Red Hat OpenStack Platform 17.1

使用 director Operator 在 Red Hat OpenShift Container Platform 集群中部署 overcloud

使用 director Operator 在 Red Hat OpenShift Container Platform 中部署和管理 Red Hat OpenStack Platform overcloud

Last Updated: 2024-05-25

Red Hat OpenStack Platform 17.1 使用 director Operator 在 Red Hat OpenShift Container Platform 集群中部署 overcloud

使用 director Operator 在 Red Hat OpenShift Container Platform 中部署和管理 Red Hat OpenStack Platform overcloud

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

了解如何在 Red Hat OpenShift Container Platform 集群中安装 RHOSP director Operator，并使用 director Operator 部署 RHOSP overcloud。 只有在您的架构被红帽服务或大客户经理批准时，才会获得对 Red Hat OpenStack Platform director Operator 的支持。在部署此功能前，请联系红帽。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

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

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

第 1 章 使用 DIRECTOR OPERATOR 创建和部署 RHOSP OVERCLOUD

Red Hat OpenShift Container Platform (RHOCP)使用 Operator 的模块化系统来扩展 RHOCP 集群的功能。Red Hat OpenStack Platform (RHOSP) director Operator (OSPdO)添加了在 RHOCP 中安装和运行 RHOSP 云的功能。OSPdO 管理一组自定义资源定义(CRD)，用于部署和管理 RHOSP 节点的基础架构和配置。OSPdO 部署的 RHOSP 云的基本架构包括以下功能：

虚拟化 control plane

Controller 节点是 OSPdO 在 Red Hat OpenShift Virtualization 中创建的虚拟机(VM)。

裸机机器置备

OSPdO 使用 RHOCP 裸机恢复机器管理为 RHOSP 云调配计算节点。

网络

OSPdO 为 RHOSP 服务配置底层网络。

基于 Heat 和 Ansible 的配置

OSPdO 在 RHOCP 中存储自定义 heat 配置，并使用 director 中的 **config-download** 功能将配置转换为 Ansible playbook。如果您更改了存储的 heat 配置，OSPdO 会自动重新生成 Ansible playbook。

CLI 客户端

OSPdO 创建一个 **openstackclient** pod，供用户运行 RHOSP CLI 命令并与其 RHOSP 云交互。

您可以使用特定于 OSPdO 的资源来置备 overcloud 基础架构，生成 overcloud 配置并创建 overcloud。要使用 OSPdO 创建 RHOSP overcloud，您必须完成以下任务：

1. 在操作 RHOCP 集群上安装 OSPdO。
2. 为基础操作系统创建 RHOCP 集群数据卷，并为远程 Git 存储库添加身份验证详细信息。
3. 使用 **OpenStackNetConfig** CRD 创建 overcloud 网络，包括 control plane 和任何隔离的网络。
4. **创建 ConfigMap** 以存储 overcloud 的任何自定义 heat 模板和环境文件。
5. 创建一个 control plane，其中包含 Controller 节点的三个虚拟机，以及一个 pod 来执行客户端操作。
6. 创建裸机 Compute 节点。
7. 创建 **OpenStackConfigGenerator** 自定义资源，以便为 overcloud 配置呈现 Ansible playbook。
8. 使用 **openstackdeploy** 将 Ansible playbook 配置应用到 overcloud 节点。

1.1. DIRECTOR OPERATOR 的自定义资源定义

Red Hat OpenStack Platform (RHOSP) director Operator (OSPdO)包括一组可用于管理 overcloud 资源的自定义资源定义(CRD)。

- 使用以下命令查看 OSPdO CRD 的完整列表：

```
$ oc get crd | grep "^openstack"
```

- 使用以下命令查看特定 CRD 的定义：

```
$ oc describe crd openstackbaremetalset
```

```
Name:         openstackbaremetalsets.osp-director.openstack.org
Namespace:
Labels:       operators.coreos.com/osp-director-operator.openstack=
Annotations:  cert-manager.io/inject-ca-from:
$(CERTIFICATE_NAMESPACE)/$(CERTIFICATE_NAME)
              controller-gen.kubebuilder.io/version: v0.3.0
API Version:  apiextensions.k8s.io/v1
Kind:         CustomResourceDefinition
...
```

- 使用以下命令查看可用于配置特定 CRD 的字段的描述：

```
$ oc explain openstackbaremetalset.spec
KIND:     OpenStackBaremetalSet
VERSION:  osp-director.openstack.org/v1beta1

RESOURCE: spec <Object>

DESCRIPTION:
    <empty>

FIELDS:
  count               <Object>
  baseImageUrl        <Object>
  deploymentSSHSecret  <Object>
  ctlplaneInterface    <Object>
  networks            <[]Object>
  ...
```

OSPdO 包括两种类型的 CRD：硬件配置和软件配置。

硬件置备 CRD

openstacknetattachment （内部）

OSPdO 用来管理 **NodeNetworkConfigurationPolicy** 和 **NodeSriovConfigurationPolicy** CRD，它们用于将网络附加到虚拟机(VM)。

openstacknetconfig

用于指定描述完整网络配置的 **openstacknet attachment** 和 openstacknet CRD。每个节点的保留 IP 和 MAC 地址集合反映在状态中。

openstackbaremetalset

用于为特定 RHOSP 角色创建一组裸机主机，如 "Compute" 和 "Storage"。

openstackcontrolplane

使用 创建 RHOSP control plane 并管理关联的 **openstackvmset** CR。

openstacknet （内部）

使用 创建用于将 IP 分配给 **openstackvmset** 和 **openstackbaremetalset** CR 的网络。

openstackipset （内部）

包含给定网络和角色的一组 IP。OSPdO 用来管理 IP 地址。

openstackprovisionservers

使用 提供自定义镜像，以使用 Metal3 置备裸机节点。

openstackvmset

用于为特定 RHOSP 角色创建一组 OpenShift Virtualization 虚拟机，如 "Controller"、"Database" 或 "NetworkController"。

软件配置 CRD

openstackconfiggenerator

在扩展或更改自定义 **ConfigMap** 时，使用 自动生成用于部署的 Ansible playbook。

openstackconfigversion

使用 代表一组可执行 Ansible playbook。

openstackdeploy

使用 执行 **openstackconfigversion** CR 中定义的 Ansible playbook 集合。

openstackclient

创建用于运行 RHOSP 部署命令的 pod。

其他资源

- 管理自定义资源定义中的资源

1.2. CRD 命名约定

每个自定义资源定义(CRD)都可以通过 **spec.names** 参数定义多个名称。您使用的名称取决于您执行的操作上下文：

- 在创建并与资源清单交互时使用 **kind** ：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackBaremetalSet
...
```

资源清单中的 **kind** 名称与相应 CRD 中的 **kind** 名称关联。

- 与多个资源交互时使用 **复数** ：

```
$ oc get openstackbaremetalsets
```

- 与单个资源进行交互时使用 **singular**

```
$ oc describe openstackbaremetalset/compute
```

- 对于任何 CLI 交互，使用 **shortName** ：

```
$ oc get osbmset
```

1.3. DIRECTOR OPERATOR 不支持的功能

Fiber Channel 后端

使用光纤通道的后端不支持 Block Storage (cinder) 镜像到卷。Red Hat OpenShift Virtualization 不支持 N_Port ID Virtualization (NPIV)。因此，需要将 LUN 从存储后端映射到控制器（其中 **cinder-volume** 默认运行）的块存储驱动程序不起作用。您必须为 **cinder-volume** 创建专用角色，并使用该

角色创建物理节点，而不是将其包含在虚拟化控制器中。如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的可组合服务和自定义角色。

基于角色的 Ansible playbook

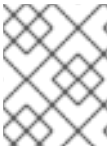
director Operator (OSPdO)不支持在置备裸机节点后运行 Ansible playbook 来配置基于角色的节点属性。这意味着您无法使用 **role_growvols_args** 额外 Ansible 变量为对象存储服务(swift)配置整个磁盘分区。基于角色的 Ansible playbook 配置仅适用于使用节点定义文件置备的裸机节点。

1.4. 其他资源

- Operator
- 在集群中添加 Operator

第 2 章 安装并准备 DIRECTOR OPERATOR

您可以在现有可正常运行的 Red Hat Openshift Container Platform (RHOCP)集群上安装 Red Hat OpenStack Platform (RHOSP) director Operator (OSPdO)。您可以在可访问 RHOCP 集群的工作站上执行 OSPdO 安装任务和所有 overcloud 创建任务。安装 OSPdO 后，您必须为基础操作系统创建数据卷，并为远程 Git 存储库添加身份验证详情。您还可以为节点设置 root 密码。如果没有设置 root 密码，您仍然可以使用 **osp-controlplane-ssh-keys** Secret 中定义的 SSH 密钥登录到节点。



注意

只有在您的架构被红帽服务或大客户经理批准时，才会获得对 Red Hat OpenStack Platform director Operator 的支持。在部署此功能前，请联系红帽。

2.1. 先决条件

- 一个正常运行的 Red Hat Openshift Container Platform (RHOCP)集群，版本 4.12 或更高版本。集群必须包含 **provisioning** 网络以及以下 Operator：
  - 一个 **裸机集群** Operator。必须启用 **baremetal** 集群 Operator。如需有关 **baremetal** 集群 Operator 的更多信息，请参阅 Bare-metal 集群 Operator。
  - OpenShift Virtualization Operator。有关安装 OpenShift Virtualization Operator 的更多信息，请参阅使用 Web 控制台安装 OpenShift Virtualization。
  - SR-IOV Network Operator。
  - Kubernetes NMState Operator。您还必须创建一个 NMState 实例来完成安装所有 NMState CRD：

    ```
    cat <<EOF | oc apply -f -
    apiVersion: nmstate.io/v1
    kind: NMState
    metadata:
      name: nmstate
      namespace: openshift-nmstate
    EOF
    ```

    有关安装 Kubernetes NMState Operator 的更多信息，请参阅安装 Kubernetes NMState Operator。
- **oc** 命令行工具已安装在您的工作站上。
- OSPdO 的远程 Git 存储库，用于存储您的 overcloud 生成的配置。
- 为 Git 存储库生成 SSH 密钥对，公钥将上传到 Git 存储库。
- 以下持久性卷来满足 OSPdO 创建的持久性卷声明：
  - 4G for **openstackclient-cloud-admin**.
  - 1G 用于 **openstackclient-hosts**。
  - 50G 用于为每个 Controller 虚拟机 OSPdO 克隆的基础镜像。
  - 每个 Controller 虚拟机至少具有 50G。如需更多信息，请参阅 Controller 节点要求

2.2. 裸机集群 OPERATOR

使用安装程序置备的基础架构(IPI)或辅助安装(AI)安装的 Red Hat Openshift Container Platform (RHOCP)集群使用 **baremetal** 平台类型并启用 **baremetal** 集群 Operator。使用用户置备的基础架构(UPI)安装的 RHOCP 集群使用 **none** 平台类型，并可能禁用了 **baremetal** 集群 Operator。

如果集群是 AI 或 IPI 类型，它使用 **metal3**，一个 Kubernetes API 用于管理裸机主机。它维护一个可用主机清单，作为 **BareMetalHost** 自定义资源定义(CRD)的实例。您可以使用裸机 Operator 执行以下任务：

- 检查主机的硬件详情，并将其报告给对应的 **BareMetalHost** CR。这包括 CPU、RAM、磁盘和 NIC 的信息。
- 使用特定镜像置备主机。
- 在调配之前或之后清理主机的磁盘内容。

要检查是否启用了 **baremetal** 集群 Operator，请导航到 **Administration > Cluster Settings > ClusterOperators > baremetal**，滚动到 **Conditions** 部分，并查看 **Disabled** 状态。

要检查 RHOCP 集群的平台类型，请导航到 **Administration > Cluster Settings > Configuration > Infrastructure**，切换到 **YAML** 视图，滚动到 **Conditions** 部分，并查看 **status.platformStatus** 值。

2.3. 安装 DIRECTOR OPERATOR

要安装 director Operator (OSPdO)，您必须为 OSPdO 创建 **openstack** 项目(**命名空间**)，并在项目中创建以下自定义资源(CR)：

- **CatalogSource**，用于标识用于 OSPdO 目录的索引镜像。
- 一个 **OperatorGroup**，它定义了 OSPdO 的 Operator 组，并将 OSPdO 限制为目标命名空间。
- 订阅，跟踪 OSPdO 目录中的更改。

流程

1. 创建 OSPdO 项目：

```
$ oc new-project openstack
```

2. 从 https://catalog.redhat.com/software/containers/search 获取最新的 **osp-director-operator-bundle** 镜像。
3. 从 https://console.redhat.com/openshift/downloads 下载 Operator Package Manager (**opm**)工具。
4. 使用 **opm** 工具创建索引镜像：

