Red Hat OpenStack Platform 13

快进升级

从 Red Hat OpenStack Platform 10 升级到 13

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 快进升级

从 Red Hat OpenStack Platform 10 升级到 13

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南提供了快进的升级过程。这个过程将 OpenStack Platform 环境从一个长生命版本升级到下一个长生命版本。在这种情况下，指南侧重于从 Red Hat OpenStack Platform 10 (Newton)升级到 13 (Queens)。

目录

第 1 章 简介

本文档提供了可帮助您将 Red Hat OpenStack Platform 环境升级到最新版本的工作流。

1.1. 开始前

注意以下几点：

- 如果您最初使用 7 或 8 版本部署了 Red Hat OpenStack Platform 环境，请注意，旧版本的 XFS 文件系统出现问题，这可能会造成升级路径和部署容器化服务。有关此问题以及如何解决问题的更多信息，请参阅文章 "XFS ftype=0 会阻止使用容器升级到 OpenStack Director 版本 "。
- 如果您的部署包含 Red Hat Ceph Storage (RHCS)节点，则每个 Ceph 对象存储守护进程(OSD)的放置组(PG)数不得超过 250 个。每个 OSD 有更多 PG 升级 Ceph 节点会导致警告状态，并可能导致升级过程失败。在开始升级过程前，您可以增加每个 OSD 的 PG 数量。有关诊断并排除这个问题的更多信息，请参阅 OpenStack FFU 从 10 到 13 超时，因为一个或多个 OSD 中分配的 Ceph PG 大于 250 个。
- 找到 **prevent_arp_spoofing** 设为 False 的所有端口。确保用于禁用端口安全性的那些端口。作为升级的一部分，**prevent_arp_spoofing** 选项被删除，且该功能由端口安全性控制。
- 在执行升级前，为您的硬件应用任何固件更新。
- 如果手动更改了部署的 overcloud （包括应用程序密码），则必须使用这些更改更新 director 部署模板，以避免升级失败。如果您有疑问，请联系红帽技术支持。

1.2. 快进升级

Red Hat OpenStack Platform **提供了快进升级** 功能。此功能通过多个 overcloud 版本提供升级路径。目标是在下一个长生命版本可用时，为用户提供处于 长**生命版本和升级的某些 OpenStack 版本中** 的机会。

本指南通过以下版本提供快进的升级路径：

旧版本	新版本
Red Hat OpenStack Platform 10	Red Hat OpenStack Platform 13

1.3. 高级别工作流

下表概述了快速升级过程所需的步骤，并估计每个升级过程步骤的持续时间和影响。

注意

这个表中的持续时间根据内部测试的最小估算值，可能不适用于所有生产环境。要准确衡量每个任务的升级持续时间，请在测试环境中使用与生产环境类似的硬件来执行这些步骤。

表 1.1. Fast forward upgrade process steps outline and impact

Step	Description	Duration
准备您的环境	对 undercloud 节点和 overcloud Controller 节点执行数据库和配置的备份。更新至最新的次版本并重新引导。验证环境。	此步骤的持续时间可能会因部署大小而异。
升级 undercloud	将 undercloud 的每个后续版本从 OpenStack Platform 10 升级到 OpenStack Platform 13。	升级 undercloud 的预计持续时间约为 60 分钟，在升级过程中 undercloud 停机。 在 undercloud 升级过程中，overcloud 仍然可以正常工作。
获取容器镜像	创建一个环境文件，其中包含各种 OpenStack 服务的容器镜像位置。	配置容器镜像源的预计持续时间约为 10 分钟。
准备 overcloud	执行相关步骤，将您的 overcloud 配置文件过渡到 OpenStack Platform 13。	准备 overcloud 进行升级的预计持续时间大约为 20 分钟。
执行快速升级	使用最新的 OpenStack Platform director 模板升级 overcloud 计划。运行通过每个后续版本的软件包和数据库升级，以便数据库模式已准备好升级到 OpenStack Platform 13。	overcloud 升级运行的预计持续时间约为 30 分钟，在升级过程中 overcloud 服务停机。 您不能在中断期间执行 OpenStack 操作。
升级 Controller 节点	将所有 Controller 节点同时升级到 OpenStack Platform 13。	Controller 节点升级的预计持续时间约为 50 分钟。 预计在 Controller 节点升级过程中会短的 overcloud 服务停机。
升级 Compute 节点	测试所选 Compute 节点上的升级。如果测试成功，请升级所有 Compute 节点。	Compute 节点升级的预计持续时间为每个节点大约 25 分钟。 在 Compute 节点升级过程中，没有预期的停机时间。
升级 Ceph Storage 节点	升级所有 Ceph Storage 节点。这包括升级到 Red Hat Ceph Storage 3 的容器化版本。	Ceph Storage 节点升级的预期持续时间为每个节点大约 25 分钟。 Ceph Storage 节点升级过程中没有预期的停机时间。
完成升级	运行 convergence 命令以刷新 overcloud 堆栈。	overcloud 聚合运行的预计持续时间至少为 1 小时，但根据您的环境可能会花费较长时间。

1.4. 在升级前检查 CEPH 集群状态

在升级环境前，您必须验证 Ceph 集群是否活跃并可按预期工作。

流程

1. 登录运行 **ceph-mon** 服务的节点。此节点通常是 Controller 节点或独立 Ceph 监控节点。
2. 查看 Ceph 集群的状态：

```
$ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph -s"
```

3. 确认集群的运行状况状态为 **HEALTH_OK**，并且所有 Object Storage Daemon (OSD)都是 active。

第 2 章 准备 OPENSTACK PLATFORM 升级

此过程准备您的 OpenStack Platform 环境。这涉及以下步骤：

- 备份 undercloud 和 overcloud。
- 将 undercloud 更新至 OpenStack Platform 10 的最新次要版本，包括最新的 Open vSwitch。
- 如果安装了较新的内核或较新的系统软件包，请重新引导 undercloud。
- 将 overcloud 更新至 OpenStack Platform 10 的最新次要版本，包括最新的 Open vSwitch。
- 如果安装了较新的内核或较新的系统软件包，请重新引导 overcloud 节点。
- 对 undercloud 和 overcloud 执行验证检查。

在进行升级前，这些步骤可确保您的 OpenStack Platform 环境处于最佳状态。

2.1. 创建裸机 UNDERCLOUD 备份

完整的 undercloud 备份包括以下数据库和文件：

- undercloud 节点上的所有 MariaDB 数据库
- undercloud 上的 MariaDB 配置文件（因此您可以准确恢复数据库）
- 配置数据： **/etc**
- 日志数据： **/var/log**
- 镜像数据： **/var/lib/glance**
- 如果使用 SSL: **/var/lib/certmonger**，证书生成数据
- 任何容器镜像数据： **/var/lib/docker** 和 **/var/lib/registry**
- 所有 swift 数据： **/srv/node**
- stack 用户主目录中的所有数据： **/home/stack**

注意

在执行备份过程前，确认 undercloud 上有足够的磁盘空间。如果不存在，存档文件至少为 3.5 GB。

流程

1. 以 **root** 用户身份登录 undercloud。
2. 备份数据库：

```
[root@director ~]# mysqldump --opt --all-databases > /root/undercloud-all-databases.sql
```

3. 创建 **备份目录**，并将目录的用户所有权改为 **stack** 用户：

```
[root@director ~]# mkdir /backup
[root@director ~]# chown stack: /backup
```

您将使用此目录存储包含 undercloud 数据库和文件系统的存档。

4. 进入 **备份目录**

```
[root@director ~]# cd /backup
```

5. 归档数据库备份和配置文件：

```
[root@director ~]# tar --xattrs --xattrs-include='*.*' --ignore-failed-read -cf \
  undercloud-backup-$(date +%F).tar \
  /root/undercloud-all-databases.sql \
  /etc \
  /var/log \
  /var/lib/glance \
  /var/lib/certmonger \
  /var/lib/docker \
  /var/lib/registry \
  /srv/node \
  /root \
  /home/stack
```

- **--ignore-failed-read** 选项跳过任何不适用于 undercloud 的目录。
- **--xattrs** 和 **--xattrs-include='** 含有扩展属性，这是存储 Object Storage (swift)和 SELinux 的元数据所必需的。

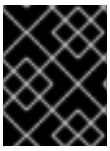
这会创建一个名为 **undercloud-backup-<date>.tar.gz** 的文件，其中 & **lt;date** > 是系统日期。将此 **tar** 文件复制到安全位置。

相关信息

- 如果您需要恢复 undercloud 备份，请参阅 附录 A, *恢复 undercloud*。

2.2. 备份 OVERCLOUD CONTROL PLANE 服务

以下流程创建 overcloud 数据库和配置的备份。overcloud 数据库和服务的备份可确保您拥有正常工作的环境的快照。如果有这个快照，则需要在操作失败时将 overcloud 恢复到其原始状态。



重要

此流程仅包含重要的 control plane 服务。它不包括计算节点工作负载的备份、Ceph Storage 节点上的数据，以及任何其他服务。

流程

1. 执行数据库备份：

   a. 登录 Controller 节点。您可以从 undercloud 访问 overcloud：

```
$ ssh heat-admin@192.0.2.100
```

b. 进入 **root** 用户：

```
$ sudo -i
```

c. 创建用于存储备份的临时目录：

```
# mkdir -p /var/tmp/mysql_backup/
```

d. 获取数据库密码，并将它存储在 **MYSQLDBPASS** 环境变量中。密码存储在 **/etc/puppet/hieradata/service_configs.json** 文件中的 **mysql::server::root_password** 变量中。使用以下命令存储密码：

```
# MYSQLDBPASS=$(sudo hiera -c /etc/puppet/hiera.yaml mysql::server::root_password)
```

e. 备份数据库：

```
# mysql -uroot -p$MYSQLDBPASS -s -N -e "select distinct table_schema from
information_schema.tables where engine='innodb' and table_schema != 'mysql';" | xargs
mysqldump -uroot -p$MYSQLDBPASS --single-transaction --databases >
/var/tmp/mysql_backup/openstack_databases-$(date +%F)-$(date +%T).sql
```

这会转储名为 **/var/tmp/mysql_backup/openstack_databases-<date>.sql 的数据库备份，**其中 & **lt;date** > 是系统日期和时间。将此数据库转储复制到安全位置。

f. 备份所有用户和权限信息：

```
# mysql -uroot -p$MYSQLDBPASS -s -N -e "SELECT CONCAT('\"SHOW GRANTS FOR
'',user,''@'',host,'';\"') FROM mysql.user where (length(user) > 0 and user NOT LIKE
'root')" | xargs -n1 mysql -uroot -p$MYSQLDBPASS -s -N -e | sed 's/$/;/' >
/var/tmp/mysql_backup/openstack_databases_grants-$(date +%F)-$(date +%T).sql
```

这会转储名为 **/var/tmp/mysql_backup/openstack_databases_grants-<date>.sql 的数据库备份，**其中 & **lt;date** > 是系统日期和时间。将此数据库转储复制到安全位置。

2. 备份 Pacemaker 配置：

a. 登录 Controller 节点。

b. 运行以下命令来创建当前 Pacemaker 配置的存档：

```
# sudo pcs config backup pacemaker_controller_backup
```

c. 将生成的存档(**pacemaker_controller_backup.tar.bz2**)复制到安全的位置。

3. 备份 OpenStack Telemetry 数据库：

a. 连接到任何控制器并获取 MongoDB 主实例的 IP：

```
# MONGOIP=$(sudo hiera -c /etc/puppet/hiera.yaml mongodb::server::bind_ip)
```

b. 创建备份：

```
# mkdir -p /var/tmp/mongo_backup/
# mongodump --oplog --host $MONGOIP --out /var/tmp/mongo_backup/
```

c. 将 **/var/tmp/mongo_backup/** 中的数据库转储复制到安全位置。

4. 备份 Redis 集群：

a. 从 HAProxy 获取 Redis 端点：

```
# REDISIP=$(sudo hiera -c /etc/puppet/hiera.yaml redis_vip)
```

b. 获取 Redis 集群的 master 密码：

```
# REDISPASS=$(sudo hiera -c /etc/puppet/hiera.yaml redis::masterauth)
```

c. 检查到 Redis 集群的连接：

```
# redis-cli -a $REDISPASS -h $REDISIP ping
```

d. 转储 Redis 数据库：

```
# redis-cli -a $REDISPASS -h $REDISIP bgsave
```

这会将数据库备份保存在默认的 **/var/lib/redis/** 目录中。将此数据库转储复制到安全位置。

5. 备份每个 Controller 节点上的文件系统：

a. 为备份创建一个目录：

```
# mkdir -p /var/tmp/filesystem_backup/
```

b. **运行以下命令** ：

```
# tar --acls --ignore-failed-read --xattrs --xattrs-include='*.*' \
    -zcvf /var/tmp/filesystem_backup/`hostname`-filesystem-`date '+%Y-%m-%d-%H-%M-
%S'`.tar \
    /etc \
    /srv/node \
    /var/log \
    /var/lib/nova \
    --exclude /var/lib/nova/instances \
    /var/lib/glance \
    /var/lib/keystone \
    /var/lib/cinder \
    /var/lib/heat \
    /var/lib/heat-config \
    /var/lib/heat-cfntools \
    /var/lib/rabbitmq \
    /var/lib/neutron \
    /var/lib/haproxy \
    /var/lib/openvswitch \
    /var/lib/redis \
    /var/lib/os-collect-config \
    /usr/libexec/os-apply-config \
    /usr/libexec/os-refresh-config \
    /home/heat-admin
```

--ignore-failed-read 选项会忽略任何缺少的目录，如果某些服务没有在自己的自定义角色中使用或分隔，这很有用。

c. 将生成的 **tar** 文件复制到安全位置。

6. 归档在 overcloud 上删除的行：

a. 检查归档的已删除实例：

```
$ source ~/overcloudrc
$ nova list --all-tenants --deleted
```

b. 如果没有存档删除的实例，请在其中一个 overcloud Controller 节点上输入以下命令来归档已删除实例：

```
# su - nova -s /bin/bash -c "nova-manage --debug db archive_deleted_rows --max_rows 1000"
```

重新运行此命令，直到您已归档所有已删除的实例。

c. 在其中一个 overcloud Controller 节点上输入以下命令来清除所有归档的已删除实例：

```
# su - nova -s /bin/bash -c "nova-manage --debug db purge --all --all-cells"
```

d. 验证没有剩余的已归档实例：

```
$ nova list --all-tenants --deleted
```

相关信息

- 如果您需要恢复 overcloud 备份，请参阅 附录 B, *恢复 overcloud*。

2.3. 为 OPENSTACK PLATFORM 10.Z 更新当前的 UNDERCLOUD 软件包

director 提供了更新 undercloud 节点上的软件包的命令。这样，您可以在当前版本的 OpenStack Platform 环境中执行次要更新。这是 **OpenStack Platform 10** 中的小更新。

注意

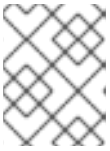
此步骤还将 undercloud 操作系统更新至 Red Hat Enterprise Linux 7 和 Open vSwitch 的最新版本 2.9。

流程

1. 以 **stack** 用户的身份登录 undercloud。

2. 停止主 OpenStack Platform 服务：

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```



注意

这会导致 undercloud 的停机时间短。overcloud 在 undercloud 升级过程中仍然可以正常工作。

3. 将 RHEL 版本设置为 RHEL 7.7：

```
$ sudo subscription-manager release --set=7.7
```

4. 更新 **python-tripleoclient** 软件包及其依赖项，以确保您具有次要版本更新的最新脚本：

```
$ sudo yum update -y python-tripleoclient
```

5. 运行 **openstack undercloud upgrade** 命令：

```
$ openstack undercloud upgrade
```

6. 等待命令完成其执行。

7. 重新引导 undercloud 以更新操作系统的内核和其他系统软件包：

```
$ sudo reboot
```

8. 稍等片刻，直到节点启动。

9. 以 **stack** 用户的身份登录 undercloud。

除了 undercloud 软件包更新外，建议您保持 overcloud 镜像最新状态，以便保持镜像配置与最新的 **openstack-tripleo-heat-template** 软件包同步。这样可确保在当前准备阶段和实际的快进升级之间成功部署和扩展操作。下面的部分演示了如何在这种情况下更新您的镜像。如果您在准备环境后立即升级您的环境，您可以跳过下一节。

2.4. 为启用 NFV 环境准备更新

如果您的环境启用了网络功能虚拟化(NFV)，请在更新 undercloud 后执行下列步骤，然后再更新 overcloud。

流程

1. 更改自定义环境文件中的 vhost 用户 socket 目录，如 **network-environment.yaml** ：

```
parameter_defaults:
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

2. 将 **ovs-dpdk-permissions.yaml** 文件添加到 **openstack overcloud deploy** 命令中，以便将 qemu 组设为 OVS-DPDK 的 **hugetlbfs** ：

```
 -e environments/ovs-dpdk-permissions.yaml
```

3. 确保所有实例的 vHost 用户端口都为 **dpdkvhostuserclient** 模式。如需更多信息，请参阅 手动更改 vhost 用户端口模式。

2.5. 为 OPENSTACK PLATFORM 10.Z 更新当前的 OVERCLOUD 镜像

undercloud 更新过程可能会从 rhosp-director-images 和 **rhosp-director-images -ipa** 软件包下载新镜像存档。此过程会在 **Red Hat OpenStack Platform 10** 中的 undercloud 上更新这些镜像。

前提条件

- 您已更新到当前 undercloud 版本的最新次版本。

流程

1. 检查 **yum** 日志以确定新镜像存档是否可用：

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

2. 如果有新存档，请将当前镜像替换为新镜像。要安装新镜像，首先从 stack 用户主目录(**/home/stack** / **images**)上的 images 目录中删除任何现有的镜像：

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

3. 在 undercloud 节点上，提供 undercloud 凭证：

```
$ source ~/stackrc
```

4. 解压存档：

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

5. 将 中最新镜像导入到 director，并将节点配置为使用新镜像：

```
$ cd ~/images
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f csv --quote none | sed "1d" | paste -s -d " ")
```

6. 要完成镜像更新，请验证新镜像是否存在：

```
$ openstack image list
$ ls -l /httpboot
```

   director 还保留旧镜像，并使用更新时的时间戳对其进行重命名。如果不再需要这些镜像，请删除它们。

director 现在更新并使用最新的镜像。您无需在更新后重启任何服务。

undercloud 现在使用更新的 OpenStack Platform 10 软件包。接下来，将 overcloud 更新为最新的次要版本。

2.6. 为 OPENSTACK PLATFORM 10.Z 更新当前的 OVERCLOUD 软件包

director 提供了 命令来更新所有 overcloud 节点上的软件包。这样，您可以在当前版本的 OpenStack Platform 环境中执行次要更新。这是 **Red Hat OpenStack Platform 10** 中的小更新。

注意

此步骤还将 overcloud 节点的操作系统更新至 Red Hat Enterprise Linux 7 和 Open vSwitch 的最新版本 2.9。

前提条件

- 您已更新到当前 undercloud 版本的最新次版本。
- 您已执行了 overcloud 的备份。

流程

1. 检查 **rhel_reg_release** 参数的订阅管理配置。如果没有设置此参数，您必须包括它并设置 7.7:

