Red Hat OpenStack Platform 17.1

升级框架(16.2 到 17.1)

从 Red Hat OpenStack Platform 16.2 原位升级到 17.1

Last Updated: 2023-09-21

Red Hat OpenStack Platform 17.1 升级框架(16.2 到 17.1)

从 Red Hat OpenStack Platform 16.2 原位升级到 17.1

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南包含有关在长生命版本间进行原位升级的框架信息。此框架包含将 OpenStack Platform 环境从一个长生命版本升级到下一个长生命版本的工具。在这种情况下，本指南侧重于从 Red Hat OpenStack Platform 16.2 (Train)升级到 17.1 (Wallaby)。

目录

使开源包含更多

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。详情请查看 CTO Chris Wright 的信息。

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。

使用直接文档反馈(DDF)功能

使用 **添加反馈** DDF 功能，用于特定句子、段落或代码块上的直接注释。

1. 以 *Multi-page HTML* 格式查看文档。
2. 请确定您看到文档右上角的 **反馈** 按钮。
3. 用鼠标指针高亮显示您想评论的文本部分。
4. 点 **添加反馈**。
5. 在**添加反馈**项中输入您的意见。
6. 可选：添加您的电子邮件地址，以便文档团队可以联系您以讨论您的问题。
7. 点 **Submit**。

第 1 章 关于用于升级的 RED HAT OPENSTACK PLATFORM 框架

用于升级的 Red Hat OpenStack Platform (RHOSP)框架是将 RHOSP 环境从一个长生命周期版本升级到下一个长生命周期版本的工作流。此工作流是一个原位解决方案，升级会在现有环境中进行。

1.1. RED HAT OPENSTACK PLATFORM 17.1 中的高级别更改

升级到 Red Hat OpenStack Platform (RHOSP) 17.1 过程中会进行以下高级别更改：

- RHOSP 升级和操作系统升级被分为两个不同的阶段。您要先升级 RHOSP，然后升级操作系统。
- 您可以将部分 Compute 节点升级到 RHEL 9.2，而其余的 Compute 节点保留在 RHEL 8.4 上。这称为多 RHEL 环境。
- 通过升级到 Red Hat Ceph Storage 5，**cephadm** 现在管理 Red Hat Ceph Storage。之前版本的 Red Hat Ceph Storage 由 **ceph-ansible** 管理。Red Hat Ceph Storage 节点可以保留在 RHOSP 17.1 版本 5 上，直到 Red Hat Ceph Storage 5 生命周期结束为止。
- overcloud 使用 Open Virtual Network (OVN)作为默认的 ML2 机制驱动程序。可以将 Open vSwitch (OVS)服务迁移到 OVN，您可以在成功完成后执行。
- undercloud 和 overcloud 在 Red Hat Enterprise Linux 9 上运行。

1.2. RED HAT ENTERPRISE LINUX 9 的更改

Red Hat OpenStack Platform 17.1 使用 Red Hat Enterprise Linux (RHEL) 9.2 作为基础操作系统。作为升级过程的一部分，您要将节点的基础操作系统升级到 RHEL 9.2。

在开始升级前，请查看以下信息以熟悉 RHEL 9：

- 有关 RHEL 9 中最新更改的更多信息，请参阅 Red Hat Enterprise Linux 9.2 发行注记 。
- 有关 Red Hat Enterprise Linux 8 和 9 之间的主要区别的更多信息，请参阅使用 RHEL 9 的注意事项。
- 有关 Red Hat Enterprise Linux 9 的常规信息，请参阅 Red Hat Enterprise Linux 9 产品文档 。
- 有关从 RHEL 8 升级到 RHEL 9 的详情，请参考从 RHEL 8 升级到 RHEL 9 。

1.3. 长生命版本的升级框架

您可以使用 Red Hat OpenStack Platform (RHOSP) **升级框架** 通过多个 overcloud 版本执行原位升级路径。目标是为您提供一个机会，在下一个 长**生命版本可用时，仍保留某些 OpenStack 版本**，这些版本被视为长生命版本。

Red Hat OpenStack Platform 升级过程还会升级节点上的 Red Hat Enterprise Linux (RHEL)版本。

本指南通过以下版本提供升级框架：

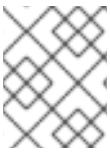
当前版本	目标版本
Red Hat OpenStack Platform 16.2.4 及更新的版本	Red Hat OpenStack Platform 17.1 latest

有关 Red Hat OpenStack Platform 生命周期支持的详细信息，请参阅 Red Hat OpenStack Platform 生命周期。

长生命版本的升级路径

在开始升级前，请熟悉可能的更新和升级路径。如果您在从主版本升级到主版本前使用早于 RHOSP 16.2.4 的环境，则必须首先将现有环境更新至最新的次版本。

例如，如果您的当前部署是 Red Hat Enterprise Linux (RHEL) 8.4 上的 Red Hat OpenStack Platform (RHOSP) 16.2.1，则必须在升级到 RHOSP 17.1 前对最新的 RHOSP 16.2 版本执行次要版本。



注意

您可以在 **/etc/rhosp-release** 和 **/etc/redhat-release** 文件中查看当前的 RHOSP 和 RHEL 版本。

表 1.1. 更新版本路径

当前版本	目标版本
RHOSP 16.2.x on RHEL 8.4	RHOSP 16.2 latest on RHEL 8.4 latest
RHOSP 17.0.x on RHEL 9.0	RHOSP 17.0 latest on RHEL 9.0 latest
RHOSP 17.0.x on RHEL 9.0	RHOSP 17.1 latest on RHEL 9.2 latest
RHOSP 17.1.x on RHEL 9.2	RHOSP 17.1 latest on RHEL 9.2 latest

如需更多信息，请参阅 执行 Red Hat OpenStack Platform 的次要更新 。

表 1.2. 升级版本路径

当前版本	目标版本
RHEL 8.4 上的 RHOSP 16.2	RHOSP 17.1 latest on RHEL 9.2 latest

红帽提供了两个选项，可将您的环境升级到下一个长生命版本：

原位升级

执行现有环境中的服务升级。本指南主要侧重于这个选项。

并行迁移

创建新的 RHOSP 17.1 环境，并将工作负载从当前环境迁移到新环境。有关 RHOSP 并行迁移的更多信息，请联系红帽全局专业服务。

重要

此表中的持续时间根据内部测试的最小估算，可能不适用于所有生产环境。例如，如果您的硬件具有低规格或扩展引导周期，则允许更多时间在这些持续时间内。要准确测量每个任务的升级持续时间，请在测试环境中与您的生产环境类似的硬件执行这些步骤。

表 1.3. 原位升级的持续时间和影响

	Duration	影响
undercloud 升级	30 分钟	无
overcloud 采用	6 分钟	无
overcloud 升级准备	20 分钟	无
Red Hat Ceph Storage 升级	30 分钟	无
Red Hat Ceph Storage 采用	30 分钟	无
Overcloud OpenStack 升级	70 分钟	无
Overcloud 系统升级	每个节点 30 分钟，也可以并行完成	无
Overcloud 计算升级	20 分钟	无
Overcloud 中断	由于工作负载从节点迁移到节点，因此中断是最小的。	

1.4. 规划和准备原位升级

在对 OpenStack Platform 环境进行原位升级前，为升级创建一个计划，并满足可能阻止成功升级的潜在障碍。

1.4.1. 熟悉 Red Hat OpenStack Platform 17.1

在执行升级前，熟悉 Red Hat OpenStack Platform 17.1 以帮助您了解生成的环境以及可能会影响升级的任何潜在的版本更改。要熟悉 Red Hat OpenStack Platform 17.1，请遵循以下建议：

- 阅读升级路径中所有版本的发行注记，并识别需要计划的任何潜在方面：
  - 包含新功能的组件
  - 已知问题

  使用以下链接为每个版本打开发行注记：
  - Red Hat OpenStack Platform 16.2，这是您的源版本
  - Red Hat OpenStack Platform 17.1，这是您的目标版本
- 阅读《使用 director 版本 17.1 安装和管理 Red Hat OpenStack Platform 指南），并熟悉本指南中任何新的要求和流程。
- 安装概念验证 Red Hat OpenStack Platform 17.1 undercloud 和 overcloud。开发目标 OpenStack Platform 版本的实践体验，并调查目标版本和当前版本之间的潜在差异。

1.4.2. Leapp 升级使用 Red Hat OpenStack Platform

长生命 Red Hat OpenStack Platform 升级需要一个从 Red Hat Enterprise Linux (RHEL) 8.4 升级到 RHEL 9.2 的基本操作系统。升级过程使用 Leapp 程序执行到 RHEL 9.2 的升级。但是，Leapp 升级的一些方面会被自定义，以确保您专门从 RHEL 8.4 升级到 RHEL 9.2。要将操作系统升级到 RHEL 9.2，请参阅 执行 undercloud 系统升级。

限制

有关可能会影响升级的潜在限制的详情，*请参考从 RHEL 8 升级到 RHEL 9* 指南中的以下部分：

- 计划升级
- 已知问题

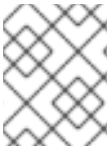
如果任何已知的限制影响您的环境，请联系红帽技术支持团队 的建议。

故障排除

有关故障排除潜在 Leapp 问题的详情，*请参考从 RHEL 8 升级到 RHEL 9* 中的 故障排除。

1.4.3. 支持的升级场景

在进行升级前，请检查您的 overcloud 是否被支持。



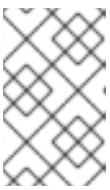
注意

如果您不确定这些列表中未提及的特定场景是否被支持，请参阅红帽 技术支持团队 的建议。

支持的场景

以下原位升级场景经过测试并被支持。

- 具有默认角色类型的标准环境：控制器、计算和 Ceph Storage OSD
- Split-Controller 可组合角色
- Ceph Storage 可组合角色
- Hyper-Converged Infrastructure：同一节点上的计算和 Ceph Storage OSD 服务
- 具有网络功能虚拟化(NFV)技术的环境：单根输入/输出虚拟化(SR-IOV)和数据平面开发套件(DPDK)
- 启用实例 HA 的环境



注意

在升级过程中，支持 nova 实时迁移。但是，由 Instance HA 启动的撤离不被支持。当您升级 Compute 节点时，节点会被完全关闭，且该节点上运行的任何工作负载都不会自动被 Instance HA 撤离。相反，您必须手动执行实时迁移。

技术预览场景

当您与这些功能结合使用时，升级的框架被视为技术预览，因此不受红帽完全支持。您应该只在概念验证环境中测试此场景，而不在生产环境中进行升级。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

- 边缘和分布式 Compute 节点(DCN)场景

1.4.4. 使用外部 Ceph 部署升级的注意事项

如果您单独部署了 Red Hat Ceph Storage 系统，然后使用 director 部署和配置 OpenStack，然后才能将 Red Hat OpenStack Platform (RHOSP)部署从 16.2 升级到 17.1，您必须将 Red Hat Ceph Storage 集群从版本 4 升级到 5。升级 Red Hat Ceph Storage 集群后，Red Hat OpenStack Platform 16 将继续操作，并与存储集群兼容。有关更多信息，请参阅 *Red Hat Ceph Storage 5 升级指南* 中的升级 Red Hat Ceph Storage 集群。

1.4.5. 可能会阻止升级的已知问题

查看以下可能影响成功升级的已知问题。

BZ#2224085 - 当指定 -debug 时，Leapp 会停留在 Interim System

如果您将操作系统从 RHEL 7.x 升级到 RHEL 8.x，或从 RHEL 8.x 升级到 RHEL 9.x，请不要使用 **--debug** 选项运行 Leapp 升级。系统 **会一直处于设置代码状态的早期控制台**，不会自动重启。要避免这个问题，**UpgradeLeappDebug** 参数默认设置为 **false**。不要在您的模板中更改这个值。

BZ#2203785 - Collectd sensubility 在 overcloud 节点重启后停止工作。

重启 overcloud 节点后，权限问题会导致 collectd-sensubility 停止工作。因此，collectd-sensubility 会停止报告容器健康状况。在从 Red Hat OpenStack Platform (RHOSP) 16.2 升级到 RHOSP 17.1 的过程中，overcloud 节点会作为 Leapp 升级的一部分重启。为确保 collectd-sensubility 继续工作，请运行以下命令：

```
sudo podman exec -it collectd setfacl -R -m u:collectd:rwx /run/podman
```

BZ#2222683 -(Edge+DirOp+OSasInfra) FFU 和 Multi-Rhel Test Execution

多 RHEL 环境中不支持以下部署：

- ShiftOnStack
- DirectorOperator

您必须将所有 Compute 主机升级到 RHEL 9.2，或者保持所有在 RHEL 8.4 上运行的计算主机。

分布式 Compute 节点(DCN)部署不支持 Multi-RHEL 环境，也不支持从 RHOSP 16.2 升级到 RHOSP 17.1。

BZ#2180542 - After upgrade, manila ceph-nfs fails to start with error: ceph-nfs start on leaf1-controller-0 returned 'error' because 'failed'

Pacemaker 控制的 **ceph-nfs** 资源需要一个运行时目录来存储某些进程数据。安装或升级 RHOSP 时会创建该目录。目前，重启 Controller 节点会删除目录，在 Controller 节点重启时不会恢复 **ceph-nfs** 服务。如果所有 Controller 节点都已重启，**ceph-nfs** 服务会永久失败。

您可以应用以下临时解决方案：

1. 如果重启 Controller 节点，登录到 Controller 节点并创建 **/var/run/ceph** 目录：
   $ mkdir -p /var/run/ceph

