Red Hat OpenStack Platform 13

过渡到容器化服务

使用 OpenStack Platform 容器化服务的基本指南

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 过渡到容器化服务

使用 OpenStack Platform 容器化服务的基本指南

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本指南提供了一些基本信息，用于帮助用户熟悉在容器中运行的 OpenStack Platform 服务。

目录

第 1 章 简介

旧版 Red Hat OpenStack Platform 使用了通过 Systemd 管理的服务。但是，最新版本的 OpenStack Platform 现在使用容器来运行服务。有些管理员可能不知道容器化 OpenStack Platform 服务如何运作，因此本指南旨在帮助您了解 OpenStack Platform 容器镜像和容器化服务。这包括：

- 如何获取和修改容器镜像
- 如何在 overcloud 中管理容器化服务
- 了解容器与 Systemd 服务的不同

其主要目标是帮助您充分了解容器化 OpenStack 平台服务，从基于 Systemd 的环境过渡到基于容器的环境。

1.1. 容器化服务和 KOLLA

每个主 Red Hat OpenStack Platform (RHOSP)服务都在容器中运行。这提供了一种将每个服务保持在与主机分离的隔离命名空间内的方法。这样做有以下影响：

- 在部署过程中，RHOSP 从红帽客户门户网站中提取并运行容器镜像。
- **podman** 命令运行管理功能，如启动和停止服务。
- 要升级容器，您必须拉取新的容器镜像，并使用更新的版本替换现有的容器。

Red Hat OpenStack Platform 使用了通过 **Kolla** 工具集来构建和管理的一组容器。

第 2 章 获取和修改容器镜像

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 2.2 节 “容器镜像准备命令用法”。

2.1. 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。

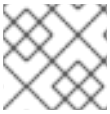
注意

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。

2.2. 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。

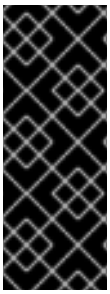
--tag

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。

2.3. 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**
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)。

Service	环境文件
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

2.4. 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。

2.5. 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。

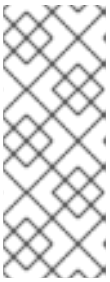
注意

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。

2.6. 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：

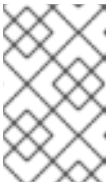
```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=satellite6.example.com:5000 \
  --prefix=acme-osp13_containers- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

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。

2.7. 修改容器镜像

红帽通过 Red Hat Container Catalog (**registry.redhat.io**)提供了一组预构建的容器镜像。可以修改这些镜像并向其中添加其他层。这对于向容器添加认证第三方驱动程序的 RPM 非常有用。

注意

为确保继续支持修改后的 OpenStack Platform 容器镜像，请确保生成的镜像符合 "Red Hat Container Support Policy"。

本例演示了如何自定义最新的 **openstack-keystone** 镜像。但是，这些说明也可以应用到其他镜像：

流程

1. 拉取您要修改的镜像。例如，对于 **openstack-keystone** 镜像：

```
$ sudo docker pull registry.redhat.io/rhosp13/openstack-keystone:latest
```

2. 检查原始镜像上的默认用户。例如，对于 **openstack-keystone** 镜像：

```
$ sudo docker run -it registry.redhat.io/rhosp13/openstack-keystone:latest whoami
root
```

注意

openstack-keystone 镜像使用 **root** 作为默认用户。其他镜像使用特定用户。例如，**openstack-glance-api** 将 **glance** 用于默认用户。

3. 创建 **Dockerfile** 以在现有容器镜像上构建额外层。以下示例从 Container Catalog 拉取最新的 OpenStack Identity (keystone)镜像，并将自定义 RPM 文件安装到镜像中：

```
FROM registry.redhat.io/rhosp13/openstack-keystone
MAINTAINER Acme
LABEL name="rhosp13/openstack-keystone-acme" vendor="Acme" version="2.1"
release="1"