```
parameter_defaults:
  ...
  rhel_reg_release: "7.7"
```

确保保存对 overcloud 订阅管理环境文件的更改。

2. 使用您的原始 **openstack overcloud deploy** 命令更新当前计划，并包含 **--update-plan-only** 选项。例如：

```
$ openstack overcloud deploy --update-plan-only \
  --templates  \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  -e /home/stack/templates/rhel-registration/environment-rhel-registration.yaml \
  [-e <environment_file>|...]
```

--update-plan-only 仅更新存储在 director 中的 Overcloud 计划。使用 **-e** 选项包括与 Overcloud 相关的环境文件及其更新路径。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更为优先。以下列表是环境文件顺序的示例：

- 任何网络隔离文件，包括 heat 模板集合中的初始化文件(**environments/network-isolation.yaml**)，然后是自定义 NIC 配置文件。
- 任何外部负载均衡环境文件。
- 任何存储环境文件。
- 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。

3. 创建 overcloud 的静态清单文件：

```
$ tripleo-ansible-inventory --ansible_ssh_user heat-admin --static-yaml-inventory
~/inventory.yaml
```

如果您使用不同于 **overcloud** 的默认 overcloud 名称，请使用 **--plan** 选项来设置 overcloud 的名称。

4. 创建一个包含在所有节点上将操作系统版本设置为 Red Hat Enterprise Linux 7.7 的任务：

```
$ cat > ~/set_release.yaml <<'EOF'
- hosts: all
  gather_facts: false
  tasks:
    - name: set release to 7.7
      command: subscription-manager release --set=7.7
      become: true
EOF
```

5. 运行 set_release.yaml playbook：

```
$ ansible-playbook -i ~/inventory.yaml -f 25 ~/set_release.yaml --limit
undercloud,Controller,Compute
```

使用 **--limit** 选项，将内容应用到所有 Red Hat OpenStack Platform 节点。

6. 使用 **openstack overcloud update** 命令在所有节点上执行软件包更新：

```
$ openstack overcloud update stack -i overcloud
```

-i 运行一个互动模式来按顺序更新每个节点。更新过程完成后，脚本会为您提供一个断点进行确认。如果没有 **-i** 选项，更新会在第一个断点保持暂停。因此，必须包括 **-i** 选项。

脚本执行以下功能：

a. 该脚本在一对一节点上运行：

   i. 对于 Controller 节点，这意味着完全软件包更新。

   ii. 对于其他节点，这意味着仅更新 Puppet 模块。

b. Puppet 同时在所有节点上运行：

   i. 对于 Controller 节点，Puppet 运行会同步配置。

   ii. 对于其他节点，Puppet 运行会更新软件包的其余部分并同步配置。

7. 更新过程开始。在此过程中，director 报告 **IN_PROGRESS** 状态，并定期提示您清除断点。例如：

```
starting package update on stack overcloud
IN_PROGRESS
IN_PROGRESS
WAITING
on_breakpoint: [u'overcloud-compute-0', u'overcloud-controller-2', u'overcloud-controller-1',
u'overcloud-controller-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear 49913767-e2dd-4772-
b648-81e198f5ed00), no=cancel update, C-c=quit interactive mode:
```

按 Enter 清除 **on_breakpoint** 列表中的最后一个节点中的断点。这将开始该节点的更新。

8. 这个脚本会自动预定义节点的更新顺序：

   - 每个 Controller 节点单独
   - 每个单独 Compute 节点
   - 每个 Ceph Storage 节点
   - 所有其他节点单独

   建议您使用这个顺序来确保更新成功，特别是：

   a. 单独清除每个 Controller 节点的断点。如果节点的服务在更新后必须重启，则每个 Controller 节点都需要一个单独的软件包更新。这可减少到其他 Controller 节点上高可用性服务的中断。

   b. 在 Controller 节点更新后，清除每个 Compute 节点的断点。您还可以键入 Compute 节点名称来清除特定节点上的断点，或使用基于 Python 的正则表达式来一次性在多个 Compute 节点上清除断点。

   c. 清除每个 Ceph Storage 节点的断点。您还可以键入 Ceph Storage 节点名称来清除特定节点上的断点，或使用基于 Python 的正则表达式来一次性清除多个 Ceph Storage 节点上的断点。

   d. 清除所有剩余的断点以更新剩余的节点。您还可以键入节点名称来清除特定节点上的断点，或使用基于 Python 的正则表达式来一次性在多个节点上清除断点。

   e. 等待所有节点完成其更新。

9. 更新命令会在更新完成后报告 **COMPLETE** 状态：

```
...
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

10. 如果您为 Controller 节点配置了隔离功能，更新过程可能会禁用它。更新过程完成后，在其中一个 Controller 节点上使用以下命令重新启用隔离：

```
$ sudo pcs property set stonith-enabled=true
```

更新过程不会自动重启 Overcloud 中的任何节点。内核和其他系统软件包的更新需要重启。检查每个节点上的 **/var/log/yum.log** 文件，以查看 **内核或 openvswitch** 软件包是否已更新其主版本或次版本。如果存在，请按照以下流程重新引导每个节点。

2.7. 重新引导控制器和可组合节点

以下流程根据可组合角色重启控制器节点和独立节点。这会排除 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：

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs cluster stop
```

3. 重新引导节点：

```
[heat-admin@overcloud-controller-0 ~]$ sudo reboot
```

4. 稍等片刻，直到节点启动。

5. 检查服务。例如：

   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：

   ```
   [heat-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：

   ```
   [heat-admin@overcloud-controller-0 ~]$ sudo systemctl status
   ```

   c. 对所有 Controller 和可组合节点重复这些步骤。

2.8. 重新引导 CEPH STORAGE (OSD) 集群

以下流程重启 Ceph Storage (OSD)节点集群。

流程

1. 登录 Ceph MON 或 Controller 节点，并暂时禁用 Ceph Storage 集群重新平衡：

```
$ sudo ceph osd set noout
$ sudo ceph osd set norebalance
```

2. 选择第一个 Ceph Storage 节点以重新引导并登录。

3. 重新引导节点：

```
$ sudo reboot
```

4. 稍等片刻，直到节点启动。

5. 登录到 Ceph MON 或 Controller 节点并检查集群状态：

```
$ sudo ceph -s
```

   确认 **pgmap** 报告的所有 **pgs** 的状态是否都正常 (**active+clean**)。

6. 从 Ceph MON 或 Controller 节点注销，重新引导下一个 Ceph Storage 节点，并检查其状态。重复此流程，直到您已重新引导所有 Ceph 存储节点。

7. 完成之后，登录 Ceph MON 或 Controller 节点，然后重新启用集群重新平衡：

```
$ sudo ceph osd unset noout
$ sudo ceph osd unset norebalance
```

8. 执行最后的状态检查，确认集群报告 **HEALTH_OK**：

```
$ sudo ceph status
```

2.9. 重新引导 COMPUTE 节点

重新引导 Compute 节点涉及以下工作流：

- 选择一个 Compute 节点来重新引导并禁用它，使其不置备新实例。
- 将实例迁移到另一个 Compute 节点，以最小化实例停机时间。
- 重新引导空的 Compute 节点并启用它。

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. 要识别您要重新引导的 Compute 节点，请列出所有 Compute 节点：

```
$ source ~/stackrc
(undercloud) $ openstack server list --name compute
```

3. 在 overcloud 中，选择 Compute 节点并禁用它：

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

4. 列出 Compute 节点上的所有实例：

```
(overcloud) $ openstack server list --host <hostname> --all-projects
```

5. 迁移您的实例。有关迁移策略的更多信息，请参阅 Compute 节点之间迁移虚拟机。
6. 登录到 Compute 节点并重新引导它：

```
[heat-admin@overcloud-compute-0 ~]$ sudo reboot
```

7. 稍等片刻，直到节点启动。
8. 启用 Compute 节点：

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set <hostname> nova-compute --enable
```

9. 验证 Compute 节点是否已启用：

```
(overcloud) $ openstack compute service list
```

2.10. 验证系统软件包

在升级前，undercloud 节点和所有 overcloud 节点都应该使用以下软件包的最新版本：

软件包	版本
openvswitch	至少 2.9
qemu-img-rhev	至少 2.10
qemu-kvm-common-rhev	至少 2.10
qemu-kvm-rhev	至少 2.10
qemu-kvm-tools-rhev	至少 2.10

流程

1. 登录到节点。
2. 运行 **yum** 检查系统软件包：

```
$ sudo yum list qemu-img-rhev qemu-kvm-common-rhev qemu-kvm-rhev qemu-kvm-tools-rhev openvswitch
```

3. 运行 **ovs-vsctl** 检查当前正在运行的版本：

```
$ sudo ovs-vsctl --version
```

4. 对所有节点重复这些步骤。

undercloud 现在使用更新的 OpenStack Platform 10 软件包。使用接下来的几个流程检查系统是否处于工作状态。

2.11. 验证 OPENSTACK PLATFORM 10 UNDERCLOUD

以下是一组步骤，用于在升级前检查 **Red Hat OpenStack Platform 10** undercloud 的功能。

流程

1. 查找 undercloud 访问详情：

```
$ source ~/stackrc
```

2. 检查失败的 Systemd 服务：

```
$ sudo systemctl list-units --state=failed 'openstack*' 'neutron*' 'httpd' 'docker'
```

3. 检查 undercloud 可用空间：

```
$ df -h
```

使用 "Undercloud 要求" 作为基础来确定您是否有足够的可用空间。

4. 如果您在 undercloud 上安装了 NTP，请检查时钟是否已同步：

```
$ sudo ntpstat
```

5. 检查 undercloud 网络服务：

```
$ openstack network agent list
```

所有代理都应 处于活动状态，其状态应为 **UP**。

6. 检查 undercloud 计算服务：

```
$ openstack compute service list
```

所有代理的状态都应 为启用状态，其状态应为 **up**

相关信息

- 以下解决方案文章演示了如何删除 OpenStack Orchestration (heat)数据库中删除的堆栈条目：https://access.redhat.com/solutions/2215131 https://access.redhat.com/solutions/2215131

2.12. 验证 OPENSTACK PLATFORM 10 OVERCLOUD

以下是一组步骤，用于在升级前检查 **Red Hat OpenStack Platform 10** overcloud 的功能。

流程

1. 查找 undercloud 访问详情：

```
$ source ~/stackrc
```

2. 检查裸机节点的状态：

```
$ openstack baremetal node list
```

所有节点均应具有有效的电源状态(**on**)和维护模式，应为 **false**。

3. 检查失败的 Systemd 服务：

```
$ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "===
$NODE ===" ; ssh heat-admin@$NODE "sudo systemctl list-units --state=failed 'openstack*'
'neutron*' 'httpd' 'docker' 'ceph*'" ; done
```

4. 检查与所有服务的 HAProxy 连接。获取 **haproxy.stats** 服务的 Control Plane VIP 地址和身份验证信息：

```
$ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh
heat-admin@$NODE sudo 'grep "listen haproxy.stats" -A 6 /etc/haproxy/haproxy.cfg'
```

5. 使用上一步中获取的连接和身份验证信息来检查 RHOSP 服务的连接状态。
如果没有启用 SSL，请在以下 cURL 请求中使用这些详情：

```
$ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi "
(frontend|backend)" | awk -F',' '{ print $1" "$2" "$18 }'
```

如果启用了 SSL，则在以下 cURL 请求中使用这些详情：

```
curl -s -u admin:<PASSWORD> "https://<HOSTNAME>:1993/;csv" | egrep -vi "
(frontend|backend)" | awk -F',' '{ print $1" "$2" "$18 }'
```

将 &lt **;PASSWORD&** gt; 和 &lt **;IP ADDRESS > 或 <HOSTNAME** > 值替换为 **haproxy.stats** 服务中的相应信息。生成的列表显示每个节点上的 OpenStack Platform 服务及其连接状态。

6. 检查 overcloud 数据库复制健康状况：

```
$ for NODE in $(openstack server list --name controller -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo clustercheck" ; done
```

7. 检查 RabbitMQ 集群健康状况：

```
$ for NODE in $(openstack server list --name controller -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo rabbitmqctl node_health_check" ;
done
```

8. 检查 Pacemaker 资源健康状况：

```
$ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh
heat-admin@$NODE "sudo pcs status"
```

查找：

- 所有集群节点 **在线**。
- 任何集群节点上都没有 **停止** 资源。
- 没有 **失败的** pacemaker 操作。

9. 检查每个 overcloud 节点上的磁盘空间：

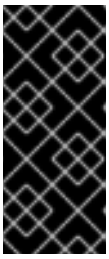
```
$ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "===
$NODE ===" ; ssh heat-admin@$NODE "sudo df -h --output=source,fstype,avail -x overlay -
x tmpfs -x devtmpfs" ; done
```

10. 检查 overcloud Ceph Storage 集群健康状态。以下命令在 Controller 节点上运行 **ceph** 工具来检查集群：

```
$ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh
heat-admin@$NODE "sudo ceph -s"
```

11. 检查 Ceph Storage OSD 是否有可用空间。以下命令在 Controller 节点上运行 **ceph** 工具来检查可用空间：

```
$ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph df"
```



重要

每个 Ceph 对象存储守护进程(OSD)的 PG 数量不得超过 250 个。每个 OSD 有更多 PG 升级 Ceph 节点会导致警告状态，并可能导致升级过程失败。在开始升级过程前，您可以增加每个 OSD 的 PG 数量。有关诊断并排除这个问题的更多信息，请参阅 OpenStack FFU 从 10 到 13 超时，因为一个或多个 OSD 中分配的 Ceph PG 大于 250 个。

12. 检查时钟是否在 overcloud 节点上同步

```
$ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo ntpstat" ; done
```

13. 查找 overcloud 访问详情：

```
$ source ~/overcloudrc
```

14. 检查 overcloud 网络服务：

```
$ openstack network agent list
```

所有代理都应 处于活动状态，其状态应为 **UP**。

15. 检查 overcloud 计算服务：

```
$ openstack compute service list
```

所有代理的状态都应 为启用状态，其状态应为 **up**

16. 检查 overcloud 卷服务：

```
$ openstack volume service list
```

所有代理的状态都应 为启用状态，其状态应为 。

相关信息

- 请参阅文章 "我如何验证我的 OpenStack 环境是通过红帽推荐的配置进行部署？"。本文提供有关如何检查您的 Red Hat OpenStack Platform 环境并对红帽的建议调整配置的一些信息。

- 参阅 "Red Hat Enterprise Linux OpenStack Platform"的"数据库大小管理"，检查并清理 overcloud 上 OpenStack Platform 服务的未使用数据库记录。

2.13. 启用 NFV 环境的最终更新

如果您的环境启用了网络功能虚拟化(NFV)，则需要在更新 undercloud 和 overcloud 后执行下列步骤。

流程

您需要迁移现有的 OVS-DPDK 实例，以确保 vhost 套接字模式从 dkdpvhostuser 变为 OVS 端口中的 dkdpvhostuserclient 模式。我们建议您对现有的实例进行快照，并根据该快照镜像重建新实例。有关实例快照的详情，请参阅管理实例快照。

快照实例并从快照引导新实例：

1. 查找 overcloud 访问详情：

```
$ source ~/overcloudrc
```

2. 查找您要生成快照的实例的服务器 ID：

```
$ openstack server list
```

3. 在生成快照前关闭源实例，以确保所有数据清理到磁盘：

```
$ openstack server stop SERVER_ID
```

4. 创建实例的快照镜像：

```
$ openstack image create --id SERVER_ID SNAPSHOT_NAME
```

5. 使用这个快照镜像引导新实例：

```
$ openstack server create --flavor DPDK_FLAVOR --nic net-id=DPDK_NET_ID --image SNAPSHOT_NAME INSTANCE_NAME
```

6. （可选）验证新实例状态为 ACTIVE ：

```
$ openstack server list
```

对需要快照和重启的所有实例重复此步骤。

2.14. 保留 YUM 历史记录

完成 overcloud 的次要更新后，保留 yum 历史记录。当您需要撤销任何可能的回滚操作的 yum 事务时，这些信息非常有用。

流程

1. 在每个节点上，运行以下命令将节点的整个 yum 历史记录保存到文件中：

```
$ sudo yum history list all > /home/heat-admin/$(hostname)-yum-history-all
```

2. 在每个节点上，运行以下命令来保存最后的 yum history 项的 ID：

```
$ sudo yum history list all | head -n 5 | tail -n 1 | awk '{print $1}' > /home/heat-admin/$(hostname)-yum-history-all-last-id
```

3. 将这些文件复制到安全位置。

2.15. 后续步骤

完成准备阶段后，您可以使用 第 3 章 *升级 undercloud* 中的步骤将 undercloud 从 10 升级到 13。

第 3 章 升级 UNDERCLOUD

此流程将 undercloud 升级至 Red Hat OpenStack Platform 13。您可以通过通过 undercloud 的每个后续版本从 OpenStack Platform 10 升级到 OpenStack Platform 13 来完成此操作。

3.1. 将 UNDERCLOUD 升级到 OPENSTACK PLATFORM 11

此流程将 undercloud 工具集和核心 Heat 模板集合升级到 OpenStack Platform 11 版本。

流程

1. 以 stack 用户身份登录 director。

2. 禁用当前的 OpenStack Platform 存储库：

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
```

3. 启用新的 OpenStack Platform 存储库：

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
```

4. 禁用 overcloud 基础镜像的更新：

```
$ sudo yum-config-manager --setopt=exclude=rhosp-director-images* --save
```

5. 停止主 OpenStack Platform 服务：

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

注意

这会导致 undercloud 的停机时间短。overcloud 在 undercloud 升级过程中仍然可以正常工作。

6. 默认的 Provisioning/Control Plane 网络已从 192.0.2.0/24 更改为 192.168.24.0/24。如果您

在之前的 undercloud.conf 文件中使用了默认网络值，则 Provisioning/Control Plane 网络被设置为 192.0.2.0/24。这意味着您需要在 undercloud.conf 文件中设置某些参数来继续使用 192.0.2.0/24 网络。这些参数：

- **local_ip**
- **network_gateway**
- **undercloud_public_vip**
- **undercloud_admin_vip**
- **network_cidr**
- **masquerade_network**
- **dhcp_start**
- **dhcp_end**

在 undercloud.conf 中设置网络值，以确保在将来的升级过程中继续使用 192.0.2.0/24 CIDR。在运行 openstack undercloud upgrade 命令前，请确保您的网络配置设置正确。

7. **运行 yum 以升级 director 的主软件包：**

```
$ sudo yum update -y instack-undercloud openstack-puppet-modules openstack-tripleo-common python-tripleoclient
```

8. **运行以下命令来升级 undercloud：**

```
$ openstack undercloud upgrade
```

9. 等待 undercloud 升级过程完成。

您已将 undercloud 升级到 OpenStack Platform 11 版本。

3.2. 将 UNDERCLOUD 升级到 OPENSTACK PLATFORM 12

此流程将 undercloud 工具集和核心 Heat 模板集合升级到 OpenStack Platform 12 发行版本。

流程

1. 以 stack 用户身份登录 director。
2. 禁用当前的 OpenStack Platform 存储库：

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-11-rpms
```