```
$ BUNDLE_IMG="registry.redhat.io/rhosp-rhel9/osp-director-operator-bundle:1.3.1"
$ INDEX_IMG="quay.io/<account>/osp-director-operator-index:x.y.z-a"
$ opm index add --bundles ${BUNDLE_IMG} --tag ${INDEX_IMG} -u podman --pull-tool
podman
```

5. 将索引镜像推送到 registry：

```
$ podman push ${INDEX_IMG}
```

6. 创建一个环境文件，以配置安装 OSPdO 所需的 **CatalogSource**、**OperatorGroup** 和 **Subscription** CR，如 **osp-director-operator.yaml**。

7. 要配置 **CatalogSource** CR，请在 **osp-director-operator.yaml** 中添加以下配置：

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: osp-director-operator-index
  namespace: openstack
spec:
  sourceType: grpc
  image: quay.io/<account>/osp-director-operator-index:x.y.z-a
```

有关如何应用 Quay 身份验证以便 Operator 部署可以拉取镜像的信息，请参阅 从私有 registry 访问 Operator 的镜像。

8. 要配置 **OperatorGroup** CR，请在 **osp-director-operator.yaml** 中添加以下配置：

```
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: "osp-director-operator-group"
  namespace: openstack
spec:
  targetNamespaces:
  - openstack
```

9. 要配置 **Subscription** CR，请在 **osp-director-operator.yaml** 中添加以下配置：

```
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: osp-director-operator-subscription
  namespace: openstack
spec:
  config:
    env:
    - name: WATCH_NAMESPACE
      value: openstack,openshift-machine-api,openshift-sriov-network-operator
  source: osp-director-operator-index
  sourceNamespace: openstack
  name: osp-director-operator
```

10. 在 **openstack** 命名空间中创建新的 **CatalogSource**、**OperatorGroup** 和 **Subscription** CR：

```
$ oc apply -f osp-director-operator.yaml
```

11. 通过列出已安装的 Operator，确认已安装 OSPdO、**osp-director-operator.openstack** ：

```
$ oc get operators
NAME                                AGE
osp-director-operator.openstack     5m
```

后续步骤

- 为基本操作系统创建数据卷

2.4. 为基本操作系统创建数据卷

您必须使用 Red Hat OpenShift Container Platform (RHOCP)集群创建数据卷，以存储 Controller 虚拟机(VM)的基本操作系统镜像。在创建 **OpenStackControlPlane** 和 **OpenStackVmSet** 自定义资源时，您可以使用 **baseImageVolumeName** 参数指定此数据卷。

先决条件

- **virtctl** 客户端工具安装在您的工作站上。要在 Red Hat Enterprise Linux (RHEL)工作站上安装这个工具，请使用以下命令：

```
$ sudo subscription-manager repos --enable=cnv-4.12-for-rhel-8-x86_64-rpms
$ sudo dnf install -y kubevirt-virtctl
```

- **virt-customize** 客户端工具已安装在您的工作站上。要在 RHEL 工作站上安装此工具，请使用以下命令：

```
$ dnf install -y libguestfs-tools-c
```

流程

1. 将红帽客户门户网站 的产品下载 部分中的 RHEL 9.2 QCOW2 镜像下载到您的工作站。
2. 可选：添加自定义 CA 证书：

```
$ sudo -s
$ export LIBGUESTFS_BACKEND=direct
$ virt-copy-in -a <local_path_to_image> <ca_certificate>.pem /etc/pki/ca-
trust/source/anchors/
```

   您可能想要添加自定义 CA 证书来保护身份服务的 LDAP 通信，或者与任何非 RHOSP 系统通信。

3. 创建一个脚本来自定义镜像来分配可预测的网络接口名称：

```
#!/bin/bash
set -eux

if [ -e /etc/kernel/cmdline ]; then
  echo 'Updating /etc/kernel/cmdline'
  sed -i -e "s/^\(.*\)net\.ifnames=0\s*\(.*\)/\1\2/" /etc/kernel/cmdline
fi

source /etc/default/grub
if grep -q "net.ifnames=0" <<< "$GRUB_CMDLINE_LINUX"; then
  echo 'Updating /etc/default/grub'
  sed -i -e "s/^\(GRUB_CMDLINE_LINUX=.*\)net\.ifnames=0\s*\(.*\)/\1\2/" /etc/default/grub
fi
```

```
if [ "$GRUB_ENABLE_BLSCFG" == "true" ]; then
  echo 'Fixing BLS entries'
  find /boot/loader/entries -type f -exec sed -i -e "s/^\(.*\)net\.ifnames=0\s*\(.*\)/\1\2/" {} \;
fi
# Always do this, on RHEL8 with BLS we still need it as the BLS entry uses $kernelopts from grubenv
echo 'Running grub2-mkconfig'
grub2-mkconfig -o /etc/grub2.cfg
grub2-mkconfig -o /etc/grub2-efi.cfg
rm -f /etc/sysconfig/network-scripts/ifcfg-ens* /etc/sysconfig/network-scripts/ifcfg-eth*
update-ca-trust extract
```

4. 运行镜像自定义脚本：

```
$ sudo -s
$ export LIBGUESTFS_BACKEND=direct
$ chmod 755 customize_image.sh
$ virt-customize -a <local_path_to_image> --run customize_image.sh --truncate /etc/machine-id
```

5. 使用 **virtctl** 将镜像上传到 OpenShift Virtualization：

```
$ virtctl image-upload dv <datavolume_name> -n openstack \
 --size=<size> --image-path=<local_path_to_image> \
 --storage-class <storage_class> --access-mode <access_mode> --insecure
```

- 将 **<datavolume_name>** 替换为数据卷的名称，如 **openstack-base-img**。
- 将 **<size>** 替换为您的环境所需的数据卷的大小，例如 **50Gi**。最小值为 50GB。
- 将 **<storage_class>** 替换为集群中所需的存储类。使用以下命令检索可用的存储类：

  ```
  $ oc get storageclass
  ```

- 将 **<access_mode>** 替换为 PVC 的访问模式。默认值为 **ReadWriteOnce**。

其他资源

- 使用 virtctl 工具上传本地磁盘镜像

后续步骤

- 为您的远程 Git 存储库添加身份验证详情

2.5. 为您的远程 GIT 存储库添加身份验证详情

director Operator (OSPdO)将呈现的 Ansible playbook 存储在远程 Git 存储库中，并使用此存储库跟踪 overcloud 配置更改。您可以使用支持 SSH 身份验证的任何 Git 存储库。您必须将 Git 存储库的详细信息作为名为 **git-secret** 的 Red Hat OpenShift Platform (RHOCP) Secret 资源提供。

先决条件

- 您的 OSPdO Git 存储库的 SSH 密钥对的私钥。

流程

1. 创建 **git-secret** Secret 资源：

```
$ oc create secret generic <secret_name> -n <namespace> \
 --from-file=git_ssh_identity=<path_to_private_SSH_key> \
 --from-literal=git_url=<git_server_URL>
```

   - 将 **<secret_name** > 替换为 secret 的名称，本例中为 **git-secret**。
   - 将 **<namespace** > 替换为在其中创建 secret 的命名空间的名称，例如 **openstack**。
   - 将 **<path_to_private_SSH_key** > 替换为访问 Git 存储库的私钥的路径。
   - 使用存储 OSPdO 配置的 git 存储库的 SSH URL 替换 **<git_server_URL>**，例如 **ssh://<user>@<server>:2202/repo.git**。

2. 验证 Secret 资源是否已创建：

```
$ oc get secret/git-secret -n openstack
```

其他资源

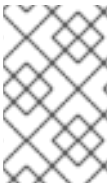
- 为 pod 提供敏感数据

后续步骤

- 使用 director Operator 创建网络

2.6. 为节点设置 ROOT 密码

要使用每个节点上的密码访问 **root** 用户，您可以在名为 **userpassword** 的 **Secret** 资源中设置一个 **root** 密码。为节点设置 root 密码是可选的。如果没有设置 **root** 密码，您仍然可以使用 **osp-controlplane-ssh-keys** Secret 中定义的 SSH 密钥登录到节点。



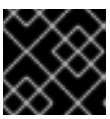
注意

如果设置 root 密码，您必须在创建 **OpenStackControlPlane** 和 **OpenStackBaremetalSet** 自定义资源时使用 **password Secret** 参数指定此 Secret 资源的名称。本指南中的示例使用 **Secret** 资源名称 **userpassword**。

流程

1. 将您选择的密码转换为 base64 值：

```
$ echo -n "p@ssw0rd!" | base64
cEBzc3cwcmQh
```



重要

n 选项会从 echo 输出中删除结尾的换行符。

2. 在您的工作站上创建一个名为 **openstack-userpassword.yaml** 的文件。在文件中包括 Secret 的以下资源规格：

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openstack
data:
  NodeRootPassword: "<password>"
```

   - 将 **<secret_name** > 替换为此 Secret 资源的名称，如 **userpassword**。
   - 将 **<password** > 替换为您的 base64 编码密码。

3. 创建 **userpassword** Secret：

```
$ oc create -f openstack-userpassword.yaml -n openstack
```

其他资源

- 为 pod 提供敏感数据

后续步骤

- 使用 director Operator 创建网络

第 3 章 使用 DIRECTOR OPERATOR 创建网络

要在 OpenShift Virtualization worker 节点上创建网络和桥接，并将虚拟机(VM)连接到这些网络，您可以定义 **OpenStackNetConfig** 自定义资源(CR)并指定 overcloud 网络的所有子网。您必须为 overcloud 创建一个 control plane 网络。您还可以选择创建额外网络来为可组合网络实施网络隔离。

3.1. 使用 OPENSTACKNETCONFIG CRD 创建 OVERCLOUD 网络

您必须使用 **OpenStackNetConfig** CRD 为 overcloud 定义至少一个 control plane 网络。您还可以选择定义 VLAN 网络，以便为可组合网络创建网络隔离，如 **InternalAPI**、**Storage** 和 **External**。每个网络定义都必须包含 IP 地址分配，以及 **OpenStackNetAttachment** CRD 的映射信息。OpenShift Virtualization 使用网络定义将任何虚拟机(VM)附加到 control plane 和 VLAN 网络。

提示

使用以下命令查看 **OpenStackNetConfig** CRD 定义和规格模式：

```
$ oc describe crd openstacknetconfig

$ oc explain openstacknetconfig.spec
```

流程

1. 在工作站上创建一个名为 **openstacknetconfig.yaml** 的文件。
2. 将以下配置添加到 **openstacknetconfig.yaml** 以创建 **OpenStackNetConfig** 自定义资源(CR)：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
```

3. 为您的网络所需的网桥配置网络附加定义。例如，将以下配置添加到 **openstacknetconfig.yaml** 中，以创建 RHOSP 网桥网络附加定义 **br-osp**，并将 **nodeNetworkConfigurationPolicy** 选项设置为创建 Linux 网桥：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
spec:
  attachConfigurations:
    br-osp: 1
      nodeNetworkConfigurationPolicy:
        nodeSelector:
          node-role.kubernetes.io/worker: ""
        desiredState:
          interfaces:
          - bridge:
              options:
                stp:
                  enabled: false
              port:
```

```
        - name: enp6s0 2
        description: Linux bridge with enp6s0 as a port
        name: br-osp 3
        state: up
        type: linux-bridge
        mtu: 1500 4
# optional DnsServers list
dnsServers:
- 192.168.25.1
# optional DnsSearchDomains list
dnsSearchDomains:
- osptest.test.metalkube.org
- some.other.domain
# DomainName of the OSP environment
domainName: osptest.test.metalkube.org
```

1. **br-osp** 的网络附加定义。
2. 在每个主机上要附加到的 NIC / 以太网设备。
3. 接口名称。
4. 单个网络数据包中传输的最大数据量。

4. 可选：要将 Jumbo Frames 用于网桥，请将网桥接口配置为使用 Jumbo Frames 并更新网桥的 **mtu** 值：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
spec:
  attachConfigurations:
    br-osp:
      nodeNetworkConfigurationPolicy:
        nodeSelector:
          node-role.kubernetes.io/worker: ""
        desiredState:
          interfaces:
          - bridge:
              options:
                stp:
                  enabled: false
              port:
              - name: enp6s0
            description: Linux bridge with enp6s0 as a port
            name: br-osp
            state: up
            type: linux-bridge
            mtu: 9000 1
          - name: enp6s0 2
            description: Configuring enp6s0 on workers
            type: ethernet
```

```
        state: up
        mtu: 9000
...
```