2. 在所有已重新引导的 Controller 节点上重复此步骤。如果在创建目录后 **ceph-nfs-pacemaker** 服务已标记为失败，请从任何 Controller 节点运行以下命令：
   $ pcs resource cleanup

BZ#2228818 -(FFU 16.2 → 17.1) nova_virtlogd 容器镜像在计算升级到 9.2 后不会更新

从 RHEL 8.4 升级到 RHEL 9.2 后，**nova_virtlogd** 容器镜像将继续运行。此容器镜像使用 Red Hat Universal Base Image (UBI) 8 容器。另外，还会创建另一个名为 **nova_virtlogd_wrapper** 的容器镜像，该镜像使用 UBI 9 容器。升级后，所有容器镜像都应该使用 UBI 9 容器。但是，**nova_virtlogd** 容器镜像不会影响环境。后续 RHOSP 发行版本中会进行修复。

BZ# 2229761 -(OSP16.2 → 17.1)在 OVN 的控制器升级过程中丢失数据包

目前，在 **ovn_controller** 和 **ovn_dbs** 的部署步骤中存在一个竞争条件，这会导致 **ovn_dbs** 在 **ovn_controller** 前升级。如果在 **ovn_dbs** 之前没有升级 **ovn_controller**，则重启到新版本时出错会导致数据包丢失。如果在 Open Virtual Network (OVN)升级过程中发生竞争条件，预计会出现一分钟网络中断。后续 RHOSP 发行版本中会进行修复。

1.4.6. 备份和恢复

在升级 Red Hat OpenStack Platform (RHOSP) 16.2 环境前，请使用以下选项之一备份 undercloud 和 overcloud control plane：

- 在执行升级前备份节点。有关在升级前备份节点的更多信息，请参阅 *Red Hat OpenStack Platform 16.2 备份和恢复 undercloud 和 control plane 节点*。
- 执行 undercloud 升级前，并在执行 overcloud 升级前备份 undercloud 节点。有关备份 undercloud 的更多信息，请参阅 *Red Hat OpenStack Platform 17.1 备份和恢复 undercloud 和 control plane 节点*中的 undercloud 节点的备份。
- 使用适合您的环境的第三方备份和恢复工具。有关认证备份和恢复工具的更多信息，请参阅 红帽生态系统目录。

1.4.7. 次版本更新

如果您在升级 RHOSP 环境前使用早于 16.2.4 的 Red Hat OpenStack Platform (RHOSP)版本，请将环境更新至当前版本的最新次版本。例如，在升级到 Red Hat OpenStack Platform 17.1 之前，将 Red Hat OpenStack Platform 16.2.3 环境更新至最新的 16.2 版本。

有关为 Red Hat OpenStack Platform 16.2 执行次要版本更新的说明，请参阅 保持 Red Hat OpenStack Platform Updated。

1.4.8. 代理配置

如果您将代理与 Red Hat OpenStack Platform 16.2 环境搭配使用，则 **/etc/environment** 文件中的代理配置会保留操作系统升级和 Red Hat OpenStack Platform 17.1 升级。

- 如需有关 Red Hat OpenStack Platform 16.2 代理配置的更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform 中的使用* 代理运行 undercloud 时的注意事项。
- 有关 Red Hat OpenStack Platform 17.1 的代理配置的更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform 中的使用* 代理运行 undercloud 时的注意事项。

1.4.9. 计划 Compute 节点升级

将 Compute 节点从 Red Hat OpenStack Platform (RHOSP) 16.2 升级到 RHOSP 17.1 后，您可以选择以下选项之一升级 Compute 主机操作系统：

- 在 Red Hat Enterprise Linux (RHEL) 8.4 上保留一部分 Compute 节点，并将剩余的计算节点升级到 RHEL 9.2。这被称为多 RHEL 环境。
- 将所有 Compute 节点升级到 RHEL 9.2，并完成环境的升级。
- 在 RHEL 8.4 上保留所有 Compute 节点。RHEL 8.4 的生命周期适用。

多 RHEL 环境的好处

您必须将所有 Compute 节点升级到 RHEL 9.2，以利用任何只在 RHOSP 17.1 中支持的硬件相关功能，如 vTPM 和安全引导。但是，您可能需要某些或所有 Compute 节点保留在 RHEL 8.4 上。例如，如果您为 RHEL 8 认证了一个应用程序，您可以在 RHEL 8.4 上保持 Compute 节点来支持应用程序，而无需阻塞整个升级。

将部分 Compute 节点升级到 RHEL 9.2 的选项可让您更好地控制升级过程。您可以在计划的维护窗口中确定升级 RHOSP 环境，并将操作系统延迟到另一个时间。需要较少的停机时间，从而最大程度降低对最终用户的影响。

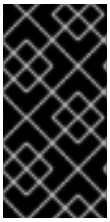
注意

如果您计划从 RHOSP 17.1 升级到 RHOSP 18.0，您必须将所有主机升级到 RHEL 9.2。如果在延长生命周期支持阶段外的主机上运行 RHEL 8.4，则必须获得 TUS 订阅。

多 RHEL 环境的限制

在多 RHEL 环境中有以下限制：

- 运行 RHEL 8 的计算节点无法使用 NVMe-over-TCP Cinder 卷。
- 您不能将不同的路径用于 RHOSP 16.2 和 17.1 上的套接字文件进行 collectd 监控。
- 您无法混合 ML2/OVN 和 ML2/OVS 机制驱动程序。例如，如果您的 RHOSP 16.2 部署包含 ML2/OVN，则您的 Multi-RHEL 环境必须使用 ML2/OVN。
- 多 RHEL 环境中不支持 FIPS。FIPS 部署是第 1 天操作。RHOSP 16.2 不支持 FIPS。因此，当您从 RHOSP 16.2 升级到 17.1 时，17.1 环境不包括 FIPS。
- 目前不支持边缘拓扑。



重要

支持的超融合基础架构环境中的所有 HCI 节点都必须使用与 Red Hat OpenStack Platform 控制器使用的版本相同的 Red Hat Enterprise Linux 版本。如果要在同一超融合基础架构环境中的 HCI 节点上使用混合状态的多个 Red Hat Enterprise 版本，请联系红帽客户体验和参与度 团队，讨论支持例外。

升级 Compute 节点

使用以下选项之一升级 Compute 节点：

- 要对 Compute 节点执行多 RHEL 升级，请参阅将 Compute 节点升级到多 RHEL 环境。

- 要将所有 Compute 节点升级到 RHEL 9.2，请参阅将 Compute 节点升级到 RHEL 9.2。
- 如果您要在 RHEL 8.4 上保留所有 Compute 节点，则不需要额外的配置。

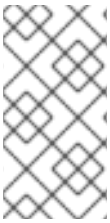
1.5. 软件仓库

本节包含 undercloud 和 overcloud 的软件仓库。在某些情况下需要启用软件仓库时，请参阅本节：

- 在注册到红帽客户门户网站时启用软件仓库。
- 启用存储库并将其同步到您的 Red Hat Satellite 服务器。
- 在注册到您的 Red Hat Satellite 服务器时启用存储库。

1.5.1. 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。从 RHOSP 16.2 升级时，在多 RHEL 环境中也支持 RHEL 8.4 Compute 节点。



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

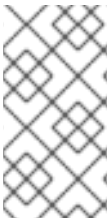
下表列出了用于安装 undercloud 的核心软件仓库。

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。

名称	软件仓库	要求说明
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。
Red Hat Enterprise Linux 8.4 for x86_64 - BaseOS (RPMs) Telecommunications Update Service (TUS)	**rhel-8-for-x86_64-baseos-tus-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 8.4 for x86_64 - AppStream (RPMs)	**rhel-8-for-x86_64-appstream-tus-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat OpenStack Platform for RHEL 8 x86_64 (RPMs)	**openstack-17.1-for-rhel-8-x86_64-rpms**	Red Hat OpenStack Platform 核心软件仓库。

1.5.2. 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。从 RHOSP 16.2 升级时，在多 RHEL 环境中也支持 RHEL 8.4 Compute 节点。



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。
Red Hat Enterprise Linux 8.4 for x86_64 - BaseOS (RPMs) Telecommunications Update Service (TUS)	**rhel-8-for-x86_64-baseos-tus-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 8.4 for x86_64 - AppStream (RPMs)	**rhel-8-for-x86_64-appstream-tus-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat OpenStack Platform for RHEL 8 x86_64 (RPMs)	**openstack-17.1-for-rhel-8-x86_64-rpms**	Red Hat OpenStack Platform 核心软件仓库。

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。

名称	软件仓库	要求说明
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。
Red Hat Enterprise Linux 8.4 for x86_64 - BaseOS (RPMs) Telecommunications Update Service (TUS)	**rhel-8-for-x86_64-baseos-tus-rpms**	x86_64 系统的基本操作系统仓库。
Red Hat Enterprise Linux 8.4 for x86_64 - AppStream (RPMs)	**rhel-8-for-x86_64-appstream-tus-rpms**	包括 Red Hat OpenStack Platform 的依赖软件包。
Red Hat OpenStack Platform for RHEL 8 x86_64 (RPMs)	**openstack-17.1-for-rhel-8-x86_64-rpms**	Red Hat OpenStack Platform 核心软件仓库。

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。

名称	软件仓库	要求说明
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。

1.5.3. Red Hat Satellite Server 6 的注意事项

如果您使用 Red Hat Satellite Server 6 为 Red Hat OpenStack Platform (RHOSP) 环境托管 RPM 和容器镜像，并计划在 RHOSP 17.1 升级过程中使用 Satellite 6 来提供内容，则必须满足以下条件：

- 您的 Satellite 服务器托管 RHOSP 16.2 RPM 和容器镜像。
- 您已将 RHOSP 16.2 环境中的所有节点注册到 Satellite 服务器。
  例如，您使用链接到 RHOSP 16.2 内容视图的激活码将节点注册到 RHOSP 16.2 内容。

RHOSP 升级建议

- 为 RHOSP 16.2 undercloud 和 overcloud 启用和同步所需的 RPM 存储库。这包括必要的 Red Hat Enterprise Linux (RHEL) 9.2 软件仓库。
- 在命令行中创建自定义产品，以托管 RHOSP 17.1 的容器镜像。
- 为 RHOSP 17.1 升级创建并提升内容视图，并在内容视图中包含以下内容：
  - RHEL 8 软件仓库：
    - Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)

      ```
      rhel-8-for-x86_64-appstream-tus-rpms
      ```

    - Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)

      ```
      rhel-8-for-x86_64-baseos-tus-rpms
      ```

    - Red Hat Enterprise Linux 8 for x86_64 - High Availability (RPMs)

```
rhel-8-for-x86_64-highavailability-eus-rpms
```

- Red Hat Fast Datapath for RHEL 8(RPMs)

```
fast-datapath-for-rhel-8-x86_64-rpms
```

- RHEL 9 软件仓库：
  - Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)

```
rhel-9-for-x86_64-appstream-eus-rpms
```

  - Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs)

```
rhel-9-for-x86_64-baseos-eus-rpms
```

- 所有 undercloud 和 overcloud RPM 存储库，包括 RHEL 9.2 软件仓库。为了避免启用 RHEL 软件仓库的问题，请确保包含正确版本的 RHEL 软件仓库，即 9.2。
- RHOSP 17.1 容器镜像。

- 将激活密钥与您为 RHOSP 17.1 升级创建的 RHOSP 17.1 内容视图相关联。
- 检查没有节点安装了 **katello-host-tools-fact-plugin** 软件包。Leapp 升级不会升级这个软件包。在 RHEL 9.2 系统上保留这个软件包会导致 **subscription-manager** 报告错误。
- 您可以将 Satellite 服务器配置为托管 RHOSP 17.1 容器镜像。如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 中的 为容器镜像准备 Satellite 服务器。
- 如果您使用 Red Hat Ceph Storage 订阅，并将 director 配置为将 **overcloud-minimal** 镜像用于 Red Hat Ceph Storage 节点，则必须在 Satellite 服务器上创建内容视图并将以下 RHEL 9.2 存储库添加到其中：
  - Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)

```
rhel-9-for-x86_64-appstream-eus-rpms
```

  - Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs)

```
rhel-9-for-x86_64-baseos-eus-rpms
```

  如需更多信息，请参阅 *Red Hat Satellite 管理内容指南* 中的导入内容和管理内容视图。

第 2 章 升级 UNDERCLOUD

将 undercloud 升级到 Red Hat OpenStack Platform 17.1。undercloud 升级使用正在运行的 Red Hat OpenStack Platform 16.2 undercloud。升级过程将 heat 堆栈导出到文件，并在升级节点上其余的服务时将 heat 转换为临时 heat。

2.1. 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-8-for-x86_64-baseos-tus-rpms \
--enable=rhel-8-for-x86_64-appstream-tus-rpms \
--enable=rhel-8-for-x86_64-highavailability-tus-rpms \
--enable=openstack-17.1-for-rhel-8-x86_64-rpms \
--enable=fast-datapath-for-rhel-8-x86_64-rpms
```

3. 在所有节点上将 **container-tools** 模块版本切换到 RHEL 8：

```
[stack@director ~]$ sudo dnf -y module switch-to container-tools:rhel8
```

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
```

2.2. 在升级前验证 RHOSP

在升级到 Red Hat OpenStack Platform (RHOSP) 17.1 之前，请使用 **tripleo-validations** playbook 验证 undercloud 和 overcloud。在 RHOSP 16.2 中，您可以通过 OpenStack Workflow Service (mistral)运行这些 playbook。