3. 启用新的 OpenStack Platform 存储库：

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
```

4. 禁用 overcloud 基础镜像的更新：

```
$ sudo yum-config-manager --setopt=exclude=rhosp-director-images* --save
```

5. 运行 yum 以升级 director 的主软件包：

```
$ sudo yum update -y python-tripleoclient
```

6. 编辑 /home/stack/undercloud.conf 文件，检查 enabled_drivers 参数是否不包含 pxe_ssh 驱动程序。这个驱动已弃用，而是使用 Virtual Baseboard Management Controller (VBMC)并从 Red Hat OpenStack Platform 中删除。有关这个新驱动程序和迁移说明的更多信息，请参阅 *Director 安装和使用指南中的* 附录 "Virtual Baseboard Management Controller (VBMC) "。
7. 运行以下命令来升级 undercloud：

```
$ openstack undercloud upgrade
```

8. **等待 undercloud 升级过程完成。**

您已将 undercloud 升级到 OpenStack Platform 12 发行版本。

3.3. 将 UNDERCLOUD 升级到 OPENSTACK PLATFORM 13

此流程将 undercloud 工具集和核心 Heat 模板集合升级到 OpenStack Platform 13 版本。

流程

1. **以 stack 用户身份登录 director。**

2. **禁用当前的 OpenStack Platform 存储库：**

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
```

3. **将 RHEL 版本设置为 RHEL 7.9:**

```
$ sudo subscription-manager release --set=7.9
```

4. **启用新的 OpenStack Platform 存储库：**

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
```

5. **重新启用 overcloud 基础镜像的更新：**

```
$ sudo yum-config-manager --setopt=exclude= --save
```

6. **运行 yum 以升级 director 的主软件包：**

```
$ sudo yum update -y python-tripleoclient
```

7. 运行以下命令来升级 **undercloud**：

```
$ openstack undercloud upgrade
```

8. 等待 **undercloud** 升级过程完成。

9. 重新引导 **undercloud** 以更新操作系统的内核和其他系统软件包：

```
$ sudo reboot
```

10. 稍等片刻，直到节点启动。

您已将 **undercloud** 升级到 **OpenStack Platform 13** 版本。

3.4. 在 UNDERCLOUD 上禁用已弃用的服务

升级 **undercloud** 后，您必须禁用已弃用的 **openstack-glance-registry** 和 **mongod** 服务。

流程

1. 以 **stack** 用户的身份登录 **undercloud**。

2. 停止并禁用 **openstack-glance-registry** 服务：

```
$ sudo systemctl stop openstack-glance-registry
$ sudo systemctl disable openstack-glance-registry
```

3. 停止并禁用 **mongod** 服务：

```
$ sudo systemctl stop mongod
$ sudo systemctl disable mongod
```

3.5. 后续步骤

undercloud 升级已完成。现在，您可以为容器镜像配置源。

第 4 章 配置容器镜像源

容器化 overcloud 需要访问具有所需容器镜像的注册表。本章介绍了如何准备 registry 和 overcloud 配置以将容器镜像用于 Red Hat OpenStack Platform。

本指南提供了几个用例，用于将 overcloud 配置为使用 registry。在尝试这些用例之前，建议熟悉如何使用镜像准备命令。如需更多信息，请参阅 第 4.2 节 “容器镜像准备命令用法”。

4.1. REGISTRY 方法

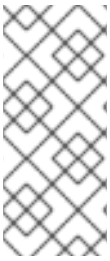
Red Hat OpenStack Platform 支持以下 registry 类型：

远程 Registry

overcloud 直接从 registry.redhat.io 中提取容器镜像。此方法是生成初始配置的最简单方法。但是，每个 overcloud 节点直接从 Red Hat Container Catalog 拉取每个镜像，这可能会导致网络拥塞和速度较慢的部署。另外，所有 overcloud 节点都需要访问互联网来访问 Red Hat Container Catalog。

本地 Registry

undercloud 使用 docker-distribution 服务作为 registry。这允许 director 同步 registry.redhat.io 中的镜像，并将它们推送到 docker-distribution registry。在创建 overcloud 时，overcloud 从 undercloud 的 docker-distribution 注册表中提取容器镜像。这个方法允许您在内部存储 registry，这可以加快部署并减少网络拥塞。但是，undercloud 只能充当基本 registry，为容器镜像提供有限的生命周期管理。



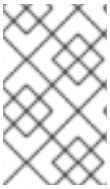
注意

docker-distribution 服务与 docker 分开。Docker 用于提取并推送镜像到 docker-distribution 注册表，不为 overcloud 提供镜像。overcloud 从 docker-distribution 注册表拉取镜像。

Satellite Server

管理容器镜像的完整应用程序生命周期，并通过 Red Hat Satellite 6 服务器发布它们。overcloud 从 Satellite 服务器拉取镜像。此方法提供了用于存储、管理和部署 Red Hat OpenStack Platform 容器的企业级解决方案。

从列表中选择一个方法，再继续配置 registry 的详情。



注意

在为多架构云构建时，不支持本地 **registry** 选项。

4.2. 容器镜像准备命令用法

本节概述如何使用 **openstack overcloud** 容器镜像准备 命令，包括关于该命令的各种选项的概念信息。

为 **Overcloud** 生成容器镜像环境文件

openstack overcloud 容器镜像准备 命令的一个主要用途是创建含有 **overcloud** 使用的镜像列表的环境文件。您可以使用 **overcloud** 部署命令包括此文件，如 **openstack overcloud deploy**。**openstack overcloud** 容器镜像准备命令将以下选项用于此功能：

--output-env-file

定义生成的环境文件名称。

以下片段是该文件的内容示例：

```
parameter_defaults:
  DockerAodhApiImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
  DockerAodhConfigImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
...
```

环境文件还包含设置为 **undercloud registry** 的 **IP** 地址和端口的 **DockerInsecureRegistryAddress** 参数。此参数将 **overcloud** 节点配置为在没有 **SSL/TLS** 认证的情况下从 **undercloud registry** 访问镜像。

为导入方法生成容器镜像列表

如果要将 **OpenStack Platform** 容器镜像导入到其他 **registry** 源，您可以生成镜像列表。列表语法主要用于将容器镜像导入到 **undercloud** 上的容器注册表，但您可以修改此列表的格式，以适应其他导入方法，如 **Red Hat Satellite 6**。

openstack overcloud 容器镜像准备命令将以下选项用于此功能：

--output-images-file

定义导入列表生成的文件名。

以下是此文件的内容示例：

```
container_images:
- imagename: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
- imagename: registry.redhat.io/rhosp13/openstack-aodh-evaluator:13.0-34
...
```

为容器镜像设置命名空间

--output-env-file 和 --output-images-file 选项都需要一个命名空间来生成生成的镜像位置。openstack overcloud 容器镜像准备命令使用以下选项来设置容器镜像的源位置，以拉取：

--namespace

定义容器镜像的命名空间。这通常是包含目录的主机名或 IP 地址。

--prefix

定义在镜像名称前添加的前缀。

因此，director 使用以下格式生成镜像名称：

- [NAMESPACE]/[PREFIX][IMAGE NAME]

设置容器镜像标签

使用 --tag 和 --tag-from-label 选项为每个容器镜像设置标签。

--tag

为来自源的所有镜像设置特定标签。如果您只使用这个选项，director 会使用该标签拉取所有容器镜像。但是，如果您将此选项与 --tag-from-label 结合使用，director 将 --tag 用作源镜像来根据标签识别特定的版本标签。默认将 --tag 选项设置为 latest。

--tag-from-label

使用指定容器镜像标签的值来发现并拉取每个镜像的 versioned 标签。director 会检查使用您为 --tag 设置的值标记的每个容器镜像，然后使用容器镜像标签 来构建新标签，director 从 registry 拉取。例如，如果您设置了 --tag-from-label {version}-{release}，director 会使用 version 和 release

标签来构造新标签。对于一个容器，版本 可能被设置为 13.0，release 可能会设置为 34，这会导致标签 13.0-34。

重要

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。此版本格式为 {version}-{release}，每个容器镜像都作为容器元数据中的标签存储。这个版本格式有助于从一个 {release} 更新至下一个版本。因此，在运行 openstack overcloud 容器镜像准备 命令时，必须始终使用 --tag-from-label {version}-{release}。不要自行使用 --tag 来拉取容器镜像。例如，使用 --tag latest 本身会在执行更新时导致问题，因为 director 需要更改标签来更新容器镜像。

4.3. 用于其他服务的容器镜像

director 仅为核心 OpenStack Platform 服务准备容器镜像。些额外的功能使用需要额外容器镜像的服务。您可以使用环境文件启用这些服务。openstack overcloud 容器镜像准备命令使用以下选项包含环境文件及其相应容器镜像：

-e

包括环境文件以启用额外容器镜像。

下表提供了使用容器镜像及其对应环境文件位于 /usr/share/openstack-tripleo-heat-templates 目录中的相应服务的示例列表。

Service	环境文件
Ceph Storage	**environments/ceph-ansible/ceph-ansible.yaml**
collectd	**environments/services-docker/collectd.yaml**
Congress	**environments/services-docker/congress.yaml**
Fluentd	**environments/services-docker/fluentd.yaml**
OpenStack Bare Metal (ironic)	**environments/services-docker/ironic.yaml**
OpenStack 数据处理(sahara)	**environments/services-docker/sahara.yaml**
OpenStack EC2-API	**environments/services-docker/ec2-api.yaml**
OpenStack Key Manager (barbican)	**environments/services-docker/barbican.yaml**

Service	环境文件
OpenStack Load Balancing-as-a-Service (octavia)	**environments/services-docker/octavia.yaml**
OpenStack Shared File System Storage (manila)	**environments/manila-{backend-name}-config.yaml** 注意：如需更多信息，请参阅 OpenStack 共享文件系统服务(manila)。
Open Virtual Network (OVN)	**environments/services-docker/neutron-ovn-dvr-ha.yaml**
Sensu	**environments/services-docker/sensu-client.yaml**

下面几节提供了包括其他服务的示例。

Ceph Storage

如果使用 overcloud 部署 Red Hat Ceph Storage 集群，您需要包含 /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml 环境文件。此文件可以在 overcloud 中启用可组合容器化服务，而 director 需要知道这些服务需要被启用来准备其镜像。

除了此环境文件外，您还需要定义 Ceph Storage 容器位置，这与 OpenStack Platform 服务不同。使用 --set 选项设置特定于 Ceph Storage 的以下参数：

--set ceph_namespace

定义 Ceph Storage 容器镜像的命名空间。这个功能与 --namespace 选项类似。

--set ceph_image

定义 Ceph Storage 容器镜像的名称。通常，这是 rhceph-3-rhel7。

--set ceph_tag

定义用于 Ceph Storage 容器镜像的标签。这个功能与 --tag 选项类似。当指定 --tag-from-label 时，从该标签开始发现 versioned 标签。

以下片段是在容器镜像文件中包含 Ceph Storage 的示例：

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  --set ceph_namespace=registry.redhat.io/rhceph \
```

```
  --set ceph_image=rhceph-3-rhel7 \
  --tag-from-label {version}-{release} \
  ...
```

OpenStack Bare Metal (ironic)

如果在 overcloud 中部署 OpenStack Bare Metal (ironic)，您需要包含 /usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml 环境文件，以便 director 能够准备镜像。以下片段是一个如何包含此环境文件的示例：

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml \
  ...
```

OpenStack 数据处理(sahara)

如果在 overcloud 中部署 OpenStack 数据处理(sahara)，您需要包含 /usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml 环境文件，以便 director 能够准备镜像。以下片段是一个如何包含此环境文件的示例：

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml \
  ...
```

OpenStack Neutron SR-IOV

如果在 overcloud 中部署 OpenStack Neutron SR-IOV，请包含 /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml 环境文件，以便 director 能够准备镜像。默认 Controller 和 Compute 角色不支持 SR-IOV 服务，因此您必须使用 -r 选项包括包含 SR-IOV 服务的自定义角色文件。以下片段是一个如何包含此环境文件的示例：

```
$ openstack overcloud container image prepare \
  ...
  -r ~/custom_roles_data.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml \
  ...
```

OpenStack Load Balancing-as-a-Service (octavia)

如果在 overcloud 中部署 OpenStack Load Balancing-as-a-Service，请包含 /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml 环境文件，以便 director 能够准备镜像。以下片段是一个如何包含此环境文件的示例：

```
$ openstack overcloud container image prepare \
  ...
```

```
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml
\
  ...
```

OpenStack Shared File System (manila)

使用格式 manila-{backend-name}-config.yaml，您可以选择受支持的后端来部署具有该后端的 Shared File System。通过包括以下任何环境文件，可以准备共享文件系统服务容器：

```
environments/manila-isilon-config.yaml
environments/manila-netapp-config.yaml
environments/manila-vmax-config.yaml
environments/manila-cephfsnative-config.yaml
environments/manila-cephfsganesha-config.yaml
environments/manila-unity-config.yaml
environments/manila-vnx-config.yaml
```

有关自定义和部署环境文件的更多信息，请参阅以下资源：

- *通过 NFS 后端指南为共享文件系统服务部署更新的环境*
- 使用 NetApp Back Ends 在共享文件系统服务 *的 NetApp 后端部署*共享文件系统服务
- 使用共享文件系统服务的 *CephFS*后端部署共享文件系统服务

4.4. 使用 RED HAT REGISTRY 作为远程 REGISTRY 源

红帽在 registry.redhat.io 上托管 overcloud 容器镜像。从远程 registry 中拉取镜像是最简单的方法，因为注册表已经配置，且所有需要是您要拉取的镜像的 URL 和命名空间。但是，在创建 overcloud 的过程中，overcloud 节点会从远程存储库拉取所有镜像，这样可将您的外部连接整合到一起。因此，不建议在生产环境中使用这个方法。对于生产环境，请改为使用以下方法之一：

- 设置本地 registry
- 在 Red Hat Satellite 6 上托管镜像

流程

1.

要在 overcloud 部署中直接从 registry.redhat.io 拉取镜像，需要一个环境文件来指定镜像参数。运行以下命令以生成容器镜像环境文件：

```
(undercloud) $ sudo openstack overcloud container image prepare \
  --namespace=registry.redhat.io/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

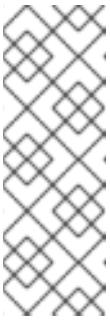
- 使用 -e 选项包括可选服务的任何环境文件。
- 使用 -r 选项包括自定义角色文件。
- 如果使用 Ceph Storage，包括额外的参数来定义 Ceph Storage 容器镜像位置：-- set ceph_namespace ,--set ceph _image,--set ceph_tag.

2. 修改 overcloud_images.yaml 文件，并包括下列参数以在部署期间与 registry.redhat.io 进行身份验证：

```
ContainerImageRegistryLogin: true
ContainerImageRegistryCredentials:
  registry.redhat.io:
    <USERNAME>: <PASSWORD>
```

- 将 & lt;USERNAME& gt ; 和 <PASSWORD > 替换为 registry.redhat.io 的凭证。

  overcloud_images.yaml 文件包含 undercloud 上的镜像位置。在您的部署中包含此文件。



注意

在运行 openstack overcloud deploy 命令前，您必须登录远程 registry：

```
(undercloud) $ sudo docker login registry.redhat.io
```

注册表配置已就绪。

4.5. 使用 UNDERCLOUD 作为本地 REGISTRY

您可以在 undercloud 上配置本地 registry，以存储 overcloud 容器镜像。

您可以使用 director 从 registry.redhat.io 拉取每个镜像，并将每个镜像推送到 undercloud 上运行的 docker-distribution registry。当使用 director 创建 overcloud 时，在 overcloud 创建过程中，节点会从 undercloud docker-distribution registry 中拉取相关镜像。

这会为您的内部网络保留容器镜像的网络流量，这并不会整合外部网络连接，并可加快部署过程。

流程

1. 查找本地 undercloud registry 的地址。地址使用以下模式：

```
<REGISTRY_IP_ADDRESS>:8787
```

使用 undercloud 的 IP 地址，之前使用 undercloud.conf 文件中的 local_ip 参数进行设置。对于下列命令，该地址被假定为 192.168.24.1:8787。

2. 登录到 registry.redhat.io ：

```
(undercloud) $ docker login registry.redhat.io --username $RH_USER --password
$RH_PASSWD
```

3. 创建模板，将镜像上传到本地 registry，以及环境文件以引用这些镜像：

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=registry.redhat.io/rhosp13 \
  --push-destination=192.168.24.1:8787 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml \
  --output-images-file /home/stack/local_registry_images.yaml
```

- 使用 -e 选项包括可选服务的任何环境文件。
- 使用 -r 选项包括自定义角色文件。

- 如果使用 Ceph Storage，包括额外的参数来定义 Ceph Storage 容器镜像位置：-- set ceph_namespace ,--set ceph _image,--set ceph_tag.

