platform 17.1 会记录来自所有容器的所有标准输出(stdout)，以及整合在 /var/log/containers/stdout 中每个容器的标准错误(stderr)。

主机会对此目录进行日志轮转以防止产生巨大的文件及占用太多磁盘空间的问题。

如果替换了容器，新的容器将输出到同一日志文件中，因为 podman 会使用容器名而非容器 ID。

您也可以使用 podman logs 命令检查容器化服务的日志。例如，要查看 keystone 容器的日志，请运行以下命令：

```
$ sudo podman logs keystone
```

访问容器

要进入容器化服务的 shell，请使用 podman exec 命令以启动 /bin/bash。例如，要进入 keystone 容器的 shell，请运行以下命令：

```
$ sudo podman exec -it keystone /bin/bash
```

要以根用户身份进入 keystone 容器的 shell，请运行以下命令：

```
$ sudo podman exec --user 0 -it <NAME OR ID> /bin/bash
```

要退出容器，请运行以下命令：

```
# exit
```

9.3. 修改 OVERCLOUD 环境

您可以修改 overcloud 以添加额外功能或更改现有操作。

流程

1. 要修改 overcloud，请在自定义环境文件和 heat 模板中进行修改，然后从您的初始 overcloud 创建中重新运行 openstack overcloud deploy 命令。例如，如果您使用 第 7.3 节 “配置和部署 overcloud” 创建 overcloud，请重新运行以下命令：

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates \
  -e ~/templates/overcloud-baremetal-deployed.yaml \
  -e ~/templates/network-environment.yaml \
  -e ~/templates/storage-environment.yaml \
  --ntp-server pool.ntp.org
```

director 会在 heat 中检查 overcloud 栈，然后根据环境文件和 heat 模板更新栈中的每一项目。director 不会重新创建 overcloud，而是更改现有 overcloud。

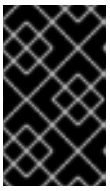
重要

从自定义环境文件中移除参数不会将参数值恢复到默认配置。您必须从 /usr/share/openstack-tripleo-heat-templates中的核心 heat 模板集合中获得默认值，然后在自定义环境文件中手动设置这些值。

2. 如果您要包括新的环境文件，则使用 `-e` 选项将其添加到 openstack overcloud deploy 命令中。例如：

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates \
  -e ~/templates/new-environment.yaml \
  -e ~/templates/network-environment.yaml \
  -e ~/templates/storage-environment.yaml \
  -e ~/templates/overcloud-baremetal-deployed.yaml \
  --ntp-server pool.ntp.org
```

此命令将环境文件中的新参数和资源纳入堆栈。



重要

不建议手动修改 overcloud 配置，因为 director 稍后可能会覆盖这些修改。

9.4. 将虚拟机导入 OVERCLOUD

您可以将虚拟机从现有 OpenStack 环境迁移到 Red Hat OpenStack Platform (RHOSP) 环境。

步骤

1. 在现有 OpenStack 环境中，通过对一个运行的服务器进行快照并下载镜像来创建一个新镜像：

```
$ openstack server image create --name <image_name> <instance_name>
$ openstack image save --file <exported_vm.qcow2> <image_name>
```

   - 将 <instance_name > 替换为实例的名称。
   - 将 <image_name > 替换为新镜像的名称。
   - 使用导出的虚拟机的名称替换 <exported_vm.qcow2>。

2. 将导出的镜像复制到 undercloud 节点：

```
$ scp exported_vm.qcow2 stack@192.168.0.2:~/.
```

3. 以 stack 用户身份登录 undercloud。

4. 查找 overcloudrc 凭证文件：

```
$ source ~/overcloudrc
```

5. 将导出的镜像上传到 overcloud 中：

```
(overcloud) $ openstack image create --disk-format qcow2  -file <exported_vm.qcow2> --container-format bare <image_name>
```

6. 启动新实例：

```
(overcloud) $ openstack server create  --key-name default --flavor m1.demo --image imported_image --nic net-id=net_id <instance_name>
```

重要

您可以使用这些命令将每个虚拟机磁盘从现有 OpenStack 环境复制到新的 Red Hat OpenStack Platform。QCOW 快照丢掉了原始的层系统。

9.5. 启动临时 HEAT 进程

在以前的 Red Hat OpenStack Platform (RHOSP)中，系统安装的 Heat 进程用于安装 overcloud。现在，我们使用临时 Heat 来安装 overcloud，这意味着 heat-api 和 heat-engine 进程由 部署、更新 和 upgrade 命令按需启动。

在以前的版本中，您使用 openstack stack 命令创建和管理堆栈。默认情况下，此命令不再可用。为了进行故障排除和调试目的，例如，如果堆栈应该失败，您必须首先启动临时 Heat 进程才能使用 openstack stack 命令。

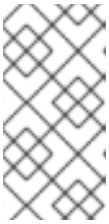
使用 openstack overcloud tripleo launch heat 命令，在部署外启用临时 heat。

流程

1. 启动临时 Heat 进程：

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-deploy/<overcloud>/heat-launcher --restore-db
```

- 将 <overcloud > 替换为 overcloud 堆栈的名称。



注意

命令会在启动 **Heat** 进程后退出，并且 **Heat** 进程在后台作为 **Podman** 容器集运行。

2. 验证 **ephemeral-heat** 进程是否正在运行：

```
(undercloud)$ sudo podman pod ps
POD ID        NAME            STATUS      CREATED         INFRA ID      # OF CONTAINERS
958b141609b2  ephemeral-heat  Running     2 minutes ago   44447995dbcf  3
```

3. 导出 **OS_CLOUD** 环境：

```
(undercloud)$ export OS_CLOUD=heat
```

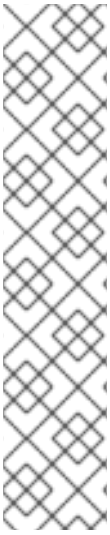
4. 列出已安装的堆栈：

```
(undercloud)$ openstack stack list
+--------------------------------------+------------+---------+-----------------+----------------------+--------------+
| ID                                   | Stack Name | Project | Stack Status    | Creation Time        | Updated Time |
+--------------------------------------+------------+---------+-----------------+----------------------+--------------+
| 761e2a54-c6f9-4e0f-abe6-c8e0ad51a76c | overcloud  | admin   | CREATE_COMPLETE | 2022-08-29T20:48:37Z | None         |
+--------------------------------------+------------+---------+-----------------+----------------------+--------------+
```

您可以使用 **openstack stack environment show** 和 **openstack stack resource list** 等命令进行调试。

5. 调试完成后，停止临时 **Heat** 进程：

```
(undercloud)$ openstack tripleo launch heat --kill
```



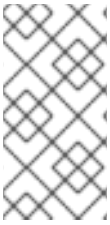
注意

有时，导出 heat 环境会失败。当使用其他凭据（如 overcloudrc ）时，可能会发生这种情况。在这种情况下，取消设置现有的环境，并提供 heat 环境。

```
(overcloud)$ unset OS_CLOUD
(overcloud)$ unset OS_PROJECT_NAME
(overcloud)$ unset OS_PROJECT_DOMAIN_NAME
(overcloud)$ unset OS_USER_DOMAIN_NAME
(overcloud)$ OS_AUTH_TYPE=none
(overcloud)$ OS_ENDPOINT=http://127.0.0.1:8006/v1/admin
(overcloud)$ export OS_CLOUD=heat
```

9.6. 运行动态清单脚本

您可以在 Red Hat OpenStack Platform (RHOSP)环境中运行基于 Ansible 的自动化。使用位于 /home/stack/overcloud-deploy/<stack> 目录中的 tripleo-ansible-inventory.yaml 清单文件来运行 ansible play 或 ad-hoc 命令。



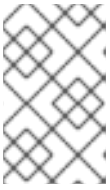
注意

如果要在 undercloud 上运行 Ansible playbook 或 Ansible 临时命令，则必须使用 /home/stack/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml 清单文件。

流程

1. 要查看您的节点清单，请运行以下命令：

```
(undercloud) $ ansible -i ./overcloud-deploy/<stack>/tripleo-ansible-inventory.yaml all --list
```



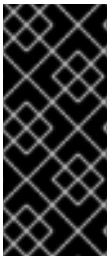
注意

将 stack 替换为部署的 overcloud 堆栈的名称。

2. 要在您的环境中执行 Ansible playbook，请运行 ansible 命令并使用 -i 选项包括到清单文件的完整路径。例如：

```
(undercloud) $ ansible <hosts> -i ./overcloud-deploy/tripleo-ansible-inventory.yaml
<playbook> <options>
```

- 将 <hosts> 替换为您要使用的主机类型：
  - controller，适用于所有 Controller 节点
  - compute，适用于所有 Compute 节点
  - overcloud，适用于所有 overcloud 子节点。例如， controller 和 compute 节点
  - "*"，适用于所有节点
- 将 & lt;options& gt; 替换为额外的 Ansible 选项。
  - 使用 --ssh-extra-args='-o StrictHostKeyChecking=no' 选项跳过主机密钥检查确认操作。
  - 使用 -u [USER] 选项更改执行 Ansible 自动化的 SSH 用户。默认的 overcloud SSH 用户由动态清单中的 ansible_ssh_user 参数自动定义。-u 选项会覆盖此参数。
  - 使用 -m [MODULE] 选项使用特定的 Ansible 模块。默认为 command，用于执行 Linux 命令。
  - 使用 -a [MODULE_ARGS] 选项为选定的模块定义参数。

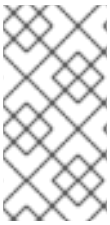


重要

overcloud 上的自定义 Ansible 自动化不是标准 overcloud 堆栈的一部分。后续执行 openstack overcloud deploy 命令可能会覆盖 overcloud 节点上的 OpenStack Platform 服务的基于 Ansible 的配置。

9.7. 删除 OVERCLOUD 堆栈

您可以删除 overcloud 堆栈，并取消置备所有堆栈节点。



注意

删除 overcloud 堆栈不会清除所有 overcloud 数据。如果您需要清除所有 overcloud 数据，请联系红帽支持。

流程

1. 以 stack 用户身份登录 undercloud 主机。

2. 查找 stackrc undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 检索堆栈中所有节点的列表及其当前状态：

```
(undercloud)$ openstack baremetal node list
+--------------------------------------+-------------+--------------------------------------+-------------+--
------------------+-------------+
| UUID                                 | Name        | Instance UUID                        | Power State |
Provisioning State | Maintenance |
+--------------------------------------+-------------+--------------------------------------+-------------+--
------------------+-------------+
| 92ae71b0-3c31-4ebb-b467-6b5f6b0caac7 | compute-0   | 059fb1a1-53ea-4060-9a47-
09813de28ea1 | power on    | active             | False       |
| 9d6f955e-3d98-4d1a-9611-468761cebabf | compute-1   | e73a4b50-9579-4fe1-bd1a-
556a2c8b504f | power on    | active             | False       |
| 8a686fc1-1381-4238-9bf3-3fb16eaec6ab | controller-0 | 6d69e48d-10b4-45dd-9776-
155a9b8ad575 | power on    | active             | False       |
| eb8083cc-5f8f-405f-9b0c-14b772ce4534 | controller-1 | 1f836ac0-a70d-4025-88a3-
bbe0583b4b8e | power on    | active             | False       |
| a6750f1f-8901-41d6-b9f1-f5d6a10a76c7 | controller-2 | e2edd028-cea6-4a98-955e-
5c392d91ed46 | power on    | active             | False       |
+--------------------------------------+-------------+--------------------------------------+-------------+--
------------------+-------------+
```

4. 删除 overcloud 堆栈并取消置备节点和网络：

```
(undercloud)$ openstack overcloud delete -b <node_definition_file> \
 --networks-file <networks_definition_file> --network-ports <stack>
```

- 将 <node_definition_file > 替换为节点定义文件的名称，如 overcloud-baremetal-deploy.yaml。

- 将 **<networks_definition_file >** 替换为网络定义文件的名称，如 **network_data_v2.yaml**。

- 将 **<stack>** 替换为您要删除的堆栈的名称。如果未指定，则默认堆栈为 **overcloud**。

5. 确认要删除 **overcloud**：

```
Are you sure you want to delete this overcloud [y/N]?
```

6. 等待 **overcloud** 删除，以及节点和网络取消置备。

7. 确认裸机节点已取消置备：