1. 使用 Jumbo Frames 更新单个网络数据包中传输的最大数据量。
2. 将网桥接口配置为使用 Jumbo Frames。

5. 定义每个 overcloud 网络。以下示例为 **InternalAPI** 流量创建一个 control plane 网络和一个隔离的网络：

```
spec:
  ...
  networks:
  - name: Control 1
    nameLower: ctlplane 2
    subnets: 3
    - name: ctlplane 4
      ipv4: 5
        allocationEnd: 172.22.0.250
        allocationStart: 172.22.0.100
        cidr: 172.22.0.0/24
        gateway: 172.22.0.1
      attachConfiguration: br-osp 6
  - name: InternalApi 7
    nameLower: internal_api
    mtu: 1350
    subnets:
    - name: internal_api
      attachConfiguration: br-osp
      vlan: 20 8
      ipv4:
        allocationEnd: 172.17.0.250
        allocationStart: 172.17.0.10
        cidr: 172.17.0.0/24
...
```

1. 网络的名称，例如 **Control**。
2. 网络名称的小写版本，例如 **ctlplane**。
3. 子网规格。
4. 子网的名称，如 **ctlplane**。
5. 带有 **allocationStart**,**allocationEnd**,**cidr**,**gateway**, 和带有 **destination** 和 **nexthop** 的路由列表的 IPv4 子网详情。
6. 将网络连接到的网络附加定义。在本例中，RHOSP 网桥 **br-osp** 连接到每个 worker 上的 NIC。
7. 可组合网络的网络定义。要使用默认的 RHOSP 网络，您必须为每个网络创建一个 **OpenStackNetConfig** 资源。有关默认 RHOSP 网络的信息，请参阅 默认 Red Hat OpenStack Platform 网络。要使用不同的网络，您必须创建一个自定义

network_data.yaml 文件。有关创建自定义 **network_data.yaml** 文件的详情，请参考 配置 overcloud 网络。

8 网络 VLAN。有关默认 RHOSP 网络的信息，请参阅 默认 Red Hat OpenStack Platform 网络。有关使用 **OpenStackNetConfig** CRD 的虚拟机桥接的更多信息，请参阅使用 OpenStackNetConfig CRD 了解虚拟机桥接。

6. 可选：为特定节点上的网络保留静态 IP 地址：

```
spec:
  ...
  reservations:
    controller-0:
      ipReservations:
        ctlplane: 172.22.0.120
    compute-0:
      ipReservations:
        ctlplane: 172.22.0.140
        internal_api: 172.17.0.40
        storage: 172.18.0.40
        tenant: 172.20.0.40
```

注意

保留优先于任何自动生成的 IP 地址。

7. 保存 **openstacknetconfig.yaml** 定义文件。
8. 创建 overcloud 网络：

```
$ oc create -f osnetconfig.yaml -n openstack
```

9. 要验证 overcloud 网络是否已创建，请查看 overcloud 网络的资源：

```
$ oc get openstacknetconfig/openstacknetconfig
```

10. 查看 **OpenStackNetConfig** API 和子资源：

```
$ oc get openstacknetconfig/openstacknetconfig -n openstack
$ oc get openstacknetattachment -n openstack
$ oc get openstacknet -n openstack
```

如果您看到错误，请检查底层 **network-attach-definition** 和节点网络配置策略：

```
$ oc get network-attachment-definitions -n openstack
$ oc get nncp
```

后续步骤

- 使用 director Operator 自定义 overcloud

3.1.1. 默认 Red Hat OpenStack Platform 网络

Network	VLAN	CIDR	分配
外部	10	10.0.0.0/24	10.0.0.10 - 10.0.0.250
InternalApi	20	172.17.0.0/24	172.17.0.10 - 172.17.0.250
存储	30	172.18.0.0/24	172.18.0.10 - 172.18.0.250
StorageMgmt	40	172.19.0.0/24	172.19.0.10 - 172.19.250
租户	50	172.20.0.0/24	172.20.0.10 - 172.20.0.250

3.2. 了解使用 OPENSTACKNETCONFIG CRD 的虚拟机桥接

使用 **OpenStackVMSet** CRD 创建虚拟机(VM)时，您必须将这些虚拟机连接到相关的 Red Hat OpenStack Platform (RHOSP)网络。您可以使用 **OpenStackNetConfig** CRD 在 Red Hat OpenShift Container Platform (RHOCP) worker 节点上创建所需的网桥，并将 Controller 虚拟机连接到 RHOSP overcloud 网络。RHOSP 需要专用的 NIC 才能部署。

OpenStackNetConfig CRD 包含一个 **attachConfigurations** 选项，它是 **nodeNetworkConfigurationPolicy** 的哈希值。**OpenStackNetConfig** 自定义资源(CR)中的每个指定的 **attachConfiguration** 都会创建一个 **NetworkAttachmentDefinition** 对象，它将网络接口数据传递到 RHOCP 集群中的 **NodeNetworkConfigurationPolicy** 资源。**NodeNetworkConfigurationPolicy** 资源使用 **nmstate** API 配置每个 RHOCP worker 节点上网络配置的最终状态。每个网络的 **NetworkAttachmentDefinition** 对象定义 Multus CNI 插件配置。当您为 **NetworkAttachmentDefinition** 对象指定 VLAN ID 时，Multus CNI 插件在网桥上启用 **vlan-filtering**。**OpenStackNetConfig** CR 中配置的每个网络都引用其中一个 **attachConfigurations**。在虚拟机中，每个网络都有一个接口。

以下示例创建了一个 **br-osp attachConfiguration**，并将 **nodeNetworkConfigurationPolicy** 选项配置为创建一个 Linux 网桥，并将网桥连接到每个 worker 上的 NIC。应用此配置时，**NodeNetworkConfigurationPolicy** 对象将每个 RHOCP worker 节点配置为与所需的最终状态匹配：每个 worker 包含一个名为 **br-osp** 的新网桥，该网桥连接到每个主机上的 **enp6s0** NIC。所有 RHOSP Controller 虚拟机都可以连接到 control plane 网络流量的 **br-osp** 网桥。

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
spec:
  attachConfigurations:
    br-osp:
      nodeNetworkConfigurationPolicy:
        nodeSelector:
          node-role.kubernetes.io/worker: ""
        desiredState:
          interfaces:
          - bridge:
              options:
                stp:
                  enabled: false
```

```
            port:
            - name: enp6s0
          description: Linux bridge with enp6s0 as a port
          name: br-osp
          state: up
          type: linux-bridge
          mtu: 1500
…
  networks:
  - name: Control
    nameLower: ctlplane
    subnets:
    - name: ctlplane
      ipv4:
        allocationEnd: 192.168.25.250
        allocationStart: 192.168.25.100
        cidr: 192.168.25.0/24
        gateway: 192.168.25.1
      attachConfiguration: br-osp
```

如果通过 VLAN 20 指定内部 API 网络，您可以设置 **attachConfiguration** 选项来修改每个 RHOCP worker 节点上的网络配置，并将 VLAN 连接到现有的 **br-osp** 网桥：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
spec:
  attachConfigurations:
    br-osp:
  ...
  networks:
  ...
  - isControlPlane: false
    mtu: 1500
    name: InternalApi
    nameLower: internal_api
    subnets:
    - attachConfiguration: br-osp
      ipv4:
        allocationEnd: 172.17.0.250
        allocationStart: 172.17.0.10
        cidr: 172.17.0.0/24
        gateway: 172.17.0.1
        routes:
        - destination: 172.17.1.0/24
          nexthop: 172.17.0.1
        - destination: 172.17.2.0/24
          nexthop: 172.17.0.1
      name: internal_api
      vlan: 20
```

br-osp 已存在，并连接到每个主机上的 **enp6s0** NIC，因此不会更改网桥本身。但是，**内部API OpenStackNet** 将 VLAN 20 与这个网络关联，这意味着 RHOSP Controller 虚拟机可以连接到内部 API 网络流量的 **br-osp** 网桥上的 VLAN 20。

使用 **OpenStackVMSet** CRD 创建虚拟机时，虚拟机会使用连接到每个网络的多个 Virtio 设备。OpenShift Virtualization 按照字母顺序对网络名称进行排序，但默认网络除外，这是第一个接口。**例如，如果您使用 OpenStackNetConfig 创建默认 RHOSP 网络，则会为 Controller 虚拟机生成以下接口配置：**

```
interfaces:
  - masquerade: {}
    model: virtio
    name: default
  - bridge: {}
    model: virtio
    name: ctlplane
  - bridge: {}
    model: virtio
    name: external
  - bridge: {}
    model: virtio
    name: internalapi
  - bridge: {}
    model: virtio
    name: storage
  - bridge: {}
    model: virtio
    name: storagemgmt
  - bridge: {}
    model: virtio
    name: tenant
```

此配置会为 Controller 节点生成以下网络到接口映射：

表 3.1. 默认网络到接口映射

Network	Interface
default	nic1
ctlplane	nic2
external	nic3
internalapi	nic4
storage	nic5
storagemgmt	nic6
tenant	nic7

注意

OpenStackVMSet 使用的角色 NIC 模板是自动生成的。您可以覆盖自定义 NIC 模板文件中的默认配置，< role>-nic-template.j2，如 controller-nic-template.j2。您必须将自定义 NIC 文件添加到包含 overcloud 配置的 tarball 文件中，该文件是使用 OpenShift ConfigMap 对象实现的。如需更多信息，请参阅第 4 章。使用 director Operator 自定义 overcloud.

其他资源

- 更新节点网络配置

3.3. OPENSTACKNETCONFIG 自定义资源文件示例

以下示例 OpenStackNetConfig 自定义资源(CR)文件定义了一个 overcloud 网络，其中包含用于默认 RHOSP 部署的 control plane 网络和隔离的 VLAN 网络。这个示例还为特定节点上的网络保留静态 IP 地址。

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
spec:
  attachConfigurations:
    br-osp:
      nodeNetworkConfigurationPolicy:
        nodeSelector:
          node-role.kubernetes.io/worker: ""
        desiredState:
          interfaces:
          - bridge:
              options:
                stp:
                  enabled: false
              port:
              - name: enp7s0
            description: Linux bridge with enp7s0 as a port
            name: br-osp
            state: up
            type: linux-bridge
            mtu: 9000
          - name: enp7s0
            description: Configuring enp7s0 on workers
            type: ethernet
            state: up
            mtu: 9000
    br-ex:
      nodeNetworkConfigurationPolicy:
        nodeSelector:
```

```
        node-role.kubernetes.io/worker: ""
      desiredState:
        interfaces:
        - bridge:
            options:
              stp:
                enabled: false
            port:
            - name: enp6s0
          description: Linux bridge with enp6s0 as a port
          name: br-ex
          state: up
          type: linux-bridge
          mtu: 1500
  # optional DnsServers list
  dnsServers:
  - 172.22.0.1
  # optional DnsSearchDomains list
  dnsSearchDomains:
  - osptest.test.metalkube.org
  - some.other.domain
  # DomainName of the OSP environment
  domainName: osptest.test.metalkube.org
networks:
- name: Control
  nameLower: ctlplane
  subnets:
  - name: ctlplane
    ipv4:
      allocationEnd: 172.22.0.250
      allocationStart: 172.22.0.10
      cidr: 172.22.0.0/24
      gateway: 172.22.0.1
    attachConfiguration: br-osp
- name: InternalApi
  nameLower: internal_api
  mtu: 1350
  subnets:
  - name: internal_api
    attachConfiguration: br-osp
    vlan: 20
    ipv4:
      allocationEnd: 172.17.0.250
      allocationStart: 172.17.0.10
      cidr: 172.17.0.0/24
      gateway: 172.17.0.1
      routes:
      - destination: 172.17.1.0/24
        nexthop: 172.17.0.1
      - destination: 172.17.2.0/24
        nexthop: 172.17.0.1
- name: External
  nameLower: external
  subnets:
  - name: external
    ipv4:
```