4. 验证是否已创建以下两个文件：

- local_registry_images.yaml，其中包含来自远程源的容器镜像信息。使用此文件将 Red Hat Container Registry (registry.redhat.io)中的镜像拉取到 undercloud。

- overcloud_images.yaml，其中包含 undercloud 上的最终镜像位置。您的部署会包含这个文件。

5. 从远程 registry 拉取容器镜像并将其推送到 undercloud registry：

```
(undercloud) $ openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

拉取所需的镜像可能需要一些时间，具体取决于您的网络和 undercloud 磁盘的速度。

注意

容器镜像大约消耗 10 GB 磁盘空间。

6. 该镜像现在存储在 undercloud 的 docker-distribution 注册表中。要查看 undercloud 的 docker-distribution registry 中的镜像列表，请运行以下命令：

```
(undercloud) $  curl http://192.168.24.1:8787/v2/_catalog | jq .repositories[]
```

注意

本身中的 _catalog 资源仅显示 100 个镜像。要显示更多镜像，请使用 ?n=<interger& gt; 查询字符串及 _catalog 资源来显示更多镜像数量：

```
(undercloud) $  curl http://192.168.24.1:8787/v2/_catalog?n=150 | jq
.repositories[]
```

要查看特定镜像的标签列表，请使用 skopeo 命令：

```
(undercloud) $ curl -s http://192.168.24.1:8787/v2/rhosp13/openstack-keystone/tags/list | jq
.tags
```

要验证标记的镜像，使用 skopeo 命令：

```
(undercloud) $ skopeo inspect --tls-verify=false
docker://192.168.24.1:8787/rhosp13/openstack-keystone:13.0-44
```

注册表配置已就绪。

4.6. 使用 SATELLITE 服务器作为 REGISTRY

Red Hat Satellite 6 提供了注册表同步功能。通过该功能可将多个镜像提取到 Satellite 服务器中，作为应用程序生命周期的一部分加以管理。Satellite 也可以作为 registry 供其他启用容器功能的系统使用。有关管理容器镜像的更多信息，请参阅 ***Red Hat Satellite 6 内容管理指南*** 中的“管理容器镜像”。

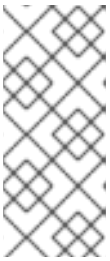
以下操作过程示例中使用了 Red Hat Satellite 6 的 hammer 命令行工具和一个名为 ACME 的示例组织。请将该组织替换为您自己 Satellite 6 中的组织。

流程

1. 创建模板以将镜像拉取到本地 registry：

```
$ source ~/stackrc
(undercloud) $ openstack overcloud container image prepare \
  --namespace=rhosp13 \
  --prefix=openstack- \
  --output-images-file /home/stack/satellite_images
```

- 使用 **-e** 选项包括可选服务的任何环境文件。
- 使用 **-r** 选项包括自定义角色文件。
- 如果使用 Ceph Storage，包括额外的参数来定义 Ceph Storage 容器镜像位置：**-- set ceph_namespace ,--set ceph _image,--set ceph_tag.**



注意

此版本的 openstack overcloud 容器镜像准备命令以 registry.redhat.io 上的 registry 为目标，以生成镜像列表。它使用与 openstack overcloud 容器镜像准备 命令不同的值。

2. 这会利用容器镜像信息创建名为 satellite_images 的文件。您将使用此文件将容器镜像同步到卫星 6 服务器。

3. 从 satellite_images 文件中删除特定于 YAML 的信息，并将其转换为仅包含镜像列表的平面文件。以下 sed 命令完成此操作：

```
(undercloud) $ awk -F ':' '{if (NR!=1) {gsub("[[:space:]]", ""); print $2}}' ~/satellite_images >
~/satellite_images_names
```

这为您提供了拉取到 Satellite 服务器的镜像列表。

4. 将 satellite_images_names 文件复制到包含 Satellite 6 hammer 工具的系统中。或者，根据 *Hammer CLI 指南*中的说明将 hammer 工具安装到 undercloud 中。

5. 运行以下 hammer 命令，在您的 Satellite 组织创建新产品(OSP13 容器)：

```
$ hammer product create \
  --organization "ACME" \
  --name "OSP13 Containers"
```

该定制产品将会包含我们的镜像。

6. 为产品添加基本容器镜像：

```
$ hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
  --url https://registry.redhat.io \
  --docker-upstream-name rhosp13/openstack-base \
  --name base
```

7. 添加 satellite_images 文件中的 overcloud 容器镜像。

```
$ while read IMAGE; do \
  IMAGENAME=$(echo $IMAGE | cut -d"/" -f2 | sed "s/openstack-//g" | sed "s/:.*//g") ; \
  hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
  --url https://registry.redhat.io \
  --docker-upstream-name $IMAGE \
  --name $IMAGENAME ; done < satellite_images_names
```

8. 同步容器镜像：

```
$ hammer product synchronize \
  --organization "ACME" \
  --name "OSP13 Containers"
```

等待 Satellite 服务器完成同步。

注意

根据具体配置情况，hammer 可能会询问您的 Satellite 服务器用户名和密码。您可以使用配置文件将 hammer 配置为自动登录。请参阅 *Hammer CLI 指南* 中的 "身份验证 "部分。

9. 如果您的 Satellite 6 服务器使用内容视图，请创建新的内容视图版本来纳入镜像。

10. 检查 base 镜像可用的标签：

```
$ hammer docker tag list --repository "base" \
  --organization "ACME" \
  --product "OSP13 Containers"
```

这会显示 OpenStack Platform 容器镜像的标签。

11. 返回到 undercloud，并为卫星服务器上的镜像生成环境文件。以下是生成环境文件的示例命令：

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=satellite6.example.com:5000 \
  --prefix=acme-osp13_containers- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

注意

此版本的 openstack overcloud 容器镜像准备命令以 Satellite 服务器为目标。它使用与上一步中使用的 openstack overcloud 容器镜像准备命令不同的值。

在运行这个命令时，包含以下数据：

- **--namespace - Satellite 服务器上 registry 的 URL 和端口。Red Hat Satellite 上的 registry 端口是 5000。例如： --namespace=satellite6.example.com:5000。**

注意

如果您使用 Red Hat Satellite 版本 6.10，则不需要指定端口。使用 443 的默认端口。如需更多信息，请参阅 "如何把 RHOSP13 部署调整为 Red Hat Satellite 6.10？"。

- **--prefix= - 前缀基于标签的 Satellite 6 约定，它使用小写字符并用下划线替换空格。前缀根据您是否使用内容视图而不同：**
  - **如果您使用了内容视图，则前缀的结构为 [组织]-[环境]-[内容视图]-[产品]-。例如： acme-production-myosp13-osp13_containers-。**
  - **如果不使用内容视图，则前缀的结构为 [组织]-[产品]-。例如： acme-osp13_containers-。**
- **--tag-from-label {version}-{release} - 识别每个镜像的最新标签。**
- **-e - 包含可选服务的任何环境文件。**

- **- R - 包含自定义角色文件。**

- **--set ceph_namespace,--set ceph_image,--set ceph_tag - If using Ceph Storage, 包括额外的参数来定义 Ceph Storage 容器镜像位置。请注意， ceph_image 现包含特定于 Satellite 的前缀。这个前缀与 --prefix 选项的值相同。例如：**

```
--set ceph_image=acme-osp13_containers-rhceph-3-rhel7
```

这将确保 overcloud 使用 Ceph 容器镜像，它利用卫星命名约定。

12. **overcloud_images.yaml 文件包含 Satellite 服务器上的镜像位置。在您的部署中包含此文件。**

注册表配置已就绪。

4.7. 后续步骤

现在，您有一个 overcloud_images.yaml 环境文件，其中包含容器镜像源列表。在将来的所有升级和部署操作中包括这个文件。

现在，您可以为升级准备 overcloud。

第 5 章 准备 OVERCLOUD 升级

本节为升级过程准备 overcloud。并非本节中的所有步骤都将适用于您的 overcloud。但是，建议逐一执行以下步骤，并确定您的 overcloud 是否需要在升级过程开始前是否需要额外的配置。

5.1. 准备 OVERCLOUD 服务停机

overcloud 升级过程会禁用主服务在关键点上。这意味着您无法在升级过程中使用任何 overcloud 服务来创建新资源。在 overcloud 中运行的工作负载在此期间保持活跃状态，这意味着实例在升级期间继续运行。

规划维护窗口非常重要，以确保用户在升级过程中无法访问 overcloud 服务。

受 overcloud 升级的影响

- OpenStack Platform 服务

不受 overcloud 升级的影响

- 在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

5.2. 选择 COMPUTE 节点进行升级测试

overcloud 升级过程允许您之一：

- 升级角色中的所有节点
- 单独节点

为确保 overcloud 升级过程正常，在升级所有 Compute 节点前，对环境中的几个单独 Compute 节点测试升级会很有用。这样可保证升级过程中不会发生重大问题，同时维持对工作负载的停机时间。

使用以下建议来帮助选择测试节点进行升级：

- 选择两个或者三个 Compute 节点进行升级测试
- 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

第 6 章 *升级 overcloud* 中的说明，在在所有 Compute 节点上运行升级前，使用 compute-0 作为 Compute 节点的示例测试升级过程。

下一步是更新您的 roles_data 文件，以确保将任何新的可组合服务添加到环境中的相关角色。要手动编辑现有的 roles_data 文件，请使用以下 OpenStack Platform 13 角色的新可组合服务列表。

注意

如果您在 Red Hat OpenStack Platform 12 或更早版本中为计算实例(Instance HA)启用了高可用性，并且希望执行到版本 13 或更高版本的快进升级，您必须首先手动禁用实例。具体步骤请参阅 禁用之前版本 中的 Instance HA。

5.3. 新的可组合服务

此版本的 Red Hat OpenStack Platform 包含新的可组合服务。如果将自定义 roles_data 文件与您自己的角色搭配使用，请将这些新的复合服务包括在相应的角色中。

所有角色

以下新服务适用于所有角色：

OS::TripleO::Services::MySQLClient

在节点上配置 MariaDB 客户端，它为其他可组合服务提供数据库配置。将该服务添加到具有独立可组合服务的所有角色中。

OS::TripleO::Services::CertmongerUser

允许 overcloud 需要来自 Certmonger 的证书。仅在启用 TLS/SSL 通信时使用。

OS::TripleO::Services::Docker

安装 docker 来管理容器化服务。

OS::TripleO::Services::ContainersLogrotateCrond

为容器日志安装 logrotate 服务。

OS::TripleO::Services::Securetty

允许在节点上配置 securetty。启用 environments/securetty.yaml 环境文件。

OS::TripleO::Services::Tuned

启用并配置 Linux 调优后台程序(调优)。

OS::TripleO::Services::AuditD

添加 auditd 守护进程并配置规则。默认禁用此选项。

OS::TripleO::Services::Collectd

添加 collectd 守护进程。默认禁用此选项。

OS::TripleO::Services::Rhsm

使用基于 Ansible 的方法配置订阅。默认禁用此选项。

OS::TripleO::Services::RsyslogSidecar

为日志记录配置 sidecar 容器。默认禁用此选项。

特定角色

以下新服务适用于特定角色：

OS::TripleO::Services::NovaPlacement

配置 OpenStack Compute (nova)放置 API。如果在当前 overcloud 中使用独立 Nova API 角色，请将此服务添加到角色。否则，将服务添加到 Controller 角色。

OS::TripleO::Services::PankoApi

配置 OpenStack Telemetry Event Storage (panko)服务。如果在当前 overcloud 中使用独立 Telemetry 角色，请将这个服务添加到角色中。否则，将服务添加到 Controller 角色。

OS::TripleO::Services::Clustercheck

任何也使用 OS::TripleO::Services::MySQL 服务的角色需要，如 Controller 或单机 数据库 角色。

OS::TripleO::Services::Iscsid

在 Controller、Compute 和 BlockStorage 角色上配置 iscsid 服务。

OS::TripleO::Services::NovaMigrationTarget

在 Compute 节点上配置迁移目标服务。

OS::TripleO::Services::Ec2Api

在 Controller 节点上启用 OpenStack Compute (nova) EC2-API 服务。默认禁用此选项。

OS::TripleO::Services::CephMgr

在 Controller 节点上启用 Ceph Manager 服务。作为 ceph-ansible 配置的一部分启用。

OS::TripleO::Services::CephMds

在 Controller 节点上启用 Ceph 元数据服务(MDS)。默认禁用此选项。

OS::TripleO::Services::CephRbdMirror

启用 RADOS 块设备(RBD)镜像服务。默认禁用此选项。

另外，有关特定自定义角色的更新 服务列表，*请参阅高级 Overcloud 自定义指南中的*"服务架构：单机 角色"部分。

除了新的可组合服务外，还要注意 OpenStack Platform 13 后已弃用的服务。

5.4. 弃用的可组合服务

如果使用自定义 roles_data 文件，请从适用的角色中删除这些服务。

OS::TripleO::Services::Core

此服务充当其他 Pacemaker 服务的核心依赖项。该服务已被移除，以适应高可用性可组合服务。

OS::TripleO::Services::VipHosts

此服务使用节点主机名和 IP 地址配置 /etc/hosts 文件。此服务现在直接集成到 director 的 Heat 模板中。

OS::TripleO::Services::FluentdClient

此服务已被 OS::TripleO::Services::Fluentd 服务替代。

OS::TripleO::Services::ManilaBackendGeneric

Manila 通用后端不再被支持。

如果使用自定义 roles_data 文件，请将这些服务从对应的角色中删除。

另外，有关特定自定义角色的更新 服务列表，*请参阅高级 Overcloud 自定义指南中的*"服务架构：单机 角色"部分。

5.5. 切换到容器化服务

快进升级过程将特定的 Systemd 服务转换为容器化服务。如果您使用 /usr/share/openstack-tripleo-heat-templates/environments/ 中的默认环境文件，则会自动进行这个过程。

如果您使用自定义环境文件在 overcloud 上启用服务，请检查 resource_registry 部分的环境文件，以及任何可组合服务的可组合服务映射。

流程

1. 查看您的自定义环境文件：

```
$ cat ~/templates/custom_environment.yaml
```

2. 检查文件内容中的 **resource_registry** 部分。

3. 检查 **resource_registry** 部分中的任何可组合服务。可组合的服务与以下命名空间：

```
OS::TripleO::Services
```

例如，以下可组合服务用于 **OpenStack Bare Metal Service (ironic) API**：

```
OS::TripleO::Services::IronicApi
```

4. 检查可组合服务是否映射到特定 **Puppet** 的 **Heat** 模板。例如：

```
resource_registry:
  OS::TripleO::Services::IronicApi: /usr/share/openstack-triple-heat-
template/puppet/services/ironic-api.yaml
```

5. 检查 **/usr/share/openstack-triple-heat-template/docker/services/ services/** 中是否存在容器化版本的 **Heat** 模板，并将服务重新 **map** 到容器化版本：

```
resource_registry:
  OS::TripleO::Services::IronicApi: /usr/share/openstack-triple-heat-
template/docker/services/ironic-api.yaml
```

另外，为 服务使用更新的环境文件，该文件位于 **/usr/share/openstack-tripleo-heat-templates/environments/** 中。例如，启用 **OpenStack Bare Metal Service (ironic)**的最新环境文件是 **/usr/share/openstack-tripleo-heat-templates/environments/services/ironic.yaml**，其中包含容器化服务映射。

如果自定义服务没有使用 **containerised** 服务，请保留到特定于 **Puppet** 的 **Heat** 模板的映射。

5.6. 弃用的参数

请注意，以下参数已弃用并已被替换。

旧参数	new Parameter
KeystoneNotificationDriver	**NotificationDriver**
controllerExtraConfig	**ControllerExtraConfig**
OvercloudControlFlavor	**OvercloudControllerFlavor**
controllerImage	**ControllerImage**
NovaImage	**ComputeImage**
NovaComputeExtraConfig	**ComputeExtraConfig**
NovaComputeServerMetadata	**ComputeServerMetadata**

NovaComputeSchedulerHints	**ComputeSchedulerHints**
	
	 
	
	注意
	
	如果您使用自定义 Compute 角色，若要使用特定于角色的 **ComputeSchedulerHints**，您需要在环境中添加以下配置，以确保设置已弃用的 **NovaComputeSchedulerHints** 参数，但未定义：
	
	`parameter_defaults:`
	`  NovaComputeSchedulerHints: {}`
	
	在使用自定义角色时，您必须添加此配置来使用特定于角色的任何 **_ROLE_SchedulerHints** 参数。

NovaComputeIPs	**ComputeIPs**
SwiftStorageServerMetadata	**ObjectStorageServerMetadata**
SwiftStorageIPs	**ObjectStorageIPs**
SwiftStorageImage	**ObjectStorageImage**
OvercloudSwiftStorageFlavor	**OvercloudObjectStorageFlavor**
NeutronDpdkCoreList	**OvsPmdCoreList**

旧参数	new Parameter
NeutronDpdkMemoryChannels	OvsDpdkMemoryChannels
NeutronDpdkSocketMemory	OvsDpdkSocketMemory
NeutronDpdkDriverType	OvsDpdkDriverType
HostCpusList	OvsDpdkCoreList

对于新参数的值，请使用双引号（没有嵌套单引号），如下例所示：

带有值的旧参数	new Parameter with Value
NeutronDpdkCoreList: "'2,3'"	OvsPmdCoreList: "2,3"
HostCpusList: "'0,1'"	OvsDpdkCoreList: "0,1"

更新自定义环境文件中的这些参数。以下参数已弃用，没有等同的。

NeutronL3HA

除了具有分布式虚拟路由(NeutronEnableDVR)的配置外，所有情况下都启用了 L3 高可用性。

CeilometerWorkers

Ceilometer 已被弃用，而是使用较新的组件(Gnocchi、Aod、Panko)。

CinderNetappEseriesHostType

所有系列支持都已弃用。

ControllerEnableSwiftStorage

应该改为使用 ControllerServices 参数。

OpenDaylightPort

使用 EndpointMap 为 OpenDaylight 定义默认端口。

OpenDaylightConnectionProtocol

此参数的值现在根据您是否要使用 TLS 部署 Overcloud 确定。

在 /home/stack 目录中运行以下 egrep 命令，以识别包含已弃用参数的任何环境文件：

```
$ egrep -r -w
'KeystoneNotificationDriver|controllerExtraConfig|OvercloudControlFlavor|controllerImage|NovaImage|N
ovaComputeExtraConfig|NovaComputeServerMetadata|NovaComputeSchedulerHints|NovaComputeIP
s|SwiftStorageServerMetadata|SwiftStorageIPs|SwiftStorageImage|OvercloudSwiftStorageFlavor|Neutr
onDpdkCoreList|NeutronDpdkMemoryChannels|NeutronDpdkSocketMemory|NeutronDpdkDriverType|
HostCpusList|NeutronDpdkCoreList|HostCpusList|NeutronL3HA|CeilometerWorkers|CinderNetappEser
iesHostType|ControllerEnableSwiftStorage|OpenDaylightPort|OpenDaylightConnectionProtocol' *
```

如果您的 OpenStack Platform 环境仍然需要这些已弃用的参数，默认的 roles_data 文件允许使用它们。但是，如果您使用自定义 roles_data 文件，且 overcloud 仍然需要这些弃用的参数，您可以通过编辑 roles_data 文件并添加到每个角色来允许访问它们：

Controller 角色

```
- name: Controller
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  ...
```

Compute Role

```
- name: Compute
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  disable_upgrade_deployment: True
  ...
```

对象存储角色