```
(undercloud) [stack@undercloud-0 ~]$ openstack baremetal node list
+--------------------------------------+-------------+--------------+------------+--------------------+----
---------+
| UUID                                 | Name        | Instance UUID | Power State | Provisioning State |
Maintenance |
+--------------------------------------+-------------+--------------+------------+--------------------+----
---------+
| 92ae71b0-3c31-4ebb-b467-6b5f6b0caac7 | compute-0   | None          | power off   |
available          | False       |
| 9d6f955e-3d98-4d1a-9611-468761cebabf | compute-1   | None          | power off   |
available          | False       |
| 8a686fc1-1381-4238-9bf3-3fb16eaec6ab | controller-0 | None          | power off   | available
| False       |
| eb8083cc-5f8f-405f-9b0c-14b772ce4534 | controller-1 | None          | power off   | available
| False       |
| a6750f1f-8901-41d6-b9f1-f5d6a10a76c7 | controller-2 | None          | power off   | available
| False       |
+--------------------------------------+-------------+--------------+------------+--------------------+----
---------+
```

8. 删除堆栈目录：

```
$ rm -rf ~/overcloud-deploy/<stack>
$ rm -rf ~/config-download/<stack>
```

注意

如果使用 openstack overcloud deploy 命令部署 overcloud 时使用了 --output-dir 和 --working-dir 选项，则堆栈的目录路径可能与默认值不同。

9.8. 管理本地磁盘分区的大小

如果您的本地磁盘分区在优化了分区大小后仍然填满，则执行以下任务之一：

- 从受影响的分区中手动删除文件。
- 添加新物理磁盘并将其添加到 LVM 卷组。如需更多信息，请参阅配置和管理逻辑卷。
- 过度置备分区以使用剩余的备用磁盘空间。这个选项是可能的，因为默认的整个磁盘 overcloud 镜像 overcloud-hardened-uefi-full.qcow2 由精简池支持。有关精简配置的逻辑卷的更多信息，请参阅 RHEL 配置和管理本地卷指南中的创建和管理精简置备卷（精简卷）。

警告

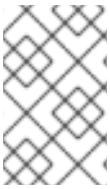
仅在无法手动删除文件或添加新物理磁盘时使用过度置备。如果空闲物理空间不足，在写入操作中过度置备可能会失败。

注意

添加新磁盘并过度置备分区需要支持例外。请联系红帽 客户体验与参与团队，讨论支持例外、如果适用或其他选项。

第 10 章 扩展 OVERCLOUD 节点

如果要在创建 overcloud 后添加或移除节点，您必须更新 overcloud。



注意

在开始横向扩展或移除 overcloud 节点之前，请确保您的裸机节点未处于维护模式。

下表介绍了对每个节点类型进行扩展的支持信息：

表 10.1. 每个节点类型的扩展支持

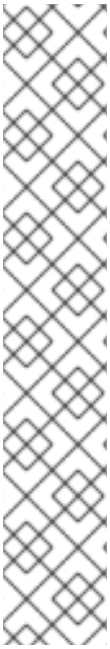
节点类型	扩展？	缩减？	备注
Controller	N	N	您可以使用 第 11 章 *替换 Controller 节点* 中的步骤替换 Controller 节点。
计算	Y	Y	
Ceph Storage 节点	Y	N	在初始创建的 overcloud 中必须至少有一个 Ceph Storage 节点。
Object Storage 节点	Y	Y	

重要

在扩展 overcloud 前，请确保至少有 10 GB 的可用空间。这些可用空间将在节点置备过程中用于保存镜像转换和缓存。

10.1. 向 OVERCLOUD 添加节点

您可以将更多节点添加到 overcloud。



注意

全新安装 Red Hat OpenStack Platform (RHOSP)不包括某些更新，如安全勘误和程序错误修复。因此，如果您要扩展使用红帽客户门户网站或 Red Hat Satellite Server 的连接环境，RPM 更新不会应用到新节点。要将最新的更新应用到 overcloud 节点，您必须执行以下操作之一：

- 在横向扩展操作后，完成节点的 overcloud 更新。
- 在 scale-out 操作前，使用 virt-customize 工具将软件包修改为基础 overcloud 镜像。有关更多信息，请参阅红帽知识库解决方案 使用 virt-customize 修改 Red Hat Linux OpenStack Platform Overcloud 镜像。

流程

1. 创建名为 newnodes.json 的新 JSON 文件，其中包含您要注册的新节点的详情：

```
{
  "nodes":[
    {
        "mac":[
            "dd:dd:dd:dd:dd:dd"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.02.24.207"
    },
    {
        "mac":[
            "ee:ee:ee:ee:ee:ee"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.02.24.208"
    }
  ]
}
```

2. 以 stack 用户身份登录 undercloud 主机。

3. 查找 stackrc undercloud 凭证文件：

```
$ source ~/stackrc
```

4. 注册新节点：

```
$ openstack overcloud node import newnodes.json
```

5. 为每个新节点启动内省过程：

```
$ openstack overcloud node introspect \
 --provide <node_1> [<node_2>] [<node_n>]
```

- 使用 --provide 选项，在内省后将所有指定的节点重置为 available 状态。

- 将 <node_1& gt ; , <node_ 2>，将直到 < node_n > 的所有节点替换为您要内省的每个节点的 UUID。

6. 为每个新节点配置镜像属性：

```
$ openstack overcloud node configure <node>
```

10.2. 扩展裸机节点

要增加现有 overcloud 中的裸机节点数量，请在 overcloud-baremetal-deploy.yaml 文件中增加节点数并重新部署 overcloud。

先决条件

- 新的裸机节点已注册、内省，并可用于调配和部署。如需更多信息，请参阅为 overcloud 注册节点 和 创建裸机节点硬件清单。

流程

1. 以 stack 用户身份登录 undercloud 主机。

2. 查找 stackrc undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 打开用于置备裸机节点的 overcloud-baremetal-deploy.yaml 节点定义文件。

4. 增加您要扩展的角色的 count 参数。例如，以下配置将 Object Storage 节点数增加到 4：

```
- name: Controller
  count: 3
- name: Compute
  count: 10
- name: ObjectStorage
  count: 4
```

5. 可选：为新节点配置预测节点放置。例如，使用以下配置在 node03 上置备新的 Object Storage 节点：

```
- name: ObjectStorage
  count: 4
  instances:
  - hostname: overcloud-objectstorage-0
    name: node00
  - hostname: overcloud-objectstorage-1
    name: node01
  - hostname: overcloud-objectstorage-2
    name: node02
  - hostname: overcloud-objectstorage-3
    name: node03
```

6. 可选：定义您要分配给新节点的任何其他属性。有关您可以在节点定义文件中配置节点属性的属性的更多信息，请参阅 裸机节点置备属性。

7. 如果您使用 Object Storage 服务(swift)和整个磁盘 overcloud 镜像，overcloud-hardened-uefi-full，请根据您的磁盘大小配置 /srv 分区的大小以及 /var 和 /srv 的存储要求。如需更多信息，请参阅为对象存储服务配置整个磁盘分区。

8. 置备 overcloud 节点：

```
$ openstack overcloud node provision \
  --stack <stack> \
  --network-config \
  --output <deployment_file> \
  /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 将 <stack> 替换为置备裸机节点的堆栈名称。如果未指定，则默认为 overcloud。
- 包含 --network-config 参数，为 cli-overcloud-node-network-config.yaml Ansible playbook 提供网络定义。
- 将 <deployment_file> 替换为用于部署命令生成的 heat 环境文件的名称，如 /home/stack/templates/overcloud-baremetal-deployed.yaml。

注意

如果您从 Red Hat OpenStack Platform 16.2 升级到 17.1，则必须在 openstack overcloud node provision 命令中在升级过程中创建或更新的 YAML 文件。例如，使用 /home/stack/tripleo-[stack]-baremetal-deploy.yaml 文件，而不是 /home/stack/templates/overcloud-baremetal-deployed.yaml 文件。有关更多信息，请参阅 执行 overcloud 的采用和 准备 Framework (16.2 到 17.1)。

9. 在一个单独的终端中监控置备进度。当置备成功时，节点状态将从 available 变为 active ：

```
$ watch openstack baremetal node list
```

10. 使用其他环境文件将生成的 overcloud-baremetal-deployed.yaml 文件添加到堆栈中，并部署 overcloud：

```
$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
  --deployed-server \
  --disable-validations \
  ...
```

10.3. 缩减裸机节点

若要缩减 overcloud 中的裸机节点数量，请标记您要从节点定义文件的堆栈中删除的节点，重新部署 overcloud，然后从 overcloud 中删除裸机节点。

先决条件

- 成功安装 undercloud。有关更多信息，请参阅在 undercloud 上安装 director。
- 成功部署 overcloud。如需更多信息，请参阅 使用预置备节点配置基本 overcloud。
- 如果要替换 Object Storage 节点，请将您要删除的节点中的数据复制到新替换节点。等待新节点上复制传递完成。在 /var/log/swift/swift.log 文件中检查复制传递进度。传递完成后，Object Storage 服务(swift)会将条目添加到类似以下示例的日志中：

```
Mar 29 08:49:05 localhost object-server: Object replication complete.
Mar 29 08:49:11 localhost container-server: Replication run OVER
Mar 29 08:49:13 localhost account-server: Replication run OVER
```

流程

1. 以 stack 用户身份登录 undercloud 主机。
2. 查找 stackrc undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 对于您要缩减的角色，减少 overcloud-baremetal-deploy.yaml 文件中的 count 参数。
4. 定义您要从堆栈中删除的每个节点的 主机名和 名称 （如果它们尚未在角色的 instances 属性中定义）。
5. 将 attribute provisioned: false 添加到您要删除的节点。例如，要从堆栈中删除节点 overcloud-objectstorage-1，请在 overcloud-baremetal-deploy.yaml 文件中包含以下代码片段：

```
- name: ObjectStorage
  count: 3
  instances:
```

```
- hostname: overcloud-objectstorage-0
  name: node00
- hostname: overcloud-objectstorage-1
  name: node01
  # Removed from cluster due to disk failure
  provisioned: false
- hostname: overcloud-objectstorage-2
  name: node02
- hostname: overcloud-objectstorage-3
  name: node03
```

重新部署 overcloud 后，堆栈中不再存在使用 provisioned: false 属性定义的节点。但是，这些节点仍然以置备状态运行。

注意

要从堆栈中临时删除节点，请使用 provisioned: false 属性部署 overcloud，然后使用 provisioned: true 属性重新部署 overcloud，以将节点返回到堆栈。

6. **从 overcloud 删除节点：**

```
$ openstack overcloud node delete \
  --stack <stack> \
  --baremetal-deployment \
   /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- **将 & lt;stack> 替换为置备裸机节点的堆栈名称。如果未指定，则默认为 overcloud。**

注意

不要将您要从堆栈中删除的节点作为命令参数包括在 openstack overcloud node delete 命令中。

7. **删除 ironic 节点：**

```
$ openstack baremetal node delete <IRONIC_NODE_UUID>
```

将 IRONIC_NODE_UUID 替换为节点的 UUID。

8. 置备 overcloud 节点以生成更新的 heat 环境文件，以包括在部署命令中：

```
$ openstack overcloud node provision \
  --stack <stack> \
  --output <deployment_file> \
  /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 将 <deployment_file> 替换为用于部署命令生成的 heat 环境文件的名称，如 /home/stack/templates/overcloud-baremetal-deployed.yaml。

9. 将 provisioning 命令生成的 overcloud-baremetal-deployed.yaml 文件添加到与其他环境文件的堆栈中，并部署 overcloud：

```
$ openstack overcloud deploy \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments \
  -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
  --deployed-server \
  --disable-validations \
  ...
```

10.4. 替换 RED HAT CEPH STORAGE 节点

您可以使用 director 来替换 director 创建的集群中的 Red Hat Ceph Storage 节点。如需更多信息，请参阅 部署 Red Hat Ceph Storage 和 Red Hat OpenStack Platform 指南。

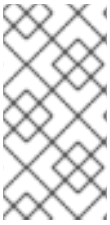
10.5. 使用跳过部署标识符

在堆栈更新操作 puppet 中，默认会获取所有清单。这可能导致耗时的操作，这可能不是必需的。

要覆盖默认操作，请使用 skip-deploy-identifier 选项。

```
openstack overcloud deploy --skip-deploy-identifier
```

如果您不希望部署命令为 DeployIdentifier 参数生成唯一标识符，则使用此选项。软件配置部署步骤仅当配置发生实际更改时才会触发。使用此选项要非常谨慎，仅当您确信不需要运行软件配置（如扩展某些角色）时方可使用。