```
      allocationEnd: 10.0.0.250
      allocationStart: 10.0.0.10
      cidr: 10.0.0.0/24
      gateway: 10.0.0.1
    attachConfiguration: br-ex
- name: Storage
  nameLower: storage
  mtu: 1500
  subnets:
  - name: storage
    ipv4:
      allocationEnd: 172.18.0.250
      allocationStart: 172.18.0.10
      cidr: 172.18.0.0/24
    vlan: 30
    attachConfiguration: br-osp
- name: StorageMgmt
  nameLower: storage_mgmt
  mtu: 1500
  subnets:
  - name: storage_mgmt
    ipv4:
      allocationEnd: 172.19.0.250
      allocationStart: 172.19.0.10
      cidr: 172.19.0.0/24
    vlan: 40
    attachConfiguration: br-osp
- name: Tenant
  nameLower: tenant
  vip: False
  mtu: 1500
  subnets:
  - name: tenant
    ipv4:
      allocationEnd: 172.20.0.250
      allocationStart: 172.20.0.10
      cidr: 172.20.0.0/24
    vlan: 50
    attachConfiguration: br-osp
reservations:
  compute-0:
    ipReservations:
      ctlplane: 172.22.0.140
      internal_api: 172.17.0.40
      storage: 172.18.0.40
      tenant: 172.20.0.40
    macReservations: {}
  controller-0:
    ipReservations:
      ctlplane: 172.22.0.120
      external: 10.0.0.20
      internal_api: 172.17.0.20
      storage: 172.18.0.20
      storage_mgmt: 172.19.0.20
      tenant: 172.20.0.20
    macReservations: {}
```

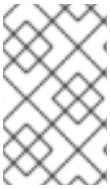
```
controller-1:
  ipReservations:
    ctlplane: 172.22.0.130
    external: 10.0.0.30
    internal_api: 172.17.0.30
    storage: 172.18.0.30
    storage_mgmt: 172.19.0.30
    tenant: 172.20.0.30
  macReservations: {}
controlplane:
  ipReservations:
    ctlplane: 172.22.0.110
    external: 10.0.0.10
    internal_api: 172.17.0.10
    storage: 172.18.0.10
    storage_mgmt: 172.19.0.10
  macReservations: {}
openstackclient-0:
  ipReservations:
    ctlplane: 172.22.0.251
    external: 10.0.0.251
    internal_api: 172.17.0.251
  macReservations: {}
```

第 4 章 使用 DIRECTOR OPERATOR 自定义 OVERCLOUD

您可以通过创建 overcloud 部署中包含的 heat 模板和环境文件来自定义 overcloud 或启用某些功能。使用 director Operator (OSPdO) overcloud 部署，您可以在运行 overcloud 部署前将这些文件存储在 ConfigMap 对象中。

4.1. 将自定义模板添加到 OVERCLOUD 配置

director Operator (OSPdO)将核心 overcloud heat 模板转换为 Ansible playbook，当您准备好在每个节点上配置 Red Hat OpenStack Platform (RHOSP)软件时，应用到置备的节点。要将自己的自定义 heat 模板和自定义角色文件添加到 overcloud 部署中，您必须将模板文件归档到 tarball 文件中，并将 tarball 文件的二进制内容包含在名为 tripleo-tarball-config 的 OpenShift ConfigMap 对象中。此 tarball 文件可以包含复杂的目录结构，以扩展一组核心模板。OSPdO 将文件和目录从 tarball 文件提取到与核心 heat 模板集相同的目录中。如果您的任何自定义模板的名称与核心集合中的模板的名称相同，则自定义模板将覆盖 core 模板。



注意

环境文件中的所有引用都必须相对于提取 tarball 的 TripleO heat 模板。

先决条件

- 要应用到置备的节点的自定义 overcloud 模板。

流程

1. 进入自定义模板的位置：

```
$ cd ~/custom_templates
```

2. 将模板归档到 gzipped tarball 中：

```
$ tar -cvzf custom-config.tar.gz *.yaml
```

3. 创建 tripleo-tarball-config ConfigMap CR，并使用 tarball 作为数据：

```
$ oc create configmap tripleo-tarball-config --from-file=custom-config.tar.gz -n openstack
```

4. 验证 **ConfigMap CR** 是否已创建：

```
$ oc get configmap/tripleo-tarball-config -n openstack
```

其他资源

- 创建和使用配置映射
- 了解 heat 模板

后续步骤

- 在 overcloud 配置中添加自定义环境文件

4.2. 在 OVERCLOUD 配置中添加自定义环境文件

要在 overcloud 中启用功能或设置参数，您必须包含 overcloud 部署的环境文件。director Operator (OSPdO)使用名为 heat-env-config 的 ConfigMap 对象存储和检索环境文件。ConfigMap 对象以以下格式存储环境文件：

```
...
data:
  <environment_file_name>: |+
    <environment_file_contents>
```

例如，以下 ConfigMap 包含两个环境文件：

```
...
data:
  network_environment.yaml: |+
    parameter_defaults:
      ComputeNetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
  cloud_name.yaml: |+
    parameter_defaults:
      CloudDomain: ocp4.example.com
      CloudName: overcloud.ocp4.example.com
      CloudNameInternal: overcloud.internalapi.ocp4.example.com
      CloudNameStorage: overcloud.storage.ocp4.example.com
      CloudNameStorageManagement: overcloud.storagemgmt.ocp4.example.com
      CloudNameCtlplane: overcloud.ctlplane.ocp4.example.com
```

将一组自定义环境文件从目录上传到 ConfigMap 对象，您可以将其作为 overcloud 部署的一部分包含。

先决条件

- 用于 overcloud 部署的自定义环境文件。

流程

1. 创建 heat-env-config ConfigMap 对象：

```
$ oc create configmap -n openstack heat-env-config \
 --from-file=~/<dir_custom_environment_files>/ \
 --dry-run=client -o yaml | oc apply -f -
```

   - 将 < dir_custom_environment_files > 替换为包含要在 overcloud 部署中使用的环境文件的目录。ConfigMap 对象存储为各个 数据 条目。

2. 验证 heat-env-config ConfigMap 对象是否包含所有必需的环境文件：

```
$ oc get configmap/heat-env-config -n openstack
```

4.3. 其他资源

- 了解 heat 模板
- 环境文件
- 创建和使用配置映射

第 5 章 使用 DIRECTOR OPERATOR 创建 OVERCLOUD 节点

Red Hat OpenStack Platform (RHOSP) overcloud 由多个节点组成，如 Controller 节点，以提供 control plane 服务和 Compute 节点来提供计算资源。要使 overcloud 具有高可用性功能，您必须有 3 个 Controller 节点和一个 Compute 节点。您可以使用 OpenStackControlPlane 自定义资源定义(CRD) 和 Compute 节点与 OpenStackBaremetalSet CRD 创建 Controller 节点。

注意

Red Hat OpenShift Container Platform (RHOCP)不会在 RHOCP worker 节点上自动发现问题，如果 worker 节点失败或出现问题，则执行托管 RHOSP Controller 虚拟机的 worker 节点自动恢复。您必须在 RHOCP 集群上启用健康检查，以便在主机 worker 节点失败时自动重新定位 Controller VM pod。有关如何在 RHOCP worker 节点上自动发现问题的详情，请参考 部署机器健康检查。

5.1. 使用 OPENSTACKCONTROLPLANE CRD 创建 CONTROL PLANE

Red Hat OpenStack Platform (RHOSP) control plane 包含管理 overcloud 的 RHOSP 服务。默认 control plane 由 3 个 Controller 节点组成。您可以使用可组合角色来管理专用控制器虚拟机(VM)上的服务。有关可组合角色的更多信息，请参阅 可组合服务和自定义角色。

定义 OpenStackControlPlane 自定义资源(CR)，将 Controller 节点创建为 OpenShift Virtualization 虚拟机(VM)。

提示

使用以下命令查看 OpenStackControlPlane CRD 定义和规格模式：

```
$ oc describe crd openstackcontrolplane

$ oc explain openstackcontrolplane.spec
```

先决条件

- 您已使用 OpenStackNetConfig CR 创建 control plane 网络以及任何其他隔离网络。

流程

1. 在工作站上创建一个名为 openstack-controller.yaml 的文件。包含 Controller 节点的资源规格。以下示例定义了由 3 个 Controller 节点组成的 control plane 规格：

```
apiVersion: osp-director.openstack.org/v1beta2
kind: OpenStackControlPlane
metadata:
  name: overcloud 1
  namespace: openstack 2
spec: 3
  openStackClientNetworks:
      - ctlplane
      - internal_api
      - external
  openStackClientStorageClass: host-nfs-storageclass
  passwordSecret: userpassword 4
  virtualMachineRoles:
    Controller:
      roleName: Controller
      roleCount: 3
      networks:
        - ctlplane
        - internal_api
        - external
        - tenant
        - storage
        - storage_mgmt
      cores: 12
      memory: 64
      rootDisk:
        diskSize: 100
        baseImageVolumeName: openstack-base-img 5
        storageClass: host-nfs-storageclass 6
        storageAccessMode:  ReadWriteMany
        storageVolumeMode: Filesystem
      # optional configure additional discs to be attached to the VMs,
      # need to be configured manually inside the VMs where to be used.
      additionalDisks:
        - name: datadisk
          diskSize: 500
          storageClass: host-nfs-storageclass
          storageAccessMode:  ReadWriteMany
          storageVolumeMode: Filesystem
  openStackRelease: "17.1"
```

1

overcloud control plane 的名称，如 overcloud。

2

OSPdO 命名空间，例如 openstack。

3

control plane 的配置。

4 可选：为具有密码的用户提供每个节点上的 root 访问权限的 Secret 资源。

5 存储控制器虚拟机的基础操作系统镜像的数据卷名称。有关 创建数据卷的更多信息，请参阅为基础操作系统创建数据卷。

6 有关配置 Red Hat OpenShift Container Platform (RHOCP)存储的详情，请参考 动态置备。

2. 保存 openstack-controller.yaml 文件。

3. 创建 control plane：

```
$ oc create -f openstack-controller.yaml -n openstack
```

4. 等待 RHOCP 创建与 OpenStackControlPlane CR 相关的资源。OSPdO 还会创建一个 OpenStackClient pod，您可以通过远程 shell 访问以运行 RHOSP 命令。

验证

1. 查看 control plane 的资源：

```
$ oc get openstackcontrolplane/overcloud -n openstack
```

2. 查看 OpenStackVMSet 资源，以验证 control plane 虚拟机集的创建：

```
$ oc get openstackvmsets -n openstack
```

3. 查看虚拟机以验证 control plane OpenShift Virtualization 虚拟机的创建：

```
$ oc get virtualmachines -n openstack
```

4. 测试对 **openstackclient** 远程 **shell** 的访问：

```
$ oc rsh -n openstack openstackclient
```

5.2. 使用 OPENSTACKBAREMETALSET CRD 创建 COMPUTE 节点

Compute 节点为您的 **Red Hat OpenStack Platform (RHOSP)**环境提供计算资源。您的 **overcloud** 中必须至少有一个 **Compute** 节点，您可以在部署后扩展 **Compute** 节点的数量。

定义一个 **OpenStackBaremetalSet** 自定义资源(CR)，从 **Red Hat OpenShift Container Platform (RHOCP)**管理的裸机创建 **Compute** 节点。

提示

使用以下命令查看 **OpenStackBareMetalSet CRD** 定义和规格模式：

```
$ oc describe crd openstackbaremetalset

$ oc explain openstackbaremetalset.spec
```

先决条件

- 您已使用 **OpenStackNetConfig CR** 创建 **control plane** 网络以及任何其他隔离网络。
- 您已创建了带有 **OpenStackControlPlane CRD** 的 **control plane**。

流程

1. 在工作站上创建一个名为 **openstack-compute.yaml** 的文件。包含 **Compute** 节点的资源规格。以下示例定义了 **1** 个 **Compute** 节点的规格：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackBaremetalSet
metadata:
  name: compute 1
  namespace: openstack 2
```

```
spec: 3
  count: 1
  baseImageUrl: http://<source_host>/rhel-9.2-x86_64-kvm.qcow2
  deploymentSSHSecret: osp-controlplane-ssh-keys
  # If you manually created an OpenStackProvisionServer, you can use it here,
  # otherwise director Operator will create one for you (with `baseImageUrl` as the image that
it server)
  # to use with this OpenStackBaremetalSet
  # provisionServerName: openstack-provision-server
  ctlplaneInterface: enp2s0
  networks:
    - ctlplane
    - internal_api
    - tenant
    - storage
  roleName: Compute
  passwordSecret: userpassword 4
```

1

Compute 节点裸机集的名称，如 compute。

2

OSPdO 命名空间，例如 openstack。

3

Compute 节点的配置。

4

可选：为具有密码的用户提供每个节点上的 root 访问权限的 Secret 资源。

2. **保存 openstack-compute.yaml 文件。**

3. **创建 Compute 节点：**

```
$ oc create -f openstack-compute.yaml -n openstack
```

验证

1. **查看 Compute 节点的资源：**

```
$ oc get openstackbaremetalset/compute -n openstack
```

2. 查看 **RHOCP** 管理的裸机机器，以验证 **Compute** 节点的创建：

```
$ oc get baremetalhosts -n openshift-machine-api
```

5.3. 使用 OPENSTACKPROVISIONSERVER CRD 创建置备服务器

置备服务器提供特定的 **Red Hat Enterprise Linux (RHEL) QCOW2** 镜像，用于为 **Red Hat OpenStack Platform (RHOSP)**置备 **Compute** 节点。为您创建的任何 **OpenStackBaremetalSet CR** 自动创建 **OpenStackProvisionServer CR**。您可以手动创建 **OpenStackProvisionServer CR**，并为您创建的任何 **OpenStackBaremetalSet CR** 提供名称。