```
- name: ObjectStorage
  uses_deprecated_params: True
  deprecated_param_metadata: 'SwiftStorageServerMetadata'
  deprecated_param_ips: 'SwiftStorageIPs'
  deprecated_param_image: 'SwiftStorageImage'
  deprecated_param_flavor: 'OvercloudSwiftStorageFlavor'
  disable_upgrade_deployment: True
  ...
```

5.7. 弃用的 CLI 选项

某些命令行选项已过时或已弃用，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。下表将已弃用的选项与 Heat 模板中的等效选项对应了起来。

表 5.1. 将已弃用的 CLI 选项映射到 Heat 模板参数

Option	Description	Heat 模板参数
--control-scale	扩展的 Controller 节点数量	**ControllerCount**
--compute-scale	扩展的 Compute 节点数量	**ComputeCount**
--ceph-storage-scale	扩展的 Ceph 节点数量	**CephStorageCount**
--block-storage-scale	扩展的 Cinder 节点数量	**BlockStorageCount**
--swift-storage-scale	扩展的 Swift 节点数量	**ObjectStorageCount**
--control-flavor	Controller 节点使用的 flavor	**OvercloudControllerFlavor**
--compute-flavor	Compute 节点使用的 flavor	**overcloudComputeFlavor**
--ceph-storage-flavor	Ceph 节点使用的 flavor	**overcloudCephStorageFlavor**
--block-storage-flavor	Cinder 节点使用的 flavor	**overcloudBlockStorageFlavor**

Option	Description	Heat 模板参数
--swift-storage-flavor	Swift 存储节点使用的 flavor	**OvercloudSwiftStorageFlavor**
--neutron-flat-networks	定义要在 neutron 插件中配置的扁平网络。默认为 "datacentre" 以允许外部网络创建	**NeutronFlatNetworks**
--neutron-physical-bridge	每个虚拟机监控程序上创建的 Open vSwitch 网桥。默认为"br-ex"。通常，不应该更改	**HypervisorNeutronPhysicalBridge**
--neutron-bridge-mappings	要使用的物理网桥映射逻辑。默认为将主机上的外部网桥(br-ex)映射到物理名称(datacentre)。您要将这个网络用于默认浮动网络	**NeutronBridgeMappings**
--neutron-public-interface	为网络节点定义网桥 br-ex 的接口	**NeutronPublicInterface**
--neutron-network-type	Neutron 的租户网络类型	**NeutronNetworkType**
--neutron-tunnel-types	Neutron 租户网络的隧道类型。要指定多个值，使用逗号分隔的字符串	**NeutronTunnelTypes**
--neutron-tunnel-id-ranges	用于租户网络分配的 GRE 隧道 ID 范围	**NeutronTunnelIdRanges**
--neutron-vni-ranges	用于租户网络分配的 VXLAN VNI ID 范围	**NeutronVniRanges**
--neutron-network-vlan-ranges	Neutron ML2 和 Open vSwitch VLAN 映射范围，以支持。默认为允许 'datacentre' 物理网络中的任何 VLAN	**NeutronNetworkVLANRanges**
--neutron-mechanism-drivers	neutron 租户网络的机制驱动程序。默认为"openvswitch"。要指定多个值，使用逗号分隔的字符串	**NeutronMechanismDrivers**
--neutron-disable-tunneling	禁用隧道功能，以便利用 Neutron 使用 VLAN 网段网络或扁平网络	没有参数映射。
--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

Option	Description	Heat 模板参数
--ntp-server	设置用于同步时间的 NTP 服务器	**NtpServer**

这些参数已从 **Red Hat OpenStack Platform** 中删除。建议您将 **CLI** 选项转换为 **Heat** 参数，并将其添加到环境文件中。

以下是名为 **deprecated_cli_options.yaml** 的文件示例，其中包含其中一些新参数：

```
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 3
  CephStorageCount: 3
  ...
```

本指南中的后续示例包括 **deprecated_cli_options.yaml** 环境文件，其中包含这些新参数。

5.8. 可组合网络

此版本的 **Red Hat OpenStack Platform** 为可组合网络引入了一项新功能。如果使用自定义 **roles_data** 文件，请编辑 文件，以将可组合网络添加到每个角色。例如，对于 **Controller** 节点：

```
- name: Controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
```

检查默认 **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** 文件，以获得更多语法示例。另外，也检查 **/usr/share/openstack-tripleo-heat-templates/roles** 中的示例角色片断。

下表提供了可组合网络到自定义独立角色的映射：

角色	需要的网络
Ceph Storage Monitor	**存储,StorageMgmt**

角色	需要的网络
Ceph Storage OSD	**存储**,**StorageMgmt**
Ceph Storage RadosGW	**存储**,**StorageMgmt**
Cinder API	**InternalApi**
Compute	**InternalApi**,**Tenant**,**Storage**
Controller	**外部**,**InternalApi**,**Storage**,**StorageMgmt**,**Tenant**
数据库	**InternalApi**
Glance	**InternalApi**
Heat	**InternalApi**
Horizon	**InternalApi**
ironic	无必需。将 Provisioning/Control Plane 网络用于 API。
Keystone	**InternalApi**
Load Balancer	**外部**,**InternalApi**,**Storage**,**StorageMgmt**,**Tenant**
Manila	**InternalApi**
消息总线	**InternalApi**
Networker	**InternalApi**,**Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	**外部**,**InternalApi**,**Tenant**
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	**存储**
Swift Storage	**StorageMgmt**
Telemetry	**InternalApi**

重要

在以前的版本中，*NetName 参数（如 InternalApiNetName）更改了默认网络的名称。这不再被支持。使用可自定义可组合网络文件。有关更多信息，*请参阅高级 Overcloud 自定义指南中的*"使用 Composable Networks "。

5.9. 准备 CEPH STORAGE 或 HCI 节点升级

由于对容器化服务升级，安装和更新 Ceph Storage 节点的方法已更改。Ceph Storage 配置现在在 ceph-ansible 软件包中使用一组 playbook，它在 undercloud 上安装。

重要的

- 如果您使用超融合部署，请参阅 第 6.7 节 “升级超融合节点” 进行升级。
- 如果您使用混合超融合部署，请参阅 第 6.8 节 “升级混合超融合节点” 进行升级。

流程

1. 如果使用 director 管理的或外部 Ceph Storage 集群，请安装 ceph-ansible 软件包：

   a. 在 undercloud 上启用 Ceph 工具存储库：

   ```
   [stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
   ```

   b. 将 ceph-ansible 软件包安装到 undercloud:

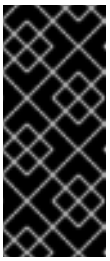
   ```
   [stack@director ~]$ sudo yum install -y ceph-ansible
   ```

2. 检查特定于 Ceph 的环境文件，并确保您的 Ceph 特定 heat 资源使用容器化服务：

   - 对于 director 管理的 Ceph Storage 集群，请确保 resource_register 中的资源指向 docker/services/ceph-ansible 中的模板：

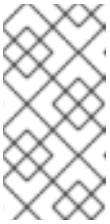
   ```
   resource_registry:
   ```

```
  OS::TripleO::Services::CephMgr: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-mgr.yaml
  OS::TripleO::Services::CephMon: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-mon.yaml
  OS::TripleO::Services::CephOSD: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-osd.yaml
  OS::TripleO::Services::CephClient: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-client.yaml
```



重要

此配置包含在 /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml 环境文件中，您可以使用 -e 包含在所有将来的部署命令中。



注意

如果环境中您要使用的环境或模板文件没有存在于 /usr/share 目录中，您必须包括文件的绝对路径。

- **对于外部 Ceph Storage 集群，请确保 resource_register 中的资源指向 docker/services/ceph-ansible 中的模板：**

```
resource_registry:
  OS::TripleO::Services::CephExternal: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-external.yaml
```

重要

此配置包含在 /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml 环境文件中，您可以使用 -e 包括在所有将来的部署命令中。

3. **对于 director 管理的 Ceph Storage 集群，请使用新的 CephAnsibleDisksConfig 参数来定义如何映射磁盘。以前的 Red Hat OpenStack Platform 版本使用 ceph::profile::params::osds hieradata 来定义 OSD 布局。将此 hieradata 转换为 CephAnsibleDisksConfig 参数的结构。以下示例说明了如何在 collocated 和 non-collocated Ceph journal 磁盘的情况下将 hieradata 转换为 CephAnsibleDisksConfig 参数的结构。**

重要

您必须设置 osd_scenario。如果未设置 osd_scenario，则可能会导致部署失败。

- **在 Ceph 日志磁盘并置的情形中，如果您的 hieradata 包含以下内容：**

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb': {}
      '/dev/sdc': {}
      '/dev/sdd': {}
```

使用 CephAnsibleDisksConfig 参数以下列方式转换 hieradata，并将 ceph::profile::params::osds 设置为 {} ：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    journal_size: 512
    osd_scenario: collocated
  ExtraConfig:
     ceph::profile::params::osds: {}
```

- **在一个场景中，如果 hieradata 包含以下内容，则日志位于更快速专用设备，并且是非并置：**

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb':
        journal: '/dev/sdn'
      '/dev/sdc':
        journal: '/dev/sdn'
      '/dev/sdd':
        journal: '/dev/sdn'
```

使用 CephAnsibleDisksConfig 参数以下列方式转换 hieradata，并将 ceph::profile::params::osds 设置为 {} ：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    dedicated_devices:
    - /dev/sdn
    - /dev/sdn
    - /dev/sdn
    journal_size: 512
    osd_scenario: non-collocated
  ExtraConfig:
    ceph::profile::params::osds: {}
```

有关 ceph-ansible 中使用的 OSD 磁盘布局选项的完整列表，请查看 /usr/share/ceph-ansible/group_vars/osds.yml.sample 中的示例文件。

4. **使用 -e 选项，包含新的 Ceph 配置环境文件及将来的部署命令。这包括以下文件：**

- **director 管理的 Ceph Storage:**
  - **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml.**
  - **具有基于 Ansible 的磁盘映射的环境文件。**
  - **任何额外环境文件及 Ceph 存储自定义。**
- **外部 Ceph 存储：**
  - **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml**
  - **任何额外环境文件及 Ceph 存储自定义。**

5.10. 使用非同同磁盘更新 CEPH 或 HCI 节点的环境变量

对于 HCI 节点，您可以在计算服务升级过程中使用旧语法进行磁盘，以及存储服务升级过程中的新语法，但可能还需要更新非同硬盘的语法。第 5.9 节 “准备 Ceph Storage 或 HCI 节点升级”

如果您要升级的节点上磁盘不相同，则它们不是同质性的。例如，在 HCI 节点和 Ceph Storage 节点的组合上的磁盘可能不是同构。

OpenStack Platform 12 及之后的版本引进了 ceph-ansible 的使用，它改变了如何将混合节点更新为非同构磁盘。这意味着，从 OpenStack Platform 12 开始，您无法使用 RoleExtraConfig 的可组合角色语法来表示磁盘。请参见以下示例。

以下示例不适用于 OpenStack Platform 12 或更高版本：

```
CephStorageExtraConfig:
  ceph::profile::params::osds:
    '/dev/sda'
    '/dev/sdb'
    '/dev/sdc'
    '/dev/sdd'

ComputeHCIExtraConfig:
  ceph::profile::params::osds:
    '/dev/sda'
    '/dev/sdb'
```

对于 OpenStack Platform 12 及更新的版本，您必须在升级前更新模板。有关如何更新非同磁盘的模板的更多信息，请参阅 *使用容器化 Red Hat Ceph 部署 Overcloud 指南中的配置 Ceph Storage* 集群设置。

5.11. 为大型 CEPH 集群增加重启延迟

在升级过程中，每个 Ceph 监控和 OSD 会按顺序停止。在成功重启同一服务前，迁移不会继续。Ansible 等待 15 秒（延迟），并检查服务启动（重试）的 5 次。如果服务没有重启，则迁移会停止，以便操作员可以干预。

根据 Ceph 集群的大小，您可能需要增加重试或延迟值。这些参数的确切名称及其默认值如下：

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

您可以更新这些参数的默认值。例如，要让集群检查 30 次，并在每个检查 Ceph OSD 的检查 20 秒之间等待 40 秒，并在每个检查 Ceph MON 的检查之间等待 10 秒，请使用 openstack overcloud deploy 命令传递以下参数：

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
    health_osd_check_retries: 30
    health_mon_check_retries: 10
    health_mon_check_delay: 20
```

5.12. 准备存储后端

有些存储后端已经从使用配置 hook 改为自己的可组合服务。如果使用自定义存储后端，请检查 环境目录中关联的环境文件是否有新的参数和资源。更新您的后端的任何自定义环境文件。例如：

- 对于 NetApp Block Storage (cinder) 后端，在部署中使用新的 environments/cinder-netapp-config.yaml。
- 对于 Dell EMC Block Storage (cinder) 后端，在部署中使用新的 environments/cinder-dellsc-config.yaml。
- 对于 Dell EqualLogic Block Storage (cinder) 后端，在部署中使用新的 environments/cinder-dellps-config.yaml。

例如，NetApp Block Storage (cinder) 后端使用以下对应版本的资源：

- OpenStack Platform 10 及以下： OS::TripleO::ControllerExtraConfigPre: ../puppet/extraconfig/pre_deploy/controller/cinder-netapp.yaml
- OpenStack Platform 11 及更高版本： OS::TripleO::Services::CinderBackendNetApp: ./puppet/services/cinder-backend-netapp.yaml

现在，您可以为这个后端使用新的 OS::TripleO::Services::CinderBackendNetApp 资源及其关联的服务模板。

5.13. 准备通过 SSL/TLS 访问 UNDERCLOUD 的公共 API

overcloud 在升级过程中需要访问 undercloud 的 OpenStack Object Storage (swift)公共 API。如果您的 undercloud 使用自签名证书，则需要将 undercloud 的证书颁发机构添加到每个 overcloud 节点。

前提条件

- undercloud 将 SSL/TLS 用于其公共 API

流程

1. director 的动态 Ansible 脚本已更新至 OpenStack Platform 12 版本，该版本使用 overcloud 计划中的 RoleNetHostnameMap Heat 参数来定义清单。但是，overcloud 当前使用 OpenStack Platform 11 模板版本，它没有 RoleNetHostnameMap 参数。这意味着您需要创建一个临时静态清单文件，您可以使用以下命令生成：

```
$ openstack server list -c Networks -f value | cut -d"=" -f2 > overcloud_hosts
```

2. 创建一个包含以下内容的 Ansible playbook (undercloud-ca.yml)：

```
---
- name: Add undercloud CA to overcloud nodes
  hosts: all
  user: heat-admin
  become: true
  vars:
    ca_certificate: /etc/pki/ca-trust/source/anchors/cm-local-ca.pem
  tasks:
    - name: Copy undercloud CA
      copy:
        src: "{{ ca_certificate }}"
        dest: /etc/pki/ca-trust/source/anchors/
    - name: Update trust
      command: "update-ca-trust extract"
    - name: Get the swift endpoint
      shell: |
        sudo hiera swift::keystone::auth::public_url | awk -F/ '{print $3}'
      register: swift_endpoint
      delegate_to: 127.0.0.1
      become: yes
      become_user: stack
    - name: Verify URL
      uri:
        url: https://{{ swift_endpoint.stdout }}/healthcheck
        return_content: yes
      register: verify
```

```
  - name: Report output
    debug:
      msg: "{{ ansible_hostname }} can access the undercloud's Public API"
    when: verify.content == "OK"
```

此 playbook 包含多个任务，在每个节点上执行以下操作：

- 将 undercloud 的证书颁发机构文件复制到 overcloud 节点。如果 undercloud 生成的默认位置为 /etc/pki/ca-trust/source/anchors/cm-local-ca.pem。
- 执行 命令，以更新 overcloud 节点上的证书颁发机构信任数据库。
- 检查 overcloud 节点上的 undercloud Object Storage Public API，并报告成功。

3. 使用以下命令运行 playbook：

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml
```

这将使用临时清单为您的 overcloud 节点提供 Ansible。

如果使用自定义证书颁发机构文件，您可以将 ca_certificate 变量改为一个位置。例如：

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml -e
ca_certificate=/home/stack/ssl/ca.crt.pem
```

4. 生成的 Ansible 输出应该会显示节点的 debug 消息。例如：

```
ok: [192.168.24.100] => {
    "msg": "overcloud-controller-0 can access the undercloud's Public API"
}
```

相关信息

- 有关在 overcloud 上运行 Ansible 自动化的更多信息，请参阅 *Director 安装和使用* 指南中的 "运行动态清单脚本 "。

5.14. 为快进升级配置注册

快进升级过程使用新方法来切换存储库。这意味着您需要从部署命令删除旧的 rhel- regration 环境文件。例如：

- environment-rhel-registration.yaml
- rhel-registration-resource-registry.yaml

快进升级过程使用脚本在升级的每个阶段更改存储库。此脚本作为 OS::TripleO::Services::TripleoPackages 可组合服务(puppet/services/tripleo-packages.yaml)包含在 OS::TripleO::Services::TripleoPackages 可组合服务(puppet/services/tripleo-packages.yaml)中。这是脚本：

```
#!/bin/bash
set -e
case $1 in
  ocata)
    subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
    ;;
  pike)
    subscription-manager repos --disable=rhel-7-server-openstack-11-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
    ;;
  queens)
    subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
    subscription-manager release --set=7.9
    subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-osd-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-mon-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-tools-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-13-deployment-tools-rpms
    ;;
  *)
    echo "unknown release $1" >&2
    exit 1
esac
```

director 将每个 OpenStack Platform 版本的上游代码名传递给脚本：

Codename	版本
ocata	OpenStack Platform 11
pike	OpenStack Platform 12
queens	OpenStack Platform 13

对 **queens** 的更改也禁用 **Ceph Storage 2** 存储库，并启用 **Ceph Storage 3 MON** 和工具存储库。这个更改不会启用 **Ceph Storage 3 OSD** 存储库，因为这些已被容器化。

在某些情况下，您可能需要使用自定义脚本。例如：

- 使用带有自定义存储库名称的 **Red Hat Satellite**。
- 使用带有自定义名称的断开连接的仓库。
- 要在每个阶段执行的附加命令。

在这些情况下，通过设置 **FastForwardCustomRepoScriptContent** 参数来包括您的自定义脚本：

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    [INSERT UPGRADE SCRIPT HERE]
```

例如，以下脚本使用一组 **Satellite 6** 激活码更改存储库：

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    set -e
    URL="satellite.example.com"
    case $1 in
      ocata)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp11 --
org=Default_Organization
        ;;
      pike)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp12 --
org=Default_Organization
```

```
        ;;
    queens)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp13 --
org=Default_Organization
        ;;
    *)
        echo "unknown release $1" >&2
        exit 1
esac
```

本指南中的示例包括 **custom_repositories_script.yaml** 环境文件，其中包含您的自定义脚本。

5.15. 检查自定义 PUPPET 参数

如果您使用 **ExtraConfig** 接口自定义 **Puppet** 参数，**Puppet** 可能会在升级过程中报告重复的声明错误。这是因为 **puppet** 模块本身提供的接口更改。

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，检查它是否具有 **ExtraConfig** 参数：

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

2. 如果结果在所选文件中显示任何角色（如 **Controller ExtraConfig** ）的 **ExtraConfig** 参数，请检查该文件中的整个参数结构。

3. 如果参数包含任何具有 **SECTION/parameter** 语法加 值 的任何 **puppet Hierdata**，则它可能已被一个参数替换为实际 **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，以查看该参数现在是否存在于 **Puppet** 类中。例如：

```
$ grep dnsmasq_local_resolv
```

如果是，请更改到新接口。

5. 以下是以语法演示更改的示例：

- 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'
```

5.16. 将网络接口模板转换为新结构

在以前的版本中，网络接口结构使用 OS::Heat::StructuredConfig 资源来配置接口：

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
```

```
    group: os-apply-config
    config:
      os_net_config:
        network_config:
          [NETWORK INTERFACE CONFIGURATION HERE]
```

现在，模板使用 **OS::Heat::SoftwareConfig** 资源进行配置：

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
                [NETWORK INTERFACE CONFIGURATION HERE]
```

此配置获取存储在 **$network_config** 变量中的接口配置，并将它注入为 **run-os-net-config.sh** 脚本的一部分。

警告

必须更新网络接口模板以使用这个新结构，并检查您的网络接口模板仍然符合 语法。不这样做可能会导致在快进升级过程中造成失败。

director 的 **Heat** 模板集合包含一个脚本，可帮助您将模板转换为这种新格式。此脚本位于 **/usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py** 中。例如：