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. 安装软件包进行验证：

```
$ sudo dnf -y update openstack-tripleo-validations python3-validations-libs validations-common
```

4. 从 mistral 复制清单：

```
$ sudo chown stack:stack /var/lib/mistral/.ssh/tripleo-admin-rsa
$ sudo cat /var/lib/mistral/<stack>/tripleo-ansible-inventory.yaml > inventory.yaml
```

- 将 <stack> 替换为堆栈的名称。

5. 运行验证：

```
$ validation run -i inventory.yaml --group pre-upgrade
```

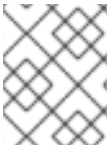
6. 查看脚本输出，以确定哪些验证成功并失败：

```
=== Running validation: "check-ftype" ===

Success! The validation passed for all hosts:
* undercloud
```

2.3. 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. 可选：备份 16.2 **containers-prepare-parameter.yaml** 文件：

```
$ cp containers-prepare-parameter.yaml \
  containers-prepare-parameter.yaml.orig
```

3. 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。

4. 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。

5. 如果您的部署包含 Red Hat Ceph Storage，请为您的部署使用的 Red Hat Ceph Storage 版本更新 **containers-prepare-parameter.yaml** 文件中的 Red Hat Ceph Storage 容器镜像参数：

```
ceph_namespace: registry.redhat.io/rhceph
ceph_image: <ceph_image_file>
ceph_tag: latest
ceph_grafana_image: <grafana_image_file>
ceph_grafana_namespace: registry.redhat.io/rhceph
ceph_grafana_tag: latest
```

   - 将 **<ceph_image_file** > 替换为部署使用的 Red Hat Ceph Storage 版本的镜像文件的名称：
     - Red Hat Ceph Storage 5: **rhceph-5-rhel8**
   - 使用部署使用的 Red Hat Ceph Storage 版本的镜像文件的名称替换：**<grafana_image_file>**
     - Red Hat Ceph Storage 5: **rhceph-5-dashboard-rhel8**

2.4. 容器镜像标记准则

Red Hat Container Registry 使用特定的版本格式来标记所有 Red Hat OpenStack Platform 容器镜像。此格式遵循每个容器的标签元数据，即 **version-release**。

version

对应于 Red Hat OpenStack Platform 的主要和次要版本。这些版本充当包含一个或多个发行版本的流。

release

对应于版本流中特定容器镜像版本的发行版本。

例如，如果 Red Hat OpenStack Platform 的最新版本为 17.1.0，容器镜像的发行版本为 **5.161**，则生成的容器镜像标签为 17.1.0-5.161。

Red Hat Container Registry 还使用一组主要和次要 **version** 标签，链接到该容器镜像版本的最新发行版本。例如，17.1 和 17.1.0 链接到 17.1.0 容器流中的最新 **版本**。如果出现 17.1 的新次要版本，17.1 标签链接到 **新次要发行版本** 流的最新发行版本，而 17.1.0 标签则继续链接到 17.1.0 流中 **的最新发行版本**。

ContainerImagePrepare 参数包含两个子参数，可用于确定要下载的容器镜像。这些子参数是 **set** 字典中的 **tag** 参数，以及 **tag_from_label** 参数。使用以下准则来确定要使用 **tag** 还是 **tag_from_label**。

- **tag** 的默认值是您的 OpenStack Platform 版本的主要版本。对于这个版本，它是 17.1。这始终对应于最新的次要版本和发行版本。

```
parameter_defaults:
  ContainerImagePrepare:
  - set:
```

```
    ...
    tag: 17.1
    ...
```

- 要更改为 OpenStack Platform 容器镜像的特定次要版本，请将标签设置为次要版本。例如，若要更改为 17.1.2，请将 **tag** 设置为 17.1.2。

```
parameter_defaults:
  ContainerImagePrepare:
  - set:
      ...
      tag: 17.1.2
      ...
```

- 在设置 **tag** 时，director 始终会在安装和更新期间下载 **tag** 中设置的版本的最新容器镜像 **release**。

- 如果没有设置 **tag**，则 director 会结合使用 **tag_from_label** 的值和最新的主要版本。

```
parameter_defaults:
  ContainerImagePrepare:
  - set:
      ...
      # tag: 17.1
      ...
    tag_from_label: '{version}-{release}'
```

- **tag_from_label** 参数根据其从 Red Hat Container Registry 中检查到的最新容器镜像发行版本的标签元数据生成标签。例如，特定容器的标签可能会使用以下 **version** 和 **release** 元数据：

```
"Labels": {
  "release": "5.161",
  "version": "17.1.0",
  ...
}
```

- **tag_from_label** 的默认值为 **{version}-{release}**，对应于每个容器镜像的版本和发行版本元数据标签。例如，如果容器镜像为 **版本** 设置了 17.1.0，并且为 **发行版本** 设置了 5.161，则生成的容器镜像标签为 17.1.0-5.161。

- **tag** 参数始终优先于 **tag_from_label** 参数。要使用 **tag_from_label**，在容器准备配置中省略 **tag** 参数。

- **tag** 和 **tag_from_label** 之间的一个关键区别是：director 仅基于主要或次要版本标签使用 **tag** 拉取镜像，Red Hat Container Registry 将这些标签链接到版本流中的最新镜像发行版本，而 director 使用 **tag_from_label** 对每个容器镜像执行元数据检查，以便 director 生成标签并拉取对应的镜像。

2.5. 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
```

2.6. 更新 UNDERCLOUD.CONF 文件

您可以继续使用 Red Hat OpenStack Platform 16.2 环境中的原始 **undercloud.conf** 文件，但必须修改该文件以保持与 Red Hat OpenStack Platform 17.1 的兼容性。有关配置 **undercloud.conf** 文件的参数的更多信息，请参阅 *使用 director 安装和管理 Red Hat OpenStack Platform* 中的 Director 配置参数。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 创建名为 **skip_rhel_release.yaml** 的文件，并将 **SkipRhelEnforcement** 参数设置为 **true** ：

```
parameter_defaults:
  SkipRhelEnforcement: true
```

3. 打开 **undercloud.conf** 文件，并在文件中的 **DEFAULT** 部分中添加以下参数：

```
container_images_file = /home/stack/containers-prepare-parameter.yaml
custom_env_files = /home/stack/skip_rhel_release.yaml
```

   - 在 **custom_env_files** 参数中添加任何其他自定义环境文件。

custom_env_files 参数定义升级所需的 **skip_rhel_release.yaml** 文件的位置。

- **container_images_file** 参数定义 **containers-prepare-parameter.yaml** 环境文件的位置，以便 director 从正确的位置拉取 undercloud 的容器镜像。

4. 检查 文件中的所有其他参数是否有任何更改。
5. 保存该文件。

2.7. 运行 DIRECTOR 升级

在 undercloud 上升级 director。

流程

- 启动 director 配置脚本以升级 director：

```
$ openstack undercloud upgrade
```

director 配置脚本升级 director 软件包并配置 director 服务，以匹配 **undercloud.conf** 文件中的设置。这个脚本会需要一些时间来完成。

注意

在继续操作前，director 配置脚本会提示确认。使用 **-y** 选项绕过此确认：

```
$ openstack undercloud upgrade -y
```

第 3 章 准备 OVERCLOUD 升级

您必须完成一些初始步骤来准备 overcloud 升级。

3.1. 准备 OVERCLOUD 服务停机时间

overcloud 升级过程在关键点上禁用主 control plane 服务。在达到这些关键点时，您无法使用任何 overcloud 服务创建新资源。overcloud 中运行的工作负载在升级过程中保持活动状态，这意味着实例在 control plane 升级过程中继续运行。在升级 Compute 节点期间，这些工作负载可以实时迁移到已升级的 Compute 节点。

规划维护窗口，以确保在升级过程中没有用户可以访问 overcloud 服务。

受 overcloud 升级影响

- OpenStack Platform 服务

不受 overcloud 升级的影响

- 升级过程中运行的实例
- Ceph Storage OSD （实例的后端存储）
- Linux 网络
- Open vSwitch 网络
- undercloud

3.2. 在 OVERCLOUD 中禁用隔离

在升级 overcloud 之前，请确保禁用隔离。

升级 overcloud 时，您可以单独升级每个 Controller 节点来保留高可用性功能。如果在您的环境中部署了隔离，overcloud 可能会检测某些节点被禁用并尝试隔离操作，这可能会导致意外的结果。

如果您在 overcloud 中启用了隔离功能，则必须在升级期间临时禁用隔离，以避免出现意外的结果。

注意

完成 Red Hat OpenStack Platform 环境的升级后，您必须在 overcloud 中重新启用隔离。有关重新启用隔离的更多信息，请参阅 overcloud 中的重新启用隔离。

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. 登录到 Controller 节点并运行 Pacemaker 命令禁用隔离：

```
$ ssh tripleo-admin@<controller_ip> "sudo pcs property set stonith-enabled=false"
```

- 将 **<controller_ip>** 替换为 Controller 节点的 IP 地址。您可以使用 **metalsmith list** 命令查找 Controller 节点的 IP 地址。

4. 在 **fencing.yaml** 环境文件中，将 **EnableFencing** 参数设置为 **false**，以确保隔离在升级过程中保持禁用。

其它资源

- 使用 STONITH 隔离 Controller 节点

3.3. UNDERCLOUD 节点数据库备份

您可以使用 **openstack undercloud backup** 命令创建在 undercloud 节点上运行的数据库备份，并使用该备份在数据库被损坏时恢复数据库状态。有关备份 undercloud 数据库的更多信息，请参阅 *Red Hat OpenStack Platform 17.1 备份和恢复 undercloud 和 control plane 节点*中的 undercloud 节点独立数据库备份。

3.4. 在自定义 ROLES_DATA 文件中更新可组合服务

本节包含有关新和已弃用的可组合服务的信息。

所有节点

所有节点已弃用了以下服务。从所有角色中删除它们。

服务	原因
OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI **OS::TripleO::Services::CinderBackendDellPs** **OS::TripleO::Services::CinderBackendVRTSHyperScale**	弃用的服务。
OS::TripleO::Services::Ec2Api	弃用的服务。
OS::TripleO::Services::Fluentd	在 **OS::TripleO::Services::Rsyslog** 的 favour 中被弃用。
OS::TripleO::Services::FluentdAlt	弃用的服务。
OS::TripleO::Services::Keepalived	弃用的服务。
OS::TripleO::Services::MistralApi **OS::TripleO::Services::MistralEngine** **OS::TripleO::Services::MistralEventEngine** **OS::TripleO::Services::MistralExecutor**	弃用的服务。

服务	原因
OS::TripleO::Services::NeutronLbaasv2Agent **OS::TripleO::Services::NeutronLbaasv2Api**	OpenStack Networking (neutron)负载均衡作为服务在 Octavia 中被弃用。
OS::TripleO::Services::NeutronML2FujitsuCfab **OS::TripleO::Services::NeutronML2FujitsuFossw**	弃用的服务。
OS::TripleO::Services::NeutronSriovHostConfig	弃用的服务。
OS::TripleO::Services::NovaConsoleauth	此服务已被删除。
OS::TripleO::Services::Ntp	弃用了 **OS::TripleO::Services::Timesync**。
OS::TripleO::Services::OpenDaylightApi **OS::TripleO::Services::OpenDaylightOvs**	Daylight 不再被支持。
OS::TripleO::Services::OpenShift::GlusterFS **OS::TripleO::Services::OpenShift::Infra** **OS::TripleO::Services::OpenShift::Master** **OS::TripleO::Services::OpenShift::Worker**	弃用的服务。
OS::TripleO::Services::PankoApi	OpenStack Telemetry 服务已弃用，而是使用 Service Telemetry Framework (STF)进行指标和监控。传统的遥测服务仅在 RHOSP 17.1 中提供，以帮助促进 STF 的转换，并将在以后的 RHOSP 版本中删除。
OS::TripleO::Services::Rear	弃用的服务。
OS::TripleO::Services::SaharaApi **OS::TripleO::Services::SaharaEngine**	弃用的服务。
OS::TripleO::Services::SensuClient **OS::TripleO::Services::SensuClientAlt**	弃用的服务。
OS::TripleO::Services::SkydiveAgent **OS::TripleO::Services::SkydiveAnalyzer**	Skydive 不再被支持。

服务	原因
OS::TripleO::Services::Tacker	Tacker 不再被支持。
OS::TripleO::Services::TripleoUI	弃用的服务。
OS::TripleO::Services::UndercloudMinionMessaging **OS::TripleO::Services::UndercloudUpgradeEphemeralHeat**	弃用的服务。
OS::TripleO::Services::Zaqar	弃用的服务。

Controller 节点

以下服务是 Controller 节点的新服务。将它们添加到 Controller 角色中。

服务	原因
OS::TripleO::Services::GlanceApiInternal	用于镜像服务(glance) API 的内部实例的服务，为需要它的管理员和服务提供位置数据，如块存储服务(cinder)和计算服务(nova)。

Compute 节点

默认情况下，Compute 角色运行 **OS::TripleO::Services::NovaLibvirt** 服务。将此服务替换为 **OS::TripleO::Services::NovaLibvirtLegacy** 服务。

3.5. 网络配置文件转换

如果您的网络配置模板包含以下功能，则必须在升级 overcloud 前手动将 NIC 模板转换为 Jinja2 Ansible 格式。自动转换不支持以下功能：

- **'get_file'**
- **'get_resource'**
- **'digest'**
- **'repeat'**
- **'resource_facade'**
- **'str_replace'**
- **'str_replace_strict'**
- **'str_split'**
- **'map_merge'**

- **'map_replace'**
- **'yaql'**
- **'equals'**
- **'if'**
- **'not'**
- **'and'**
- **'or'**
- **'filter'**
- **'make_url'**
- **'contains'**

有关手动转换 NIC 模板的更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 中的手动将 NIC 模板转换为 Jinja2 Ansible 格式 。

3.6. 检查自定义 PUPPET 参数

如果您使用 **ExtraConfig** 接口来自定义 Puppet 参数，则 Puppet 可能会在升级过程中报告重复声明错误。这是因为 puppet 模块本身提供的接口有变化。

此流程演示了如何检查环境文件中的任何自定义 **ExtraConfig** hieradata 参数。

流程

1. 选择一个环境文件，检查它是否有 **ExtraConfig** 参数：