OpenStackProvisionServer CRD 为特定的 **RHEL QCOW2** 镜像在 **Red Hat OpenShift Container Platform (RHOCP)**置备网络上创建一个 **Apache** 服务器。

流程

1. 在工作站上创建一个名为 **openstack-provision.yaml** 的文件。包含 **Provisioning** 服务器的资源规格。以下示例使用特定的 **RHEL 9.2 QCOW2** 镜像定义了 **Provisioning** 服务器的规格：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackProvisionServer
metadata:
  name: openstack-provision-server 1
  namespace: openstack 2
spec:
  baseImageUrl: http://<source_host>/rhel-9.2-x86_64-kvm.qcow2 3
  port: 8080 4
```

1. 标识 **OpenStackProvisionServer CR** 的名称。

2. **OSPdO** 命名空间，例如 **openstack**。

3. **Provisioning** 服务器的 **RHEL QCOW2** 镜像的初始来源。在创建服务器时，从此远程源下载镜像。

4

要进一步描述您可以配置 OpenStackProvisionServer CR 的值，请查看 OpenStackProvisionServer CRD 规格模式：

```
$ oc describe crd openstackprovisionserver
```

2. 保存 openstack-provision.yaml 文件。

3. 创建 Provisioning 服务器：

```
$ oc create -f openstack-provision.yaml -n openstack
```

4. 验证 Provisioning 服务器的资源是否已创建：

```
$ oc get openstackprovisionserver/openstack-provision-server -n openstack
```

第 6 章 使用 DIRECTOR OPERATOR 配置和部署 OVERCLOUD

在为 overcloud 置备虚拟和裸机节点后，您可以配置 overcloud 节点。您必须创建一个 OpenStackConfigGenerator 资源来生成 Ansible playbook，将节点注册到红帽客户门户网站或 Red Hat Satellite，然后创建一个 OpenStackDeploy 资源来将配置应用到节点。

6.1. 使用 OPENSTACKCONFIGGENERATOR CRD 创建用于 OVERCLOUD 配置的 ANSIBLE PLAYBOOK

置备 overcloud 基础架构后，您必须创建一组 Ansible playbook，以便在 overcloud 节点上配置 Red Hat OpenStack Platform (RHOSP)。您可以使用 OpenStackConfigGenerator 自定义资源定义 (CRD)创建这些 playbook。OpenStackConfigGenerator CRD 使用 RHOSP director config-download 功能将 heat 配置转换为 playbook。

提示

使用以下命令查看 OpenStackConfigGenerator CRD 定义和规格模式：

```
$ oc describe crd openstackconfiggenerator

$ oc explain openstackconfiggenerator.spec
```

先决条件

- 您已创建了带有 OpenStackControlPlane CRD 的 control plane。
- 已使用 OpenStackBaremetalSets CRD 创建 Compute 节点。
- 您已创建了包含自定义 heat 模板的 ConfigMap 对象。
- 您已创建了包含自定义环境文件的 ConfigMap 对象。

流程

1. 在工作站上创建一个名为 openstack-config-generator.yaml 的文件。包含用于生成 Ansible playbook 的资源规格。以下示例定义了生成 playbook 的规格：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackConfigGenerator
metadata:
  name: default (1)
  namespace: openstack
spec:
  enableFencing: true (2)
  gitSecret: git-secret (3)
  imageURL: registry.redhat.io/rhosp-rhel8/openstack-tripleoclient:17.1
  heatEnvConfigMap: heat-env-config (4)
  # List of heat environment files to include from tripleo-heat-templates/environments
  heatEnvs: (5)
  - ssl/tls-endpoints-public-dns.yaml
  - ssl/enable-tls.yaml
  tarballConfigMap: tripleo-tarball-config (6)
```

(1) 配置生成器的名称，默认为。

(2) 设置为 true 以启用自动创建所需的 heat 环境文件以启用隔离。生产环境 RHOSP 环境必须启用隔离。运行 Pacemaker 的虚拟机需要 fence-agents-kubevirt 软件包。

(3) 设置为包含 Git 身份验证凭据的 ConfigMap 对象，默认为 git-secret。

(4) 默认情况下，包含自定义环境文件的 ConfigMap 对象（默认为 heat-env-config ）。

(5) tripleo-heat-templates/environments 目录中由 TripleO 提供的默认 heat 环境文件列表，用于生成 playbook。

(6) 默认情况下，包含带有自定义 heat 模板的 tarball 的 ConfigMap 对象默认为 tripleo-tarball-config。

2. 可选： 要更改 OpenStackConfigGenerator CR 用来创建临时 heat 服务的容器镜像位置，请在 openstack-config-generator.yaml 文件中添加以下配置：

```
spec:
  ...
  ephemeralHeatSettings:
    heatAPIImageURL: <heat_api_image_location>
    heatEngineImageURL: <heat_engine_image_location>
    mariadbImageURL: <mariadb_image_location>
    rabbitImageURL: <rabbitmq_image_location>
```

- 将 <heat_api_image_location > 替换为托管 heat API 镜像的目录的路径，openstack-heat-api。
- 将 <heat_engine_image_location > 替换为托管 heat 引擎镜像的目录的路径，即 openstack-heat-engine。
- 将 <mariadb_image_location > 替换为托管 MariaDB 镜像的目录的路径，openstack-mariadb。
- 将 <rabbitmq_image_location > 替换为托管 RabbitMQ 镜像的目录的路径，openstack-rabbitmq。

3. 可选：要在 debug 模式中为生成配置创建 Ansible playbook，请在 openstack-config-generator.yaml 文件中添加以下配置：

```
spec:
  ...
  interactive: true
```

有关在交互模式中调试 OpenStackConfigGenerator pod 的更多信息，请参阅 调试配置生成。

4. 保存 openstack-config-generator.yaml 文件。

5. 创建 Ansible 配置生成器：

```
$ oc create -f openstack-config-generator.yaml -n openstack
```

6. 验证是否已创建配置生成器的资源：

-

```
$ oc get openstackconfiggenerator/default -n openstack
```

6.2. 注册 OVERCLOUD 的操作系统

在 director Operator (OSPdO)配置 overcloud 节点之前，您必须将所有节点的操作系统注册到红帽客户门户网站或 Red Hat Satellite Server，并为节点启用存储库。

作为 OpenStackControlPlane CR 的一部分，OSPdO 创建一个 OpenStackClient pod，您可以通过远程 Shell (RSH)访问以运行 Red Hat OpenStack Platform (RHOSP)命令。此 pod 还包含一个名为 /home/cloud-admin/ctlplane-ansible-inventory 的 Ansible 清单脚本。

要注册节点，您可以将 redhat_subscription Ansible 模块与 OpenStackClient pod 中的清单脚本搭配使用。

流程

1. 打开到 OpenStackClient pod 的 RSH 连接：

```
$ oc rsh -n openstack openstackclient
```

2. 进入 cloud-admin 主目录：

```
$ cd /home/cloud-admin
```

3. 创建一个使用 redhat_subscription 模块注册节点的 playbook。例如，以下 playbook 注册 Controller 节点：

```
---
- name: Register Controller nodes
  hosts: Controller
  become: yes
  vars:
    repos:
      - rhel-9-for-x86_64-baseos-eus-rpms
      - rhel-9-for-x86_64-appstream-eus-rpms
      - rhel-9-for-x86_64-highavailability-eus-rpms
      - openstack-17.1-for-rhel-9-x86_64-rpms
      - fast-datapath-for-rhel-9-x86_64-rpms
      - rhceph-6-tools-for-rhel-9-x86_64-rpms
  tasks:
    - name: Register system 1
      redhat_subscription:
```

```
    username: myusername
    password: p@55w0rd!
    org_id: 1234567
    release: 9.2
    pool_ids: 1a85f9223e3d5e43013e3d6e8ff506fd
- name: Disable all repos (2)
  command: "subscription-manager repos --disable *"
- name: Enable Controller node repos (3)
  command: "subscription-manager repos --enable {{ item }}"
  with_items: "{{ repos }}"
```

(1) **注册节点的任务。**

(2) **禁用任何启用了自动的存储库的任务。**

(3) **仅启用与 Controller 节点相关的软件仓库的任务。存储库通过 repos 变量列出。**

4. **将 overcloud 节点注册到所需的软件仓库：**

```
$ ansible-playbook -i /home/cloud-admin/ctlplane-ansible-inventory ./rhsm.yaml
```

其他资源

- **redhat_subscription - 使用 subscription-manager 命令管理 RHSM 的注册和订阅**
- **手动运行基于 Ansible 的注册**
- **overcloud 软件仓库**

6.3. 使用 DIRECTOR OPERATOR 应用 OVERCLOUD 配置

只有在创建了 control plane、置备裸机 Compute 节点并生成 Ansible playbook 以在每个节点上配置软件后，才可使用 director Operator (OSPdO)配置 overcloud。当您创建 OpenStackDeploy 自定义资源(CR)时，OSPdO 会创建一个作业，该作业将运行 Ansible playbook 来配置 overcloud。

提示

使用以下命令查看 **OpenStackDeploy CRD** 定义和规格模式：

```
$ oc describe crd openstackdeploy

$ oc explain openstackdeploy.spec
```

先决条件

- 您已创建了带有 **OpenStackControlPlane CRD** 的 **control plane**。
- 已使用 **OpenStackBarementalSets CRD** 创建 **Compute** 节点。
- 您已使用 **OpenStackConfigGenerator CRD** 为 **overcloud** 创建 **Ansible playbook** 配置。

流程

1. 检索最新的 **OpenStackConfigVersion** 对象的 **hash/digest**，它代表应该用于配置 **overcloud** 的 **Ansible playbook**：

```
$ oc get -n openstack --sort-by {.metadata.creationTimestamp} openstackconfigversion -o
json
```

2. 在工作站上创建一个名为 **openstack-deployment.yaml** 的文件，并包含 **Ansible playbook** 的资源规格：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: default
spec:
  configVersion: <config_version>
  configGenerator: default
```

- 将 **<config_version >** 替换为在第 **1** 步中检索到的 **Ansible playbook hash/digest**，例如 **n5fch96h548h75hf4hbdhb8hfdh676h57bh96h5c5h59hf4h88h.. .**

3. 保存 **openstack-deployment.yaml** 文件。

4. 创建 OpenStackDeploy 资源：

```
$ oc create -f openstack-deployment.yaml -n openstack
```

当部署运行时，它会创建一个 Kubernetes 作业来执行 Ansible playbook。您可以查看作业的日志以监视 Ansible playbook 运行：

```
$ oc logs -f jobs/deploy-openstack-default
```

您还可以通过登录到 openstackclient pod 来手动访问执行的 Ansible playbook。您可以在 /home/cloud-admin/work/directory 中找到用于当前部署的 ansible playbook 和 ansible.log 文件。

6.4. 调试配置生成

要调试配置生成操作，您可以将 OpenStackConfigGenerator CR 设置为使用交互模式。在交互模式中，OpenStackConfigGenerator CR 会创建环境来启动 playbook，但不自动呈现 playbook。

先决条件

- 您的 OpenStackConfigGenerator CR 以互动模式创建：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackConfigGenerator
metadata:
  name: default
  namespace: openstack
spec:
  …
  interactive: true
```

- 启动前缀 generate-config 的 OpenStackConfigGenerator pod。

流程

1. 打开到 OpenStackConfigGenerator pod 的远程 Shell (RSH)连接：

```
$ oc rsh $(oc get pod -o name -l job-name=generate-config-default)
```

2. 检查文件和 playbook 渲染：

```
$ ls -la /home/cloud-admin/
...
config 1
config-custom 2
config-passwords 3
create-playbooks.sh 4
process-heat-environment.py 5
tht-tars 6
```

1. 存储 OSPdO 自动渲染的文件的目录。

2. 存储通过 heatEnvConfigMap 选项指定的环境文件的目录。

3. 存储 OSPdO 创建的 overcloud 服务密码的目录。

4. 呈现 Ansible playbook 的脚本。

5. create-playbooks 用来复制未文档的 heat 客户端合并映射参数的内部脚本。

6. 存储通过 tarballConfigMap 选项指定的 tarball 的目录。

第 7 章 使用 DIRECTOR OPERATOR 部署 RHOSP 超融合基础架构(HCI)

您可以使用 director Operator (OSPdO)来部署带有超融合基础架构(HCI)的 overcloud。具有 HCI 的 overcloud 将计算和 Red Hat Ceph Storage OSD 服务在同一节点上并置。