# switch to root and install a custom RPM, etc.
USER root
COPY custom.rpm /tmp
RUN rpm -ivh /tmp/custom.rpm
```

```
# switch the container back to the default user
USER root
```

4. 构建并标记新镜像。例如，使用保存在 **/home/stack/keystone** 目录中的本地 **Dockerfile** 进行构建，并将其标记为 undercloud 的本地 registry：

```
$ docker build /home/stack/keystone -t "192.168.24.1:8787/rhosp13/openstack-keystone-acme:rev1"
```

5. 将生成的镜像推送到 undercloud 的本地 registry：

```
$ docker push 192.168.24.1:8787/rhosp13/openstack-keystone-acme:rev1
```

6. 编辑 overcloud 容器镜像环境文件（通常为 **overcloud_images.yaml**）并更改适当的参数，以使用自定义容器镜像。

警告

容器目录发布容器镜像，其中包含内置于其中的完整软件堆栈。当容器目录发布包含更新和安全修复的容器镜像时，您现有的自定义容器将 **不包括** 这些更新，且需要使用从 Catalog 中的新镜像版本重建。

第 3 章 使用容器部署和更新 OVERCLOUD

本章提供了如何创建基于容器的 overcloud 并保持更新的信息。

3.1. 部署 OVERCLOUD

此流程演示了如何使用最小配置部署 overcloud。结果将是一个基本的双节点 overcloud (1 个 Controller 节点、1 个 Compute 节点)。

流程

1. Source **stackrc** 文件：

```
$ source ~/stackrc
```

2. 运行 **deploy** 命令，并包含包含 overcloud 镜像位置的文件（通常为 **overcloud_images.yaml**）：

```
(undercloud) $ openstack overcloud deploy --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  --ntp-server pool.ntp.org
```

3. 等待 overcloud 完成部署。

3.2. 更新 OVERCLOUD

有关更新容器化 overcloud 的信息，请参阅 *保持 Red Hat OpenStack Platform 更新* 指南。

第 4 章 使用容器化服务

本章提供了一些管理容器的命令示例，以及如何排除您的 OpenStack Platform 容器

4.1. 管理容器化服务

overcloud 在容器中运行大多数 OpenStack Platform 服务。在某些情况下，您可能需要控制主机上的单个服务。本节提供了一些常见的 **docker** 命令，您可以在 overcloud 节点上运行来管理容器化服务。有关使用 **docker** *管理容器的更多信息，请参阅开始使用* 容器指南中的 Docker 格式 容器。

注意

在运行这些命令前，请检查您是否已登录到 overcloud 节点，而不是在 undercloud 上运行这些命令。

列出容器和镜像

列出正在运行的容器：

```
$ sudo docker ps
```

另外，要列出已停止或失败的容器，请添加 **--all** 选项：

```
$ sudo docker ps --all
```

列出容器镜像：

```
$ sudo docker images
```

检查容器属性

要查看容器或容器镜像的属性，请使用 **docker inspect** 命令。例如，检查 **keystone** 容器：

```
$ sudo docker inspect keystone
```

管理基本容器操作

要重启容器化服务，请使用 **docker restart** 命令。例如，要重启 **keystone** 容器：

```
$ sudo docker restart keystone
```

要停止容器化服务，请使用 **docker stop** 命令。例如，停止 **keystone** 容器：

```
$ sudo docker stop keystone
```

要启动已停止的容器化服务，请使用 **docker start** 命令。例如，要启动 **keystone** 容器：

```
$ sudo docker start keystone
```

注意

在重启容器后，针对其中的服务配置文件所做的所有更改都会恢复。这是因为容器基于 **/var/lib/config-data/puppet-generated/** 中节点本地文件系统上的文件重新生成服务配置。例如，如果您编辑了 **keystone** 容器中的 **/etc/keystone/keystone.conf**，并重启了该容器，则该容器会使用节点的本地文件系统上的 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 来重新生成配置，以覆盖重启之前在该容器中所做的所有更改。

监控容器

要检查容器化服务的日志，请使用 **docker logs** 命令。例如，查看 **keystone** 容器的日志：

```
$ sudo docker logs keystone
```

访问容器

要进入容器化服务的 shell，请使用 **docker exec** 命令启动 **/bin/bash**。例如，输入 **keystone** 容器的 shell：