```
$ grep ExtraConfig ~/templates/custom-config.yaml
```

2. 如果结果在所选文件中显示 **ExtraConfig** 参数（如 **ControllerExtraConfig**），请检查该文件中的完整参数结构。

3. 如果参数包含 **SECTION/parameter** 语法后跟一个 **值** 的任何 puppet Hierdata，则它可能已被替换为实际 Puppet 类的参数。例如：

```
parameter_defaults:
  ExtraConfig:
    neutron::config::dhcp_agent_config:
      'DEFAULT/dnsmasq_local_resolv':
        value: 'true'
```

4. 检查 director 的 Puppet 模块，以查看参数现在存在于 Puppet 类中。例如：

```
$ grep dnsmasq_local_resolv
```

如果是，请更改为新接口。

5. 以下是演示语法更改的示例：

- 示例 1：

```
parameter_defaults:
  ExtraConfig:
    neutron::config::dhcp_agent_config:
      'DEFAULT/dnsmasq_local_resolv':
        value: 'true'
```

更改：

```
parameter_defaults:
  ExtraConfig:
    neutron::agents::dhcp::dnsmasq_local_resolv: true
```

- 示例 2：

```
parameter_defaults:
  ExtraConfig:
    ceilometer::config::ceilometer_config:
      'oslo_messaging_rabbit/rabbit_qos_prefetch_count':
        value: '32'
```

更改：

```
parameter_defaults:
  ExtraConfig:
    oslo::messaging::rabbit::rabbit_qos_prefetch_count: '32'
```

3.7. 升级前进行最后的检查

在开始升级前，请对所有准备步骤进行最终检查。

3.7.1. upgrade 命令概述

升级过程涉及您在进程的特定阶段运行的不同命令。

重要

本节仅包含每个命令的信息。您必须以特定顺序运行这些命令，并提供特定于 overcloud 的选项。等待您收到说明，以便在适当的步骤中运行这些命令。

3.7.1.1. OpenStack overcloud 升级准备

此命令执行 overcloud 升级的初始准备步骤，其中包括将 undercloud 上的当前 overcloud 计划替换为新的 OpenStack Platform 17.1 overcloud 计划和更新的环境文件。此命令的功能与 **openstack overcloud deploy** 命令类似，并使用许多相同的选项。

在运行 **openstack overcloud upgrade prepare** 命令前，您必须执行 overcloud 采用。有关 overcloud 采用的更多信息，请参阅 执行 overcloud 采用。

3.7.1.2. OpenStack overcloud 升级运行

此命令执行升级过程。director 根据新的 OpenStack Platform 17.1 overcloud 计划创建一组 Ansible playbook，并在整个 overcloud 上运行快速转发任务。这包括通过 16.2 到 17.1 的每个 OpenStack Platform 版本运行升级过程。

除了标准升级过程外，这个命令还可以对 overcloud 节点上的操作系统执行 Leapp 升级。使用 **--tags** 选项运行这些任务。

为 Leapp 升级任务标签

system_upgrade

合并了 **system_upgrade_prepare**、**system_upgrade_run** 和 **system_upgrade_reboot** 的任务。

system_upgrade_prepare

使用 Leapp 准备操作系统升级的任务。

system_upgrade_run

运行 Leapp 和升级操作系统的任务。

system_upgrade_reboot

重启系统并完成操作系统升级的任务。

3.7.1.3. OpenStack overcloud external-upgrade run

此命令在标准升级过程外执行升级任务。director 根据新的 OpenStack Platform 17.1 overcloud 计划创建一组 Ansible playbook，并使用 **--tags** 选项运行特定的任务。

用于容器管理的外部任务标签

container_image_prepare

将容器镜像拉取到 undercloud registry 的任务，并为 overcloud 使用准备镜像。

3.7.2. 升级参数

您可以使用升级参数修改升级过程的行为。

参数	描述
UpgradeInitCommand	在所有 overcloud 节点上运行的命令或脚本片段以初始化升级过程。例如，仓库开关。
UpgradeInitCommonCommand	升级过程所需的常用命令。这通常不应该由 Operator 修改，并在 major-upgrade-composable-steps.yaml 和 major-upgrade-converge.yaml 环境文件中设置和取消设置。
UpgradeLeappCommandOptions	附加到 Leapp 命令的其他命令行选项。
UpgradeLeappDebug	运行 Leapp 时打印调试输出。默认值为 **true**。
UpgradeLeappDevelSkip	在 development/testing 中运行 Leapp 时，通过设置 env 变量跳过 Leapp 检查。例如：LEAPP_DEVEL_SKIP_RHSM=1。

参数	描述
UpgradeLeappEnabled	使用 Leapp 进行操作系统升级。默认值为 **false**。
UpgradeLeappPostRebootDelay	等待机器重启并响应 test 命令的最大（秒）。默认值为 **120**。
UpgradeLeappRebootTimeout	通过 Leapp 进行 OS 升级阶段的超时（秒）。默认值为 **3600**。
UpgradeLeappToInstall	Leapp 升级后要安装的软件包列表。
UpgradeLeappToRemove	Leapp 升级过程中要删除的软件包列表。

3.7.3. 部署中包含的自定义文件

如果部署中的任何 overcloud 节点都是专用的 Object Storage (swift)节点，您必须复制默认的 **roles_data.yaml** 文件，并编辑 **ObjectStorage** 以删除 **deprecated_server_resource_name: 'SwiftStorage'**。然后使用 **--roles-file** 选项将文件传递到 **openstack overcloud upgrade prepare** 命令。

3.7.4. 要包括在部署中的新环境文件

除了常规的 overcloud 环境文件外，还必须包含新的环境文件，以便于升级到 Red Hat OpenStack Platform (RHOSP) 17.1。

File	备注
/home/stack/templates/upgrades-environment.yaml	此文件包含特定于升级的参数。此文件只在升级的持续时间内是必需的。
/home/stack/containers-prepare-parameter.yaml	包含源和准备步骤的文件。这是与 undercloud 升级一起使用的相同文件。
/home/stack/templates/ceph.yaml	此文件包含 Ceph Storage 覆盖所需的参数。

运行以下命令，将这些文件添加到环境文件列表的末尾：

- **OpenStack overcloud 升级准备**
- **OpenStack overcloud 部署**

3.7.5. 从部署中删除的环境文件

删除特定于 OpenStack Platform Red Hat OpenStack Platform 16.2 的任何环境文件：

- Red Hat OpenStack Platform 16.2 容器镜像列表
- Red Hat OpenStack Platform 16.2 客户门户网站或 Satellite **rhel-registration** 脚本

运行以下命令，从包含的环境文件列表中删除这些文件：

- **OpenStack overcloud 升级准备**
- **OpenStack overcloud 部署**

3.7.6. 升级 IPA 服务

如果环境中无处启用了 TLS，请为 Nova Host Manager 角色添加额外的权限，以允许创建 DNS 区域条目。

先决条件

检查您的环境中是否包含 Nova 主机管理权限：

```
$ ipa privilege-show "Nova Host Management"
```

如果您已有这个权限，请跳过以下步骤。

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. 添加 **Nova 主机管理权限** ：

```
$ kinit admin
$ ipa privilege-add-permission 'Nova Host Management' --permission 'System: Modify Realm Domains'
```

4. 创建名为 **ipa_environment.yaml** 的环境文件，并包含以下配置：

```
resource_registry:
  OS::TripleO::Services::IpaClient: /usr/share/openstack-tripleo-heat-templates/deployment/ipa/ipaservices-baremetal-ansible.yaml

parameter_defaults:
  IdMServer: $IPA_FQDN
  IdMDomain: $IPA_DOMAIN
  IdMInstallClientPackages: False
```

5. 保存环境文件。

3.7.7. 升级清单

使用以下清单来确定升级 overcloud 的就绪情况：

项	complete
验证正常工作的 overcloud。	Y / N

项	complete
对 overcloud control plane 执行 Relax-and-Recover (ReaR)备份。如需更多信息，请参阅 Red Hat OpenStack Platform 16.2 备份和恢复 undercloud 和 control plane 节点	Y / N
创建在 undercloud 节点上运行的数据库的备份。如需更多信息，请参阅 *Red Hat OpenStack Platform 17.1 备份和恢复 undercloud 和 control plane 节点*中的创建 undercloud 节点备份。	Y / N
更新了 Red Hat OpenStack Platform 17.1 存储库的注册详情，并将环境文件转换为使用基于 Ansible 的方法。	Y / N
更新了网络配置模板。	Y / N
使用 Red Hat OpenStack Platform 17.1 的新环境文件更新了您的环境文件列表。	Y / N
可选： 如果您的部署包含专用 Object Storage (swift)节点： 复制 **roles_data.yaml** 文件，删除了 **deprecated_server_resource_name: 'SwiftStorage'**，并将该文件传递给 **openstack overcloud upgrade prepare** 命令。	Y / N
删除了仅与 Red Hat OpenStack Platform 16.2 相关的旧环境文件，如旧的 Red Hat registration 和容器镜像位置文件。	Y / N

第 4 章 执行 OVERCLOUD 采用

overcloud 采用涉及以下任务：

- 在每个堆栈上，将网络和主机调配配置导出到 overcloud。
- 定义新容器和其他兼容性配置。

4.1. 运行 OVERCLOUD 升级准备

升级需要运行 **openstack overcloud upgrade prepare** 命令，该命令执行以下任务：

- 将 overcloud 计划更新至 OpenStack Platform 17.1
- 为升级准备节点

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. 验证 undercloud 升级过程中导出的以下文件是否包含 overcloud 升级的预期配置。您可以在 **~/overcloud-deploy** 目录中找到以下文件：
   - **tripleo-<stack>-passwords.yaml**
   - **tripleo-<stack>-network-data.yaml**
   - **tripleo-<stack>-virtual-ips.yaml**
   - **tripleo-<stack>-baremetal-deployment.yaml**

注意

如果在 undercloud 升级后没有生成这些文件，请联系红帽支持。

4. 在主堆栈上，将 password **.yaml** 文件复制到 **~/overcloud-deploy/$(<stack>)** 目录。在环境中的每个堆栈中重复此步骤：

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-passwords.yaml ~/overcloud-
deploy/<stack>/<stack>-passwords.yaml
```

   - 将 **<stack>** 替换为您的堆栈的名称。

5. 在主堆栈上，将 **network-data.yaml** 文件复制到 stack 用户的主目录并部署网络。在环境中的每个堆栈中重复此步骤：

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-network-data.yaml ~/
$ mkdir ~/overcloud_adopt
$ openstack overcloud network provision --debug \
```

```
--output /home/stack/overcloud_adopt/generated-networks-deployed.yaml tripleo-<stack>-network-data.yaml
```

有关更多信息，请参阅*使用 director 安装和管理 Red Hat OpenStack Platform* 中的置备和部署 overcloud。

6. 在主堆栈上，将 **virtual-ips.yaml** 文件复制到 stack 用户的主目录，并置备网络 VIP。在环境中的每个堆栈中重复此步骤：

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-virtual-ips.yaml ~/
$ openstack overcloud network vip provision --debug \
--stack <stack> --output \
/home/stack/overcloud_adopt/generated-vip-deployed.yaml tripleo-<stack>-virtual-ips.yaml
```

7. 在主堆栈上，将 **baremetal-deployment.yaml** 文件复制到 stack 用户的主目录，并置备 overcloud 节点。在环境中的每个堆栈中重复此步骤：

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-baremetal-deployment.yaml ~/
$ openstack overcloud node provision --debug --stack <stack> \
--output /home/stack/overcloud_adopt/baremetal-deployment.yaml \
tripleo-<stack>-baremetal-deployment.yaml
```

8. 完成以下步骤以准备容器：

   a. 备份用于 undercloud 升级的 **containers-prepare-parameter.yaml** 文件：

```
$ cp containers-prepare-parameter.yaml \
containers-prepare-parameter.yaml.orig
```

   b. 在运行脚本以更新 **containers-prepare-parameter.yaml** 文件前，定义以下环境变量：

   - **NAMESPACE**: UBI9 镜像的命名空间。例如，**NAMESPACE='"namespace":"example.redhat.com:5002",'**
   - **EL8_NAMESPACE** ：UBI8 镜像的命名空间。
   - **NEUTRON_DRIVER** ：要使用的驱动程序，并确定要使用的 OpenStack Networking (neutron)容器。设置为用于部署原始堆栈的容器类型。例如，设置为 **NEUTRON_DRIVER='"neutron_driver":"ovn",'** 以使用基于 OVN 的容器。
   - **EL8_TAGS**: UBI8 镜像的标签，例如 **EL8_TAGS='"tag":"rhel_8_rhosp17.1",'**。
   - **EL9_TAGS**: UBI9 镜像的标签，例如 **EL9_TAGS='"tag":"rhel_9_rhosp17.1",'**。
   - **CONTROL_PLANE_ROLES** ：使用 **--role** 选项的 control plane 角色列表，如 **--role ControllerOpenstack, --role Database, --role Messaging, --role Networker, --role CephStorage**。要查看环境中的 control plane 角色列表，请运行以下命令：