注意

如果 puppet 清单或 hierdata 发生变化，puppet 也会重新应用所有清单，即使指定了 --skip-deploy-identifier。

10.6. 将节点列入黑名单

您可以阻止 overcloud 节点获得更新的部署内容。这在某些情况下非常有用，比如，您想要扩展新节点，并阻止现有节点获得核心 heat 模板集合中更新的参数和资源集合。这意味着列入黑名单的节点将完全不受栈操作的影响。

在环境文件中使用 DeploymentServerBlacklist 参数可创建黑名单。

设置黑名单

DeploymentServerBlacklist 参数是服务器名称列表。可以将其写入新的环境文件，或将参数值添加到现有的自定义环境文件，然后将此文件传递给部署命令：

```
parameter_defaults:
  DeploymentServerBlacklist:
    - overcloud-compute-0
    - overcloud-compute-1
    - overcloud-compute-2
```

注意

参数值中的服务器名称是由 OpenStack Orchestration (heat) 规定的名称，并非实际的服务器主机名。

将此环境文件包含到 openstack overcloud deploy 命令中：

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates \
  -e server-blacklist.yaml \
  [OTHER OPTIONS]
```

heat 会将列表中的任何服务器列入黑名单，阻止其获得更新的 heat 部署内容。在栈操作完成后，黑名单中的服务器不会发生任何变化。您也可以在操作过程中关闭或停止 os-collect-config 代理。

警告

- *将节点列入黑名单时要非常谨慎。在使用黑名单前，必须完全清楚在有黑名单的情况下如何应用所要求的更改。在使用黑名单功能时，有可能造成栈停止工作，或对 overcloud 执行不正确的配置。例如，如果集群配置更改应用到 Pacemaker 集群的所有成员，那么在执行更改时将 Pacemaker 集群的某个成员列入黑名单就会导致集群出现问题。*
- *不要在更新或升级过程中使用黑名单。这些过程本身有一些方法可将更改操作与特定服务器进行隔离。*
- *将服务器加入黑名单后，不允许再对这些节点进行更改操作，除非将服务器从黑名单中移除。这包括更新、升级、扩展、缩减和节点替换等操作。例如，如果在使用新的 Compute 节点扩展 overcloud 时将现有 Compute 节点列入黑名单，则列入黑名单的节点会错过添加到 /etc/hosts 和 /etc/ssh/ssh_known_hosts 中的信息。这可能导致实时迁移失败，具体取决于目标主机。在下一次 overcloud 部署过程中，利用添加到 /etc/hosts 和 /etc/ssh/ssh_known_hosts 中的信息更新 Compute 节点，这些节点不再列入黑名单。不要手动修改 /etc/hosts 和 /etc/ssh/ssh_known_hosts 文件。要修改 /etc/hosts 和 /etc/ssh/ssh_known_hosts 文件，请运行 overcloud 部署命令，如清除黑名单一节中所述。*

清除黑名单

要清除黑名单以便对其中节点执行后续的栈操作，可编辑 DeploymentServerBlacklist，使其成为空阵列：

```
parameter_defaults:
  DeploymentServerBlacklist: []
```

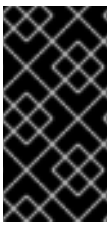
警告

不要省略 DeploymentServerBlacklist 参数。如果省略该参数，overcloud 部署将使用先前保存的参数值。

第 11 章 替换 CONTROLLER 节点

在一些情况下，高可用性集群中的 Controller 节点可能会出现故障。在这种情况下，您需要把这个节点从集群中删除，并替换为一个新的 Controller 节点。

完成本节中的步骤来替换 Controller 节点。在 Controller 节点替换过程中，需要运行 openstack overcloud deploy 命令，以使用替换 Controller 节点的请求来更新 overcloud。



重要

以下操作过程仅适用于高可用性环境。在只使用一个 Controller 节点的情况下不要使用此过程。

11.1. 准备替换 CONTROLLER 节点

在替换 overcloud 控制器节点前，务必要检查Red Hat OpenStack Platform 环境的当前状态；此检查有助于避免在替换控制器节点的过程中出现混乱。使用以下初步检查列表，确定是否可以安全地执行 Controller 节点替换。在 undercloud 上对这些检查运行所有命令。

步骤

1. 在 undercloud 中检查 overcloud 栈的当前状态：

```
$ source stackrc
$ openstack overcloud status
```

只有 overcloud 堆栈的部署状态为 DEPLOY_SUCCESS 时才继续。

2. 安装数据库客户端工具：

```
$ sudo dnf -y install mariadb
```

3. 配置数据库的 root 用户访问权限：

```
$ sudo cp /var/lib/config-data/puppet-generated/mysql/root/.my.cnf /root/.
```

4. 对 **undercloud 数据库进行备份：**

```
$ mkdir /home/stack/backup
$ sudo mysqldump --all-databases --quick --single-transaction | gzip >
/home/stack/backup/dump_db_undercloud.sql.gz
```

5. 检查**您的 undercloud 是否包含 10 GB 可用存储，可在置备新节点时容纳镜像缓存和转换：**

```
$ df -h
```

6. **如果您要为新 Controller 节点重复使用 IP 地址，请确保删除旧 Controller 使用的端口：**

```
$ openstack port delete <port>
```

7. **在运行的 Controller 节点上检查 Pacemaker 的状态。例如，运行的 Controller 节点的 IP 地址是 192.168.0.47，使用以下命令查看 Pacemaker 的状态：**

```
$ ssh tripleo-admin@192.168.0.47 'sudo pcs status'
```

输出显示了在现有节点上运行的所有服务，以及在故障节点上停止的服务。

8. 检查 **overcloud 的 MariaDB 集群中各个节点的以下参数：**

   - **wsrep_local_state_comment: Synced**
   - **wsrep_cluster_size: 2**

   使用以下命令检查各个运行的 Controller 节点的这些参数。在本例中，Controller 节点 IP 地址是 192.168.0.47 和 192.168.0.46：

```
$ for i in 192.168.0.46 192.168.0.47 ; do echo "*** $i ***" ; ssh tripleo-admin@$i "sudo
podman exec \$(sudo podman ps --filter name=galera-bundle -q) mysql -e \"SHOW
STATUS LIKE 'wsrep_local_state_comment'; SHOW STATUS LIKE
'wsrep_cluster_size';\""; done
```

9. 检查 **RabbitMQ 状态。例如，运行的 Controller 节点的 IP 地址是 192.168.0.47，使用以下**

命令查看 RabbitMQ 的状态：

```
$ ssh tripleo-admin@192.168.0.47 "sudo podman exec \$(sudo podman ps -f
name=rabbitmq-bundle -q) rabbitmqctl cluster_status"
```

running_nodes 键应该只显示两个可用的节点，而不会显示有故障的节点。

10. *如果启用了隔离服务，则将其禁用。例如，如果 192.168.0.47 是运行中 Controller 节点的 IP 地址，则使用以下命令检查隔离服务的状态：*

```
$ ssh tripleo-admin@192.168.0.47 "sudo pcs property show stonith-enabled"
```

运行以下命令可禁用隔离服务：

```
$ ssh tripleo-admin@192.168.0.47 "sudo pcs property set stonith-enabled=false"
```

11. *登录到失败的 Controller 节点，并停止运行的所有 nova zFCP 容器：*

```
$ sudo systemctl stop tripleo_nova_api.service
$ sudo systemctl stop tripleo_nova_api_cron.service
$ sudo systemctl stop tripleo_nova_conductor.service
$ sudo systemctl stop tripleo_nova_metadata.service
$ sudo systemctl stop tripleo_nova_scheduler.service
```

12. *可选：如果您使用 Bare Metal Service (ironic)作为 virt 驱动程序，则必须为其 instances.host 被设置为要删除的控制器的任何裸机实例手动更新单元数据库中的服务条目。联系红帽支持以获取帮助。*

注意

当使用 Bare Metal Service (ironic)作为 virt 驱动程序时，这个手动更新会临时解决这个问题，以确保节点重新平衡，直到 BZ2017980 完成为止。

11.2. 删除 CEPH MONITOR 守护进程

如果您的 Controller 节点正在运行 Ceph 监控服务，请完成以下步骤以删除 ceph-mon 守护进程。

注意

在集群中添加新的 Controller 节点，也会自动添加新的 Ceph 监控器守护进程。

流程

1. 连接到要替换的 Controller 节点：

```
$ ssh tripleo-admin@192.168.0.47
```

2. 列出 Ceph mon 服务：

```
$ sudo systemctl --type=service | grep ceph
ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@crash.controller-0.service          loaded active
running Ceph crash.controller-0 for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-0.mufglq.service     loaded
active running Ceph mgr.controller-0.mufglq for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mon.controller-0.service            loaded active
running Ceph mon.controller-0 for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@rgw.rgw.controller-0.ikaevh.service loaded
active running Ceph rgw.rgw.controller-0.ikaevh for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
```

3. 停止 Ceph mon 服务：

```
$ sudo systemctl stop ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mon.controller-
0.service
```

4. 禁用 Ceph mon 服务：

```
$ sudo systemctl disable ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mon.controller-
0.service
```

5. 从要替换的 Controller 节点断开连接。

6. 使用 SSH 连接到同一集群中的另一 Controller 节点：

```
$ ssh tripleo-admin@192.168.0.46
```

7. 在此过程中修改并应用 Ceph 规格文件，以操作您必须导出该文件：

```
$ sudo cephadm shell --ceph orch ls --export > spec.yaml
```

8. 从集群中删除该监控器：

```
$ sudo cephadm shell -- ceph mon remove controller-0
  removing mon.controller-0 at [v2:172.23.3.153:3300/0,v1:172.23.3.153:6789/0], there will be
2 monitors
```

9. 从 Controller 节点断开连接，再重新登录到您要从集群中删除的 Controller 节点：

```
$ ssh tripleo-admin@192.168.0.47
```

10. 列出 Ceph mgr 服务：

```
$ sudo systemctl --type=service | grep ceph
ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@crash.controller-0.service          loaded active
running Ceph crash.controller-0 for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-0.mufglq.service     loaded
active running Ceph mgr.controller-0.mufglq for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@rgw.rgw.controller-0.ikaevh.service loaded
active running Ceph rgw.rgw.controller-0.ikaevh for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
```

11. 停止 Ceph mgr 服务：

```
$ sudo systemctl stop ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-
0.mufglq.service
```

12. 禁用 Ceph mgr 服务：

```
$ sudo systemctl disable ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-
0.mufglq.service
```

13. 启动 cephadm shell：

```
$ sudo cephadm shell
```

14. *验证 **Controller** 节点的 **Ceph mgr** 服务是否已从集群中移除：*

```
$ ceph -s
cluster:
    id:     b9b53581-d590-41ac-8463-2f50aa985001
    health: HEALTH_OK

  services:
    mon: 2 daemons, quorum controller-2,controller-1 (age 2h)
    mgr: controller-2(active, since 20h), standbys: controller1-1
    osd: 15 osds: 15 up (since 3h), 15 in (since 3h)

  data:
    pools:   3 pools, 384 pgs
    objects: 32 objects, 88 MiB
    usage:   16 GiB used, 734 GiB / 750 GiB avail
    pgs:     384 active+clean
```

*如果 **Ceph mgr** 服务被成功移除，则节点不会被列出。*

15. *导出 **Red Hat Ceph Storage** 规格：*

```
$ ceph orch ls --export > spec.yaml
```

16. *在 **spec.yaml** 规范文件中，从 **service_type: mon** 和 **service_type: mgr** 中删除主机的所有实例，如 **controller-0**。*

17. *重新应用 **Red Hat Ceph Storage** 规格：*

```
$ ceph orch apply -i spec.yaml
```

18. *验证删除的主机上没有保留 **Ceph** 守护进程：*

```
$ ceph orch ps controller-0
```

注意

如果存在守护进程，使用以下命令删除它们：

```
$ ceph orch host drain controller-0
```

在运行 **ceph orch host drain** 命令前，备份 **/etc/ceph** 的内容。在运行 **ceph orch host drain** 命令后恢复内容。您必须在运行 **ceph orch host drain** 命令前备份，直到 https://bugzilla.redhat.com/show_bug.cgi?id=2153827 解析为止。

19. 从 **Red Hat Ceph Storage** 集群中删除 **controller-0** 主机：

```
$ ceph orch host rm controller-0
  Removed host 'controller-0'
```

20. 退出 **cephadm shell**：

```
$ exit
```

其它资源

- 有关使用 **systemd** 控制 **Red Hat Ceph Storage** 服务的更多信息，请参阅了解 **Ceph** 的进程管理。
- 如需有关编辑和应用 **Red Hat Ceph Storage** 规格文件的更多信息，请参阅使用服务规格部署 **Ceph** 监控守护进程。

11.3. 为 CONTROLLER 节点替换准备集群

在替换节点前，请确保 **Pacemaker** 没有在该节点上运行，然后从 **Pacemaker** 集群中删除该节点。

流程

1. 要查看 **Controller** 节点的 **IP** 地址列表，请运行以下命令：