```
$ /usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py \
    --script-dir /usr/share/openstack-tripleo-heat-templates/network/scripts \
    [NIC TEMPLATE] [NIC TEMPLATE] ...
```

重要

在使用此脚本时，请确保您的模板不包含任何注释行。这在解析旧模板结构时可能会导致错误。

如需更多信息，请参阅 "网络隔离 "。

5.17. 检查 DPDK 和 SR-IOV 配置

本节用于使用 NFV 技术，如数据平面开发套件(DPDK)集成和单根输入/输出虚拟化(SR-IOV)。如果您的 overcloud 不使用这些功能，请忽略本节。

注意

在 Red Hat OpenStack Platform 10 中，不需要将 first-boot 脚本文件替换为 host-config-and-reboot.yaml，这是 OpenStack Platform 13 模板。在整个升级过程中维护第一引导脚本可避免进行额外的重启。

5.17.1. 升级 DPDK 环境

对于使用 DPDK 的环境，请检查特定的服务映射，以确保成功过渡到容器化环境。

流程

1. 由于过渡到容器化服务，所以会自动升级 DPDK 服务的快速升级。如果将自定义环境文件用于 DPDK，请手动调整这些环境文件以映射到容器化服务。

```
OS::TripleO::Services::ComputeNeutronOvsDpdk:
  /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-dpdk-agent.yaml
```

注意

另外，也可使用最新的 NFV 环境文件 /usr/share/openstack-tripleo-heat-templates/environments/services/services/neutron-ovs-dpdk.yaml。

2. 将 OpenStack Network (Neutron)代理服务映射到适当的容器化模板：

- 如果您正在使用 DPDK 的默认 Compute 角色，请将 ComputeNeutronOvsAgent 服务映射到核心 heat 模板集合的 docker/services 目录中的 neutron-ovs-dpdk-agent.yaml 文件。

```
resource_registry:
  OS::TripleO::Services::ComputeNeutronOvsAgent:
    /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-dpdk-
agent.yaml
```

- 如果您在 DPDK 中使用自定义角色，则应当存在一个自定义的可组合服务，如 ComputeNeutronOvsDpdkAgentCustom。将这个服务映射到 docker 目录中的 neutron-ovs-dpdk-agent.yaml 文件。

3. 在 DPDK 角色定义中添加以下服务和额外参数：

```
RoleParametersDefault:
  VhostuserSocketGroup: "hugetlbfs"
  TunedProfileName: "cpu-paritioning"

ServicesDefault:
  - OS::TripleO::Services::ComputeNeutronOvsDPDK
```

4. 删除以下服务：

```
ServicesDefault:
  - OS::TripleO::Services::NeutronLinuxbridgeAgent
  - OS::TripleO::Services::NeutronVppAgent
  - OS::TripleO::Services::Tuned
```

5.17.2. 升级 SR-IOV 环境

对于使用 SR-IOV 的环境，请检查以下服务映射，以确保成功过渡到容器化环境。

流程

1. SR-IOV 服务的快速升级会因为过渡到容器化服务而自动进行。如果您要为 SR-IOV 使用自定义环境文件，请确保这些服务正确映射到容器化服务。

```
OS::TripleO::Services::NeutronSriovAgent:
  /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-sriov-agent.yaml
```

```
OS::TripleO::Services::NeutronSriovHostConfig:
  /usr/share/openstack-tripleo-heat-templates/puppet/services/neutron-sriov-host-config.yaml
```

注意

另外，也可使用最后一个 NFV 环境文件 **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml**。

2. 确保 **roles_data.yaml** 文件包含所需的 SR-IOV 服务。

如果您要为 SR-IOV *使用默认* Compute 角色，请在 OpenStack Platform 13 中包括此角色中的相应服务。

- 将 **roles_data.yaml** 文件从 **/usr/share/openstack-tripleo-heat-templates** 复制到您的自定义模板目录，例如 **/home/stack/templates**。
- 将以下服务添加到默认计算角色中：
  - **OS::TripleO::Services::NeutronSriovAgent**
  - **OS::TripleO::Services::NeutronSriovHostConfig**
- 从默认的 **Compute** 角色中删除以下服务：
  - **OS::TripleO::Services::NeutronLinuxbridgeAgent**
  - **OS::TripleO::Services::Tuned**

如果您要为 SR-IOV *使用自定义* Compute 角色，则 **NeutronSriovAgent** 服务应当存在。添加 **NeutronSriovHostConfig** 服务，该服务在 **Red Hat OpenStack Platform 13** 中引入。

注意

在运行 **ffwd-upgrade** 命令以下部分 **准备** 和聚合时，应该包括 **roles_data.yaml** 文件。

5.18. 准备预置备节点升级

预置备节点是在 **director** 管理之外创建的节点。使用预置备节点的 **overcloud** 需要升级前的一些额外步骤。

前提条件

- **overcloud** 使用预置备节点。

流程

1. 运行以下命令，在 **OVERCLOUD_HOSTS** 环境变量中保存节点 **IP** 地址列表：

```
$ source ~/stackrc
$ export OVERCLOUD_HOSTS=$(openstack server list -f value -c Networks | cut -d "=" -f 2 |
tr '\n' ' ')
```

2. 运行以下脚本：

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

3. 继续升级。

- 在将 **openstack overcloud upgrade run** 命令与预置备节点搭配使用时，请包含 **--ssh-user tripleo-admin** 参数。
- 升级 **Compute** 或 **Object Storage** 节点时，使用以下方法：

a. 将 **-U** 选项与 **upgrade-non-controller.sh** 脚本一起使用，并指定 **stack** 用户。这是因为预置备节点的默认用户是 **stack**，而不是 **heat-admin**。

b.

使用带 **--upgrade** 选项的节点 IP 地址。这是因为节点没有使用 director 的 Compute (nova)和 Bare Metal (ironic)服务来管理，且没有节点名称。

例如：

```
$ upgrade-non-controller.sh -U stack --upgrade 192.168.24.100
```

相关信息

- 有关预置备节点的更多信息，请参阅 *Director 安装和使用* 指南中的 "使用预置备节点配置基本 Overcloud "。

5.19. 后续步骤

overcloud 准备阶段已完成。现在，您可以使用 第 6 章 *升级 overcloud* 中的步骤将 overcloud 从 10 升级到 13。

第 6 章 升级 OVERCLOUD

本节升级 overcloud。这包括以下工作流：

- 运行快速升级准备命令
- 运行 fast forward upgrade 命令
- 升级 Controller 节点
- 升级 Compute 节点
- 升级 Ceph Storage 节点
- 模拟快进升级。

开始此工作流后，在完成所有步骤之前，您不应完全掌控 overcloud 的 OpenStack 服务。这意味着工作负载在所有节点都成功升级到 OpenStack Platform 13 之前无法管理。这些工作负载本身将保持不变，并继续运行。任何 overcloud 工作负载的更改或添加需要等到快速的升级完成。

6.1. 快进升级命令

快进升级过程涉及您使用某些流程阶段运行的不同命令。下表包含每个命令的一些基本信息。

重要

此列表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。等待您收到说明以在适当的步骤中运行这些命令。

openstack overcloud ffwd-upgrade prepare

此命令执行 overcloud 升级的初始准备步骤，其中包括将 undercloud 上当前的 overcloud 计划替换为新的 OpenStack Platform 13 overcloud 计划和更新的环境文件。此命令的功能类似于

openstack overcloud deploy 命令，并使用许多相同的选项。

openstack overcloud ffwd-upgrade run

此命令执行快速的升级过程。director 根据新的 OpenStack Platform 13 overcloud 计划创建一组 Ansible playbook，并在整个 overcloud 上运行快速转发任务。这包括通过每个 OpenStack Platform 版本从 10 到 13 运行升级过程。

OpenStack overcloud 升级运行

此命令针对单个节点或角色中的多个节点执行特定于节点的升级配置。director 根据 overcloud 计划创建一组 Ansible playbook，并根据所选节点运行任务，这会使用适当的 OpenStack Platform 13 配置来配置节点。此命令还提供以每个角色为基础对更新进行阶段的方法。例如，您首先运行这个命令来升级 Controller 节点，然后再次运行 命令来升级 Compute 节点和 Ceph Storage 节点。

OpenStack overcloud ceph-upgrade 运行

此命令执行 Ceph Storage 版本升级。运行 openstack overcloud 升级针对 Ceph Storage 节点运行 openstack overcloud 升级前 运行此命令。director 使用 ceph-ansible 来执行 Ceph Storage 版本升级。

openstack overcloud ffwd-upgrade converge

此命令执行 overcloud 升级中的最后一步。最后一步是将 overcloud Heat 堆栈与 OpenStack Platform 13 overcloud 计划及您更新的环境文件同步。这样可确保生成的 overcloud 与新的 OpenStack Platform 13 overcloud 的配置相匹配。此命令的功能类似于 openstack overcloud deploy 命令，并使用许多相同的选项。

您必须以特定顺序运行这些命令。按照本章中的其余部分，使用这些命令完成快速升级。

注意

如果您将自定义名称用于 overcloud，请为每个命令使用 --stack 选项设置自定义名称。

6.2. 执行 OVERCLOUD 的快进升级

快进升级需要运行两个命令来执行以下任务：

- 将 overcloud 计划更新为 OpenStack Platform 13。

- 为节点做好快进升级准备。
- 通过在快进升级中每个后续版本的升级步骤运行，包括：
  - 每个 OpenStack Platform 服务版本特定的任务。
  - 将存储库更改为快速升级中每个后续 OpenStack Platform 版本。
  - 更新升级数据库所需的某些软件包。
  - 为每个后续版本执行数据库升级。
- 为 overcloud 准备最终升级到 OpenStack Platform 13。

流程

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 使用适合部署的所有相关选项和环境文件运行快速升级准备命令：

```
$ openstack overcloud ffwd-upgrade prepare \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/deprecated_cli_options.yaml \
  -e /home/stack/templates/custom_repositories_script.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

包含以下与您的环境相关的选项：

- 自定义配置环境文件(-e)。例如：
  - 容器镜像位置(overcloud_images.yaml)的环境文件。请注意，升级命令可能会显示有关使用 --container-registry-file 的警告。您可以忽略这个警告，因为这个选项已弃用，而是使用 -e 作为容器镜像环境文件。
  - 如果适用，使用 deprecated_cli_options.yaml 将已弃用的 CLI 选项映射到 Heat 参数的环境文件。
  - 如果适用，使用 custom_repositories_script.yaml 的自定义存储库脚本环境文件。
  - 如果使用 Ceph Storage 节点，相关的环境文件。
  - 任何与您环境相关的额外环境文件。
- 如果使用自定义堆栈名称，请使用 --stack 选项传递名称。
- 如果适用，您的自定义角色(roles_data)文件使用 --roles-file。

重要

提示会询问您是否要执行 ffwd-upgrade 命令。输入 yes。

注意

您可以多次运行 openstack ffwd-upgrade 准备 命令。如果命令失败，您可以修复模板中的问题，然后重新运行 命令。

3. overcloud 计划更新 OpenStack Platform 13 版本。等待快进升级准备完成。

4. 在进行升级前，先创建 overcloud 的快照或备份。

5. 运行 **fast forward upgrade** 命令：

```
$ openstack overcloud ffwd-upgrade run
```

- 如果使用自定义堆栈名称，请使用 **--stack** 选项传递名称。

重要

提示会询问您是否要执行 **ffwd-upgrade** 命令。输入 **yes**。

注意

您可以多次运行 **openstack ffwd-upgrade** 命令。如果命令失败，您可以修复模板中的问题，然后重新运行 命令。

6. 等待快速升级完成。

在这个阶段：

- 工作负载仍然在运行
- **overcloud** 数据库已升级至 **OpenStack Platform 12** 版本
- 所有 **overcloud** 服务都被禁用
- **Ceph Storage** 节点仍然为 **2** 版本

这意味着 **overcloud** 现在处于执行访问 **OpenStack Platform 13** 的标准升级步骤的状态。

6.3. 升级 CONTROLLER 和自定义角色节点

使用以下步骤将所有 Controller 节点、分割 Controller 服务和其他自定义节点升级到 OpenStack Platform 13。此过程涉及运行 openstack overcloud upgrade run 命令，并包括 --nodes 选项，将操作限制为仅所选节点：

```
$ openstack overcloud upgrade run --nodes [ROLE]
```

将 [ROLE] 替换为角色的名称或以逗号分隔的角色列表。

如果您的 overcloud 使用单体 Controller 节点，请针对 Controller 角色运行此命令。

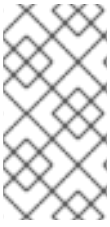
如果您的 overcloud 使用分割 Controller 服务，请按照以下顺序升级节点角色：

- 所有使用 Pacemaker 的角色。例如： 控制器OpenStack 、数据库、消息传递 和遥测。
- Networker 节点
- 任何其他自定义角色

不要 升级以下节点：

- 基于 DPDK 或 Hyper-Converged Infratructure (HCI) Compute 节点的 Compute 节点
- CephStorage 节点

您将在以后的阶段升级这些节点。



注意

此流程中的命令使用 --skip-tags 验证 选项，因为 OpenStack Platform 服务在 overcloud 上不活跃且无法验证。

流程

1. **Source stackrc 文件：**

```
$ source ~/stackrc
```

2. **如果使用单体 Controller 节点，请针对 Controller 角色运行 upgrade 命令：**

```
$ openstack overcloud upgrade run --nodes Controller --skip-tags validation
```

- **如果您使用自定义堆栈名称，请使用 --stack 选项传递名称。**

3. **如果使用 Controller 服务在多个角色间分割：**

   a. **使用 Pacemaker 服务为角色运行 upgrade 命令：**

```
$ openstack overcloud upgrade run --nodes ControllerOpenStack --skip-tags validation
$ openstack overcloud upgrade run --nodes Database --skip-tags validation
$ openstack overcloud upgrade run --nodes Messaging --skip-tags validation
$ openstack overcloud upgrade run --nodes Telemetry --skip-tags validation
```

   - **如果您使用自定义堆栈名称，请使用 --stack 选项传递名称。**

   b. **对 Networker 角色运行 upgrade 命令：**

```
$ openstack overcloud upgrade run --nodes Networker --skip-tags validation
```

   - **如果您使用自定义堆栈名称，请使用 --stack 选项传递名称。**

   c. **对剩余的自定义角色运行 upgrade 命令，但 Compute 或 CephStorage 角色除外：**

```
$ openstack overcloud upgrade run --nodes ObjectStorage --skip-tags validation
```

   - **如果您使用自定义堆栈名称，请使用 --stack 选项传递名称。**

在这个阶段：

- 工作负载仍然在运行
- overcloud 数据库已升级至 OpenStack Platform 13 版本
- Controller 节点已升级至 OpenStack Platform 13
- 所有 Controller 服务都被启用
- Compute 节点仍然需要升级
- Ceph Storage 节点仍然处于 2 版本，需要升级

警告

虽然启用了 Controller 服务，但在禁用 Compute 节点和 Ceph Storage 服务时不要执行任何工作负载操作。这会导致孤立的虚拟机。等待整个环境升级。

6.4. 升级测试 COMPUTE 节点

此过程升级了为测试选择的 Compute 节点。此过程涉及运行 openstack overcloud upgrade run 命令，并包括 --nodes 选项，以限制仅测试节点的操作。此流程使用 --nodes compute-0 作为命令中的示例。

流程

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 运行 **upgrade** 命令：

```
$ openstack overcloud upgrade run --nodes compute-0 --skip-tags validation
```

注意

命令使用 --skip-tags 验证，因为 OpenStack Platform 服务在 overcloud 上不活跃，且无法验证。

- **如果使用自定义堆栈名称，请使用 --stack 选项传递名称。**

3. **等待测试节点升级完成。**

6.5. 升级所有 COMPUTE 节点

重要的

- **如果您使用超融合部署，请参阅 第 6.7 节 “升级超融合节点” 进行升级。**
- **如果您使用混合超融合部署，请参阅 第 6.8 节 “升级混合超融合节点” 进行升级。**

此过程会将所有剩余的 Compute 节点升级到 OpenStack Platform 13。此过程涉及运行 openstack overcloud upgrade run 命令，并包括 --nodes Compute 选项，以限制仅向 Compute 节点限制操作。

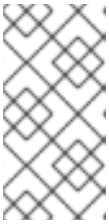
流程

1. **Source stackrc 文件：**

```
$ source ~/stackrc
```

2. 运行 **upgrade** 命令：

```
$ openstack overcloud upgrade run --nodes Compute --skip-tags validation
```



注意

命令使用 **--skip-tags** 验证，因为 **OpenStack Platform** 服务在 **overcloud** 上不活跃，且无法验证。

- 如果您使用的是自定义堆栈名称，请使用 **--stack** 选项传递名称。
- 如果您使用的是自定义 **Compute** 角色，请确保使用 **--nodes** 选项包含角色名称。

3. 等待 **Compute** 节点升级完成。

在这个阶段：

- 工作负载仍然在运行
- **Controller** 节点和 **Compute** 节点已升级到 **OpenStack Platform 13**
- **Ceph Storage** 节点仍然处于 **2** 版本，需要升级

6.6. 升级所有 CEPH STORAGE 节点

重要的

- 如果您使用超融合部署，请参阅 第 6.7 节 “升级超融合节点” 进行升级。
- 如果您使用混合超融合部署，请参阅 第 6.8 节 “升级混合超融合节点” 进行升级。

此过程升级 **Ceph Storage** 节点。进程涉及：

- 运行 **openstack overcloud upgrade run** 命令并包括 **--nodes CephStorage** 选项，以限制仅对 **Ceph Storage** 节点的操作。

- 运行 **openstack overcloud ceph-upgrade run** 命令以对容器化 **Red Hat Ceph Storage 3** 集群执行升级。

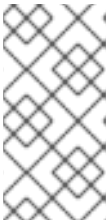
流程

1. **Source stackrc** 文件：

```
$ source ~/stackrc
```

2. 运行 **upgrade** 命令：

```
$ openstack overcloud upgrade run --nodes CephStorage --skip-tags validation
```



注意

命令使用 --skip-tags 验证，因为 **OpenStack Platform** 服务在 **overcloud** 上不活跃，且无法验证。

- **如果使用自定义堆栈名称，请使用 --stack** 选项传递名称。

3. **等待节点升级完成。**

4. 运行 **Ceph Storage upgrade** 命令。例如：

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e <ENVIRONMENT FILE> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/deprecated_cli_options.yaml \
  -e /home/stack/templates/custom_repositories_script.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  --ceph-ansible-playbook '/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml,/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml'
```

包含以下与您的环境相关的选项：

- 自定义配置环境文件(-e)。例如：
  - 容器镜像位置(overcloud_images.yaml)的环境文件。请注意，升级命令可能会显示有关使用 --container-registry-file 的警告。您可以忽略这个警告，因为这个选项已弃用，而是使用 -e 作为容器镜像环境文件。
  - 如果适用，使用 deprecated_cli_options.yaml 将已弃用的 CLI 选项映射到 Heat 参数的环境文件。
  - 如果适用，使用 custom_repositories_script.yaml 的自定义存储库脚本环境文件。
  - Ceph Storage 节点的相关环境文件。
  - 任何与您环境相关的额外环境文件。
- 如果使用自定义堆栈名称，请使用 --stack 选项传递名称。
- 如果适用，您的自定义角色(roles_data)文件使用 --roles-file。
- 以下 ansible playbook：
- /usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-containerized-ceph-daemons.yml
- /usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml

5. 等待 Ceph Storage 节点升级完成。

6.7. 升级超融合节点

如果您只使用来自 ComputeHCI 角色的超融合节点，且没有使用专用计算节点或专用 Ceph 节点，请完成以下步骤来升级节点：

流程

1. 查找 **stackrc** 文件：