```
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/$STACK/tripleo-ansible-inventory.yaml \
| grep -vi compute
```

   - **COMPUTE_ROLES** ：使用 **--role** 选项的 Compute 角色列表，如 **--Compute-1**。要查看环境中的 Compute 角色列表，请运行以下命令：

```
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/$STACK/tripleo-ansible-inventory.yaml \
| grep -i compute
```

- **CEPH_TAGS** ：如果您部署 Red Hat Ceph Storage，请指定 Red Hat Ceph Storage 5 容器镜像。例如：" **ceph_tag":"5-404"**。

c. 运行以下脚本以更新 **containers-prepare-parameter.yaml** 文件：

```
$ python3 /usr/share/openstack-tripleo-heat-templates/tools/multi-rhel-container-image-prepare.py \
    ${COMPUTE_ROLES} \
    ${CONTROL_PLANE_ROLES} \
    --enable-multi-rhel \
    --excludes collectd \
    --excludes nova-libvirt \
    --minor-override "{${EL8_TAGS}${EL8_NAMESPACE}${CEPH_TAGS}${NEUTRON_DRIVER}\"no_tag\":\"not_used\"}" \
    --major-override "{${EL9_TAGS}${NAMESPACE}${CEPH_TAGS}${NEUTRON_DRIVER}\"no_tag\":\"not_used\"}" \
    --output-env-file \
   /home/stack/containers-prepare-parameter.yaml
```

9. 如果使用 Red Hat Ceph Storage，请创建一个名为 **ceph_params.yaml** 的文件，并包含以下内容：

```
parameter_defaults:
  CephSpecFqdn: true
  CephConfigPath: "/etc/ceph"
  CephAnsibleRepo: "rhceph-5-tools-for-rhel-8-x86_64-rpms"
  DeployedCeph: true
```

注意

如果您的 Red Hat Ceph Storage 部署包含短名称，您必须将 **CephSpecFqdn** 参数设置为 **false**。如果设置为 **true**，则清单会生成一个短名称和域名，从而导致 Red Hat Ceph Storage 升级失败。

10. 在 templates 目录中创建一个名为 **upgrade-environment.yaml** 的环境文件，并包含以下内容：

```
parameter_defaults:
  ExtraConfig:
    nova::workarounds::disable_compute_service_check_for_ffu: true
  DnsServers: ["<dns_servers>"]
    DockerInsecureRegistryAddress: <undercloud_FQDN>
  UpgradeInitCommand: |
    sudo subscription-manager repos --disable=*
      if $( grep -q  9.2  /etc/os-release )
      then
        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
        sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-
tools-for-rhel-9-x86_64-rpms
        sudo subscription-manager release --set=9.2
      else
        sudo subscription-manager repos --enable=rhel-8-for-x86_64-baseos-tus-rpms --
enable=rhel-8-for-x86_64-appstream-tus-rpms --enable=rhel-8-for-x86_64-highavailability-
tus-rpms --enable=openstack-17.1-for-rhel-8-x86_64-rpms --enable=fast-datapath-for-rhel-8-
x86_64-rpms
        sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-
tools-for-rhel-8-x86_64-rpms
        sudo subscription-manager release --set=8.4
      fi

      if $(sudo podman ps | grep -q ceph )
      then
        sudo dnf -y install cephadm
      fi
```

- 将 **<dns_servers>** 替换为 DNS 服务器 IP 地址的逗号分隔列表，例如 **["10.0.0.36", "10.0.0.37"]**。

- 将 **<undercloud_FQDN** > 替换为 undercloud 主机的完全限定域名(FQDN)，例如 **"undercloud-0.ctlplane.redhat.local:8787"**。
  有关您可以在环境文件中配置的升级参数的更多信息，请参阅升级 参数。

11. 在 undercloud 上，在 templates 目录中创建一个名为 **overcloud_upgrade_prepare.sh** 的文件。您必须为环境中的每个堆栈创建此文件。此文件包含 overcloud 部署文件的原始内容，以及与您的环境相关的环境文件。例如：

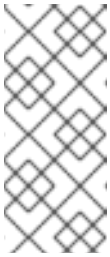
```
#!/bin/bash
openstack overcloud upgrade prepare --yes \
  --timeout 460 \
  --templates /usr/share/openstack-tripleo-heat-templates \
  --ntp-server 192.168.24.1 \
  --stack <stack> \
  -r /home/stack/roles_data.yaml \
  -e /home/stack/templates/internal.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  -e /home/stack/templates/network/network-environment.yaml \
  -e /home/stack/templates/inject-trust-anchor.yaml \
  -e /home/stack/templates/hostnames.yml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/nodes_data.yaml \
  -e /home/stack/templates/debug.yaml \
  -e /home/stack/templates/firstboot.yaml \
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/overcloud-params.yaml \
  -e /home/stack/overcloud-deploy/overcloud/overcloud-network-environment.yaml \
  -e /home/stack/overcloud_adopt/baremetal-deployment.yaml \
  -e /home/stack/overcloud_adopt/generated-networks-deployed.yaml \
  -e /home/stack/overcloud_adopt/generated-vip-deployed.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-hw-machine-type-
```

```
upgrade.yaml \
 -e /home/stack/skip_rhel_release.yaml \
 -e ~/containers-prepare-parameter.yaml
```

a. 在原始 **network-environment.yaml** 文件中(**/home/stack/templates/network/network-environment.yaml**)，删除所有指向 **OS::TripleO::*::Net::SoftwareConfig** 的 resource_registry 资源。

b. 在 **overcloud_upgrade_prepare.sh** 文件中，包含与您的环境相关的以下选项：

- 环境文件 (**upgrades-environment.yaml**) 带有特定于升级的参数 (**-e**)。
- 包含新容器镜像位置(**-e**)的环境文件(**containers-prepare-parameter.yaml**)。在大多数情况下，这是 undercloud 使用的环境文件。
- 环境文件(**skip_rhel_release.yaml**)，带有发行版本参数(-e)。
- 与您的部署相关的任何自定义配置文件(**-e**)。
- 如果适用，使用 **--roles-file** 的自定义角色(**roles_data**)文件。
- 对于 Ceph 部署，环境文件(**ceph_params.yaml**)以及 Ceph 参数(-e)。
- 在 overcloud 采用过程中生成的文件(network**-deployed.yaml**,**vip-deployed.yaml**,**baremetal-deployment.yaml**) (-e)。
- 如果适用，带有您的 IPA 服务(-e)的环境文件(**ipa-environment.yaml**)。
- 如果使用自定义堆栈名称，请使用 **--stack** 选项传递名称。



注意

您必须在模板中包含 **nova-hw-machine-type-upgrade.yaml** 文件，直到所有 RHEL 8 Compute 节点都升级到环境中的 RHEL 9。如果排除了此文件，则 **/var/log/containers/nova** 目录中的 **nova_compute.log** 中会出现一个错误。将所有 RHEL 8 Compute 节点升级到 RHEL 9 后，您可以从配置中删除此文件并更新堆栈。

c. 如果您在要升级的部署中通过 NFS 启用共享文件系统服务(manila)，则必须在 **overcloud_upgrade_prepare.sh** 脚本文件末尾指定额外的环境文件。您必须在脚本末尾添加环境文件，因为它会覆盖之前在脚本中指定的另一个环境文件：

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/manila-cephfsganesha-config.yaml
```

12. 为环境中的每个堆栈运行升级准备命令：

```
$ source stackrc
$ chmod 755 /home/stack/overcloud_upgrade_prepare.sh
$ sh /home/stack/overcloud_upgrade_prepare.sh
```

13. 等待升级准备完成。

14. 下载容器镜像：

```
$ openstack overcloud external-upgrade run --stack <stack> --tags
container_image_prepare
```

第 5 章 使用 DIRECTOR 部署的 CEPH 部署升级 OVERCLOUD

如果您的环境包含 director 部署的 Red Hat Ceph Storage 部署，您必须将部署升级到 Red Hat Ceph Storage 5。随着版本 5 的升级，**cephadm** 现在管理 Red Hat Ceph Storage，而不是 **ceph-ansible**。

5.1. 安装 CEPH-ANSIBLE

如果使用 director 部署 Red Hat Ceph Storage，您必须完成此步骤。使用 Red Hat OpenStack Platform 升级 Red Hat Ceph Storage 需要 **ceph-ansible** 软件包。

流程

1. 启用 Ceph 5 Tools 存储库：

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhceph-5-tools-for-rhel-8-x86_64-rpms
```

2. 安装 **ceph-ansible** 软件包：

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

5.2. 升级到 RED HAT CEPH STORAGE 5

将 Red Hat Ceph Storage 节点从版本 4 升级到 5 版本。

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 **ceph** 标签运行 Red Hat Ceph Storage 外部升级过程：

```
$ openstack overcloud external-upgrade run \
  --skip-tags "ceph_health,opendev-validation, ceph_ansible_remote_tmp" \
  --stack <stack> \
  --tags ceph,facts 2>&1
```

   - 将 **<stack>** 替换为您的堆栈的名称。

4. 创建 ceph-admin 用户并分发适当的密钥环：

```
ANSIBLE_LOG_PATH=/home/stack/cephadm_enable_user_key.log \
ANSIBLE_HOST_KEY_CHECKING=false \
ansible-playbook -i /home/stack/overcloud-deploy/<stack>/config-download/<stack>/tripleo-ansible-inventory.yaml \
  -b -e ansible_python_interpreter=/usr/libexec/platform-python /usr/share/ansible/tripleo-playbooks/ceph-admin-user-playbook.yml \
 -e tripleo_admin_user=ceph-admin \
 -e distribute_private_key=true \
  --limit ceph_osd,ceph_mon,Undercloud
```

5. 更新 Red Hat Ceph Storage 节点上的软件包：

```
$ openstack overcloud upgrade run \
    --stack <stack> \
    --skip-tags ceph_health,opendev-validation,ceph_ansible_remote_tmp \
    --tags setup_packages --limit ceph_osd,ceph_mon,Undercloud \
    --playbook /home/stack/overcloud-deploy/<stack>/config-
download/<stack>/upgrade_steps_playbook.yaml 2>&1
```

注意

Ceph 监控服务(CephMon)在 Controller 节点上运行。此命令包含 **ceph_mon** 标签，它也会更新 Controller 节点上的软件包。

6. 配置 Red Hat Ceph Storage 节点以使用 **cephadm** ：

```
$ openstack overcloud external-upgrade run \
    --skip-tags ceph_health,opendev-validation,ceph_ansible_remote_tmp \
    --stack <stack> \
    --tags cephadm_adopt  2>&1
```

7. 修改 **overcloud_upgrade_prepare.sh** 文件，将 **ceph-ansible** 文件替换为 **cephadm** heat 环境文件。

```
#!/bin/bash
openstack overcloud upgrade prepare  --yes \
  --timeout 460 \
 --templates /usr/share/openstack-tripleo-heat-templates \
  --ntp-server 192.168.24.1 \
  --stack <stack> \
  -r /home/stack/roles_data.yaml \
  -e /home/stack/templates/internal.yaml \
  …
  -e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-
only.yaml \
  -e ~/containers-prepare-parameter.yaml
```

注意

本例使用 **environments/cephadm/cephadm-rbd-only.yaml** 文件，因为没有部署 RGW。如果您计划部署 RGW，请在升级 RHOSP 环境后使用 **environments/cephadm/cephadm.yaml**，然后运行堆栈更新。

8. 如果您之前在运行 overcloud 升级准备时添加了 **overcloud_upgrade_prepare.sh** 文件，请修改 overcloud_upgrade_prepare.sh 文件，以删除以下环境文件：

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/manila-
cephfsganesha-config.yaml
```

9. 保存该文件。

10. 运行 upgrade preparation 命令：

```
$ source stackrc
$ chmod 755 /home/stack/overcloud_upgrade_prepare.sh
sh /home/stack/overcloud_upgrade_prepare.sh
```

Red Hat Ceph Storage 集群现在升级到版本 5。这会产生以下影响：

- 您不再使用 **ceph-ansible** 管理 Red Hat Ceph Storage。相反，Ceph 编排器管理 Red Hat Ceph Storage 集群。有关 Ceph 编排器的更多信息，请参阅 Ceph 操作指南。
- 在大多数情况下，您不再需要执行堆栈更新来更改 Red Hat Ceph Storage 集群。相反，您可以直接在集群上运行第 2 天的 Red Hat Ceph Storage 操作，如 Ceph 操作指南中所述。您还可以扩展 Red Hat Ceph Storage 集群节点，如 *部署 Red Hat Ceph Storage 和 Red Hat OpenStack Platform* 中的扩展 Ceph Storage 集群 中所述。
- 检查 Red Hat Ceph Storage 集群的健康状态。如需有关监控集群健康状况的更多信息，请参阅 *部署 Red Hat Ceph Storage 和 Red Hat OpenStack Platform* 中的监控 Red Hat Ceph Storage 节点。
- 不要在 openstack deployment 命令（如 **openstack overcloud deploy**）中包含环境文件，如 **environments/ceph-ansible/ceph-ansible.yaml**。如果您的部署包含 **ceph-ansible** 环境文件，请将它们替换为以下选项之一：