```
(undercloud)$ metalsmith -c Hostname -c "IP Addresses" list
+------------------------+-----------------------+
```

```
| Hostname              | IP Addresses          |
+-----------------------+-----------------------+
| overcloud-compute-0   | ctlplane=192.168.0.44 |
| overcloud-controller-0 | ctlplane=192.168.0.47 |
| overcloud-controller-1 | ctlplane=192.168.0.45 |
| overcloud-controller-2 | ctlplane=192.168.0.46 |
+-----------------------+-----------------------+
```

2. 登录节点并确认 **pacemaker** 状态。如果 **pacemaker** 正在运行，请使用 **pcs cluster** 命令停止 **pacemaker**。本例停止 **overcloud-controller-0** 上的 **pacemaker** ：

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs status | grep -w Online | grep -w overcloud-controller-0"
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs cluster stop overcloud-controller-0"
```

注意

如果节点物理不可用或停止，则不需要执行前面的操作，因为 **pacemaker** 已在该节点上停止。

3. 在节点上停止 **Pacemaker** 后，从 **pacemaker** 集群中删除该节点。以下示例登录到 **overcloud-controller-1** 以删除 **overcloud-controller-0** ：

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs cluster node remove overcloud-controller-0"
```

如果要替换的节点无法访问（例如，由于硬件故障），请运行 **pcs** 命令并使用附加 **--skip-offline** 和 **--force** 选项从集群中强制删除该节点：

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs cluster node remove overcloud-controller-0 --skip-offline --force"
```

4. 从 **pacemaker** 集群中删除节点后，从 **pacemaker** 中的已知主机列表中删除该节点：

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs host deauth overcloud-controller-0"
```

无论节点是否可访问，都可运行该命令。

5.

要确保新 Controller 节点在替换后使用正确的 STONITH 隔离设备，请输入以下命令从节点中删除设备：

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs stonith delete
<stonith_resource_name>"
```

- 将 <stonith_resource_name > 替换为与节点对应的 STONITH 资源的名称。资源名称使用 < resource_agent>-<host_mac> 格式。您可以在 fence.yaml 文件的 FencingConfig 部分查找资源代理和主机 MAC 地址。

6. overcloud 数据库必须在替换过程中继续运行。为了确保 Pacemaker 不会在此过程中停止 Galera，可选择一个运行中的 Controller 节点，然后使用该 Controller 节点的 IP 地址在 undercloud 上运行以下命令：

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs resource unmanage galera-
bundle"
```

7. 从集群中删除替换 Controller 节点的 OVN 北向数据库服务器：

   a. 获取要替换的 OVN 北向数据库服务器的服务器 ID：

```
$ ssh tripleo-admin@<controller_ip> sudo podman exec ovn_cluster_north_db_server
ovs-appctl -t /var/run/ovn/ovnnb_db.ctl cluster/status OVN_Northbound 2>/dev/null|grep -
A4 Servers:
```

将 <controller_ip> 替换为任何活跃 Controller 节点的 IP 地址。

您应该看到类似如下的输出：

```
Servers:
96da (96da at tcp:172.17.1.55:6643) (self) next_index=26063 match_index=26063 466b
(466b at tcp:172.17.1.51:6643) next_index=26064 match_index=26063 last msg 2936
ms ago
ba77 (ba77 at tcp:172.17.1.52:6643) next_index=26064 match_index=26063 last msg
2936 ms ago
```

在本例中，172.17.1.55 是被替换的 Controller 节点的内部 IP 地址，因此北向数据库服务器 ID 为 96da。

b. 使用您在上一步中获得的服务器 ID，运行以下命令来删除 OVN 北向数据库服务器：

```
$ ssh tripleo-admin@172.17.1.52 sudo podman exec ovn_cluster_north_db_server ovs-appctl -t /var/run/ovn/ovnnb_db.ctl cluster/kick OVN_Northbound 96da
```

在本例中，您要将 172.17.1.52 替换为任何活跃的 Controller 节点的 IP 地址，并将 96da 替换为 OVN 北向数据库服务器的服务器 ID。

8. 从集群中移除替换 Controller 节点的 OVN 南向数据库服务器：

a. 获取要替换的 OVN 南向数据库服务器的服务器 ID：

```
$ ssh tripleo-admin@<controller_ip> sudo podman exec ovn_cluster_south_db_server ovs-appctl -t /var/run/ovn/ovnsb_db.ctl cluster/status OVN_Southbound 2>/dev/null|grep -A4 Servers:
```

将 <controller_ip> 替换为任何活跃 Controller 节点的 IP 地址。

您应该看到类似如下的输出：

```
Servers:
e544 (e544 at tcp:172.17.1.55:6644) last msg 42802690 ms ago
17ca (17ca at tcp:172.17.1.51:6644) last msg 5281 ms ago
6e52 (6e52 at tcp:172.17.1.52:6644) (self)
```

在本例中，172.17.1.55 是要替换的 Controller 节点的内部 IP 地址，因此南向数据库服务器 ID 为 e544。

b. 使用您在上一步中获得的服务器 ID，运行以下命令来删除 OVN 南向数据库服务器：

```
$ ssh tripleo-admin@172.17.1.52 sudo podman exec ovn_cluster_south_db_server ovs-appctl -t /var/run/ovn/ovnsb_db.ctl cluster/kick OVN_Southbound e544
```

在本例中，您要将 172.17.1.52 替换为任何活跃的 Controller 节点的 IP 地址，并将 e544 替换为 OVN 南向数据库服务器的服务器 ID。

9. 运行以下清理命令以防止重新加入集群。

将 **<replaced_controller_ip>** 替换为您要替换的 **Controller** 节点的 **IP** 地址：

```
$ ssh tripleo-admin@<replaced_controller_ip> sudo systemctl disable --now
tripleo_ovn_cluster_south_db_server.service tripleo_ovn_cluster_north_db_server.service

$ ssh tripleo-admin@<replaced_controller_ip> sudo rm -rfv /var/lib/openvswitch/ovn/.ovn*
/var/lib/openvswitch/ovn/ovn*.db
```

11.4. 替换 BOOTSTRAP CONTROLLER 节点

如果要替换用于 **bootstrap** 操作的 **Controller** 节点并保留节点名称，请完成以下步骤，以在替换过程后设置 **bootstrap Controller** 节点的名称。

步骤

1. 运行以下命令，查找 **bootstrap Controller** 节点的名称：

```
ssh tripleo-admin@<controller_ip> "sudo hiera -c /etc/puppet/hiera.yaml
pacemaker_short_bootstrap_node_name"
```

- 将 **<controller_ip>** 替换为任何活跃 **Controller** 节点的 **IP** 地址。

2. 检查您的环境文件是否包含 **ExtraConfig** 和 **AllNodesExtraMapData** 参数。如果没有设置参数，请创建以下环境文件 **~/templates/bootstrap-controller.yaml** 并添加以下内容：

```
parameter_defaults:
  ExtraConfig:
    pacemaker_short_bootstrap_node_name: NODE_NAME
    mysql_short_bootstrap_node_name: NODE_NAME
  AllNodesExtraMapData:
    ovn_dbs_bootstrap_node_ip: NODE_IP
    ovn_dbs_short_bootstrap_node_name: NODE_NAME
```

- 将 **NODE_NAME** 替换为在替换过程后您要在 **bootstrap** 操作中使用的现有 **Controller** 节点的名称。

- 将 **NODE_IP** 替换为映射到 **NODE_NAME** 中命名的控制器的 **IP** 地址。要获取名称，请运行以下命令：

```
sudo hiera -c /etc/puppet/hiera.yaml ovn_dbs_node_ips
```

如果您的环境文件已经包含 **ExtraConfig** 和 **AllNodesExtraMapData** 参数，请只添加此步骤中显示的行。

有关对 **bootstrap Controller** 节点替换进行故障排除的信息，请参阅文章如果相同的主机名用于新节点，则第 1 步中替换第一个 **Controller** 节点会失败。

11.5. 取消置备和删除 CONTROLLER 节点

您可以取消置备和删除 **Controller** 节点。

流程

1. **Source stackrc** 文件：

```
$ source ~/stackrc
```

2. 识别 **overcloud-controller-0** 节点的 **UUID**：

```
(undercloud)$ NODE=$(metalsmith -c UUID -f value show overcloud-controller-0)
```

3. 把节点设为维护模式：

```
$ openstack baremetal node maintenance set $NODE
```

4. 复制 **overcloud-baremetal-deploy.yaml** 文件：

```
$ cp /home/stack/templates/overcloud-baremetal-deploy.yaml
/home/stack/templates/unprovision_controller-0.yaml
```

5. 在 **unprovision_controller-0.yaml** 文件中，降低 **Controller** 数量来取消置备您要替换的 **Controller** 节点。在本例中，计数从 **3** 减小到 **2**。将 **controller-0** 节点移到 **instances** 字典中，并将 **provisioned** 参数设置为 **false** ：

```
- name: Controller
  count: 2
  hostname_format: controller-%index%
  defaults:
    resource_class: BAREMETAL.controller
    networks:
      [ ... ]
  instances:
  - hostname: controller-0
    name: <IRONIC_NODE_UUID_or_NAME>
    provisioned: false
- name: Compute
  count: 2
  hostname_format: compute-%index%
  defaults:
    resource_class: BAREMETAL.compute
    networks:
      [ ... ]
```

6. *运行 **node unprovision** 命令：*

```
$ openstack overcloud node delete \
  --stack overcloud \
  --baremetal-deployment /home/stack/templates/unprovision_controller-0.yaml
```

```
The following nodes will be unprovisioned:
+--------------+------------------------+--------------------------------------+
| hostname     | name                   | id                                   |
+--------------+------------------------+--------------------------------------+
| controller-0 | baremetal-35400-leaf1-2 | b0d5abf7-df28-4ae7-b5da-9491e84c21ac |
+--------------+------------------------+--------------------------------------+

Are you sure you want to unprovision these overcloud nodes and ports [y/N]?
```

7. ***可选：删除 ironic** 节点：*

```
$ openstack baremetal node delete <IRONIC_NODE_UUID>
```

- ***将 IRONIC_NODE_UUID 替换为节点的 UUID。***

11.6. 将新的控制器节点部署到 OVERCLOUD

要将新控制器节点部署到 overcloud，请完成以下步骤。

先决条件

- 必须注册、检查并标记新的 Controller 节点以进行置备。如需更多信息，请参阅置备裸机 overcloud 节点

流程

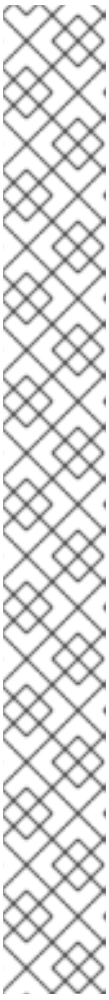
1. 以 stack 用户身份登录 undercloud 主机。

2. 查找 stackrc undercloud 凭证文件：

   $ source ~/stackrc

3. 使用原始 overcloud-baremetal-deploy.yaml 环境文件置备 overcloud：

```
$ openstack overcloud node provision
  --stack overcloud
  --network-config
  --output /home/stack/templates/overcloud-baremetal-deployed.yaml
  /home/stack/templates/overcloud-baremetal-deploy.yaml
```



注意

如果要使用相同的调度、放置或 IP 地址，您可以编辑 overcloud-baremetal-deploy.yaml 环境文件。在 instances 部分中，为新的 controller-0 实例设置主机名、名称和网络。例如：

```
- name: Controller
  count: 3
  hostname_format: controller-%index%
  defaults:
    resource_class: BAREMETAL.controller
    networks:
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
      template: templates/multiple_nics/multiple_nics_dvr.j2
      default_route_network:
```

```
    - external
instances:
- hostname: controller-0
  name: baremetal-35400-leaf1-2
  networks:
  - network: external
    subnet: external_subnet
    fixed_ip: 10.0.0.224
  - network: internal_api
    subnet: internal_api_subnet01
    fixed_ip: 172.17.0.97
  - network: storage
    subnet: storage_subnet01
    fixed_ip: 172.18.0.24
  - network: storage_mgmt
    subnet: storage_mgmt_subnet01
    fixed_ip: 172.19.0.129
  - network: tenant
    subnet: tenant_subnet01
    fixed_ip: 172.16.0.11
- name: Compute
  count: 2
  hostname_format: compute-%index%
  defaults:
    [ ... ]
```

置备节点时，从 overcloud-baremetal-deploy.yaml 文件中删除 instances 部分。

4. ***要在新 Controller 节点上创建 cephadm 用户，请导出包含新主机信息的基本 Ceph 规格：***