7.1. 先决条件

- 您的计算 HCI 节点需要额外的磁盘才能用作 OSD。
- 您已在正常运行的 Red Hat OpenShift Container Platform (RHOCP)集群上安装并准备好 OSPdO。如需更多信息，请参阅安装和准备 director Operator。
- 已使用 OpenStackNetConfig 自定义资源定义(CRD) （包括 control plane 和任何隔离网络）创建了 overcloud 网络。如需更多信息，请参阅使用 director Operator 创建网络。
- 您已创建了 ConfigMap 来存储 overcloud 的任何自定义 heat 模板和环境文件。如需更多信息，请参阅使用 director Operator 自定义 overcloud。
- 您已为 overcloud 创建 control plane 和裸机 Compute 节点。如需更多信息，请参阅使用 director Operator 创建 overcloud 节点。
- 您已创建了并应用 OpenStackConfigGenerator cusstom 资源，以呈现用于 overcloud 配置的 Ansible playbook。

7.2. 使用 DIRECTOR OPERATOR 的 COMPUTE HCI 角色创建 ROLES_DATA.YAML 文件

要在 overcloud 中包含计算 HCI 角色的配置，您必须将 Compute HCI 角色包含在 overcloud 部署的 roles_data.yaml 文件中。

注意

确保使用 roles_data.yaml 作为文件名。

流程

1. **访问 openstackclient 的远程 shell：**

```
$ oc rsh -n openstack openstackclient
```

2. **取消设置 OS_CLOUD 环境变量：**

```
$ unset OS_CLOUD
```

3. **进入 cloud-admin 目录：**

```
$ cd /home/cloud-admin/
```

4. **使用 Controller 和 ComputeHCI 角色生成一个新的 roles_data.yaml 文件：**

```
$ openstack overcloud roles generate -o roles_data.yaml Controller ComputeHCI
```

5. **退出 openstackclient pod:**

```
$ exit
```

6. **将自定义 roles_data.yaml 文件从 openstackclient pod 复制到您的自定义 templates 目录中：**

```
$ oc cp openstackclient:/home/cloud-admin/roles_data.yaml
custom_templates/roles_data.yaml -n openstack
```

其他资源

- **创建 roles_data 文件**

后续步骤

- **在 director Operator 中配置 HCI 网络**

7.3. 在 DIRECTOR OPERATOR 中配置 HCI 网络

在工作站上创建目录以存储自定义模板和环境文件，并为计算 HCI 角色配置 NIC 模板。

流程

1. 为您的自定义模板创建一个目录：

```
$ mkdir custom_templates
```

2. 在 custom_templates 目录中创建一个名为 multiple_nics_vlans_dvr.j2 的自定义模板文件。

3. 将裸机节点的 NIC 的配置添加到 multiple_nics_vlans_dvr.j2 文件中。有关 NIC 配置文件示例，请参阅 HCI Compute 节点的自定义 NIC heat 模板。

4. 为自定义环境文件创建一个目录：

```
$ mkdir custom_environment_files
```

5. 在 custom_environment_files 目录中的 network-environment.yaml 环境文件中映射 overcloud 角色的 NIC 模板：

```
parameter_defaults:
  ComputeHCINetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
```

其他资源

- 自定义网络接口模板

后续步骤

- 将自定义模板添加到 overcloud 配置

7.4. HCI COMPUTE 节点的自定义 NIC HEAT 模板

以下示例是包含 HCI Compute 裸机节点的 NIC 配置的 heat 模板。heat 模板中的配置将网络映射到以下网桥和接口：

网络	Bridge	Interface
Control Plane, Storage, Internal API	N/A	**nic3**
External, Tenant	**br-ex**	**nic4**

要在部署中使用以下模板，请将示例复制到 workstation 上的 custom_templates 目录中的 multiple_nics_vlans_dvr.j2。您可以为裸机节点的 NIC 配置修改此配置。

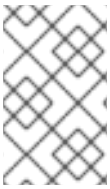
示例

```
{% set mtu_list = [ctlplane_mtu] %}
{% for network in role_networks %}
{{ mtu_list.append(lookup('vars', networks_lower[network] ~ '_mtu')) }}
{%- endfor %}
{% set min_viable_mtu = mtu_list | max %}
network_config:
# BMH provisioning interface used for ctlplane
- type: interface
  name: nic1
  mtu: 1500
  use_dhcp: false
  dns_servers: {{ ctlplane_dns_nameservers }}
  domain: {{ dns_search_domains }}
  addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  routes: {{ ctlplane_host_routes }}
# Disable OCP cluster interface
- type: interface
  name: nic2
  mtu: 1500
  use_dhcp: false
{% for network in networks_all if network not in networks_skip_config|default([]) %}
{% if network == 'External' %}
- type: ovs_bridge
  name: {{ neutron_physical_bridge_name }}
  mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
  dns_servers: {{ ctlplane_dns_nameservers }}
  use_dhcp: false
{% if network in role_networks %}
  addresses:
  - ip_netmask:
      {{ lookup('vars', networks_lower[network] ~ '_ip') }}/{{ lookup('vars', networks_lower[network] ~
'_cidr') }}
  routes: {{ lookup('vars', networks_lower[network] ~ '_host_routes') }}
{% endif %}
  members:
  - type: interface
```

```
    name: nic3
    mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
    primary: true
{% endif %}
{% endfor %}
- type: ovs_bridge
  name: br-tenant
  mtu: {{ min_viable_mtu }}
  use_dhcp: false
  members:
  - type: interface
    name: nic4
    mtu: {{ min_viable_mtu }}
    use_dhcp: false
    primary: true
{% for network in networks_all if network not in networks_skip_config|default([]) %}
{% if network not in ["External"] and network in role_networks %}
  - type: vlan
    mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
    vlan_id: {{ lookup('vars', networks_lower[network] ~ '_vlan_id') }}
    addresses:
    - ip_netmask:
        {{ lookup('vars', networks_lower[network] ~ '_ip') }}/{{ lookup('vars', networks_lower[network] ~
'_cidr') }}
    routes: {{ lookup('vars', networks_lower[network] ~ '_host_routes') }}
{% endif %}
{% endfor %}
```

7.5. 将自定义模板添加到 OVERCLOUD 配置

director Operator (OSPdO)将核心 overcloud heat 模板转换为 Ansible playbook，当您准备好在每个节点上配置 Red Hat OpenStack Platform (RHOSP)软件时，应用到置备的节点。要将自己的自定义 heat 模板和自定义角色文件添加到 overcloud 部署中，您必须将模板文件归档到 tarball 文件中，并将 tarball 文件的二进制内容包含在名为 tripleo-tarball-config 的 OpenShift ConfigMap 对象中。此 tarball 文件可以包含复杂的目录结构，以扩展一组核心模板。OSPdO 将文件和目录从 tarball 文件提取到与核心 heat 模板集相同的目录中。如果您的任何自定义模板的名称与核心集合中的模板的名称相同，则自定义模板将覆盖 core 模板。



注意

环境文件中的所有引用都必须相对于提取 **tarball** 的 **TripleO heat** 模板。

先决条件

- 要应用到置备的节点的自定义 **overcloud** 模板。

流程

1. 进入自定义模板的位置：

```
$ cd ~/custom_templates
```

2. 将模板归档到 gzipped tarball 中：

```
$ tar -cvzf custom-config.tar.gz *.yaml
```

3. 创建 tripleo-tarball-config ConfigMap CR，并使用 tarball 作为数据：

```
$ oc create configmap tripleo-tarball-config --from-file=custom-config.tar.gz -n openstack
```

4. 验证 ConfigMap CR 是否已创建：

```
$ oc get configmap/tripleo-tarball-config -n openstack
```

其他资源

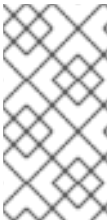
- 创建和使用配置映射
- 了解 heat 模板

后续步骤

- 在 overcloud 配置中添加自定义环境文件

7.6. 用于在 DIRECTOR OPERATOR 中配置超融合基础架构(HCI)存储的自定义环境文件

以下示例是包含计算 HCI 节点的 Red Hat Ceph Storage 配置的环境文件。此配置将 OSD 节点映射到 sdb、sdc 和 sdd 设备，并使用 is_hci 选项启用 HCI。



注意

您可以修改此配置，以适应裸机节点的存储配置。使用 "Ceph Placement Groups (PG) per Pool Calculator" 来确定 **CephPoolDefaultPgNum** 参数的值。

要在部署中使用此模板，请将示例的内容复制到 **workstation** 上的 **custom_environment_files** 目录中的 **compute-hci.yaml** 中。

```
resource_registry:
  OS::TripleO::Services::CephMgr: deployment/cephadm/ceph-mgr.yaml
  OS::TripleO::Services::CephMon: deployment/cephadm/ceph-mon.yaml
  OS::TripleO::Services::CephOSD: deployment/cephadm/ceph-osd.yaml
  OS::TripleO::Services::CephClient: deployment/cephadm/ceph-client.yaml

parameter_defaults:
  CephDynamicSpec: true
  CephSpecFqdn: true
  CephConfigOverrides:
    rgw_swift_enforce_content_length: true
    rgw_swift_versioning_enabled: true
    osd:
      osd_memory_target_autotune: true
      osd_numa_auto_affinity: true
    mgr:
      mgr/cephadm/autotune_memory_target_ratio: 0.2

  CinderEnableIscsiBackend: false
  CinderEnableRbdBackend: true
  CinderBackupBackend: ceph
  CinderEnableNfsBackend: false
  NovaEnableRbdBackend: true
  GlanceBackend: rbd
  CinderRbdPoolName: "volumes"
  NovaRbdPoolName: "vms"
  GlanceRbdPoolName: "images"
  CephPoolDefaultPgNum: 32
  CephPoolDefaultSize: 2
```

7.7. 在 OVERCLOUD 配置中添加自定义环境文件

要在 **overcloud** 中启用功能或设置参数，您必须包含 **overcloud** 部署的环境文件。**director Operator (OSPdO)**使用名为 **heat-env-config** 的 **ConfigMap** 对象存储和检索环境文件。**ConfigMap** 对象以以下格式存储环境文件：

```
...
data:
  <environment_file_name>: |+
    <environment_file_contents>
```

例如，以下 ConfigMap 包含两个环境文件：

```
...
data:
  network_environment.yaml: |+
    parameter_defaults:
      ComputeNetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
  cloud_name.yaml: |+
    parameter_defaults:
      CloudDomain: ocp4.example.com
      CloudName: overcloud.ocp4.example.com
      CloudNameInternal: overcloud.internalapi.ocp4.example.com
      CloudNameStorage: overcloud.storage.ocp4.example.com
      CloudNameStorageManagement: overcloud.storagemgmt.ocp4.example.com
      CloudNameCtlplane: overcloud.ctlplane.ocp4.example.com
```

将一组自定义环境文件从目录上传到 ConfigMap 对象，您可以将其作为 overcloud 部署的一部分包含。

先决条件

- 用于 overcloud 部署的自定义环境文件。

流程

1. 创建 heat-env-config ConfigMap 对象：

```
$ oc create configmap -n openstack heat-env-config \
 --from-file=~/<dir_custom_environment_files>/ \
 --dry-run=client -o yaml | oc apply -f -
```

   - 将 < dir_custom_environment_files > 替换为包含要在 overcloud 部署中使用的环境文件的目录。ConfigMap 对象存储为各个 数据 条目。

2. 验证 heat-env-config ConfigMap 对象是否包含所有必需的环境文件：

```
$ oc get configmap/heat-env-config -n openstack
```

7.8. 创建 HCI COMPUTE 节点并部署 OVERCLOUD

Compute 节点为您的 Red Hat OpenStack Platform (RHOSP)环境提供计算资源。您的 overcloud

中必须至少有一个 Compute 节点，您可以在部署后扩展 Compute 节点的数量。

定义一个 OpenStackBaremetalSet 自定义资源(CR)，从 Red Hat OpenShift Container Platform (RHOCP)管理的裸机创建 Compute 节点。

提示

使用以下命令查看 OpenStackBareMetalSet CRD 定义和规格模式：

```
$ oc describe crd openstackbaremetalset

$ oc explain openstackbaremetalset.spec
```

先决条件

- 您已使用 OpenStackNetConfig CR 创建 control plane 网络以及任何其他隔离网络。
- 您已创建了带有 OpenStackControlPlane CRD 的 control plane。