```
$ sudo docker exec -it keystone /bin/bash
```

以 root 用户身份输入 **keystone** 容器的 shell：

```
$ sudo docker exec --user 0 -it <NAME OR ID> /bin/bash
```

退出容器：

```
# exit
```

4.2. 容器化服务故障排除

如果容器化服务在 overcloud 部署期间或之后失败，请使用以下建议来确定故障的根本原因：

注意

在运行这些命令前，请检查您是否已登录到 overcloud 节点，而不是在 undercloud 上运行这些命令。

检查容器日志

每个容器都会保留其主进程的标准输出内容。此输出作为日志作用，可以帮助确定容器运行期间实际发生的情况。例如，要查看 **keystone** 容器的日志，请使用以下命令：

```
$ sudo docker logs keystone
```

在大多数情况下，此日志提供了容器失败的原因。

检查容器

在某些情况下，您可能需要验证容器的相关信息。例如，请使用以下命令来查看 **keystone** 容器的相关数据：

```
$ sudo docker inspect keystone
```

这提供了一个包含低级配置数据的 JSON 对象。您可以通过管道将这些输出内容传递给 **jq** 命令，以对特定数据进行解析。例如，要查看 **keystone** 容器的加载情况，请运行以下命令：

```
$ sudo docker inspect keystone | jq .[0].Mounts
```

您还可以使用 **--format** 选项将数据解析到一行中，这在针对一组容器数据运行命令时非常有用。例如，要重建用于运行 **keystone** 容器的选项，请使用包含 **--format** 选项的以下 **inspect** 命令：

```
$ sudo docker inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range .Mounts}} -v
{{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}' keystone
```

注意

--format 选项会按照 Go 语法来创建查询。

将这些选项与 **docker run** 命令结合使用来重新创建容器以进行故障排除：

```
$ OPTIONS=$( sudo docker inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range
.Mounts}} -v {{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}'
keystone )
$ sudo docker run --rm $OPTIONS /bin/bash
```

在容器内运行命令

在某些情况下，您可能需要通过特定的 Bash 命令从容器中获取信息。在这种情况下，使用以下 **docker** 命令在运行中的容器内执行命令。例如，要在 **keystone** 容器中运行命令：

```
$ sudo docker exec -ti keystone <COMMAND>
```

注意

-ti 选项会通过交互式伪终端来运行命令。

将 **<COMMAND>** 替换为您的所需命令。例如，每个容器都有一个健康检查脚本，用于验证服务的连接状况。您可以使用以下命令为 **keystone** 运行这个健康检查脚本：

```
$ sudo docker exec -ti keystone /openstack/healthcheck
```

要访问容器的 shell，请运行 **docker exec**，将 **/bin/bash** 用作命令：

```
$ sudo docker exec -ti keystone /bin/bash
```

导出容器

当容器出现故障时，您可能需要调查文件中包含的所有内容。在这种情况下，您可以将容器的整个文件系统导出为 **tar** 归档。例如，要导出 **keystone** 容器的文件系统，请运行以下命令：