Red Hat Ceph Storage 部署	原始 ceph-ansible 文件	Cephadm 文件替换
仅限 Ceph RADOS 块设备 (RBD)	任何 **ceph-ansible** 环境文件	**environments/cephadm/cephadm-rbd-only.yaml**
RBD 和 Ceph 对象网关(RGW)	任何 **ceph-ansible** 环境文件	**environments/cephadm/cephadm.yaml**
Ceph 仪表板	**environments/ceph-ansible/ceph-dashboard.yaml**	**environments/cephadm/** 中的相应文件
Ceph MDS	**environments/ceph-ansible/ceph-mds.yaml**	**environments/cephadm/** 中的相应文件

第 6 章 准备网络功能虚拟化 (NFV)

如果使用网络功能虚拟化 (NFV)，您必须完成一些准备 overcloud 升级。

6.1. 网络功能虚拟化(NFV)环境文件

在典型的 NFV 环境中，您可以启用服务，如下所示：

- 单根输入/输出虚拟化(SR-IOV)
- 数据平面开发套件(DPDK)

您不需要对这些服务进行任何特定的重新配置，以适应升级到 Red Hat OpenStack Platform 17.1。但是，请确保启用 NFV 功能的环境文件满足以下要求：

- 启用 NFV 功能的默认环境文件位于 Red Hat OpenStack Platform 17.1 **openstack-tripleo-heat-templates** 的 **environments/services** 目录中。如果您在 Red Hat OpenStack Platform 16.2 部署中包含 **openstack-tripleo-heat-templates** 中的默认 NFV 环境文件，请验证 Red Hat OpenStack Platform 17.1 中相应功能的正确环境文件位置：
  - Open vSwitch (OVS)网络和 SR-IOV: **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml**
  - Open vSwitch (OVS)网络和 DPDK: **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-dpdk.yaml**
- 要在从 Red Hat OpenStack Platform 16.2 升级到 Red Hat OpenStack Platform 17.1 期间维护 OVS 兼容性，您必须包含 **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml** 环境文件。在运行涉及环境文件的部署和升级命令时，您必须在 **neutron-ovs.yaml** 文件后包含任何与 NFV 相关的环境文件。例如，在运行 **openstack overcloud upgrade** 准备使用 OVS 和 NFV 环境文件时，按以下顺序包含文件：
- OVS 环境文件
- SR-IOV 环境文件
- DPDK 环境文件

```
$ openstack overcloud upgrade prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-dpdk.yaml \
  ...
```

注意

NFV 工作负载的迁移限制： 在升级过程中无法从 OVS-DPDK Compute 节点实时迁移实例。或者，您可以在升级过程中从 OVS-DPDK Compute 节点冷迁移实例。

第 7 章 升级 OVERCLOUD

在环境中的每一堆栈上升级整个 overcloud 的 Red Hat OpenStack Platform 内容。

7.1. 在每个堆栈中的所有节点上升级 RHOSP

从主堆栈开始，将每个堆栈的所有 overcloud 节点升级到 Red Hat OpenStack Platform (RHOSP) 17.1。

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. 在主堆栈中的所有节点上升级 RHOSP：

```
$ openstack overcloud upgrade run --yes --stack <stack> --debug --limit
allovercloud,undercloud --playbook all
```

- 将 <stack> 替换为您要升级节点的 overcloud 堆栈的名称。
  对 RHOSP 部署中的每个堆栈重复此步骤。

第 8 章 升级 UNDERCLOUD 操作系统

您必须将 undercloud 操作系统从 Red Hat Enterprise Linux 8.4 升级到 Red Hat Enterprise Linux 9.2。系统升级执行以下任务：

- 确保网络接口命名在系统升级后保持不变
- 使用 Leapp 升级 RHEL 原位升级
- 重启 undercloud

8.1. 在 UNDERCLOUD 上设置 SSH ROOT 权限参数

Leapp 升级会检查 **/etc/ssh/sshd_config** 文件中是否存在 **PermitRootLogin** 参数。您必须将此参数明确设置为 **yes** 或 **no**。

为安全起见，将此参数设置为 **no**，以禁用对 undercloud 上 root 用户的 SSH 访问。

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. 检查 **PermitRootLogin** 参数的 **/etc/ssh/sshd_config** 文件：

```
$ sudo grep PermitRootLogin /etc/ssh/sshd_config
```

3. 如果参数不在 **/etc/ssh/sshd_config** 文件中，编辑该文件并设置 **PermitRootLogin** 参数：

```
PermitRootLogin no
```

4. 保存该文件。

8.2. 验证 SSH 密钥大小

Red Hat Enterprise Linux (RHEL) 9.1 至少需要 SSH 密钥大小为 2048 位。您必须检查 Red Hat OpenStack Platform (RHOSP) director 上的当前 SSH 密钥是否至少为 2048 位。如果您没有满足 RHEL 9.x 要求的 SSH 密钥，您可能会丢失对 overcloud 的访问。

如果您的 SSH 密钥大小在所需的 2048 位下，则必须完成这个步骤。

流程

1. 验证您的 SSH 密钥大小：

```
ssh-keygen -l -f ~/.ssh/id_rsa.pub
```

输出示例：

```
1024 SHA256:Xqz0Xz0/aJua6B3qRD7VsLr6n/V3zhmnGSkcFR6FlJw
stack@director.example.local (RSA)
```

2. 如果您的 SSH 密钥小于 2048 位，请在继续操作前使用 **ssh_key_rotation.yaml** ansible playbook 替换 SSH 密钥。

```
ansible-playbook \
-i ~/overcloud-deploy/<stack>/tripleo-ansible-inventory.yaml \
/usr/share/ansible/tripleo-playbooks/ssh_key_rotation.yaml \
--extra-vars "keep_old_key_authorized_keys=<true|false> backup_folder_path=
</path/to/backup_folder>"
```

- 将 **<stack** > 替换为 overcloud 堆栈的名称。
- 根据您的备份需求替换 &lt **;true|false** >。如果没有包含此变量，则默认值为 **true**。
- 将 **</path/to/backup_folder&** gt; 替换为备份路径。如果没有包含此变量，则默认值为 **/home/{{ ansible_user_id }}/backup_keys/{{ ansible_date_time.epoch }}**

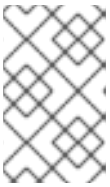
8.3. 执行 UNDERCLOUD 系统升级

将 undercloud 操作系统升级到 Red Hat Enterprise Linux (RHEL) 9.2。作为此升级的一部分，您可以创建一个名为 **system_upgrade.yaml** 的文件，您可以使用它来启用适当的软件仓库以及所需的 Red Hat OpenStack Platform 选项和内容来安装 Leapp。您可以使用此文件同时升级 control plane 节点和 Compute 节点。

流程

1. 以 **stack** 用户身份登录 undercloud。
2. 在 templates 目录中创建一个名为 **system_upgrade.yaml** 的文件，并包含以下内容：

```
parameter_defaults:
  UpgradeLeappDevelSkip: "LEAPP_UNSUPPORTED=1
LEAPP_DEVEL_SKIP_CHECK_OS_RELEASE=1 LEAPP_NO_NETWORK_RENAMING=1
LEAPP_DEVEL_TARGET_RELEASE=9.2"
  UpgradeLeappDebug: false
  UpgradeLeappEnabled: true
  LeappActorsToRemove:
['checkifcfg','persistentnetnamesdisable','checkinstalledkernels','biosdevname']
  LeappRepoInitCommand: |
     sudo subscription-manager repos --disable=*
     subscription-manager repos --enable rhel-8-for-x86_64-baseos-tus-rpms --enable rhel-8-
for-x86_64-appstream-tus-rpms --enable openstack-17.1-for-rhel-8-x86_64-rpms
      subscription-manager release --set=8.4
  UpgradeLeappCommandOptions:
"--enablerepo=rhel-9-for-x86_64-baseos-eus-rpms --enablerepo=rhel-9-for-x86_64-
appstream-eus-rpms --enablerepo=rhel-9-for-x86_64-highavailability-eus-rpms --
enablerepo=openstack-17.1-for-rhel-9-x86_64-rpms --enablerepo=fast-datapath-for-rhel-9-
x86_64-rpms"
```



注意

如果您的部署包含 Red Hat Ceph Storage 节点，您必须添加 **CephLeappRepoInitCommand** 参数，并指定 Red Hat Ceph Storage 节点的源操作系统版本。例如：

```
CephLeappRepoInitCommand:
...
  subscription-manager release --set=8.6
```

3. 在 **system_upgrade.yaml** 文件中添加 **LeappInitCommand** 参数，以指定适用于您的环境的额外要求，例如，如果您需要定义基于角色的覆盖：

```
  LeappInitCommand: |
    sudo subscription-manager repos --disable=*
    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

    leapp answer --add --section check_vdo.confirm=True

    dnf -y remove irb
```

4. 如果您使用基于内核的 NIC 名称，请在 **system_upgrade.yaml** 文件中添加以下参数，以确保在整个升级过程中保留 NIC 名称：

```
parameter_defaults:
  NICsPrefixesToUdev: ['en']
...
```

5. 运行 Leapp 升级：

```
$ openstack undercloud upgrade --yes --system-upgrade \
/home/stack/system_upgrade.yaml
```

注意

如果您需要再次运行 Leapp 升级，您必须首先将软件仓库重置为 RHEL 8。

6. 重新引导 undercloud：

```
$ sudo reboot
```

第 9 章 升级 CONTROL PLANE 操作系统

升级 control plane 节点上的操作系统。升级包括以下任务：

- 使用系统升级参数运行 overcloud upgrade prepare 命令
- 运行 overcloud 系统升级，它使用 Leapp 升级 RHEL 原位升级
- 重新引导节点

9.1. 升级 CONTROL PLANE 节点

要将环境中的 control plane 节点升级到 Red Hat Enterprise Linux 9.2，您必须一次升级三分之一 control plane 节点，从 bootstrap 节点开始。

您可以使用 **openstack overcloud upgrade run** 命令升级 control plane 节点。这个命令执行以下操作：

- 对操作系统执行 Leapp 升级。
- 作为 Leapp 升级的一部分执行重启。

每个节点会在系统升级过程中重启。在此停机期间，Pacemaker 集群和 Red Hat Ceph Storage 集群的性能会降级，但没有中断。

本例包含以下带有可组合角色的节点：

- **controller-0**
- **controller-1**
- **controller-2**
- **database-0**
- **database-1**
- **database-2**
- **networker-0**
- **networker-1**
- **networker-2**
- **ceph-0**
- **ceph-1**
- **ceph-2**

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. 运行以下脚本，但没有 **CONTROL_PLANE_ROLES** 参数：确保您包含用于运行 overcloud 升级准备容器时使用的变量。

```
python3 \
/usr/share/openstack-tripleo-heat-templates/tools/multi-rhel-container-image-prepare.py \
    ${COMPUTE_ROLES} \
    --enable-multi-rhel \
    --excludes collectd \
    --excludes nova-libvirt \
    --minor-override \
"
{${EL8_TAGS}${EL8_NAMESPACE}${CEPH_TAGS}${NEUTRON_DRIVER}\"no_tag\":\"not_
used\"}" \
    --major-override \
    "
{${EL9_TAGS}${NAMESPACE}${CEPH_TAGS}${NEUTRON_DRIVER}\"no_tag\":\"not_used
\"}" \
    --output-env-file \
/home/stack/containers-prepare-parameter.yaml
```

注意

CONTROL_PLANE_ROLES 参数定义 control plane 角色的列表。从脚本中删除此参数为升级到 RHEL 9.2 准备 control plane 角色。如果脚本中包含 **CONTROL_PLANE_ROLES** 参数，则 control plane 角色会保留在 RHEL 8.4 上。

4. 在 **skip_rhel_release.yaml** 文件中，将 **SkipRhelEnforcement** 参数设置为 **false** ：

```
parameter_defaults:
  SkipRhelEnforcement: false
```

5. 更新 **overcloud_upgrade_prepare.sh** 文件：

```
$ openstack overcloud upgrade prepare --yes \
    ...
    -e /home/stack/system_upgrade.yaml \
    -e /home/stack/containers-prepare-parameter.yaml \
    -e /home/stack/skip_rhel_release.yaml \
    ...
```

- 使用特定于升级的参数(-e)包括 **system_upgrade.yaml** 文件。
- 包含 **containers-prepare-parameter.yaml** 文件，并删除了 control plane 角色(-e)。
- 包含 **skip_rhel_release.yaml** 文件，其中包含发行版本参数(-e)。

6. 运行 **overcloud_upgrade_prepare.sh** 脚本：

```
$ sh /home/stack/overcloud_upgrade_prepare.sh
```

7. 获取系统升级所需的任何新的或修改的容器：

```
$ openstack overcloud external-upgrade run  \
    --stack <stack> \
    --tags container_image_prepare 2>&1
```

8. 升级第一个 control plane 节点三分之一：

```
$ openstack overcloud upgrade run --yes \
    --stack <stack> \
    --tags system_upgrade \
    --limit <controller-0>,<database-0>,<messaging-0>,<networker-0>,<ceph-0>
```

- 将 **<stack>** 替换为您的堆栈的名称。
- 将 &**lt;controller-0>,<database-0&** gt;**, <messaging-0** &**gt; , <networker-0 > , <ceph-0** > 替换为您自己的节点名称。