```
$ openstack overcloud ceph spec --stack overcloud \
  /home/stack/templates/overcloud-baremetal-deployed.yaml \
  -o ceph_spec_host.yaml
```

注意

如果您的环境使用自定义角色，请包含 --roles-data 选项。

5. ***将 cephadm 用户添加到新的 Controller 节点：***

```
$ openstack overcloud ceph user enable \
  --stack overcloud ceph_spec_host.yaml
```

6. 登录 **Controller** 节点，并将新角色添加到 **Ceph** 集群：

```
$ sudo cephadm shell \
  -- ceph orch host add controller-3 <IP_ADDRESS> <LABELS>
192.168.24.31 _admin mon mgr
Inferring fsid 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
Using recent ceph image undercloud-0.ctlplane.redhat.local:8787/rh-
osbs/rhceph@sha256:3075e8708792ebd527ca14849b6af4a11256a3f881ab09b837d7af0f8b2
102ea
Added host 'controller-3' with addr '192.168.24.31'
```

- 将 **<IP_ADDRESS>** 替换为 **Controller** 节点的 **IP** 地址。
- 将 **<LABELS>** 替换为任何所需的 **Ceph** 标签。

7. 重新运行 **openstack overcloud deploy** 命令：

```
$ openstack overcloud deploy --stack overcloud --templates \
   -n /home/stack/templates/network_data.yaml \
   -r /home/stack/templates/roles_data.yaml \
   -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
   -e /home/stack/templates/overcloud-networks-deployed.yaml \
   -e /home/stack/templates/overcloud-vips-deployed.yaml \
   -e /home/stack/templates/bootstrap_node.yaml \
   -e [ ... ]
```

注意

如果替换的 **Controller** 节点是 **bootstrap** 节点，请包含 **bootstrap_node.yaml** 环境文件。

11.7. 在新控制器节点上部署 CEPH 服务

在置备一个新的 **Controller** 节点并且 **Ceph** 监控服务运行后，您可以在 **Controller** 节点上部署 **mgr**、**rgw** 和 **osd Ceph** 服务。

先决条件

- 新的 **Controller** 节点已调配，并且正在运行 **Ceph** 监控服务。

流程

1. **修改 spec.yml 环境文件，将前面的 Controller 节点名称替换为新的 Controller 节点名称：**

```
$ cephadm shell -- ceph orch ls --export > spec.yml
```

注意

不要使用基本的 Ceph 环境文件 ceph_spec_host.yaml，因为它不包含所有必要的集群信息。

2. **应用修改后的 Ceph 规格文件：**

```
$ cat spec.yml | sudo cephadm shell -- ceph orch apply -i -
Inferring fsid 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
Using recent ceph image undercloud-0.ctlplane.redhat.local:8787/rh-
osbs/rhceph@sha256:3075e8708792ebd527ca14849b6af4a11256a3f881ab09b837d7af0f8b2
102ea
Scheduled crash update...
Scheduled mgr update...
Scheduled mon update...
Scheduled osd.default_drive_group update...
Scheduled rgw.rgw update...
```

3. **验证新监控器的可见性：**

```
$ sudo cephadm --ceph status
```

.

11.8. CONTROLLER 节点替换后清理

完成节点替换后，您可以完成 Controller 集群。

流程

1. **登录 Controller 节点。**

2. 启用 Galera 集群的 Pacemaker 管理，并在新节点上启动 Galera：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs resource refresh galera-bundle
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs resource manage galera-bundle
```

3. 启用隔离：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs property set stonith-enabled=true
```

4. 执行最后的状态检查来确保服务在正确运行：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs status
```

注意

如果有任何服务失败，请使用 **pcs resource refresh** 命令来解决问题并重新启动失败的服务。

5. 退出 director：

```
[tripleo-admin@overcloud-controller-0 ~]$ exit
```

6. 查找 **overcloudrc** 文件，以便您可以跟 **overcloud** 交互：

```
$ source ~/overcloudrc
```

7. 检查 **overcloud** 环境中的网络代理：

```
(overcloud) $ openstack network agent list
```

8. 如果出现任何旧节点的代理，请删除它们：

```
(overcloud) $ for AGENT in $(openstack network agent list --host overcloud-controller-
1.localdomain -c ID -f value) ; do openstack network agent delete $AGENT ; done
```

9.

如有必要，将您的路由器添加到新节点上的 3 层代理主机。使用以下示例命令，通过 UUID 2d1c1dc1-d9d4-4fa9-b2c8-f29cd1a649d4 将名为 r1 的路由器添加到 L3 代理中：

```
(overcloud) $ openstack network agent add router --l3 2d1c1dc1-d9d4-4fa9-b2c8-
f29cd1a649d4 r1
```

10. 清理 cinder 服务。

    a. 列出 cinder 服务：

    ```
    (overcloud) $ openstack volume service list
    ```

    b. 登录到控制器节点，连接到 cinder-api 容器，并使用 cinder-manage service remove 命令删除左侧服务：

    ```
    [tripleo-admin@overcloud-controller-0 ~]$ sudo podman exec -it cinder_api cinder-
    manage service remove cinder-backup <host>
    [tripleo-admin@overcloud-controller-0 ~]$ sudo podman exec -it cinder_api cinder-
    manage service remove cinder-scheduler <host>
    ```

11. 清理 RabbitMQ 集群。

    a. 登录 Controller 节点。

    b. 使用 podman exec 命令启动 bash，并验证 RabbitMQ 集群的状态：

    ```
    [tripleo-admin@overcloud-controller-0 ~]$ sudo podman exec -it rabbitmq-bundle-
    podman-0 bash
    [root@overcloud-controller-0 /]$ rabbitmqctl cluster_status
    ```

    c. 使用 rabbitmqctl 命令忘记替换的控制器节点：

    ```
    [root@controller-0 /]$ rabbitmqctl forget_cluster_node <node_name>
    ```

12. 如果替换了 bootstrap Controller 节点，则必须在替换过程后移除环境文件 ~/templates/bootstrap-controller.yaml，或者从现有环境文件中移除 pacemaker_short_bootstrap_node_name 和 mysql_short_bootstrap_node_name 参数。此

步骤可防止 director 在后续替换中尝试覆盖 Controller 节点名称。如需更多信息，请参阅 替换 bootstrap Controller 节点。

13. 如果您在 overcloud 上使用 Object Storage 服务(swift)，则必须在更新 overcloud 节点后同步 swift 环。使用类似以下示例的脚本，将 ring 文件从之前存在的 Controller 节点（本例中为 Controller 节点 0）分发到所有 Controller 节点，再重启这些节点上的对象存储服务容器：

```
#!/bin/sh
set -xe

SRC="tripleo-admin@overcloud-controller-0.ctlplane"
ALL="tripleo-admin@overcloud-controller-0.ctlplane tripleo-admin@overcloud-controller-1.ctlplane tripleo-admin@overcloud-controller-2.ctlplane"
```

- 获取当前的环文件集合：

```
ssh "${SRC}" 'sudo tar -czvf - /var/lib/config-data/puppet-generated/swift_ringbuilder/etc/swift/{*.builder,*.ring.gz,backups/*.builder}' > swift-rings.tar.gz
```

- 将 ring 上传到所有节点，将它们放在正确的位置，然后重新启动 swift 服务：

```
for DST in ${ALL}; do
  cat swift-rings.tar.gz | ssh "${DST}" 'sudo tar -C / -xvzf -'
  ssh "${DST}" 'sudo podman restart swift_copy_rings'
  ssh "${DST}" 'sudo systemctl restart tripleo_swift*'
done
```

第 12 章 重新引导节点

您可能需要在 undercloud 和 overcloud 中重新引导节点。

注意

如果您在 overcloud 中启用了实例 HA （高可用性），并且需要关闭或重新引导 Compute 节点，请参阅 Chapter 3。在 undercloud 和带有实例 HA 的 overcloud 上执行维护，为实例 配置高可用性。

使用以下步骤了解如何重新引导不同的节点类型。

- 如果重新引导一个角色中的所有节点，建议单独重新引导各节点。如果您同时重新引导角色中的所有节点，则重启操作过程中可能会发生服务停机时间。
- 如果重新引导 OpenStack Platform 环境中的所有节点，则按照以下顺序重新引导节点：

建议的节点重新引导顺序

1. 重新引导 undercloud 节点。
2. 重新引导 Controller 节点和其他可组合节点。
3. 重新引导独立 Ceph MON 节点。
4. 重新引导 Ceph Storage 节点。
5. 重新引导 Object Storage 服务(swift)节点。
6. 重新引导 Compute 节点。

12.1. 重新引导 UNDERCLOUD 节点

完成以下步骤以重新引导 undercloud 节点。

步骤

1. 以 stack 用户的身份登录 undercloud。
2. 重新引导 undercloud：

```
$ sudo reboot
```

3. 稍等片刻，直到节点启动。

12.2. 重新引导 CONTROLLER 和可组合节点

根据可组合角色重新引导 Controller 节点和独立节点，并排除 Compute 节点和 Ceph Storage 节点。

流程

1. 登录您要重新引导的节点。
2. 可选：如果节点使用 Pacemaker 资源，请停止集群：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs cluster stop
```

3. 重新引导节点：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo reboot
```

4. 稍等片刻，直到节点启动。

验证

1. 验证服务是否已启用。

a. 如果该节点使用 Pacemaker 服务，请检查该节点是否已重新加入集群：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs status
```

b. 如果该节点使用 Systemd 服务，请检查是否所有服务都已启用：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo systemctl status
```

c. 如果该节点使用容器化服务，则检查节点上的所有容器是否已激活：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo podman ps
```

12.3. 重新引导独立 CEPH MON 节点

完成以下步骤以重新引导独立 Ceph MON 节点。

步骤

1. 登录到一个 Ceph MON 节点。
2. 重新引导节点：

```
$ sudo reboot
```

3. 等待节点被引导并重新加入 MON 集群。

对集群中的每个 MON 节点重复这些步骤。

12.4. 重新引导 CEPH STORAGE (OSD) 集群

完成以下步骤以重新引导 Ceph Storage (OSD) 节点集群。

先决条件

-

在运行 ceph-mon 服务的 Ceph monitor 或 Controller 节点上，检查 Red Hat Ceph Storage 集群状态是否健康，pg 状态为 active+clean ：

```
$ sudo cephadm -- shell ceph status
```

如果 Ceph 集群处于健康状态，它会返回 HEALTH_OK 状态。

如果 Ceph 集群状态不健康，它将返回 HEALTH_WARN 或 HEALTH_ERR 的状态。有关故障排除指南，请参阅 Red Hat Ceph Storage 5 故障排除指南 或 Red Hat Ceph Storage 6 故障排除指南。

流程

1. 登录到运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点，并临时禁用 Ceph Storage 集群重新平衡：

```
$ sudo cephadm shell -- ceph osd set noout
$ sudo cephadm shell -- ceph osd set norebalance
```

注意

如果您有多堆栈或分布式计算节点(DCN)架构，您必须在设置 noout 和 norebalance 标志时指定 Ceph 集群名称。例如： sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring。

2. 选择第一个要重新引导的 Ceph Storage 节点并登录到该节点。

3. 重新引导节点：

```
$ sudo reboot
```

4. 稍等片刻，直到节点启动。

5. 登录节点并检查 Ceph 集群状态：

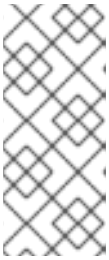
```
$ sudo cephadm -- shell ceph status
```

确认 pgmap 报告的所有 pgs 的状态是否都正常 (active+clean)。

6. 注销节点，重新引导下一个节点，并检查其状态。重复此过程，直到您已重新引导所有 Ceph Storage 节点。

7. 完成后，登录到运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点，并启用 Ceph 集群重新平衡：

```
$ sudo cephadm shell -- ceph osd unset noout
$ sudo cephadm shell -- ceph osd unset norebalance
```



注意

如果您有多堆栈或分布式计算节点(DCN)架构，您必须在取消设置 noout 和 norebalance 标志时指定 Ceph 集群名称。例如： sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring

8. 执行最后的状态检查，确认集群报告 HEALTH_OK：

```
$ sudo cephadm shell ceph status
```

12.5. 重新引导 OBJECT STORAGE 服务 (SWIFT) 节点

以下流程重启 Object Storage 服务(swift)节点。对集群中的每个 Object Storage 节点完成以下步骤。

流程

1. 登录 Object Storage 节点。

2. 重新引导节点：