```
$ source ~/stackrc
```

2. 运行 **upgrade** 命令：

```
$ openstack overcloud upgrade run --roles ComputeHCI
```

如果您使用自定义堆栈名称，请使用 **--stack** 选项将名称传递给 **upgrade** 命令。

3. 运行 **Ceph Storage upgrade** 命令。例如：

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

包含以下与您的环境相关的选项：

- 自定义配置环境文件(-e)。例如：
  - 容器镜像位置(overcloud_images.yaml)的环境文件。请注意，升级命令可能会显示有关使用 **--container-registry-file** 的警告。您可以忽略这个警告，因为这个选项已弃用，而是使用 **-e** 作为容器镜像环境文件。
  - 如果适用，使用 **deprecated_cli_options.yaml** 将已弃用的 **CLI** 选项映射到 **Heat** 参数的环境文件。
  - 如果适用，使用 **custom_repositories_script.yaml** 的自定义存储库脚本环境文

件。

- Ceph Storage 节点的相关环境文件。

- 如果使用自定义堆栈名称，请使用 --stack 选项传递名称。

- 如果适用，您的自定义角色(roles_data)文件使用 --roles-file。

- 以下 ansible playbook：

- /usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-containerized-ceph-daemons.yml

- /usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml

4. 等待 Ceph Storage 节点升级完成。

6.8. 升级混合超融合节点

如果您在 ComputeHCI 角色等超融合节点外还使用专用计算节点或专用 ceph 节点，请完成以下步骤来升级节点：

流程

1. 查找 stackrc 文件：

```
$ source ~/stackrc
```

2. 为 Compute 节点运行 upgrade 命令：

```
$ openstack overcloud upgrade run --roles Compute
If using a custom stack name, pass the name with the --stack option.
```

3. 等待节点升级完成。

4. 为 **ComputeHCI** 节点运行 **upgrade** 命令：

```
$ openstack overcloud upgrade run --roles ComputeHCI
If using a custom stack name, pass the name with the --stack option.
```

5. 等待节点升级完成。

6. 为 **Ceph Storage** 节点运行 **upgrade** 命令：

```
$ openstack overcloud upgrade run --roles CephStorage
```

7. 等待 **Ceph Storage** 节点升级完成。

8. 运行 **Ceph Storage upgrade** 命令。例如：

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

包含以下与您的环境相关的选项：

- 自定义配置环境文件(**-e**)。例如：
  - 容器镜像位置(**overcloud_images.yaml**)的环境文件。请注意，升级命令可能会显示有关使用 **--container-registry-file** 的警告。您可以忽略这个警告，因为这个选项已弃用，而是使用 **-e** 作为容器镜像环境文件。
  - 如果适用，使用 **deprecated_cli_options.yaml** 将已弃用的 **CLI** 选项映射到 **Heat** 参数的环境文件。

- 如果适用，使用 custom_repositories_script.yaml 的自定义存储库脚本环境文件。
- Ceph Storage 节点的相关环境文件。
- 任何与您环境相关的额外环境文件。

- 如果使用自定义堆栈名称，请使用 --stack 选项传递名称。
- 如果适用，您的自定义角色(roles_data)文件使用 --roles-file。
- 以下 ansible playbook：
- /usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-containerized-ceph-daemons.yml
- /usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml

9. 等待 Ceph Storage 节点升级完成。

在这个阶段：

- 所有节点已升级至 OpenStack Platform 13，工作负载仍在运行

虽然环境现已升级，但您必须执行一个最后一步才能完成升级。

6.9. 模拟快进升级

快进升级需要最后一步来更新 overcloud 堆栈。这可确保堆栈的资源结构与 OpenStack Platform 13 的常规部署保持一致，并允许您在以后执行标准 openstack overcloud deploy 功能。

流程

1. **Source stackrc 文件：**

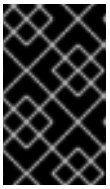
```
$ source ~/stackrc
```

2. 运行 **fast forward upgrade finalization 命令：**

```
$ openstack overcloud ffwd-upgrade converge \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/deprecated_cli_options.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <OTHER ENVIRONMENT FILES>
```

包含以下与您的环境相关的选项：

- **自定义配置环境文件(-e)。例如：**
  - **容器镜像位置(overcloud_images.yaml)的环境文件。请注意，升级命令可能会显示有关使用 --container-registry-file 的警告。您可以忽略这个警告，因为这个选项已弃用，而是使用 -e 作为容器镜像环境文件。**
  - **如果适用，使用 deprecated_cli_options.yaml 将已弃用的 CLI 选项映射到 Heat 参数的环境文件。**
  - **如果使用 Ceph Storage 节点，相关的环境文件。**
  - **任何与您环境相关的额外环境文件。**
- **如果使用自定义堆栈名称，请使用 --stack 选项传递名称。**
- **如果适用，您的自定义角色(roles_data)文件使用 --roles-file。**



重要

提示会询问您是否要执行 ffwd-upgrade 命令。输入 yes。

3. 等待快速升级最终完成。

6.10. 后续步骤

overcloud 升级已完成。现在，您可以使用 第 8 章 *执行 Post Upgrade steps* 中的步骤执行所有相关的 overcloud 配置。对于将来的部署操作，请确保包含与 OpenStack Platform 13 环境相关的所有环境文件，包括在升级过程中创建或转换的新环境文件。

第 7 章 升级后重启 OVERCLOUD

升级 Red Hat OpenStack 环境后，重启 overcloud。重启会将节点更新为任何关联的内核、系统级别和容器组件更新。这些更新可能会提供性能和安全性优势。

计划停机时间来执行以下重启过程。

7.1. 重新引导控制器和可组合节点

以下流程根据可组合角色重启控制器节点和独立节点。这会排除 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：

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs cluster stop
```

3. 重新引导节点：

```
[heat-admin@overcloud-controller-0 ~]$ sudo reboot
```

4. 稍等片刻，直到节点启动。
5. 检查服务。例如：
   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：

```
[heat-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：

```
[heat-admin@overcloud-controller-0 ~]$ sudo systemctl status
```

c. 对所有 Controller 和可组合节点重复这些步骤。

7.2. 重新引导 CEPH STORAGE (OSD) 集群

以下流程重启 Ceph Storage (OSD)节点集群。

流程

1. 登录 Ceph MON 或 Controller 节点，并暂时禁用 Ceph Storage 集群重新平衡：

```
$ sudo ceph osd set noout
$ sudo ceph osd set norebalance
```

2. 选择第一个 Ceph Storage 节点以重新引导并登录。

3. 重新引导节点：

```
$ sudo reboot
```

4. 稍等片刻，直到节点启动。

5. 登录到 Ceph MON 或 Controller 节点并检查集群状态：

```
$ sudo ceph -s
```

确认 pgmap 报告的所有 pgs 的状态是否都正常 (active+clean)。

6. 从 Ceph MON 或 Controller 节点注销，重新引导下一个 Ceph Storage 节点，并检查其状态。重复此流程，直到您已重新引导所有 Ceph 存储节点。

7. 完成之后，登录 Ceph MON 或 Controller 节点，然后重新启用集群重新平衡：

```
$ sudo ceph osd unset noout
$ sudo ceph osd unset norebalance
```

8. 执行最后的状态检查，确认集群报告 **HEALTH_OK**：

```
$ sudo ceph status
```

7.3. 重新引导 COMPUTE 节点

重新引导 **Compute** 节点涉及以下工作流：

- 选择一个 **Compute** 节点来重新引导并禁用它，使其不置备新实例。
- 将实例迁移到另一个 **Compute** 节点，以最小化实例停机时间。
- 重新引导空的 **Compute** 节点并启用它。

流程

1. 以 **stack** 用户的身份登录 **undercloud**。
2. 要识别您要重新引导的 **Compute** 节点，请列出所有 **Compute** 节点：

```
$ source ~/stackrc
(undercloud) $ openstack server list --name compute
```

3. 在 **overcloud** 中，选择 **Compute** 节点并禁用它：

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

4. 列出 **Compute** 节点上的所有实例：

```
(overcloud) $ openstack server list --host <hostname> --all-projects
```

5. 迁移您的实例。有关迁移策略的更多信息，请参阅 Compute 节点之间迁移虚拟机。

6. 登录到 Compute 节点并重新引导它：

```
[heat-admin@overcloud-compute-0 ~]$ sudo reboot
```

7. 稍等片刻，直到节点启动。

8. 启用 Compute 节点：

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set <hostname> nova-compute --enable
```

9. 验证 Compute 节点是否已启用：

```
(overcloud) $ openstack compute service list
```

7.4. 重新引导计算 HCI 节点

以下流程重启计算超融合基础架构(HCI)节点。

流程

1. 登录 Ceph MON 或 Controller 节点，并暂时禁用 Ceph Storage 集群重新平衡：

```
$ sudo ceph osd set noout
$ sudo ceph osd set norebalance
```

2. 以 stack 用户的身份登录 undercloud。

3. 列出所有的 Compute 节点及其 UUID：

```
$ source ~/stackrc
(undercloud) $ openstack server list --name compute
```

确定您要重新引导的 Compute 节点的 UUID。

4. 在 undercloud 中，选择 Compute 节点并禁用它：

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service set [hostname] nova-compute --disable
```

5. 列出 Compute 节点上的所有实例：

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

6. 使用以下命令之一迁移您的实例：

   a. 将实例迁移到您选择的特定主机：

   ```
   (overcloud) $ openstack server migrate [instance-id] --live [target-host]--wait
   ```

   b. 让 nova-scheduler 自动选择目标主机：

   ```
   (overcloud) $ nova live-migration [instance-id]
   ```

   c. 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。

7. 等待迁移完成。

8. 确认迁移成功完成：

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

9. **继续迁移实例，直到所选 Compute 节点中不剩任何实例。**

10. **登录到 Ceph MON 或 Controller 节点并检查集群状态：**

```
$ sudo ceph -s
```

确认 pgmap 报告的所有 pgs 的状态是否都正常 (active+clean)。

11. **重新引导 Compute HCI 节点：**

```
$ sudo reboot
```

12. **稍等片刻，直到节点启动。**

13. **再次启用 Compute 节点：**

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set [hostname] nova-compute --enable
```

14. **验证 Compute 节点是否已启用：**

```
(overcloud) $ openstack compute service list
```

15. **注销节点，重新引导下一个节点，并检查其状态。重复此流程，直到您已重新引导所有 Ceph 存储节点。**

16. **完成后，登录 Ceph MON 或 Controller 节点，然后再次启用集群重新平衡：**

```
$ sudo ceph osd unset noout
$ sudo ceph osd unset norebalance
```

17. **执行最后的状态检查，确认集群报告 HEALTH_OK：**

```
$ sudo ceph status
```

第 8 章 执行 POST UPGRADE STEPS

这个过程会在完成主升级过程后执行最后的步骤。这包括更改镜像以及快速升级过程完成后的额外配置步骤或注意事项。

8.1. 验证 UNDERCLOUD

以下是检查 undercloud 功能的一组步骤。

流程

1. 查找 undercloud 访问详情：

```
$ source ~/stackrc
```

2. 检查失败的 Systemd 服务：

```
(undercloud) $ sudo systemctl list-units --state=failed 'openstack*' 'neutron*' 'httpd' 'docker'
```

3. 检查 undercloud 可用空间：

```
(undercloud) $ df -h
```

使用 "Undercloud 要求" 作为基础来确定您是否有足够的可用空间。

4. 如果您在 undercloud 上安装了 NTP，请检查时钟是否已同步：

```
(undercloud) $ sudo ntpstat
```

5. 检查 undercloud 网络服务：

```
(undercloud) $ openstack network agent list
```

所有代理都应 处于活动状态，其状态应为 UP。

6. 检查 **undercloud** 计算服务：

```
(undercloud) $ openstack compute service list
```

所有代理的状态都应 为启用状态，其状态应为 **up**

相关信息

- 以下解决方案文章演示了如何删除 OpenStack Orchestration (heat)数据库中删除的堆栈条目：**https://access.redhat.com/solutions/2215131**
  https://access.redhat.com/solutions/2215131

8.2. 验证容器化 OVERCLOUD

以下是检查容器化 **overcloud** 功能的一组步骤。

流程

1. 查找 **undercloud** 访问详情：

```
$ source ~/stackrc
```

2. 检查裸机节点的状态：

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

所有节点均应具有有效的电源状态(on)和维护模式，应为 **false**。

3. 检查失败的 **Systemd** 服务：

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo systemctl list-units --state=failed
'openstack*' 'neutron*' 'httpd' 'docker' 'ceph*'" ; done
```

4. 检查失败的容器化服务：

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'exited=1' --all" ; done
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'status=dead' -f
'status=restarting'" ; done
```

5. **检查与所有服务的 HAProxy 连接。获取 haproxy.stats 服务的 Control Plane VIP 地址和身份验证详情：**

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -
d= -f2); ssh heat-admin@$NODE sudo 'grep "listen haproxy.stats" -A 6 /var/lib/config-
data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg'
```

在以下 cURL 请求中使用这些详情：

```
(undercloud) $ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi
"(frontend|backend)" | cut -d, -f 1,2,18,37,57 | column -s, -t
```

将 <PASSWORD & gt; 和 <IP ADDRESS > 详情替换为 haproxy.stats 服务的实际详情。生成的列表显示每个节点上的 OpenStack Platform 服务及其连接状态。

注意

如果节点运行 Redis 服务，则只有一个节点为该服务显示 ON 状态。这是因为 Redis 是一个主动 - 被动服务，它一次仅在一个节点上运行。

6. **检查 overcloud 数据库复制健康状况：**

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks |
cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec
clustercheck clustercheck" ; done
```

7. **检查 RabbitMQ 集群健康状况：**

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks |
cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec $(ssh
heat-admin@$NODE "sudo docker ps -f 'name=.*rabbitmq.*' -q") rabbitmqctl
node_health_check" ; done
```

8. 检查 **Pacemaker** 资源健康状况：

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo pcs status"
```

查找：

- 所有集群节点 在线。
- 任何集群节点上都没有 停止 资源。
- 没有 失败的 pacemaker 操作。

9. 检查每个 **overcloud** 节点上的磁盘空间：

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo df -h --output=source,fstype,avail -x overlay -x tmpfs -x devtmpfs" ; done
```

10. 检查 **overcloud Ceph Storage** 集群健康状态。以下命令在 **Controller** 节点上运行 **ceph** 工具来检查集群：

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph -s"
```

11. 检查 **Ceph Storage OSD** 是否有可用空间。以下命令在 **Controller** 节点上运行 **ceph** 工具来检查可用空间：

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph df"
```

12. 检查时钟是否在 **overcloud** 节点上同步

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo ntpstat" ; done
```

13. 查找 **overcloud** 访问详情：

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

14. 检查 **overcloud** 网络服务：

```
(overcloud) $ openstack network agent list
```

所有代理都应 处于活动状态，其状态应为 **UP**。

15. 检查 **overcloud** 计算服务：

```
(overcloud) $ openstack compute service list
```

所有代理的状态都应 为启用状态，其状态应为 **up**

16. 检查 **overcloud** 卷服务：

```
(overcloud) $ openstack volume service list
```

所有代理的状态都应 为启用状态，其状态应为 。

相关信息

- 请参阅文章 **"我如何验证我的 OpenStack 环境是通过红帽推荐的配置进行部署？"**。本文提供有关如何检查您的 **Red Hat OpenStack Platform** 环境并对红帽的建议调整配置的一些信息。

8.3. 升级 OVERCLOUD 镜像

您需要将当前的 **overcloud** 镜像替换为新版本。新镜像确保 **director** 可以使用最新版本的 **OpenStack Platform** 软件内省和置备节点。

前提条件

- 您已将 undercloud 升级到最新版本。

流程

1. 查找 undercloud 访问详情：

```
$ source ~/stackrc
```

2. 从 stack 用户主目录(/home/ stack / images)上的 images 目录中删除任何现有的镜像：

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

3. 解压存档：

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-13.0.tar; do tar -xvf $i; done
$ cd ~
```

4. 将最新的镜像导入到 director：

```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

5. 将节点配置为使用新镜像：

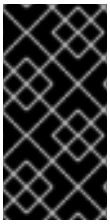
```
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f value)
```

6. 验证新镜像是否存在：

```
$ openstack image list
$ ls -l /httpboot
```

重要

在部署 overcloud 节点时，确保 overcloud 镜像版本对应于对应的 heat 模板版本。例如，仅使用 OpenStack Platform 13 heat 模板的 OpenStack Platform 13 镜像。



重要

新的 overcloud-full 镜像替换旧的 overcloud-full 镜像。如果对旧镜像进行了更改，您必须对新镜像重复更改，特别是未来要部署新节点时。

8.4. 测试部署

虽然 overcloud 已升级，但建议运行测试部署，以确保以后成功部署操作。

流程

1. Source stackrc 文件：

```
$ source ~/stackrc
```

2. 运行部署命令并包括与 overcloud 相关的所有环境文件：

```
$ openstack overcloud deploy \
  --templates \
  -e <ENVIRONMENT FILE>
```

包含以下与您的环境相关的选项：

- 使用 -e 自定义配置环境文件。
- 如果适用，您的自定义角色(roles_data)文件使用 --roles-file。

3. 等待部署完成。

8.5. 总结

本到快进的升级过程到此结束。

附录 A. 恢复 UNDERCLOUD

以下恢复过程会假定 undercloud 节点失败，且处于不可恢复的状态。这个过程包括在全新安装中恢复数据库和关键文件系统。它假设如下：

- 您已重新安装了最新版本的 Red Hat Enterprise Linux 7。
- 硬件布局相同。
- 机器的主机名和 undercloud 设置相同。
- 备份存档已复制到 根目录。

流程

1. 以 root 用户身份登录 undercloud。
2. 使用 Content Delivery Network 注册您的系统，在提示时输入您的客户门户网站用户名和密码：

```
[root@director ~]# subscription-manager register
```

3. 附加 Red Hat OpenStack Platform 权利：

```
[root@director ~]# subscription-manager attach --pool=Valid-Pool-Number-123456
```

4. 禁用所有默认的软件仓库，然后启用所需的 Red Hat Enterprise Linux 软件仓库：

```
[root@director ~]# subscription-manager repos --disable=*
[root@director ~]# subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-openstack-10-rpms
```

5. 在系统上执行更新，确保您有最新的基本系统软件包：

```
-
```

```
[root@director ~]# yum update -y
[root@director ~]# reboot
```