```
$ sudo docker export keystone -o keystone.tar
```

这个命令会创建 **keystone.tar** 归档，以供您提取和研究。

第 5 章 SYSTEMD 服务与容器化服务比较

本章提供了一些参考材料，用于显示容器化服务与 Systemd 服务有何不同。

5.1. SYSTEMD 服务命令与容器化服务命令

下表显示了基于 Systemd 的命令和对应的 Docker 之间的一些相似性。这有助于识别您要执行的服务操作类型。

功能	基于 systemd	基于 Docker
列出所有服务	**systemctl list-units -t service**	**docker ps --all**
列出活跃服务	**systemctl list-units -t service --state active**	**Docker ps**
检查服务的状态	**systemctl status openstack-nova-api**	**docker ps --filter "name=nova_api$" --all**
停止服务	**systemctl stop openstack-nova-api**	**docker stop nova_api**
启动服务	**systemctl start openstack-nova-api**	**docker start nova_api**
重启服务	**systemctl restart openstack-nova-api**	**docker restart nova_api**
显示服务配置	**systemctl show openstack-nova-api** **systemctl cat openstack-nova-api**	**docker inspect nova_api**
显示服务日志	**journalctl -u openstack-nova-api**	**docker logs nova_api**

5.2. SYSTEMD 服务与容器化服务

下表显示了基于 Systemd 的 OpenStack 服务及其基于容器的等效服务。

OpenStack 服务	systemd 服务	Docker 容器

OpenStack 服务	systemd 服务	Docker 容器
Aodh	**openstack-aodh-evaluator** **openstack-aodh-listener** **openstack-aodh-notifier** **httpd (openstack-aodh-api)**	**aodh_listener** **aodh_api** **aodh_notifier** **aodh_evaluator**
Ceilometer	**openstack-ceilometer-central** **openstack-ceilometer-collector** **openstack-ceilometer-notification** **httpd (openstack-ceilometer-api)**	**ceilometer_agent_notification** **ceilometer_agent_central**
cinder	**openstack-cinder-api** **openstack-cinder-scheduler** **openstack-cinder-volume**	**cinder_scheduler** **cinder_api** **openstack-cinder-volume-docker-0**
Glance	**openstack-glance-api** **openstack-glance-registry**	**glance_api**
Gnocchi	**openstack-gnocchi-metricd** **openstack-gnocchi-statsd** **httpd (openstack-gnocchi-api)**	**gnocchi_statsd** **gnocchi_api** **gnocchi_metricd**
Heat	**openstack-heat-api-cfn** **openstack-heat-api-cloudwatch** **openstack-heat-api** **openstack-heat-engine**	**heat_api_cfn** **heat_engine** **heat_api**
Horizon	**httpd (openstack-dashboard)**	**Horizon**
keystone	**httpd (openstack-keystone)**	**Keystone**

OpenStack 服务	systemd 服务	Docker 容器
neutron	**neutron-dhcp-agent** **neutron-l3-agent** **neutron-metadata-agent** **neutron-openvswitch-agent** **neutron-server**	**neutron_ovs_agent** **neutron_l3_agent** **neutron_metadata_agent** **neutron_dhcp** **neutron_api**
Nova	**openstack-nova-api** **openstack-nova-conductor** **openstack-nova-consoleauth** **openstack-nova-novncproxy** **openstack-nova-scheduler**	**nova_metadata** **nova_api** **nova_conductor** **nova_vnc_proxy** **nova_consoleauth** **nova_api_cron** **nova_scheduler** **nova_placement**
Panko		**panko_api**

OpenStack 服务	systemd 服务	Docker 容器
swift	**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**	**swift_proxy** **swift_account_server** **swift_container_auditor** **swift_object_expirer** **swift_object_updater** **swift_container_replicator** **swift_account_auditor** **swift_object_replicator** **swift_container_server** **swift_rsync** **swift_account_reaper** **swift_account_replicator** **swift_object_auditor** **swift_object_server** **swift_container_update**

5.3. SYSTEMD 日志位置和容器化日志位置

下表显示了基于 Systemd 的 OpenStack 日志及其容器的等效日志。所有基于容器的日志位置都在物理主机上可用，并挂载到容器。

OpenStack 服务	systemd 服务日志	Docker 容器日志
Aodh	**/var/log/aodh/**	**/var/log/containers/aodh/** **/var/log/containers/httpd/aodh-api/**