```
$ sudo reboot
```

3. 稍等片刻，直到节点启动。

4. 对集群中的每个 Object Storage 节点重复重启。

12.6. 重新引导 COMPUTE 节点

为确保 Red Hat OpenStack Platform 环境中 实例的停机时间最少，迁移实例工作流 概述了从您要重新引导的 Compute 节点迁移实例的步骤。

迁移实例工作流

1. 决定是否在重新引导节点前将实例迁移到另一个 Compute 节点。
2. 选择并禁用您要重新引导的 Compute 节点，使其不置备新实例。
3. 将实例迁移到另一个 Compute 节点中。
4. 重新引导空的 Compute 节点。
5. 启用空的 Compute 节点。

先决条件

- 重启 Compute 节点之前，必须决定是否在节点重启过程中将实例迁移到另一个 Compute 节点。

  查看在 Compute 节点之间迁移虚拟机实例时可能会遇到的迁移限制列表。如需更多信息，请参阅为实例创建配置 Compute Service 中的迁移限制。

  注意

  如果您有 Multi-RHEL 环境，并且希望将虚拟机从运行 RHEL 9.2 的 Compute 节点迁移到运行 RHEL 8.4 的 Compute 节点，则只支持冷迁移。有关冷迁移的更多信息，请参阅配置 计算服务 以进行实例创建中的冷迁移实例。

- 如果您无法迁移实例，则可设置以下核心模板参数以在 Compute 节点重启后控制实例的状

态：

NovaResumeGuestsStateOnHostBoot

*确定重新引导后是否将实例返回 Compute 节点上的相同状态。设为 **False** 时，实例保持关闭，必须手动启动。默认值为 **False**。*

NovaResumeGuestsShutdownTimeout

*重启前等待实例被关闭的时间（以秒为单位）。建议不要将此值设置为 **0**。默认值为 **300**。*

*有关 **overcloud** 参数及其用法的更多信息，请参阅 **Overcloud 参数**。*

流程

1. *以 **stack** 用户的身份登录 **undercloud**。*

2. *检索 **Compute** 节点列表，以识别您要重新引导的节点的主机名：*

```
(undercloud)$ source ~/overcloudrc
(overcloud)$ openstack compute service list
```

*识别您要重新引导的 **Compute** 节点的主机名。*

3. *在您要重新引导的 **Compute** 节点上禁用 **Compute** 服务：*

```
(overcloud)$ openstack compute service list
(overcloud)$ openstack compute service set <hostname> nova-compute --disable
```

   - *将 **<hostname >** 替换为 **Compute** 节点的主机名。*

4. *列出 **Compute** 节点上的所有实例：*

```
(overcloud)$ openstack server list --host <hostname> --all-projects
```

5. 可选：要将实例迁移到另一个 Compute 节点，请完成以下步骤：

a. 如果您决定将实例迁移至另一个 Compute 节点，则使用以下命令之一：

- 要将实例迁移到其他主机，请运行以下命令：

```
(overcloud) $ openstack server migrate <instance_id> --live <target_host> --wait
```

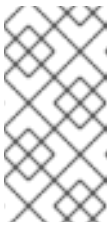
  - 将 <instance_id& gt; 替换为您的实例 ID。
  - 将 <target_host > 替换为您要将实例迁移到的主机。

- 让 nova-scheduler 自动选择目标主机：

```
(overcloud) $ nova live-migration <instance_id>
```

- 一次性实时迁移所有实例：

```
$ nova host-evacuate-live <hostname>
```



注意

nova 命令可能会引发一些弃用警告，这些警告信息可以被安全忽略。

b. 稍等片刻，直至迁移完成。

c. 确认迁移成功完成：

```
(overcloud) $ openstack server list --host <hostname> --all-projects
```

d. 继续迁移实例，直到 Compute 节点上没有保留任何实例。

6. *登录到 Compute 节点并重启节点：*

```
[tripleo-admin@overcloud-compute-0 ~]$ sudo reboot
```

7. *稍等片刻，直到节点启动。*

8. *重新启用 Compute 节点：*

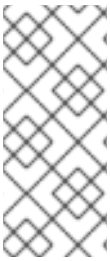
```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set <hostname>  nova-compute --enable
```

9. *确认是否已启用 Compute 节点：*

```
(overcloud) $ openstack compute service list
```

第 13 章 关闭并启动 UNDERCLOUD 和 OVERCLOUD

如果必须对 undercloud 和 overcloud 执行维护，您必须按照特定顺序关闭和启动 undercloud 和 overcloud 节点，以确保启动 overcloud 时出现的问题很少。



注意

如果您在 overcloud 中启用了实例 HA （高可用性），并且需要关闭或重新引导 Compute 节点，请参阅 Chapter 3。在 为实例 配置高可用性 中的在 undercloud 和带有实例 HA 的 overcloud 上执行维护。

先决条件

- 运行 undercloud 和 overcloud

13.1. UNDERCLOUD 和 OVERCLOUD 关闭顺序

要关闭 Red Hat OpenStack Platform 环境，您必须按照以下顺序关闭 overcloud 和 undercloud：

1. 关闭 overcloud Compute 节点上的实例
2. 关闭 Compute 节点
3. 停止 Controller 节点上的所有高可用性和 OpenStack Platform 服务
4. 关闭 Ceph Storage 节点
5. 关闭 Controller 节点
6. 关闭 undercloud

13.2. 关闭 OVERCLOUD COMPUTE 节点上的实例

作为关闭 Red Hat OpenStack Platform 环境的一部分，在关闭 Compute 节点之前关闭 Compute 节点上的所有实例。

先决条件

- *具有活跃 Compute 服务的 overcloud*

步骤

1. *以 stack 用户身份登录 undercloud。*
2. *提供 overcloud 的凭据文件：*

```
$ source ~/overcloudrc
```

3. *查看 overcloud 中运行的实例：*

```
$ openstack server list --all-projects
```

4. *停止 overcloud 中的每个实例：*

```
$ openstack server stop <INSTANCE>
```

对每个实例重复这一步，直到停止 overcloud 中的所有实例。

13.3. 关闭 COMPUTE 节点

作为关闭 Red Hat OpenStack Platform 环境的一部分，登录并关闭每个 Compute 节点。

先决条件

- *关闭 Compute 节点上的所有实例：*

步骤

1. 以 root 用户身份登录 Compute 节点。

2. 关闭该节点：

```
# shutdown -h now
```

3. 对每个 Compute 节点执行这些步骤，直到关闭所有 Compute 节点。

13.4. 停止 CONTROLLER 节点上的服务

作为关闭 Red Hat OpenStack Platform 环境的一部分，在关闭 Controller 节点前停止节点上的服务。这包括 Pacemaker 和 systemd 服务。

先决条件

- 具有活跃 Pacemaker 服务的 overcloud

步骤

1. 以 root 用户身份登录 Controller 节点。

2. 停止 Pacemaker 集群。

```
# pcs cluster stop --all
```

此命令停止所有节点上的集群。

3. 等待 Pacemaker 服务停止并检查服务是否已停止。

   a. 检查 Pacemaker 状态：

```
# pcs status
```

b. 检查 Podman 中没有 Pacemaker 服务在运行：

```
# podman ps --filter "name=.*-bundle.*"
```

4. 停止 Red Hat OpenStack Platform 服务：

```
# systemctl stop 'tripleo_*'
```

5. 等待服务停止，检查 Podman 中服务不再运行：

```
# podman ps
```

13.5. 关闭 CEPH STORAGE 节点

作为关闭 Red Hat OpenStack Platform 环境的一部分，禁用 Ceph Storage 服务，然后登录并关闭每个 Ceph Storage 节点。

先决条件

- 正常运行的 Ceph Storage 集群
- Ceph MON 服务在单机 Ceph MON 节点或 Controller 节点上运行

步骤

1. 以 root 用户身份登录运行 Ceph MON 服务的节点，如 Controller 节点或单机 Ceph MON 节点。

2. 检查集群的运行状况。在以下示例中，podman 命令在 Controller 节点上的 Ceph MON 容器中运行状态检查：

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

确保状态为 HEALTH_OK。

3.

为集群设置 noout、norecover、norebalance、nobackfill、nodown 和 pause 标志。在以下示例中，podman 命令通过 Controller 节点上的 Ceph MON 容器设置这些标志：

```
# sudo podman exec -it ceph-mon-controller-0 ceph osd set noout
# sudo podman exec -it ceph-mon-controller-0 ceph osd set norecover
# sudo podman exec -it ceph-mon-controller-0 ceph osd set norebalance
# sudo podman exec -it ceph-mon-controller-0 ceph osd set nobackfill
# sudo podman exec -it ceph-mon-controller-0 ceph osd set nodown
# sudo podman exec -it ceph-mon-controller-0 ceph osd set pause
```

4. 关闭每个 Ceph Storage 节点：

   a. 以 root 用户身份登录 Ceph Storage 节点。

   b. 关闭该节点：

   ```
   # shutdown -h now
   ```

   c. 对每个 Ceph Storage 节点执行这些步骤，直到关闭所有 Ceph Storage 节点。

5. 关闭任何单机 Ceph MON 节点：

   a. 以 root 用户身份登录单机 Ceph MON 节点。

   b. 关闭该节点：

   ```
   # shutdown -h now
   ```

   c. 对每个单机 Ceph MON 节点执行这些步骤，直到关闭所有单机 Ceph MON 节点。

其他资源

- “关闭并启动整个 Ceph 集群的步骤是什么？”

13.6. 关闭 CONTROLLER 节点

作为关闭 Red Hat OpenStack Platform 环境的一部分，登录并关闭每个 Controller 节点。

先决条件

- *停止 Pacemaker 集群*
- *停止 Controller 节点上的所有 Red Hat OpenStack Platform 服务*

步骤

1. *以 root 用户身份登录 Controller 节点。*
2. *关闭该节点：*

```
# shutdown -h now
```

3. *对每个 Controller 节点执行这些步骤，直到关闭所有 Controller 节点。*

13.7. 关闭 UNDERCLOUD

作为关闭 Red Hat OpenStack Platform 环境的一部分，登录到 undercloud 节点并关闭 undercloud。

先决条件

- *正在运行的 undercloud*

步骤

1. *以 stack 用户身份登录 undercloud。*
2. *关闭 undercloud：*

```
$ sudo shutdown -h now
```

13.8. 执行系统维护

在完全关闭 undercloud 和 overcloud 后，对环境中的系统执行任何维护，然后启动 undercloud 和 overcloud。

13.9. UNDERCLOUD 和 OVERCLOUD 启动顺序

要启动 Red Hat OpenStack Platform 环境，您必须按照以下顺序启动 undercloud 和 overcloud：

1. 启动 undercloud。
2. 启动 Controller 节点。
3. 启动 Ceph Storage 节点。
4. 启动 Compute 节点。
5. 启动 overcloud Compute 节点上的实例。

13.10. 启动 UNDERCLOUD

作为启动 Red Hat OpenStack Platform 环境的一部分，启动 undercloud 节点，登录到 undercloud，再检查 undercloud 服务。

先决条件

- undercloud 已关闭。

流程

- 打开 undercloud 并等待 undercloud 引导。

验证

1. 以 stack 用户身份登录 undercloud 主机。

2. 查找 stackrc undercloud 凭证文件：

```
$ source ~/stackrc
```

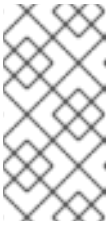
3. 检查 undercloud 上的服务：

```
$ systemctl list-units 'tripleo_*'
```

4. 验证名为 tripleo-ansible-inventory.yaml 的静态清单文件：

```
$ validation run --group pre-introspection -i <inventory_file>
```

- 将 <inventory_file > 替换为 Ansible 清单文件的名称和位置，如 ~/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml。



注意

当您运行验证时，输出中的 Reasons 列仅限于 79 个字符。要查看验证结果已满，请查看验证日志文件。

5. 检查所有服务和容器是否活跃且健康：

```
$ validation run --validation service-status --limit undercloud -i <inventory_file>
```

其他资源

- 使用验证框架

13.11. 启动 CONTROLLER 节点

作为启动 Red Hat OpenStack Platform 环境的一部分，打开每个 Controller 节点电源，并检查节点上的非 Pacemaker 服务。

先决条件

- **Controller 节点已关机。**

流程

- **打开每个 Controller 节点电源。**

验证

1. **以 root 用户身份登录每个 Controller 节点。**
2. *检查* **Controller 节点上的服务：**