9. 登录到每个升级的节点，并验证每个节点的集群是否正在运行：

```
$ sudo pcs status
```

在升级了第二个 control plane 节点的三分之一后，重复这个验证步骤，并在升级最后三分之一 control plane 节点后。

10. 升级第二个 control plane 节点的三分之一：

```
$ openstack overcloud upgrade run --yes \
    --stack <stack> \
    --tags system_upgrade \
    --limit <controller-1>,<database-1>,<messaging-1>,<networker-1>,<ceph-1>
```

- 将 &**lt;controller-1&gt**; ,**<database-1&** gt;**, <messaging** -1&**gt; , <networker** -**1> , <ceph** -1> 替换为您自己的节点名称。

11. 升级 control plane 节点的最后三分之一：

```
$ openstack overcloud upgrade run --yes \
    --stack <stack> \
    --tags system_upgrade \
    --limit <controller-2>,<database-2>,<messaging-2>,<networker-2>,<ceph-2>
```

- 将 &**lt;controller-2&gt**; ,**<database-2&** gt;**, <messaging** -2&**gt; , <networker** -**2> , <ceph** -2> 替换为您自己的节点名称。

第 10 章 升级 COMPUTE 节点操作系统

您可以将所有 Compute 节点上的操作系统升级到 RHEL 9.2，或升级一些 Compute 节点，同时剩余的在 RHEL 8.4 上。

先决条件

- 参阅 Compute 节点升级的规划。

10.1. 为升级测试选择 COMPUTE 节点

通过 overcloud 升级过程，您可以：

- 升级角色中的所有节点。
- 单独升级单个节点。

为确保 overcloud 平稳升级过程，在升级所有 Compute 节点前，在环境中测试几个单独 Compute 节点上的升级会很有用。这样可确保升级过程中不会发生重大问题，同时为工作负载保持最小的停机时间。

使用以下建议来帮助为升级选择测试节点：

- 选择两个或三个 Compute 节点用于升级测试。
- 选择没有运行任何关键实例的节点。
- 如有必要，将关键实例从所选测试 Compute 节点迁移到其他 Compute 节点。查看支持哪些迁移场景：

源 Compute 节点 RHEL 版本	目标 Compute 节点 RHEL 版本	支持/不支持
RHEL 8	RHEL 8	支持
RHEL 8	RHEL 9	支持
RHEL 9	RHEL 9	支持
RHEL 9	RHEL 8	不支持

10.2. 将所有 COMPUTE 节点升级到 RHEL 9.2

将所有 Compute 节点升级到 RHEL 9.2，以利用最新的功能并减少停机时间。

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. 在 **container-image-prepare.yaml** 文件中，确保仅包含 **ContainerImagePrepare** 参数中指定的标签，并且 **MultiRhelRoleContainerImagePrepare** 参数已被删除。例如：

```
parameter_defaults:
  ContainerImagePrepare:
  - tag_from_label: "{version}-{release}"
    set:
      namespace:
      name_prefix:
      name_suffix:
      tag:
      rhel_containers: false
      neutron_driver: ovn
      ceph_namespace:
      ceph_image:
      ceph_tag:
```

4. 在 **roles_data.yaml** 文件中，将 **OS::TripleO::Services::NovaLibvirtLegacy** 服务替换为 RHEL 9.2 所需的 **OS::TripleO::Services::NovaLibvirt** 服务。

5. 运行 **openstack overcloud upgrade prepare** 命令，并使用特定于升级的参数包括 **system_upgrade.yaml** 文件：

```
$ openstack overcloud upgrade prepare --yes
…
-e /home/stack/system_upgrade.yaml
…
```

6. 将 Compute 节点上的操作系统升级到 RHEL 9.2。使用您要升级的节点列表使用 **--limit** 选项。以下示例升级 **compute-0**、**compute-1** 和 **compute-2** 节点。

```
$ openstack overcloud upgrade run --yes --tags system_upgrade --stack <stack> --limit
compute-0,compute-1,compute-2
```

   - 将 **<stack>** 替换为您的堆栈的名称。

7. 将 Compute 节点上的容器升级到 RHEL 9.2。使用您要升级的节点列表使用 **--limit** 选项。以下示例升级 **compute-0**、**compute-1** 和 **compute-2** 节点。

```
$ openstack overcloud upgrade run --yes --stack <stack>  --limit compute-0,compute-
1,compute-2
```

10.3. 将 COMPUTE 节点升级到多 RHEL 环境

您可以将部分 Compute 节点升级到 RHEL 9.2，而其余的 Compute 节点保留在 RHEL 8.4 上。这个升级过程涉及以下基本步骤：

1. 选择您要升级到 RHEL 9.2 的节点，以及您希望在 RHEL 8.4 上保留哪些节点。为您要为每个批处理节点创建的每个角色选择一个角色名称，如 **ComputeRHEL-9.2** 和 **ComputeRHEL-8.4**。

2. 创建存储要升级到 RHEL 9.2 的节点的角色，或存储您要保留在 RHEL 8.4 上的节点。这些角色可以保持为空，直到您准备好将 Compute 节点移到新角色为止。您可以根据需要创建任意数量的角色，并在决定时划分节点。例如：

   - 如果您的环境使用名为 **ComputeSRIOV** 的角色，您需要运行 Canary 测试来升级到 RHEL

9.2，您可以创建一个新的 **ComputeSRIOVRHEL9** 角色，并将 Canary 节点移到新角色。

- 如果您的环境使用名为 **ComputeOffload** 的角色，而您想要将该角色中的大多数节点升级到 RHEL 9.2，但在 RHEL 8.4 上保留几个节点，您可以创建一个新的 **ComputeOffloadRHEL8** 角色来存储 RHEL 8.4 节点。然后，您可以选择原始 **ComputeOffload** 角色中的节点升级到 RHEL 9.2。

3. 将节点从每个 Compute 角色移到新角色。
4. 将特定 Compute 节点上的操作系统升级到 RHEL 9.2。您可以从相同的角色或多个角色升级批处理中的节点。

注意

在 Multi-RHEL 环境中，部署应该继续使用 pc-i440fx 机器类型。不要将默认值更新为 Q35。迁移到 Q35 机器类型是在所有 Compute 节点升级到 RHEL 9.2 后要遵循的独立、升级后的步骤。有关迁移 Q35 机器类型的更多信息，请参阅 升级到 RHOSP 17 后为主机更新默认机器类型。

使用以下步骤将 Compute 节点升级到 Multi-RHEL 环境：

- 为 Multi-RHEL Compute 节点创建角色
- 升级 Compute 节点操作系统

10.3.1. 为 Multi-RHEL Compute 节点创建角色

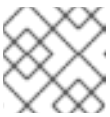
创建新角色以存储您要升级到 RHEL 9.2 或保留在 RHEL 8.4 上的节点，并将节点移到新角色中。

流程

1. 为您的环境创建相关角色。在 **role_data.yaml** 文件中，复制源 Compute 角色以用于新角色。对所需的每个额外角色重复此步骤。角色可以保持空，直到您准备好将 Compute 节点移到新角色。

- 如果要创建 RHEL 8 角色：

```
name: <ComputeRHEL8>
 description: |
  Basic Compute Node role
 CountDefault: 1
 rhsm_enforce: False
...
ServicesDefault:
...
- OS::TripleO::Services::NovaLibvirtLegacy
```



注意

包含 RHEL 8.4 上剩余的节点的角色必须包含 **NovaLibvirtLegacy** 服务。

- 将 **<ComputeRHEL8>** 替换为 RHEL 8.4 角色的名称。
- 如果要创建 RHEL 9 角色：

```
name: <ComputeRHEL9>
 description: |
Basic Compute Node role
 CountDefault: 1
...
ServicesDefault:
...
- OS::TripleO::Services::NovaLibvirt
```

注意

包含升级到 RHEL 9.2 的节点的角色必须包含 **NovaLibvirt** 服务。将 **OS::TripleO::Services::NovaLibvirtLegacy** 替换为 **OS::TripleO::Services::NovaLibvirt**。

- 将 <ComputeRHEL9> 替换为 RHEL 9.2 角色的名称。

2. 将 **overcloud_upgrade_prepare.sh** 文件复制到 **copy_role_Compute_param.sh** 文件中：

```
$ cp overcloud_upgrade_prepare.sh copy_role_Compute_param.sh
```

3. 编辑 **copy_role_Compute_param.sh** 文件，使其包含 **copy_role_params.py** 脚本。此脚本会生成环境文件，其中包含新角色的额外参数和资源。例如：

```
/usr/share/openstack-tripleo-heat-templates/tools/copy_role_params.py --rolename-src
<Compute_source_role> --rolename-dst <Compute_destination_role> \
 -o <Compute_new_role_params.yaml> \

-e /home/stack/templates/internal.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
-e /home/stack/templates/network/network-environment.yaml \
-e /home/stack/templates/inject-trust-anchor.yaml \
-e /home/stack/templates/hostnames.yml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
\
-e /home/stack/templates/nodes_data.yaml \
-e /home/stack/templates/debug.yaml \
-e /home/stack/templates/firstboot.yaml \
   -e /home/stack/overcloud-params.yaml \
 -e /home/stack/overcloud-deploy/overcloud/overcloud-network-environment.yaml \
-e /home/stack/overcloud_adopt/baremetal-deployment.yaml \
-e /home/stack/overcloud_adopt/generated-networks-deployed.yaml \
-e /home/stack/overcloud_adopt/generated-vip-deployed.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-hw-machine-type-
upgrade.yaml \
-e ~/containers-prepare-parameter.yaml
```

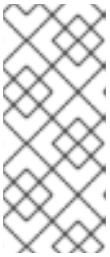
- 将 **<Compute_source_role** > 替换为您要复制的源 Compute 角色的名称。
- 将 **<Compute_destination_role** > 替换为新角色的名称。
- 使用 -o 选项定义包含新角色 source Compute 角色的所有非默认值的输出文件的名称。将 **<Compute_new_role_params.yaml** > 替换为您的输出文件的名称。

4. 运行 **copy_role_Compute_param.sh** 脚本：

```
$ sh /home/stack/copy_role_Compute_param.sh
```

5. 将 Compute 节点从源角色移到新角色中：

```
python3
/usr/share/openstack-tripleo-heat-templates/tools/baremetal_transition.py  --baremetal-
deployment /home/stack/tripleo-<stack>-baremetal-deployment.yaml  --src-role
<Compute_source_role>  --dst-role <Compute_destination_role> <Compute-0> <Compute-
1> <Compute-2>
```



注意

此工具包括您在 undercloud 升级过程中导出的原始 **/home/stack/tripleo-<stack>-baremetal-deployment.yaml** 文件。该工具在 **/home/stack/tripleo-<stack>-baremetal-deployment.yaml** 文件中复制并重命名源角色定义。然后，它会更改 **hostname_format**，以防止与新创建的目标角色冲突。然后，该工具将节点从源角色移到目的地角色，并更改 **count** 值。

- 将 **<stack>** 替换为您的堆栈的名称。
- 将 < **Compute_source_role** > 替换为包含您要移至新角色的节点的源 Compute 角色的名称。
- 将 **<Compute_destination_role** > 替换为新角色的名称。
- 将 **<Compute-0 > & lt;Compute** -1> <Compute-2> 替换为您要移动到新角色的节点的名称。

6. 重新置备节点，以使用新角色位置更新堆栈中的环境文件：

```
$ openstack overcloud node provision --stack <stack> --output
/home/stack/overcloud_adopt/baremetal-deployment.yaml /home/stack/tripleo-<stack>-
baremetal-deployment.yaml
```

注意

输出 **baremetal-deployment.yaml** 文件是在 overcloud 采用过程中 **overcloud_upgrade_prepare.sh** 文件中使用的相同文件。

7. 在 **COMPUTE_ROLES** 参数中包含 RHEL 8.4 上剩余的任何 Compute 角色，并运行以下脚本。例如，如果您有一个名为 **ComputeRHEL8** 的角色，其中包含在 RHEL 8.4 中剩余的节点，**COMPUTE_ROLES = --role ComputeRHEL8**。

```
python3
/usr/share/openstack-tripleo-heat-templates/tools/multi-rhel-container-image-prepare.py \
    ${COMPUTE_ROLES} \
    --enable-multi-rhel \
    --excludes collectd --excludes nova-libvirt \
    --minor-override
    "
{${EL8_TAGS}${EL8_NAMESPACE}${CEPH_TAGS}${NEUTRON_DRIVER}\"no_tag\":\"not_
used\"}" --major-override
```

```
    "
{${EL9_TAGS}${NAMESPACE}${CEPH_TAGS}${NEUTRON_DRIVER}\"no_tag\":\"not_used
\"}" \
    --output-env-file
/home/stack/containers-prepare-parameter.yaml
```

8. 重复此步骤以创建额外的角色，并将额外的 Compute 节点移到这些新角色中。

10.3.2. 升级 Compute 节点操作系统

将所选 Compute 节点上的操作系统升级到 RHEL 9.2。您可以同时从不同角色升级多个节点。

先决条件

确保您已为您的环境创建了必要的角色。有关为多 RHEL 环境创建角色的更多信息，请参阅为多 RHEL Compute 节点创建角色。

流程

1. 在 **skip_rhel_release.yaml** 文件中，将 **SkipRhelEnforcement** 参数设置为 **false** ：

```
parameter_defaults:
  SkipRhelEnforcement: false
```