Ceilometer	**/var/log/ceilometer/**	**/var/log/containers/ceilometer/**

OpenStack 服务	systemd 服务日志	Docker 容器日志
cinder	**/var/log/cinder/**	**/var/log/containers/cinder/** **/var/log/containers/httpd/cinder-api/**
Glance	**/var/log/glance/**	**/var/log/containers/glance/**
Gnocchi	**/var/log/gnocchi/**	**/var/log/containers/gnocchi/** **/var/log/containers/httpd/gnocchi-api/**
Heat	**/var/log/heat/**	**/var/log/containers/heat/** **/var/log/containers/httpd/heat-api/** **/var/log/containers/httpd/heat-api-cfn/**
Horizon	**/var/log/horizon/**	**/var/log/containers/horizon/** **/var/log/containers/httpd/horizon/**
keystone	**/var/log/keystone/**	**/var/log/containers/keystone** **/var/log/containers/httpd/keystone/**
数据库	**/var/log/mariadb/** **/var/log/mongodb/** **/var/log/mysqld.log**	**/var/log/containers/mysql/**
neutron	**/var/log/neutron/**	**/var/log/containers/neutron/** **/var/log/containers/httpd/neutron-api/**
Nova	**/var/log/nova/**	**/var/log/containers/nova/** **/var/log/containers/httpd/nova-api/** **/var/log/containers/httpd/nova-placement/**

OpenStack 服务	systemd 服务日志	Docker 容器日志
Panko		**/var/log/containers/panko/** **/var/log/containers/httpd/panko-api/**
rabbitmq	**/var/log/rabbitmq/**	**/var/log/containers/rabbitmq/**
redis	**/var/log/redis/**	**/var/log/containers/redis/**
swift	**/var/log/swift/**	**/var/log/containers/swift/**

5.4. SYSTEMD 配置与容器化配置

下表显示了基于 Systemd 的 OpenStack 配置及其容器的等效配置。所有基于容器的配置位置都在物理主机上可用，并挂载到容器，并将（通过 **kolla**）合并到各个容器中的配置中。

OpenStack 服务	systemd 服务配置	Docker 容器配置
Aodh	**/etc/aodh/**	**/var/lib/config-data/puppet-generated/aodh/**
Ceilometer	**/etc/ceilometer/**	**/var/lib/config-data/puppet-generated/ceilometer/etc/ceilometer/**
cinder	**/etc/cinder/**	**/var/lib/config-data/puppet-generated/cinder/etc/cinder/**
Glance	**/etc/glance/**	**/var/lib/config-data/puppet-generated/glance_api/etc/glance/**
Gnocchi	**/etc/gnocchi/**	**/var/lib/config-data/puppet-generated/gnocchi/etc/gnocchi/**
hapoxy	**/etc/haproxy/**	**/var/lib/config-data/puppet-generated/haproxy/etc/haproxy/**

OpenStack 服务	systemd 服务配置	Docker 容器配置
Heat	**/etc/heat/**	**/var/lib/config-data/puppet-generated/heat/etc/heat/** **/var/lib/config-data/puppet-generated/heat_api/etc/heat/** **/var/lib/config-data/puppet-generated/heat_api_cfn/etc/heat/**
Horizon	**/etc/openstack-dashboard/**	**/var/lib/config-data/puppet-generated/horizon/etc/openstack-dashboard/**
keystone	**/etc/keystone/**	**/var/lib/config-data/puppet-generated/keystone/etc/keystone/**
数据库	**/etc/my.cnf.d/** **/etc/my.cnf**	**/var/lib/config-data/puppet-generated/mysql/etc/my.cnf.d/**
neutron	**/etc/neutron/**	**/var/lib/config-data/puppet-generated/neutron/etc/neutron/**
Nova	**/etc/nova/**	**/var/lib/config-data/puppet-generated/nova/etc/nova/** **/var/lib/config-data/puppet-generated/nova_placement/etc/nova/**
Panko		**/var/lib/config-data/puppet-generated/panko/etc/panko**
rabbitmq	**/etc/rabbitmq/**	**/var/lib/config-data/puppet-generated/rabbitmq/etc/rabbitmq/**
redis	**/etc/redis/** **/etc/redis.conf**	**/var/lib/config-data/puppet-generated/redis/etc/redis/** **/var/lib/config-data/puppet-generated/redis/etc/redis.conf**

OpenStack 服务	systemd 服务配置	Docker 容器配置
swift	**/etc/swift/**	**/var/lib/config-data/puppet-generated/swift/etc/swift/** **/var/lib/config-data/puppet-generated/swift_ringbuilder/etc/swift/**