```
$ systemctl -t service
```

只有基于非 Pacemaker 的服务正在运行。

3. **等待 Pacemaker 服务启动并检查服务是否已启动：**

```
$ pcs status
```

注意

如果您的环境使用 Instance HA，Pacemaker 资源不会在您启动 Compute 节点或使用 pcs stonith confirm <compute_node> 命令执行手动 unfence 操作前启动。您必须在使用 Instance HA 的每个 Compute 节点上运行此命令。

13.12. 启动 CEPH STORAGE 节点

作为启动 Red Hat OpenStack Platform 环境的一部分，打开 Ceph MON 和 Ceph Storage 节点电源，并启用 Ceph Storage 服务。

先决条件

- **已关闭电源的 Ceph Storage 集群**

- Ceph MON 服务在已关闭电源的单机 Ceph MON 节点或已打开电源的 Controller 节点上启用

步骤

1. 如果您的环境有单机 Ceph MON 节点，请打开每个 Ceph MON 节点电源。

2. 打开每个 Ceph Storage 节点电源。

3. 以 root 用户身份登录运行 Ceph MON 服务的节点，如 Controller 节点或单机 Ceph MON 节点。

4. 检查集群节点的状态：在以下示例中，podman 命令在 Controller 节点上的 Ceph MON 容器中运行状态检查：

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

确保每个节点都已打开电源并连接。

5. 为集群取消设置 noout、norecover、norebalance、nobackfill、nodown 和 pause 标志。在以下示例中，podman 命令通过 Controller 节点上的 Ceph MON 容器取消设置这些标志：

```
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset noout
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset norecover
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset norebalance
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset nobackfill
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset nodown
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset pause
```

验证

1. 检查集群的运行状况。在以下示例中，podman 命令在 Controller 节点上的 Ceph MON 容器中运行状态检查：

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

确保状态为 HEALTH_OK。

其他资源

- “关闭并启动整个 Ceph 集群的步骤是什么？”

13.13. 启动 COMPUTE 节点

作为启动 Red Hat OpenStack Platform 环境的一部分，打开每个 Compute 节点电源并检查节点上的服务。

先决条件

- 关闭 Compute 节点电源

步骤

1. 打开每个 Compute 节点电源。

验证

1. 以 root 用户身份登录每个 Compute。
2. 检查 Compute 节点上的服务：

```
$ systemctl -t service
```

13.14. 启动 OVERCLOUD COMPUTE 节点上的实例

作为启动 Red Hat OpenStack Platform 环境的一部分，启动 Compute 节点上的实例。

先决条件

- 具有活跃节点的活跃 overcloud

步骤

1. *以 **stack** 用户身份登录 **undercloud**。*

2. *提供 **overcloud** 的凭据文件：*

```
$ source ~/overcloudrc
```

3. *查看 **overcloud** 中运行的实例：*

```
$ openstack server list --all-projects
```

4. *启动 **overcloud** 中的实例：*

```
$ openstack server start <INSTANCE>
```

第 14 章 DIRECTOR 错误故障排除

在 director 过程的某些阶段可能发生错误。本节包含一些有关诊断常见问题的信息。

14.1. 节点注册故障排除

由于节点详细信不正确的问题而通常导致节点注册问题。在这些情况下，请验证包含节点详细信息的模板文件并更正导入的节点详细信息。

步骤

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 使用 --validate-only 选项运行节点导入命令。此选项无需执行导入即可验证您的节点模板：

```
(undercloud) $ openstack overcloud node import --validate-only ~/nodes.json
Waiting for messages on queue 'tripleo' with no timeout.

Successfully validated environment file
```

3. 要修复导入节点的错误详细信息，请运行 openstack baremetal 命令以更新节点详细信息。下例显示如何更改网络详细信息：

   a. 识别导入节点的已分配端口 UUID：

```
$ source ~/stackrc
(undercloud) $ openstack baremetal port list --node [NODE UUID]
```

   b. 更新 MAC 地址：

```
(undercloud) $ openstack baremetal port set --address=[NEW MAC] [PORT UUID]
```

   c. 在节点上配置新的 IPMI 地址：

```
(undercloud) $ openstack baremetal node set --driver-info ipmi_address=[NEW IPMI
ADDRESS] [NODE UUID]
```

14.2. 硬件内省故障排除

如果检查 RAM 磁盘没有响应，裸机置备检查器服务 ironic-inspector 会在默认的一小时后超时。超时可能会指示检查 RAM 磁盘中的一个错误，但通常因为环境错误配置而导致超时。

您可以诊断和解决常见的环境错误配置问题，以确保内省进程完成运行。

流程

1. 查找 stackrc undercloud 凭证文件：

```
$ source ~/stackrc
```

2. 确保您的节点处于 manageable 状态。内省不检查处于 available 状态的节点，该状态意味着用于部署。如果要检查处于 available 状态的节点，请在内省前将节点状态更改为 manageable 状态：

```
(undercloud)$ openstack baremetal node manage <node_uuid>
```

3. 要在内省调试过程中配置对内省 RAM 磁盘的临时访问，请使用 sshkey 参数将公共 SSH 密钥附加到 /httpboot/inspector.ipxe 文件中的 内核 配置中：

```
kernel http://192.2.0.1:8088/agent.kernel ipa-inspection-callback-
url=http://192.168.0.1:5050/v1/continue ipa-inspection-collectors=default,extra-hardware,logs
systemd.journald.forward_to_console=yes BOOTIF=${mac} ipa-debug=1 ipa-inspection-
benchmarks=cpu,mem,disk selinux=0 sshkey="<public_ssh_key>"
```

4. 在节点上运行内省：

```
(undercloud)$ openstack overcloud node introspect <node_uuid> --provide
```

内省完成后，使用 --provide 选项将节点状态更改为 available。

5. 从 dnsmasq 日志中识别节点的 IP 地址：

```
(undercloud)$ sudo tail -f /var/log/containers/ironic-inspector/dnsmasq.log
```

6. 如果出错，则使用根用户和临时访问详细信息访问节点：

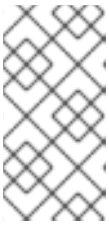
```
$ ssh root@192.168.24.105
```

在内省期间访问节点以运行诊断命令并排除内省故障。

7. 要停止内省过程，请运行以下命令：

```
(undercloud)$ openstack baremetal introspection abort <node_uuid>
```

您也可以等待操作过程超时。



注意

Red Hat OpenStack Platform director 在初始中止后重试内省三次。在每次尝试时均运行 **openstack baremetal introspection abort** 命令以完全中止内省。

14.3. OVERCLOUD 创建和部署故障排除

使用 OpenStack Orchestration (heat) 服务对 overcloud 进行初始创建。如果 overcloud 部署失败，则使用 OpenStack 客户端和服务日志文件诊断失败的部署。

步骤

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 启动临时 Heat 进程：

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-deploy/overcloud/heat-launcher --restore-db
(undercloud)$ export OS_CLOUD=heat
```

3. 查看失败的详情：

```
(undercloud)$ openstack stack failures list <overcloud> --long
```

- 用您的 overcloud 的名称替换 <overcloud>。

4. 识别失败的堆栈：

```
(undercloud)$ openstack stack list --nested --property status=FAILED
```

5. 从 undercloud 中删除临时 Heat 进程：

```
(undercloud)$ openstack tripleo launch heat --kill
```

14.4. 节点置备故障排除

OpenStack Orchestration (heat) 服务控制置备过程。如果节点置备失败，则使用 OpenStack 客户端和服务日志文件诊断问题。

步骤

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 检查裸机恢复服务以查看所有注册节点及其当前状态：

```
(undercloud) $ openstack baremetal node list

+----------+------+--------------+-------------+-----------------+-------------+
| UUID     | Name | Instance UUID | Power State | Provision State | Maintenance |
+----------+------+--------------+-------------+-----------------+-------------+
| f1e261...| None | None          | power off   | available       | False       |
| f0b8c1...| None | None          | power off   | available       | False       |
+----------+------+--------------+-------------+-----------------+-------------+
```

可用于置备的所有节点都应设置以下状态：

- *Maintenance 设置为 False。*
- *在置备前，Provision State 设置为 available。*

3. *如果节点的 Maintenance 没有设置为 False 或 Provision State 设置为 available，使用以下表来找出问题以及相关的解决方案。*

问题	原因	解决方案
Maintenance 自动将自身设置为 **True**。	director 无法访问节点的电源管理。	检查节点电源管理的凭据。
Provision State 设置为 **available**，但节点未置备。	此问题在裸机部署启动前发生。	检查节点硬件详情是否在要求内。
节点的 Provision State 设置为 **wait call-back**。	此节点的节点置备过程尚未完成。	等到此状态更改。否则，连接到节点的虚拟控制台并检查输出。
Provision State 处于 **active**，Power State 处于 **power on**，但节点无响应。	节点置备已成功完成，并在部署后配置步骤中出问题。	诊断节点配置过程。连接到节点的虚拟控制台并检查输出。
Provision State 为 **error** 或 **deploy failed**。	节点置备已失败。	使用 **openstack baremetal node show** 命令查看裸机节点详细信息，并检查 **last_error** 字段，其中包含错误说明。

其他资源

- *裸机节点置备状态*

14.5. 置备期间 IP 地址冲突故障排除

如果目标主机分配的 IP 地址已经使用，则内省和部署任务将失败。要防止这些失败，可对 Provisioning 网络执行端口扫描，以确定发现 IP 范围和主机 IP 范围是否可用。

步骤

1. **安装 nmap:**

```
$ sudo dnf install nmap
```

2. **使用 nmap 命令扫描 IP 地址范围中的活动地址。这个示例会扫描 192.168.24.0/24 这个范围，使用 Provisioning 网络的 IP 子网值（使用 CIDR 位掩码符号 ）替换它：**

```
$ sudo nmap -sn 192.168.24.0/24
```

3. **复查 nmap 扫描的输出。例如，您应查看 undercloud 的 IP 地址，以及子网上存在的任何其他主机：**

```
$ sudo nmap -sn 192.168.24.0/24

Starting Nmap 6.40 ( http://nmap.org ) at 2015-10-02 15:14 EDT
Nmap scan report for 192.168.24.1
Host is up (0.00057s latency).
Nmap scan report for 192.168.24.2
Host is up (0.00048s latency).
Nmap scan report for 192.168.24.3
Host is up (0.00045s latency).
Nmap scan report for 192.168.24.5
Host is up (0.00040s latency).
Nmap scan report for 192.168.24.9
Host is up (0.00019s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.45 seconds
```

如果这些活跃的 IP 地址和 undercloud.conf 中指定的 IP 地址范围有冲突，则必须在内省或部署overcloud 节点前修改 IP 地址范围或释放一些 IP 地址。

14.6. OVERCLOUD 配置故障排除

Red Hat OpenStack Platform director 使用 Ansible 配置 overcloud。完成以下步骤，对 overcloud 上的 Ansible playbook 错误 (config-download) 进行诊断。

流程

1. **确保 stack 用户有权访问 undercloud 上 ~/config-download/overcloud 目录中的文件：**

```
$ sudo setfacl -R -m u:stack:rwx ~/config-download/overcloud
```

2. 进入 config-download 文件的工作目录：

```
$ cd ~/config-download/overcloud
```

3. 搜索 ansible.log 文件以查找故障点：

```
$ less ansible.log
```

记录失败的步骤。

4. 在工作目录中查找 config-download playbook 中失败的步骤以确定发生的操作。

14.7. 容器配置故障排除

Red Hat OpenStack Platform director 使用 podman 管理容器，并使用 puppet 创建容器配置。此步骤显示如何在出错时对容器进行诊断。

访问主机

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 获取包含容器故障的节点的 IP 地址。

```
(undercloud) $ metalsmith list
```

3. 登录该节点：

```
(undercloud) $ ssh tripleo-admin@192.168.24.60
```

识别故障容器

1. 查看所有容器：

```
$ sudo podman ps --all
```

识别故障容器。故障容器通常以非零的状态推出。

检查容器日志

1. 每个容器都会保留其主进程的标准输出内容。使用此输出内容作为日志，帮助确定容器运行过程中实际上发生了什么。例如，要查看 **keystone** 容器的日志，请运行以下命令：

```
$ sudo podman logs keystone
```

在大多数情况下，此日志包含有关容器故障原因的信息。

2. 主机还包含已失败服务的 **stdout** 日志。可在 **/var/log/containers/stdouts/** 中查找 **stdout** 日志。例如，要查看出现故障的 **keystone** 容器的日志，请运行以下命令：

```
$ cat /var/log/containers/stdouts/keystone.log
```