流程

1. 在工作站上创建一个名为 openstack-hcicompute.yaml 的文件。包含 HCI Compute 节点的资源规格。例如，3 HCI Compute 节点的规格如下：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackBaremetalSet
metadata:
  name: computehci (1)
  namespace: openstack (2)
spec: (3)
  count: 3
  baseImageUrl: http://<source_host>/rhel-9.2-x86_64-kvm.qcow2
  deploymentSSHSecret: osp-controlplane-ssh-keys
  ctlplaneInterface: enp8s0
  networks:
    - ctlplane
    - internal_api
    - tenant
    - storage
    - storage_mgmt
  roleName: ComputeHCI
  passwordSecret: userpassword (4)
```

1. HCI Compute 节点裸机集的名称，如 computehci。
2. OSPdO 命名空间，例如 openstack。
3. HCI Compute 节点的配置。
4. 可选：为具有密码的用户提供每个节点上的 root 访问权限的 Secret 资源。

2. 保存 openstack-hcicompute.yaml 文件。
3. 创建 HCI Compute 节点：

```
$ oc create -f openstack-hcicompute.yaml -n openstack
```

4. 验证 HCI Compute 节点的资源是否已创建：

```
$ oc get openstackbaremetalset/computehci -n openstack
```

5. 要验证 HCI Compute 节点的创建，请查看 RHOCP 管理的裸机机器：

```
$ oc get baremetalhosts -n openshift-machine-api
```

6. 使用 OpenStackConfigGenerator CRD 为 overcloud 配置创建 Ansible playbook。如需更多信息，请参阅使用 OpenStackConfigGenerator CRD 为 overcloud 配置创建 Ansible playbook。
7. 注册 overcloud 的操作系统。如需更多信息，请参阅 注册 overcloud 的操作系统。
8. 应用 overcloud 配置。如需更多信息，请参阅使用 director Operator 应用 overcloud 配置。

第 8 章 使用 DIRECTOR OPERATOR 部署外部 RED HAT CEPH STORAGE 集群

您可以使用 director Operator (OSPdO)部署连接到外部 Red Hat Ceph Storage 集群的 overcloud。

先决条件

- 您有一个外部 Red Hat Ceph Storage 集群。
- 您已在正常运行的 Red Hat OpenShift Container Platform (RHOCP)集群上安装并准备好 OSPdO。如需更多信息，请参阅安装和准备 director Operator。
- 已使用 OpenStackNetConfig 自定义资源定义(CRD) （包括 control plane 和任何隔离网络）创建了 overcloud 网络。如需更多信息，请参阅使用 director Operator 创建网络。
- 您已创建了 ConfigMap 来存储 overcloud 的任何自定义 heat 模板和环境文件。如需更多信息，请参阅使用 director Operator 自定义 overcloud。
- 您已为 overcloud 创建 control plane 和裸机 Compute 节点。如需更多信息，请参阅使用 director Operator 创建 overcloud 节点。
- 您已创建了并应用 OpenStackConfigGenerator 自定义资源，以便为 overcloud 配置呈现 Ansible playbook。

8.1. 在 DIRECTOR OPERATOR 中为 COMPUTE 角色配置网络

在工作站上创建目录以存储自定义模板和环境文件，并为 Compute 角色配置 NIC 模板。

流程

1. 为您的自定义模板创建一个目录：

```
$ mkdir custom_templates
```

2. 在 custom_templates 目录中创建一个名为 multiple_nics_vlans_dvr.j2 的自定义模板文件。

3. 将裸机 Compute 节点的 NIC 配置添加到 multiple_nics_vlans_dvr.j2 文件中。如需 NIC 配置文件示例，请参阅 Compute 节点 的自定义 NIC heat 模板。

4. 为自定义环境文件创建一个目录：

```
$ mkdir custom_environment_files
```

5. 在 custom_environment_files 目录中的 network-environment.yaml 环境文件中映射 overcloud 角色的 NIC 模板：

```
parameter_defaults:
  ComputeNetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
```

其他资源

- 自定义网络接口模板

8.2. COMPUTE 节点的自定义 NIC HEAT 模板

以下示例是 heat 模板，其中包含连接到外部 Red Hat Ceph Storage 集群的 overcloud 中 Compute 裸机节点的 NIC 配置。heat 模板中的配置将网络映射到以下网桥和接口：

网络	Bridge	interface
Control Plane, Storage, Internal API	N/A	**nic3**
External, Tenant	**br-ex**	**nic4**

要在部署中使用以下模板，请将示例复制到 workstation 上的 custom_templates 目录中的 multiple_nics_vlans_dvr.j2。您可以为裸机节点的 NIC 配置修改此配置。

示例

```
{% set mtu_list = [ctlplane_mtu] %}
```

```
{% for network in role_networks %}
{{ mtu_list.append(lookup('vars', networks_lower[network] ~ '_mtu')) }}
{%- endfor %}
{% set min_viable_mtu = mtu_list | max %}
network_config:
# BMH provisioning interface used for ctlplane
- type: interface
  name: nic1
  mtu: 1500
  use_dhcp: false
  dns_servers: {{ ctlplane_dns_nameservers }}
  domain: {{ dns_search_domains }}
  addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  routes: {{ ctlplane_host_routes }}
# Disable OCP cluster interface
- type: interface
  name: nic2
  mtu: 1500
  use_dhcp: false
{% for network in networks_all if network not in networks_skip_config|default([]) %}
{% if network == 'External' %}
- type: ovs_bridge
  name: {{ neutron_physical_bridge_name }}
  mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
  dns_servers: {{ ctlplane_dns_nameservers }}
  use_dhcp: false
{% if network in role_networks %}
  addresses:
  - ip_netmask:
      {{ lookup('vars', networks_lower[network] ~ '_ip') }}/{{ lookup('vars', networks_lower[network] ~
'_cidr') }}
  routes: {{ lookup('vars', networks_lower[network] ~ '_host_routes') }}
{% endif %}
  members:
  - type: interface
    name: nic3
    mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
    primary: true
{% endif %}
{% endfor %}
- type: ovs_bridge
  name: br-tenant
  mtu: {{ min_viable_mtu }}
  use_dhcp: false
  members:
  - type: interface
    name: nic4
    mtu: {{ min_viable_mtu }}
    use_dhcp: false
    primary: true
{% for network in networks_all if network not in networks_skip_config|default([]) %}
{% if network not in ["External"] and network in role_networks %}
  - type: vlan
    mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
    vlan_id: {{ lookup('vars', networks_lower[network] ~ '_vlan_id') }}
```

```
    addresses:
    - ip_netmask:
        {{ lookup('vars', networks_lower[network] ~ '_ip') }}/{{ lookup('vars', networks_lower[network] ~
'_cidr') }}
    routes: {{ lookup('vars', networks_lower[network] ~ '_host_routes') }}
{% endif %}
{% endfor %}
```

8.3. 将自定义模板添加到 OVERCLOUD 配置

director Operator (OSPdO)将核心 overcloud heat 模板转换为 Ansible playbook，当您准备好在每个节点上配置 Red Hat OpenStack Platform (RHOSP)软件时，应用到置备的节点。要将自己的自定义 heat 模板和自定义角色文件添加到 overcloud 部署中，您必须将模板文件归档到 tarball 文件中，并将 tarball 文件的二进制内容包含在名为 tripleo-tarball-config 的 OpenShift ConfigMap 对象中。此 tarball 文件可以包含复杂的目录结构，以扩展一组核心模板。OSPdO 将文件和目录从 tarball 文件提取到与核心 heat 模板集相同的目录中。如果您的任何自定义模板的名称与核心集合中的模板的名称相同，则自定义模板将覆盖 core 模板。

注意

环境文件中的所有引用都必须相对于提取 tarball 的 TripleO heat 模板。

先决条件

- 要应用到置备的节点的自定义 overcloud 模板。

流程

1. 进入自定义模板的位置：

```
$ cd ~/custom_templates
```

2. 将模板归档到 gzipped tarball 中：

```
$ tar -cvzf custom-config.tar.gz *.yaml
```

3. 创建 tripleo-tarball-config ConfigMap CR，并使用 tarball 作为数据：

```
$ oc create configmap tripleo-tarball-config --from-file=custom-config.tar.gz -n openstack
```

4. 验证 **ConfigMap CR** 是否已创建：

```
$ oc get configmap/tripleo-tarball-config -n openstack
```

其他资源

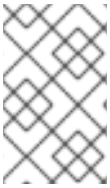
- 创建和使用配置映射
- 了解 heat 模板

后续步骤

- 在 overcloud 配置中添加自定义环境文件

8.4. 用于在 DIRECTOR OPERATOR 中配置外部 CEPH STORAGE 使用的自定义环境文件

要与外部 Red Hat Ceph Storage 集群集成，请包括环境文件以及类似以下示例所示的值。这个示例在 overcloud 节点上启用 CephExternal 和 CephClient 服务，并为不同的 RHOSP 服务设置池。



注意

您可以修改此配置来适合您的存储配置。

要在部署中使用此模板，请将示例的内容复制到 workstation 上的 custom_environment_files 目录中的 ceph-ansible-external.yaml 中。

```
resource_registry:
  OS::TripleO::Services::CephExternal: deployment/cephadm/ceph-client.yaml

parameter_defaults:
  CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19' 1
  CephClientKey: 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==' 2
  CephExternalMonHost: '172.16.1.7, 172.16.1.8' 3
  ExternalCeph: true

  # the following parameters enable Ceph backends for Cinder, Glance, Gnocchi and Nova
  NovaEnableRbdBackend: true
```

```
CinderEnableRbdBackend: true
CinderBackupBackend: ceph
GlanceBackend: rbd
# Uncomment below if enabling legacy telemetry
# GnocchiBackend: rbd
# If the Ceph pools which host VMs, Volumes and Images do not match these
# names OR the client keyring to use is not named 'openstack',  edit the
# following as needed.
NovaRbdPoolName: vms
CinderRbdPoolName: volumes
CinderBackupRbdPoolName: backups
GlanceRbdPoolName: images
# Uncomment below if enabling legacy telemetry
# GnocchiRbdPoolName: metrics
CephClientUserName: openstack

# finally we disable the Cinder LVM backend
CinderEnableIscsiBackend: false
```

1

外部 Red Hat Ceph Storage 集群的文件系统 ID。

2

用于外部 Red Hat Ceph Storage 集群的 Red Hat Ceph Storage 客户端密钥。

3

外部 Red Hat Ceph Storage 集群中所有 MON 主机的 IP 地址的逗号分隔列表。

其他资源

- **将 overcloud 与现有 Red Hat Ceph Storage 集群集成**
- **Red Hat Container Registry 身份验证**

8.5. 在 OVERCLOUD 配置中添加自定义环境文件

要在 overcloud 中启用功能或设置参数，您必须包含 overcloud 部署的环境文件。director Operator (OSPdO)使用名为 heat-env-config 的 ConfigMap 对象存储和检索环境文件。ConfigMap 对象以以下格式存储环境文件：

```
...
data:
```

```
  <environment_file_name>: |+
    <environment_file_contents>
```

例如，以下 ConfigMap 包含两个环境文件：

```
...
data:
  network_environment.yaml: |+
    parameter_defaults:
      ComputeNetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
  cloud_name.yaml: |+
    parameter_defaults:
      CloudDomain: ocp4.example.com
      CloudName: overcloud.ocp4.example.com
      CloudNameInternal: overcloud.internalapi.ocp4.example.com
      CloudNameStorage: overcloud.storage.ocp4.example.com
      CloudNameStorageManagement: overcloud.storagemgmt.ocp4.example.com
      CloudNameCtlplane: overcloud.ctlplane.ocp4.example.com
```

将一组自定义环境文件从目录上传到 ConfigMap 对象，您可以将其作为 overcloud 部署的一部分包含。

先决条件

- **用于 overcloud 部署的自定义环境文件。**

流程

1. **创建 heat-env-config ConfigMap 对象：**

```
$ oc create configmap -n openstack heat-env-config \
 --from-file=~/<dir_custom_environment_files>/ \
 --dry-run=client -o yaml | oc apply -f -
```

   - **将 < dir_custom_environment_files > 替换为包含要在 overcloud 部署中使用的环境文件的目录。ConfigMap 对象存储为各个 数据 条目。**

2. **验证 heat-env-config ConfigMap 对象是否包含所有必需的环境文件：**

```
$ oc get configmap/heat-env-config -n openstack
```

8.6. 创建 COMPUTE 节点并部署 OVERCLOUD

Compute 节点为您的 Red Hat OpenStack Platform (RHOSP)环境提供计算资源。您的 overcloud 中必须至少有一个 Compute 节点，您可以在部署后扩展 Compute 节点的数量。

定义一个 OpenStackBaremetalSet 自定义资源(CR)，从 Red Hat OpenShift Container Platform (RHOCP)管理的裸机创建 Compute 节点。

提示

使用以下命令查看 OpenStackBareMetalSet CRD 定义和规格模式：