6. 确保同步 undercloud 上的时间。例如：

```
[root@director ~]# yum install -y ntp
[root@director ~]# systemctl start ntpd
[root@director ~]# systemctl enable ntpd
[root@director ~]# ntpdate pool.ntp.org
[root@director ~]# systemctl restart ntpd
```

7. 将 undercloud 备份存档复制到 undercloud 的根目录。以下步骤使用 undercloud-backup-$TIMESTAMP.tar 作为文件名，其中 $TIMESTAMP 是归档的时间戳的 Bash 变量。

8. 安装数据库服务器和客户端工具：

```
[root@director ~]# yum install -y mariadb mariadb-server
```

9. 启动数据库：

```
[root@director ~]# systemctl start mariadb
[root@director ~]# systemctl enable mariadb
```

10. 增加允许的数据包以适应数据库备份的大小：

```
[root@director ~]# mysql -uroot -e"set global max_allowed_packet = 1073741824;"
```

11. 从存档中提取数据库和数据库配置：

```
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar
etc/my.cnf.d/*server*.cnf
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar root/undercloud-all-
databases.sql
```

12. 恢复数据库备份：

```
[root@director ~]# mysql -u root < /root/undercloud-all-databases.sql
```

13. **提取 root 配置文件的临时版本：**

```
[root@director ~]# tar -xvf undercloud-backup-$TIMESTAMP.tar root/.my.cnf
```

14. **获取旧的 root 数据库密码：**

```
[root@director ~]# OLDPASSWORD=$(sudo cat root/.my.cnf | grep -m1 password | cut -d'='
-f2 | tr -d """)
```

15. **重置 root 数据库密码：**

```
[root@director ~]# mysqladmin -u root password "$OLDPASSWORD"
```

16. **将 root 配置文件从临时目录移动到 根目录 ：**

```
[root@director ~]# mv ~/root/.my.cnf ~/.
[root@director ~]# rmdir ~/root
```

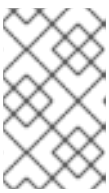
17. **获取旧用户权限列表：**

```
[root@director ~]# mysql -e 'select host, user, password from mysql.user;'
```

18. **删除列出的每个主机的旧用户权限。例如：**

```
[root@director ~]# HOST="192.0.2.1"
[root@director ~]# USERS=$(mysql -Nse "select user from mysql.user WHERE user !=
\"root\" and host = \"$HOST\";" | uniq | xargs)
[root@director ~]# for USER in $USERS ; do mysql -e "drop user \"$USER\"@\"$HOST\"" ||
true ;done
[root@director ~]# for USER in $USERS ; do mysql -e "drop user $USER" || true ;done
[root@director ~]# mysql -e 'flush privileges'
```

对通过主机 IP 和任何主机("%")访问的所有用户执行此操作。



注意

HOST 参数中的 IP 地址是 control plane 中的 undercloud IP 地址。

19. **重启数据库：**

```
[root@director ~]# systemctl restart mariadb
```

20. **创建 stack 用户：**

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

21. **为该用户设置密码：**

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

22. **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
```

23. **恢复 stack 用户主目录：**

```
# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar home/stack
```

24. **安装 policycoreutils-python 软件包：**

```
[root@director ~]# yum -y install policycoreutils-python
```

25. **安装 openstack-glance 软件包并恢复其数据和文件权限：**

```
[root@director ~]# yum install -y openstack-glance
[root@director ~]# tar --xattrs --xattrs-include='*.*' -xvC / -f undercloud-backup-
$TIMESTAMP.tar var/lib/glance/images
[root@director ~]# chown -R glance: /var/lib/glance/images
[root@director ~]# restorecon -R /var/lib/glance/images
```

26. **安装 openstack-swift 软件包并恢复其数据和文件权限：**

```
[root@director ~]# yum install -y openstack-swift
[root@director ~]# tar --xattrs --xattrs-include='*.*' -xvC / -f undercloud-backup-
```

```
$TIMESTAMP.tar srv/node
[root@director ~]# chown -R swift: /srv/node
[root@director ~]# restorecon -R /srv/node
```

27. **安装 openstack-keystone 软件包并恢复其配置数据：**

```
[root@director ~]# yum -y install openstack-keystone
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/keystone
[root@director ~]# restorecon -R /etc/keystone
```

28. **安装 openstack-heat 和 restore 配置：**

```
[root@director ~]# yum install -y openstack-heat*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/heat
[root@director ~]# restorecon -R /etc/heat
```

29. **安装 puppet 并恢复其配置数据：**

```
[root@director ~]# yum install -y puppet hiera
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/puppet/hieradata/
```

30. **如果在 undercloud 中使用 SSL，请刷新 CA 证书。根据 undercloud 配置，使用用户提供的证书的步骤或自动生成的证书的步骤：**

- **如果 undercloud 配置了用户提供的证书，请完成以下步骤：**

  a. **提取证书：**

  ```
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/instack-certs/undercloud.pem
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/ca-trust/source/anchors/*
  ```

  b. **恢复 SELinux 上下文并管理文件系统标记：**

  ```
  [root@director ~]# restorecon -R /etc/pki
  [root@director ~]# semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.*)?"
  [root@director ~]# restorecon -R /etc/pki/instack-certs
  ```

c. **更新证书：**

```
[root@director ~]# update-ca-trust extract
```

- **如果您使用 certmonger 自动 undercloud 的证书，请完成以下步骤：**

a. **提取证书、CA 证书和 certmonger 文件：**

```
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar
var/lib/certmonger/*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/tls/*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/ca-
trust/source/anchors/*
```

b. **恢复 SELinux 上下文：**

```
[root@director ~]# restorecon -R /etc/pki
[root@director ~]# restorecon -R /var/lib/certmonger
```

c. **删除 /var/lib/certmonger/lock 文件：**

```
[root@director ~]# rm -f /var/lib/certmonger/lock
```

31. **切换到 stack 用户：**

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

32. **安装 python-tripleoclient 软件包：**

```
$ sudo yum install -y python-tripleoclient
```

33. **运行 undercloud 安装命令。确保在 stack 用户的主目录中运行它：**

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

安装完成后，undercloud 会自动恢复与 overcloud 的连接。节点继续轮询 OpenStack

Orchestration (heat)是否有待处理的任务。

附录 B. 恢复 OVERCLOUD

B.1. 恢复 OVERCLOUD CONTROL PLANE 服务

以下流程恢复 overcloud 数据库和配置的备份。在这种情况下，建议在所有三个 Controller 节点上同时打开三个终端窗口。另外，建议您选择 Controller 节点来执行高可用性操作。此流程将这个 Controller 节点称为 ***bootstrap Controller* 节点**。

重要

此流程只恢复 control plane 服务。它不包括在 Ceph Storage 节点上恢复 Compute 节点工作负载或数据。

注意

红帽支持使用原生 SDN 备份 Red Hat OpenStack Platform，如 Open vSwitch (OVS)和默认的 Open Virtual Network (OVN)。有关第三方 SDN 的信息，请参阅第三方 SDN 文档。

流程

1. 停止 Pacemaker 并删除所有容器化服务。

   a. 登录到 bootstrap Controller 节点并停止 pacemaker 集群：

   ```
   # sudo pcs cluster stop --all
   ```

   b. 等待集群完全关闭：

   ```
   # sudo pcs status
   ```

   c. 在所有 Controller 节点上，删除所有 OpenStack 服务的容器：

   ```
   # docker stop $(docker ps -a -q)
   # docker rm $(docker ps -a -q)
   ```

2. 如果您要从失败的主发行版本升级中恢复，您可能需要反转所有节点上的 yum 事务。这包括

每个节点中的以下内容：

a. 为之前的版本启用软件仓库。例如：

```
# sudo subscription-manager repos --enable=rhel-7-server-openstack-10-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
```

b. 启用以下 Ceph 软件仓库：

```
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-2-tools-rpms
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-2-mon-rpms
```

c. 检查 yum 历史记录：

```
# sudo yum history list all
```

识别升级过程中发生的事务。这些操作在其中一个 Controller 节点上发生（在升级过程中选择为 bootstrap 节点的 Controller 节点）。如果您需要查看特定的事务，使用 history info 子命令查看它：

```
# sudo yum history info 25
```

注意

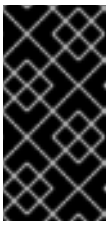
要强制 yum history list all 显示从每个事务运行的命令，请在 yum.conf 文件中设置 history_list_view=commands。

d. 恢复升级后发生的任何 yum 事务。例如：

```
# sudo yum history undo 25
# sudo yum history undo 24
# sudo yum history undo 23
...
```

确保从最后一个事务开始，再按降序排列。您还可以使用回滚选项在一个执行时恢复多个事务。例如，以下命令将最后事务中的事务回滚至 23：

```
# sudo yum history rollback 23
```



重要

建议对每个事务使用 撤消 而不是 回滚，以便您可以验证每个事务的逆转。

e.

相关的 yum 事务被反向后，在所有节点上仅启用原始 OpenStack Platform 存储库。例如：

```
# sudo subscription-manager repos --disable=rhel-7-server-openstack-*-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-10-rpms
```

f.

禁用以下 Ceph 软件仓库：

```
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
```

3.

恢复数据库：

a.

将数据库备份复制到 bootstrap Controller 节点。

b.

停止到所有 Controller 节点上的数据库端口的外部连接：

```
# MYSQLIP=$(hiera -c /etc/puppet/hiera.yaml mysql_bind_host)
# sudo /sbin/iptables -I INPUT -d $MYSQLIP -p tcp --dport 3306 -j DROP
```

这会隔离到节点的所有数据库流量。

c.

暂时禁用数据库复制。在所有 Controller 节点上编辑 /etc/my.cnf.d/galera.cnf 文件。

```
# vi /etc/my.cnf.d/galera.cnf
```

进行以下更改：

- 注释掉 **wsrep_cluster_address** 参数。
- 将 **wsrep_provider** 设置为 **none**

d. 保存 **/etc/my.cnf.d/galera.cnf** 文件。

e. 确保所有 **Controller** 节点上都禁用 **MariaDB** 数据库。在升级到 **OpenStack Platform 13** 时，**MariaDB** 服务移至您之前禁用的容器化服务。确保服务没有作为主机上的进程运行：

```
# mysqladmin -u root shutdown
```

注意

您可以从禁用数据库的 **HAProxy** 收到警告。

f. 移动现有的 **MariaDB** 数据目录，并在所有 **Controller** 节点上准备新数据目录，

```
# mv /var/lib/mysql/ /var/lib/mysql.old
# mkdir /var/lib/mysql
# chown mysql:mysql /var/lib/mysql
# chmod 0755 /var/lib/mysql
# mysql_install_db --datadir=/var/lib/mysql --user=mysql
# chown -R mysql:mysql /var/lib/mysql/
# restorecon -R /var/lib/mysql
```

g. 在所有 **Controller** 节点上手动启动数据库：

```
# mysqld_safe --skip-grant-tables --skip-networking --wsrep-on=OFF &
```

h. 在所有 **Controller** 节点上获取旧的密码重置数据库密码：

```
# OLDPASSWORD=$(sudo cat .my.cnf | grep -m1 password | cut -d'=' -f2 | tr -d "'")
# mysql -uroot -e"use mysql;update user set
password=PASSWORD($OLDPASSWORD)"
```

i. 停止所有 **Controller** 节点上的数据库：

```
# /usr/bin/mysqladmin -u root shutdown
```

j. **在 bootstrap Controller 节点上手动启动数据库，无需 --skip-grant-tables 选项：**

```
# mysqld_safe --skip-networking --wsrep-on=OFF &
```

k. **在 bootstrap Controller 节点上，恢复 OpenStack 数据库。这会稍后复制到其他 Controller 节点：**

```
# mysql -u root < openstack_database.sql
```

l. **在 bootstrap 控制器节点上，恢复用户和权限：**

```
# mysql -u root < grants.sql
```

m. **使用以下命令关闭 bootstrap Controller 节点：**

```
# mysqladmin shutdown
```

n. **启用数据库复制。在所有 Controller 节点上编辑 /etc/my.cnf.d/galera.cnf 文件。**

```
# vi /etc/my.cnf.d/galera.cnf
```

进行以下更改：

- **取消注释 wsrep_cluster_address 参数。**
- **将 wsrep_provider 设置为 /usr/lib64/galera/libgalera_smm.so**

o. **保存 /etc/my.cnf.d/galera.cnf 文件。**

p. **在 bootstrap 节点上运行数据库：**

```
# /usr/bin/mysqld_safe --pid-file=/var/run/mysql/mysqld.pid --
```

```
socket=/var/lib/mysql/mysql.sock --datadir=/var/lib/mysql --log-
error=/var/log/mysql_cluster.log --user=mysql --open-files-limit=16384 --wsrep-cluster-
address=gcomm:// &
```

--wsrep-cluster-address 选项中没有节点会强制 Galera 创建新集群，并使 bootstrap 节点成为 master 节点。

q. 检查节点的状态：

```
# clustercheck
```

此命令应该报告 Galera 集群节点已同步。检查 /var/log/mysql_cluster.log 文件中的错误。

r. 在剩余的 Controller 节点上，启动数据库：

```
$ /usr/bin/mysqld_safe --pid-file=/var/run/mysql/mysqld.pid --
socket=/var/lib/mysql/mysql.sock --datadir=/var/lib/mysql --log-
error=/var/log/mysql_cluster.log  --user=mysql --open-files-limit=16384 --wsrep-cluster-
address=gcomm://overcloud-controller-0,overcloud-controller-1,overcloud-controller-2 &
```

在 --wsrep-cluster-address 选项中包含节点会将节点添加到新集群中，并从 master 同步内容。

s. 定期检查每个节点的状态：

```
# clustercheck
```

当所有节点都完成其同步操作时，这个命令应该 为每个节点报告 Galera 集群节点。

t. 停止所有节点上的数据库：

```
$ mysqladmin shutdown
```

u. 从用于恢复对数据库的访问的服务从每个节点中删除防火墙规则：

```
# sudo /sbin/iptables -D INPUT -d $MYSQLIP -p tcp --dport 3306 -j DROP
```

4. **恢复 Pacemaker 配置：**

   a. **将 Pacemaker 归档复制到 bootstrap 节点。**

   b. **登录到 bootstrap 节点。**

   c. **运行配置恢复命令：**

```
# pcs config restore pacemaker_controller_backup.tar.bz2
```

5. **恢复文件系统：**

   a. **将每个 Controller 节点的 backup tar 文件复制到一个临时目录中，并解压缩所有数据：**

```
# mkdir /var/tmp/filesystem_backup/
# cd /var/tmp/filesystem_backup/
# mv <backup_file>.tar.gz .
# tar --xattrs --xattrs-include='*.*' -xvzf <backup_file>.tar.gz
```

注意

不要直接提取到 / 目录。这会覆盖您当前的文件系统。建议将 文件提取到临时目录中。

   b. **恢复 os-*-config 文件并重启 os-collect-config ：**

```
# cp -rf /var/tmp/filesystem_backup/var/lib/os-collect-config/* /var/lib/os-collect-config/.
# cp -rf /var/tmp/filesystem_backup/usr/libexec/os-apply-config/* /usr/libexec/os-apply-config/.
# cp -rf /var/tmp/filesystem_backup/usr/libexec/os-refresh-config/* /usr/libexec/os-refresh-config/.
# systemctl restart os-collect-config
```

   c. **恢复 Puppet hieradata 文件：**

```
# cp -r /var/tmp/filesystem_backup/etc/puppet/hieradata /etc/puppet/hieradata
# cp -r /var/tmp/filesystem_backup/etc/puppet/hiera.yaml /etc/puppet/hiera.yaml
```

d. **保存这个目录，以防需要任何配置文件。**

6. **恢复 redis 资源：**

a. **将 Redis 转储复制到每个 Controller 节点。**

b. **将 Redis 转储移动到每个 Controller 上的原始位置：**

```
# mv dump.rdb /var/lib/redis/dump.rdb
```

c. **恢复 Redis 目录的权限：**

```
# chown -R redis: /var/lib/redis
```

7. **删除以下任何目录的内容：**

```
# rm -rf /var/lib/config-data/puppet-generated/*
# rm /root/.ffu_workaround
```

8. **恢复 OpenStack Object Storage (swift)服务的权限：**

```
# chown -R swift: /srv/node
# chown -R swift: /var/lib/swift
# chown -R swift: /var/cache/swift
```

9. **登录 undercloud，并从 OpenStack Platform 10 部署中运行原始 openstack overcloud deploy 命令。确保包含与原始部署相关的所有环境文件。**

10. **等待部署完成。**

11. **恢复 overcloud control plane 数据后，检查每个相关服务是否已启用并在正确运行：**

a. 对于控制器节点上的高可用性服务：

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

b. 对于控制器和计算节点上的系统服务：

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

下面几节提供了应启用的服务参考。

B.2. 恢复的高可用性服务

以下是在恢复后在 OpenStack Platform 10 Controller 节点上活跃的高可用性服务列表。如果禁用其中任何服务，使用以下命令启用它们：

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

Controller 服务
galera
hapoxy
openstack-cinder-volume
rabbitmq
redis

B.3. 恢复的 CONTROLLER 服务

以下是在恢复后在 OpenStack Platform 10 Controller 节点上应激活的核心 Systemd 服务列表。如果禁用其中任何服务，使用以下命令启用它们：

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

Controller 服务
httpd
memcached
neutron-dhcp-agent
neutron-l3-agent
neutron-metadata-agent
neutron-openvswitch-agent
neutron-ovs-cleanup
neutron-server
ntpd
openstack-aodh-evaluator
openstack-aodh-listener
openstack-aodh-notifier
openstack-ceilometer-central
openstack-ceilometer-collector
openstack-ceilometer-notification
openstack-cinder-api
openstack-cinder-scheduler
openstack-glance-api
openstack-glance-registry
openstack-gnocchi-metricd
openstack-gnocchi-statsd
openstack-heat-api-cfn
openstack-heat-api-cloudwatch

Controller 服务
openstack-heat-api
openstack-heat-engine
openstack-nova-api
openstack-nova-conductor
openstack-nova-consoleauth
openstack-nova-novncproxy
openstack-nova-scheduler
openstack-swift-account-auditor
openstack-swift-account-reaper
openstack-swift-account-replicator
openstack-swift-account
openstack-swift-container-auditor
openstack-swift-container-replicator
openstack-swift-container-updater
openstack-swift-container
openstack-swift-object-auditor
openstack-swift-object-expirer
openstack-swift-object-replicator
openstack-swift-object-updater
openstack-swift-object
openstack-swift-proxy
openvswitch
os-collect-config

Controller 服务
ovs-delete-transient-ports
ovs-vswitchd
ovsdb-server
pacemaker

B.4. 恢复的 OVERCLOUD COMPUTE 服务

以下是在恢复后在 OpenStack Platform 10 Compute 节点上应激活的核心 Systemd 服务列表。如果禁用其中任何服务，使用以下命令启用它们：

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

Compute Services
neutron-openvswitch-agent
neutron-ovs-cleanup
ntpd
openstack-ceilometer-compute
openstack-nova-compute
openvswitch
os-collect-config