检查容器

在某些情况下，您可能需要验证容器的相关信息。例如，请使用以下命令来查看 **keystone** 容器的相关数据：

```
$ sudo podman inspect keystone
```

此命令返回一个包含低级配置数据的 **JSON** 对象。您可以通过管道将这些输出内容传递给 **jq** 命令，以对特定数据进行解析。例如，要查看 **keystone** 容器的加载情况，请运行以下命令：

```
$ sudo podman inspect keystone | jq .[0].Mounts
```

您还可以使用 **--format** 选项将数据解析到一行中，这在针对一组容器数据运行命令时非常有用。例如，要重建用于运行 **keystone** 容器的选项，请使用包含 **--format** 选项的以下 **inspect** 命令：

```
$ sudo podman inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range .Mounts}} -v
{{.Source}}:{{.Destination}}:{{ join .Options "," }}{{end}} -ti {{.Config.Image}}' keystone
```

注意

--format 选项会按照 Go 语法来创建查询。

将这些选项和 podman run 命令一起使用以重新创建容器用于故障排除目的：

```
$ OPTIONS=$( sudo podman inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range
.Mounts}} -v {{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}'
keystone )
$ sudo podman run --rm $OPTIONS /bin/bash
```

在容器内运行命令

在某些情况下，您可能需要通过特定的 Bash 命令从容器中获取信息。在此情况下，使用以下 podman 命令以在运行中容器内执行命令。例如，运行 podman exec 命令以在 keystone 容器内运行命令：

```
$ sudo podman exec -ti keystone <COMMAND>
```

注意

-ti 选项会通过交互式伪终端来运行命令。

- 用您要运行的命令替换 <COMMAND>。例如，每个容器都有一个健康检查脚本，用于验证服务的连接状况。您可以使用以下命令为 keystone 运行这个健康检查脚本：

```
$ sudo podman exec -ti keystone /openstack/healthcheck
```

要访问容器的 shell，请运行 podman exec，并将 /bin/bash 用作您要在容器内运行的命令：

```
$ sudo podman exec -ti keystone /bin/bash
```

查看容器文件系统

1. 要查看故障容器的文件系统，请运行 podman mount 命令。例如，要查看出现故障的 keystone 容器的文件系统，请运行以下命令：

```
$ sudo podman mount keystone
```

这会提供一个挂载位置，用于查看文件系统内容：

```
/var/lib/containers/storage/overlay/78946a109085aeb8b3a350fc20bd8049a08918d74f573396d7358270e711c610/merged
```

这对于查看容器内的 Puppet 报告很有用。您可以在容器挂载内的 var/lib/puppet/ 目录中查找这些报告。

导出容器

当容器出现故障时，您可能需要调查文件中包含的所有内容。在这种情况下，您可以将容器的整个文件系统导出为 tar 归档。例如，要导出 keystone 容器的文件系统，请运行以下命令：

```
$ sudo podman export keystone -o keystone.tar
```

这个命令会创建 keystone.tar 归档，以供您提取和研究。

14.8. COMPUTE 节点故障排除

Compute 节点使用 Compute 服务来执行基于虚拟机监控程序的操作。这意味着，对 Compute 节点进行故障排除可以解决与这个服务相关的问题。

步骤

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 获取包含故障的 Compute 节点的 IP 地址：

```
(undercloud) $ openstack server list
```

3. 登录该节点：

```
(undercloud) $ ssh tripleo-admin@192.168.24.60
```

4. 切换到 root 用户：

```
$ sudo -i
```

5. 查看容器状态：

```
$ sudo podman ps -f name=nova_compute
```

6. Compute 节点的主日志文件为 /var/log/containers/nova/nova-compute.log。如果 Compute 节点通信出现问题，请使用此文件开始诊断。

7. 如果需要在 Compute 节点上进行维护工作，把主机上存在的实例迁移到另外一个可以正常工作的 Compute 节点上，然后禁用需要进行维护的节点。

14.9. 创建 SOSREPORT

如果您需要联系红帽以获得 Red Hat OpenStack 平台的产品支持，可能需要生成一份 sosreport。有关创建 sosreport 的更多信息，请参阅：

- "How to collect all required logs for Red Hat Support to investigate an OpenStack issue"

14.10. 日志位置

在故障排除时，使用以下日志手机 undercloud 和 overcloud 的信息。

表 14.1. undercloud 和 overcloud 节点上的日志

信息	日志位置
容器化服务日志	/var/log/containers/
容器化服务中的标准输出	/var/log/containers/stdouts
Ansible 配置日志	~/ansible.log

表 14.2. undercloud 节点上的其他日志

信息	日志位置
openstack overcloud deploy 的命令历史记录	**/home/stack/.tripleo/history**
Undercloud 安装日志	**/home/stack/install-undercloud.log**

表 14.3. overcloud 节点上的其他日志

信息	日志位置
Cloud-Init 日志	**/var/log/cloud-init.log**
高可用性日志	**/var/log/pacemaker.log**

第 15 章 UNDERCLOUD 和 OVERCLOUD 服务的提示

本节提供有关在 undercloud 上调整和管理特定 OpenStack 服务的建议。

15.1. 调优部署性能

Red Hat OpenStack Platform director 使用 OpenStack Orchestration (heat) 开展主要部署和置备功能。Heat 使用一系列 worker 执行部署任务。为计算默认 worker 数，director 的 heat 配置将 undercloud 的总 CPU 线程计数减半。在本实例中，线程数是指 CPU 内核数乘以超线程值。例如，如果 undercloud 的 CPU 具有 16 个线程，则 heat 默认生成 8 个 worker。director 配置会默认使用最小和最大值：

服务	最小值	最大值
OpenStack Orchestration (heat)	4	24

但是，您可以使用环境文件中的 HeatWorkers 参数手动设置 worker 数：

heat-workers.yaml

```
parameter_defaults:
  HeatWorkers: 16
```

undercloud.conf

```
custom_env_files: heat-workers.yaml
```

15.2. 更改 HAPROXY 的 SSL/TLS 密码规则

如果在 undercloud 中启用了 SSL/TLS （请参阅 第 4.2 节 “undercloud 配置参数”），您可能需要强化与 HAProxy 配置一起使用的 SSL/TLS 密码和规则。这种加强有助于避免 SSL/TLS 漏洞，如

POODLE 漏洞。

使用 hieradata_override undercloud 配置选项，设置下列 hieradata：

tripleo::haproxy::ssl_cipher_suite

HAProxy 中使用的密码套件。

tripleo::haproxy::ssl_options

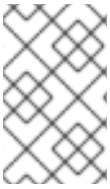
HAProxy 中使用的 SSL/TLS 规则。

例如，您可能需要使用下列密码和规则：

- *Cipher: ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS*
- *规则：no-sslv3 no-tls-tickets*

创建一个包含以下内容的 hieradata 覆盖文件 (haproxy-hiera-overrides.yaml)：

```
tripleo::haproxy::ssl_cipher_suite: ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS
tripleo::haproxy::ssl_options: no-sslv3 no-tls-tickets
```



注意

cipher 集合是一个连续行。

设置 undercloud.conf 文件中的 hieradata_override 参数，以便使用在运行 openstack undercloud install 之前创建的 hieradata 覆盖文件：

```
[DEFAULT]
...
hieradata_override = haproxy-hiera-overrides.yaml
...
```

第 16 章 电源管理驱动

虽然 IPMI 是 director 用来进行电源管理的主要方法，但是 director 也支持其它电源管理类型。此附录包含 director 支持的电源管理功能列表。为 overcloud 注册节点时使用这些电源管理设置。有关更多信息，请参阅为 overcloud 注册节点。

16.1. 智能平台管理接口 (IPMI)

使用基板管理控制器 (BMC) 时的标准电源管理方法。

pm_type

将这个选项设置为 ipmi。

pm_user; pm_password

IPMI 的用户名和密码。

pm_addr

IPMI 控制器的 IP 地址。

pm_port（可选）

连接到 IPMI 控制器的端口。

16.2. REDFISH

由分布式管理任务组 (DMTF) 开发的，IT 基础架构的标准 RESTful API

pm_type

将这个选项设置为 redfish。

pm_user; pm_password

Redfish 的用户名和密码。

pm_addr

Redfish 控制器的 IP 地址。

pm_system_id

系统资源的规范路径（canonical path）。此路径必须包含 root 服务、版本以及系统的路径/唯一 ID。例如：/redfish/v1/Systems/CX34R87.

redfish_verify_ca

如果基板管理控制器 (BMC) 中的 Redfish 服务没有配置为使用由认可证书颁发机构 (CA) 签名的有效 TLS 证书，ironic 中的 Redfish 客户端无法连接到 BMC。将 redfish_verify_ca 选项设置为 false 来静默错误。但请注意：禁用 BMC 身份验证会破坏您的 BMC 的访问安全性。

16.3. DELL REMOTE ACCESS CONTROLLER (DRAC)

DRAC 是一个提供远程电源功能的接口，这些功能包括电源管理和服务器监控。

pm_type

将这个选项设置为 idrac。

pm_user; pm_password

DRAC 的用户名和密码。

pm_addr

DRAC 主机的 IP 地址。

16.4. INTEGRATED LIGHTS-OUT (ILO)

iLO 是惠普提供的一个远程电源功能的接口，这些功能包括电源管理和服务器监控。

pm_type

将这个选项设置为 ilo。

pm_user; pm_password

iLO 的用户名和密码。

pm_addr

iLO 接口的 IP 地址。

- 要启用这个驱动程序，请将 ilo 添加到 undercloud.conf 的 enabled_hardware_types

选项中，然后重新运行 openstack undercloud install。

- HP 节点的 ILO 固件版本至少为 1.85（2015 年 5 月 13 日）才能成功内省。director 已成功测试了使用此 ILO 固件版本的节点。

- 不支持使用共享 iLO 端口。

16.5. FUJITSU INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)

Fujitsu iRMC 是一个 BMC（Baseboard Management Controller），它集成了 LAN 连接以及扩展的功能。这个驱动提供了对连接到 iRMC 上的裸机系统的电源管理功能。

重要

需要 iRMC S4 或更高版本。

pm_type

将这个选项设置为 irmc。

pm_user; pm_password

iRMC 接口的用户名和密码。

重要

iRMC 用户必须具有 ADMINISTRATOR 权限。

pm_addr

iRMC 接口的 IP 地址。

pm_port（可选）

iRMC 操作使用的端口。默认值是 443。

pm_auth_method（可选）

iRMC 操作的验证方法。使用 basic 或 digest。默认值为 basic。

pm_client_timeout（可选）

iRMC 操作的超时时间（以秒为单位）。默认值是 60 秒。

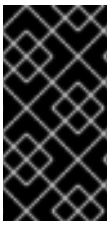
pm_sensor_method（可选）

获得感应器数据的方法。使用 ipmitool 或 scci。默认值是 ipmitool。

- *要启用此驱动程序，将 irmc 添加到 undercloud.conf 的 enabled_hardware_types 选项中，并重新运行 openstack undercloud install 命令。*

16.6. MANUAL-MANAGEMENT 驱动程序

使用 manual-management 驱动程序控制没有电源管理的裸机设备。director 无法控制已注册的裸机设备，您必须在内省和部署过程中的特定点执行手动电源操作。



重要

此选项仅适用于测试和评估目的。不建议用于 Red Hat OpenStack Platform 企业环境。

pm_type

将此选项设置为 manual-management。

- *这个驱动不使用任何验证信息，因为它不控制电源管理。*
- *要启用此驱动程序，将 manual-management 添加到 undercloud.conf 的 enabled_hardware_types 选项中，并重新运行 openstack undercloud install 命令。*
- *在 instackenv.json 节点清单文件中，为您要手动管理的节点将 pm_type 设置为 manual-management。*

内省

- 在节点上执行内省时，先手动启动节点，再运行 openstack overcloud node introspect 命令。确保节点通过 PXE 引导。

- 如果您启用了节点清理，在出现 Introspection completed 消息后手动重新引导节点，并在运行 openstack baremetal node list 命令时为每个节点 进行清理。确保节点通过 PXE 引导。

- 在内省和清理过程完成后，关闭节点。

Deployment

- 在执行 overcloud 部署时，使用 openstack baremetal node list 命令检查节点状态。等待节点状态从 deploying 变为 wait call-back，然后手动启动节点。确保节点通过 PXE 引导。

- 在 overcloud 置备过程完成后，节点将关闭。您必须从磁盘引导节点以启动配置过程。若要检查调配的完成，请使用 openstack baremetal node list 命令检查节点状态，然后等待每个节点的节点 状态更改处于活动状态。当节点状态为 active 时，手动引导置备的 overcloud 节点。