2. 运行 **openstack overcloud upgrade prepare** 命令，以设置新角色所需的参数和资源。

```
$ openstack overcloud upgrade prepare --yes \
...
-e /home/stack/system_upgrade.yaml  \
-e /home/stack/skip_rhel_release.yaml \
-e /home/stack/<Compute_new_role_params.yaml> \
...
```

   - 使用特定于升级的参数(-e)包括 **system_upgrade.yaml** 文件。
   - 包含 **skip_rhel_release.yaml** 文件，其中包含 release 参数。
   - 包含包含新角色所需参数(-e)的环境文件。将 **<Compute_new_role_params.yaml** > 替换为您为新角色创建的环境文件的名称。
   - 如果您要同时从多个角色升级节点，请包含您创建的每个新角色的环境文件。

3. 可选：迁移您的实例。有关迁移策略的更多信息，请参阅在 Compute 节点间迁移虚拟机和准备迁移。

4. 升级特定 Compute 节点上的操作系统。使用您要升级的节点列表使用 **--limit** 选项。以下示例将 **computerhel9-0**, **computerhel9-1**, **computerhel9-2**, 和 **computesriov-42** 节点从 **ComputeRHEL9** 和 **ComputeSRIOV** 介绍升级。

```
$ openstack overcloud upgrade run --yes --tags system_upgrade --stack <stack> --limit
computerhel9-0,computerhel9-1,computerhel9-2,computesriov-42
```

   - 将 <stack> 替换为您的堆栈的名称。

5. 将 Compute 节点上的容器升级到 RHEL 9.2。使用您要升级的节点列表使用 **--limit** 选项。以下示例将 **computerhel9-0**, **computerhel9-1**, **computerhel9-2**, 和 **computesriov-42** 节点从 **ComputeRHEL9** 和 **ComputeSRIOV** 介绍升级。

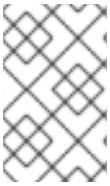
```
$ openstack overcloud upgrade run --yes --stack <stack>  --limit computerhel9-
0,computerhel9-1,computerhel9-2,computesriov-42
```

第 11 章 执行升级后的操作

完成 overcloud 升级后，您必须执行一些升级后的配置，以确保您的环境被完全支持并准备好进行将来的操作。

11.1. 升级 OVERCLOUD 镜像

您必须将您当前的 overcloud 镜像替换为新版本。新镜像可确保 director 可以使用最新版本的 Red Hat OpenStack Platform 软件内省和置备您的节点。



注意

如果重新部署 overcloud，则必须使用 overcloud 镜像的新版本。有关安装 overcloud 镜像的更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform 中的安装 overcloud 镜像*。

先决条件

- 您已将 undercloud 升级到最新版本。

流程

1. 从 **stack** 用户的主目录（**/home/stack/images**）的 **images** 中删除任何存在的镜像。

```
$ rm -rf ~/images/*
```

2. 解压存档：

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-17.1.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-17.1.tar; do tar -xvf $i; done
$ cd ~
```

11.2. 更新 CPU 固定参数

在完成升级到 Red Hat OpenStack Platform 17.1 后，您必须将 CPU 固定配置从 **NovaVcpuPinSet** 参数迁移到以下参数：

NovaComputeCpuDedicatedSet
设置专用（固定）CPU。

NovaComputeCpuSharedSet
设置共享（未固定）CPU。

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. 如果您的 Compute 节点支持并发多线程(SMT)，但创建了带有 **hw:cpu_thread_policy=isolated** 策略的实例，您必须执行以下选项之一：
   - 取消设置 **hw:cpu_thread_policy** 线程策略并调整实例大小：

i. 提供 overcloud 身份验证文件：

```
$ source ~/overcloudrc
```

ii. 取消设置类别的 **hw:cpu_thread_policy** 属性：

```
(overcloud) $ openstack flavor unset --property hw:cpu_thread_policy <flavor>
```

注意

- 取消设置 **hw:cpu_thread_policy** 属性可将策略设置为默认的 **prefer** 策略，这会将实例设置为使用启用了 SMT 的 Compute 节点（如果可用）。您还可以将 **hw:cpu_thread_policy** 属性设置为 **require**，这将为启用 SMT 的 Compute 节点设置硬要求。
- 如果 Compute 节点没有 SMT 架构或足够的 CPU 内核，则调度将失败。要防止这种情况，将 **hw:cpu_thread_policy** 设置为 **prefer** 而不是 **require**。默认 **首选** 策略确保线程同级在可用时被使用。
- 如果使用 **hw:cpu_thread_policy=isolate**，则必须禁用 SMT，或使用不支持 SMT 的平台。

iii. 将实例转换为使用新的线程策略。

```
(overcloud) $ openstack server resize --flavor <flavor> <server>
(overcloud) $ openstack server resize confirm <server>
```

使用 **hw:cpu_thread_policy=isolated** 策略为所有固定实例重复此步骤。

- 从 Compute 节点迁移实例，并在 Compute 节点上禁用 SMT：

i. 提供 overcloud 身份验证文件：

```
$ source ~/overcloudrc
```

ii. 禁用 Compute 节点接受新虚拟机：

```
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service set <hostname> nova-compute --disable
```

iii. 从 Compute 节点迁移所有实例。有关实例迁移的更多信息，请参阅 在 Compute 节点之间迁移虚拟机实例。

iv. 重新引导 Compute 节点，并在 Compute 节点的 BIOS 中禁用 SMT。

v. 引导 Compute 节点。

vi. 重新启用 Compute 节点：

```
(overcloud) $ openstack compute service set <hostname> nova-compute --enable
```

3. Source **stackrc** 文件：

```
$ source ~/stackrc
```

4. 编辑包含 **NovaVcpuPinSet** 参数的环境文件。

5. 将 CPU 固定配置从 **NovaVcpuPinSet** 参数迁移到 **NovaComputeCpuDedicatedSet** 和 **NovaComputeCpuSharedSet** ：

   - 将 **NovaVcpuPinSet** 的值迁移到之前用于固定实例的主机的 **NovaComputeCpuDedicatedSet**。
   - 将 **NovaVcpuPinSet** 的值迁移到之前用于未固定实例的主机的 **NovaComputeCpuSharedSet**。
   - 如果没有为 NovaVcpuPinSet 设置值，则所有 Compute 节点内核都应分配给 **NovaComputeCpuDedicatedSet** 或 **NovaComputeCpuSharedSet**，具体取决于您要在节点上托管的实例类型。

   例如，您之前的环境文件可能包含以下固定配置：

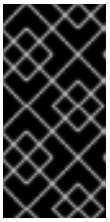
```
parameter_defaults:
  ...
  NovaVcpuPinSet: 1,2,3,5,6,7
  ...
```

   要将配置迁移到固定配置，请设置 **NovaComputeCpuDedicatedSet** 参数，然后取消设置 **NovaVcpuPinSet** 参数：

```
parameter_defaults:
  ...
  NovaComputeCpuDedicatedSet: 1,2,3,5,6,7
  NovaVcpuPinSet: ""
  ...
```

   要将配置迁移到未固定配置，请设置 **NovaComputeCpuSharedSet** 参数，然后取消设置 **NovaVcpuPinSet** 参数：

```
parameter_defaults:
  ...
  NovaComputeCpuSharedSet: 1,2,3,5,6,7
  NovaVcpuPinSet: ""
  ...
```



重要

确保 **NovaComputeCpuDedicatedSet** 或 **NovaComputeCpuSharedSet** 的配置与 **NovaVcpuPinSet** 中定义的配置匹配。要更改其中任一个的配置，或配置 **NovaComputeCpuDedicatedSet** 或 **NovaComputeCpuSharedSet**，请确保在更新配置前，使用固定配置的 Compute 节点不会运行任何实例。

6. 保存该文件。

7. 运行部署命令，以使用新的 CPU 固定参数更新 overcloud。

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

```
--stack _STACK NAME_ \
--templates \
...
-e /home/stack/templates/<compute_environment_file>.yaml
...
```

其他资源

- 在 Compute 节点上配置 CPU 固定

11.3. 升级到 RHOSP 17 后为主机更新默认机器类型

实例的机器类型是一个虚拟芯片组，提供某些默认设备，如 PCIe 图形卡或以太网控制器。云用户可以使用带有 **hw_machine_type** 元数据属性的镜像为其实例指定机器类型。

云管理员可以使用 Compute 参数 **NovaHWMachineType** 配置每个 Compute 节点架构，使其具有默认的机器类型，以应用到托管在该架构上的实例。如果在启动实例时不提供 **hw_machine_type** 镜像属性，则主机架构的默认机器类型将应用到实例。Red Hat OpenStack Platform (RHOSP) 17 基于 RHEL 9。RHEL 9 中弃用了 **pc-i440fx** QEMU 机器类型，因此在 RHEL 9 上运行的 **x86_64** 实例的默认机器类型已从 **pc** 改为 **q35**。根据 RHEL 9 中的这个变化，机器类型 **x86_64** 的默认值也从 RHOSP 16 中的 **pc** 改为 **q35** （在 RHOSP 17 中）。

在 RHOSP 16.2 及之后的版本中，Compute 服务会在实例启动时记录实例元数据中的实例类型。这意味着，现在可以在 RHOSP 部署生命周期内更改 **NovaHWMachineType**，而不影响现有实例的机器类型。

Compute 服务记录了不在 **SHELVED_OFFLOADED** 状态的实例类型。因此，在升级到 RHOSP 17 后，您必须手动记录处于 **SHELVED_OFFLOADED** 状态的机器类型，并验证环境或特定单元中的所有实例是否已记录机器类型。使用机器类型更新每个实例的系统元数据后，您可以将 **NovaHWMachineType** 参数更新为 RHOSP 17 默认 **q35**，而不影响现有实例的机器类型。

先决条件

- 将所有 Compute 节点升级到 RHEL 9.2。有关升级 Compute 节点的更多信息，请参阅 将所有 Compute 节点升级到 RHEL 9.2。

流程

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

```
$ source ~/stackrc
```

3. 以 **heat-admin** 用户身份登录 Controller 节点：

```
(undercloud)$ metalsmith list
$ ssh heat-admin@<controller_ip>
```

将 **<controller_ip** > 替换为 Controller 节点的 IP 地址。

4. 检索没有设置机器类型的实例列表：

```
[heat-admin@<controller_ip> ~]$ sudo podman exec -i -u root nova_api \
  nova-manage libvirt list_unset_machine_type
```

5. 检查 **nova-hw-machine-type-upgrade.yaml** 文件中的 **NovaHWMachineType** 参数，以获取实例主机的默认机器类型。RHOSP 16.2 中 **NovaHWMachineType** 参数的默认值如下：
   x86_64=pc-i440fx-rhel7.6.0,aarch64=virt-rhel7.6.0,ppc64=pseries-rhel7.6.0,ppc64le=pseries-rhel7.6.0

6. 使用默认实例类型更新每个实例的系统元数据：

```
[heat-admin@<controller_ip> ~]$ sudo podman exec -i -u root nova_api \
  nova-manage libvirt update_machine_type <instance_uuid> <machine_type>
```

   - 将 **<instance_uuid** > 替换为实例的 UUID。
   - 将 **<machine_type>** 替换为要记录的实例类型。

7. 确认为所有实例记录了机器类型：

```
[heat-admin@<controller_ip> ~]$ sudo podman exec -i -u root nova_api \
  nova-status upgrade check
```

   如果实例在没有机器类型的情况下找到，这个命令会返回一个警告。如果收到这个警告，请从第 4 步重复这个过程。

8. 将 Compute 环境文件中的 **NovaHWMachineType** 默认值更改为 **x86_64=q35** 并部署 overcloud。

验证

1. 创建具有默认机器类型的实例：

```
(overcloud)$ openstack server create --flavor <flavor> \
  --image <image> --network <network> \
  --wait defaultMachineTypeInstance
```

   - 将 **<flavor** > 替换为实例的类别名称或 ID。
   - 将 **<image** > 替换为没有设置 **hw_machine_type** 的镜像的名称或 ID。
   - 将 & **lt;network** > 替换为要连接实例的网络的名称或 ID。

2. 验证实例机器类型是否已设置为默认值：

```
[heat-admin@<controller_ip> ~]$ sudo podman exec -i -u root nova_api \
  nova-manage libvirt get_machine_type <instance_uuid>
```

   将 **<instance_uuid** > 替换为实例的 UUID。

3. 硬重启一个类型为 **x86_64=pc-i440fx** 的实例：

```
(overcloud)$ openstack server reboot --hard <instance_uuid>
```

   将 **<instance_uuid** > 替换为实例的 UUID。

4. 验证实例机器类型尚未更改：

```
[heat-admin@<controller_ip> ~]$ sudo podman exec -i -u root nova_api \
  nova-manage libvirt get_machine_type <instance_uuid>
```

将 **<instance_uuid>** 替换为实例的 UUID。

11.4. 在 OVERCLOUD 中重新启用隔离

在升级 overcloud 之前，您在 overcloud 中禁用隔离功能。升级环境后，在节点失败时重新启用隔离来保护数据。

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. 登录到 Controller 节点并运行 Pacemaker 命令重新启用隔离：

```
$ ssh tripleo-admin@<controller_ip> "sudo pcs property set stonith-enabled=true"
```

   - 将 **<controller_ip>** 替换为 Controller 节点的 IP 地址。您可以使用 **openstack server list** 命令查找 Controller 节点的 IP 地址。

4. 在 **fencing.yaml** 环境文件中，将 **EnableFencing** 参数设置为 **true**。

其它资源

- 使用 STONITH 隔离 Controller 节点