```
$ oc describe crd openstackbaremetalset

$ oc explain openstackbaremetalset.spec
```

先决条件

- 您已使用 OpenStackNetConfig CR 创建 control plane 网络以及任何其他隔离网络。
- 您已创建了带有 OpenStackControlPlane CRD 的 control plane。

流程

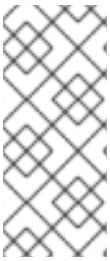
1. 使用 OpenStackBaremetalSet CRD 创建 Compute 节点。如需更多信息，请参阅使用 OpenStackBaremetalSet CRD 创建 Compute 节点。
2. 使用 OpenStackConfigGenerator CRD 为 overcloud 配置创建 Ansible playbook。如需更多信息，请参阅使用 OpenStackConfigGenerator CRD 为 overcloud 配置创建 Ansible playbook。
3. 注册 overcloud 的操作系统。如需更多信息，请参阅 注册 overcloud 的操作系统。
4. 应用 overcloud 配置。如需更多信息，请参阅使用 director Operator 应用 overcloud 配置。

第 9 章 访问使用 DIRECTOR OPERATOR 部署的 OVERCLOUD

使用 director Operator (OSPdO)部署 overcloud 后，您可以使用 openstack 客户端工具访问和运行命令。overcloud 的主要访问点是通过 OSPdO 部署的 OpenStackClient pod，作为您创建的 OpenStackControlPlane 资源的一部分。

9.1. 访问 OPENSTACKCLIENT POD

OpenStackClient 容器集是针对 overcloud 运行命令的主要访问点。此 pod 包含对 overcloud 执行操作所需的客户端工具和身份验证详情。要从工作站访问 pod，您必须使用工作站上的 oc 命令连接到 pod 的远程 shell。



注意

当您访问没有 director Operator (OSPdO)部署的 overcloud 时，您通常会运行 source ~/overcloudrc 命令来设置环境变量以访问 overcloud。对于使用 OSPdO 部署的 overcloud，您不需要这一步。

流程

1. 访问 openstackclient 的远程 shell：

```
$ oc rsh -n openstack openstackclient
```

2. 进入 cloud-admin 主目录：

```
$ cd /home/cloud-admin
```

3. 运行 openstack 命令。例如，您可以使用以下命令创建默认网络：

```
$ openstack network create default
```

其他资源

- 创建和管理实例
- 配置 Red Hat OpenStack Platform 网络

9.2. 访问 OVERCLOUD 仪表板

您可以使用与标准 overcloud 相同的方法访问使用 director Operator (OSPdO)部署的 overcloud 仪表板：使用 Web 浏览器访问 control plane 保留的虚拟 IP 地址。

流程

1. 可选： 要以 admin 用户身份登录，请从 tripleo-passwords secret 中的 AdminPassword 参数获取 admin 密码：

```
$ oc get secret tripleo-passwords -o jsonpath='{.data.tripleo-overcloud-passwords\.yaml}' |
base64 -d
```

2. 从 OpenStackNetConfig CR 检索为 control plane 保留的 IP 地址：

```
spec:
  ...
  reservations:
    controlplane:
      ipReservations:
        ctlplane: 172.22.0.110
        external: 10.0.0.10
        internal_api: 172.17.0.10
        storage: 172.18.0.10
        storage_mgmt: 172.19.0.10
```

3. 打开 Web 浏览器。

4. 在 URL 字段中输入 control plane 的 IP 地址。

5. 使用您的用户名和密码登录控制面板。

第 10 章 使用 DIRECTOR OPERATOR 扩展 COMPUTE 节点

如果 overcloud 需要更多或较少的计算资源，您可以根据您的要求扩展 Compute 节点的数量。

10.1. 使用 DIRECTOR OPERATOR 将 COMPUTE 节点添加到 OVERCLOUD

要添加更多 Compute 节点到 overcloud，您必须增加 compute OpenStackBaremetalSet 资源的节点数。置备新节点时，您可以创建一个新的 OpenStackConfigGenerator 资源来生成一组新的 Ansible playbook，然后使用 OpenStackConfigVersion 创建或更新 OpenStackDeploy 对象，以将 Ansible 配置重新应用到 overcloud。

流程

1. 检查您在 openshift-machine-api 命名空间中有足够的主机处于 ready 状态：

```
$ oc get baremetalhosts -n openshift-machine-api
```

有关管理裸机主机的更多信息，请参阅 管理裸机主机。

2. 增加 compute OpenStackBaremetalSet 资源的 count 参数：

```
$ oc patch openstackbaremetalset compute --type=merge --patch '{"spec":{"count":3}}' -n openstack
```

OpenStackBaremetalSet 资源使用 Red Hat Enterprise Linux 基础操作系统自动置备新节点。

3. 等待置备过程完成。定期检查节点以确定节点的就绪情况：

```
$ oc get baremetalhosts -n openshift-machine-api
$ oc get openstackbaremetalset
```

4. 使用 OpenStackConfigGenerator 生成 Ansible playbook，并应用 overcloud 配置。如需更多信息，请参阅使用 director Operator 配置和部署 overcloud。

其他资源

- 管理裸机主机

10.2. 使用 DIRECTOR OPERATOR 从 OVERCLOUD 中删除 COMPUTE 节点

要从 overcloud 中删除 Compute 节点，您必须禁用 Compute 节点，将其标记为删除，并减少 compute OpenStackBaremetalSet 资源的节点数。

注意

如果您使用同一角色中的新节点扩展 overcloud，节点将重复利用主机名，从最低 ID 后缀和对应的 IP 保留开始。

先决条件

- Compute 节点上的工作负载已迁移到其他 Compute 节点。如需更多信息，请参阅在 Compute 节点间迁移虚拟机实例。

流程

1. 访问 openstackclient 的远程 shell：

```
$ oc rsh -n openstack openstackclient
```

2. 识别您要删除的 Compute 节点：

```
$ openstack compute service list
```

3. 禁用节点上的 Compute 服务，以防止节点调度新实例：

```
$ openstack compute service set <hostname> nova-compute --disable
```

4. 注解裸机节点，以防止 Metal^3 启动节点：

```
$ oc annotate baremetalhost <node> baremetalhost.metal3.io/detached=true
$ oc logs --since=1h <metal3-pod> metal3-baremetal-operator | grep -i detach
$ oc get baremetalhost <node> -o json | jq .status.operationalStatus
"detached"
```

- 将 <node > 替换为 BareMetalHost 资源的名称。

- 将 <metal3-pod > 替换为 metal3 pod 的名称。

5. 以 root 用户身份登录 Compute 节点，并关闭裸机节点：

```
[root@compute-0 ~]# shutdown -h now
```

如果 Compute 节点无法访问，请完成以下步骤：

a. 以 root 用户身份登录 Controller 节点。

b. 如果启用了 Instance HA，请禁用 Compute 节点的 STONITH 设备：

```
[root@controller-0 ~]# pcs stonith disable <stonith_resource_name>
```

- 将 <stonith_resource_name > 替换为与节点对应的 STONITH 资源的名称。资源名称使用 <resource_agent>-<host_mac> 格式。您可以在 fence.yaml 文件的 FencingConfig 部分查找资源代理和主机 MAC 地址。

c. 使用 IPMI 关闭裸机节点。如需更多信息，请参阅您的硬件厂商文档。

6. 检索与您要删除的节点对应的 BareMetalHost 资源：

```
$ oc get openstackbaremetalset compute -o json | jq '.status.baremetalHosts | to_entries[] | "\
(.key) => \(.value | .hostRef)"'
"compute-0, openshift-worker-3"
"compute-1, openshift-worker-4"
```

7. 要在 OpenStackBaremetalSet 资源中将 annotatedForDeletion 参数的状态更改为 true，请使用 osp-director.openstack.org/delete-host=true 注解 BareMetalHost 资源：

```
$ oc annotate -n openshift-machine-api bmh/openshift-worker-3 osp-
director.openstack.org/delete-host=true --overwrite
```

8. **可选：确认 OpenStackBaremetalSet 资源中 annotatedForDeletion 状态已更改为 true ：**

```
$ oc get openstackbaremetalset compute -o json -n openstack | jq .status
{
  "baremetalHosts": {
    "compute-0": {
      "annotatedForDeletion": true,
      "ctlplaneIP": "192.168.25.105/24",
      "hostRef": "openshift-worker-3",
      "hostname": "compute-0",
      "networkDataSecretName": "compute-cloudinit-networkdata-openshift-worker-3",
      "provisioningState": "provisioned",
      "userDataSecretName": "compute-cloudinit-userdata-openshift-worker-3"
    },
    "compute-1": {
      "annotatedForDeletion": false,
      "ctlplaneIP": "192.168.25.106/24",
      "hostRef": "openshift-worker-4",
      "hostname": "compute-1",
      "networkDataSecretName": "compute-cloudinit-networkdata-openshift-worker-4",
      "provisioningState": "provisioned",
      "userDataSecretName": "compute-cloudinit-userdata-openshift-worker-4"
    }
  },
  "provisioningStatus": {
    "readyCount": 2,
    "reason": "All requested BaremetalHosts have been provisioned",
    "state": "provisioned"
  }
}
```

9. **减少 compute OpenStackBaremetalSet 资源的 count 参数：**

```
$ oc patch openstackbaremetalset compute --type=merge --patch '{"spec":{"count":1}}' -n
openstack
```

当您减少 OpenStackBaremetalSet 资源的资源数时，您可以触发对应的控制器来处理资源删除，这会导致以下操作：

- **director Operator 从已删除节点的 OpenStackIPSet 和 OpenStackNetConfig 中删除对应的 IP 保留。**
- **director Operator 将 OpenStackNet 资源中的 IP 保留条目标记为已删除。**

```
$ oc get osnet ctlplane -o json -n openstack | jq .reservations
```

```
{
  "compute-0": {
    "deleted": true,
    "ip": "172.22.0.140"
  },
  "compute-1": {
    "deleted": false,
    "ip": "172.22.0.100"
  },
  "controller-0": {
    "deleted": false,
    "ip": "172.22.0.120"
  },
  "controlplane": {
    "deleted": false,
    "ip": "172.22.0.110"
  },
  "openstackclient-0": {
    "deleted": false,
    "ip": "172.22.0.251"
  }
```

10. **可选：要使已删除的 OpenStackBaremetalSet 资源的 IP 保留可供其他角色使用，请在 OpenStackNetConfig 对象中将 spec.preserveReservations 参数的值设置为 false。**

11. **访问 openstackclient 的远程 shell：**

```
$ oc rsh openstackclient -n openstack
```

12. **从 overcloud 中删除 Compute 服务条目：**

```
$ openstack compute service list
$ openstack compute service delete <service-id>
```

13. **检查 overcloud 中的 Compute 网络代理条目，如果存在它们，则将其删除：**

```
$ openstack network agent list
$ for AGENT in $(openstack network agent list --host <scaled-down-node> -c ID -f value) ;
do openstack network agent delete $AGENT ; done
```

14. **从 openstackclient 退出：**

```
$ exit
```

第 11 章 使用 DIRECTOR OPERATOR 执行 RHOSP OVERCLOUD 的次要更新

更新 openstackclient pod 后，通过运行 overcloud 和容器镜像准备部署、更新节点并运行 overcloud 更新聚合部署来更新 overcloud。在次版本更新过程中，提供了 control plane API。

Red Hat OpenStack Platform (RHOSP)环境的次要更新涉及更新 overcloud 节点上的 RPM 软件包和容器。您可能还需要更新某些服务的配置。data plane 和 control plane 在次版本更新过程中被完全支持。您必须完成以下步骤来更新 RHOSP 环境：

1. 为次要更新准备您的 RHOSP 环境。
2. 可选：更新 ovn-controller 容器。
3. 更新包含 Pacemaker 服务的 Controller 节点和可组合节点。
4. 更新 Compute 节点。
5. 更新 Red Hat Ceph Storage 节点。
6. 更新 Red Hat Ceph Storage 集群。
7. 重新引导 overcloud 节点。

先决条件

- 您有 RHOSP 部署的备份。如需更多信息，请参阅 备份和恢复 director Operator 部署 overcloud。

11.1. 准备 DIRECTOR OPERATOR 以进行次要更新

要准备 Red Hat OpenStack Platform (RHOSP)环境以使用 director Operator (OSPdO)执行次要更新，请完成以下任务：

1. 将 RHOSP 环境锁定到 Red Hat Enterprise Linux (RHEL)发行版本。

2. 更新 RHOSP 软件仓库。

3. 更新容器镜像准备文件。

4. 在 overcloud 中禁用隔离。

11.1.1. 将 RHOSP 环境锁定到 RHEL 发行版本

Red Hat OpenStack Platform (RHOSP) 17.1 支持 Red Hat Enterprise Linux (RHEL) 9.2。在执行更新前，将 overcloud 存储库锁定到 RHEL 9.2 版本，以避免将操作系统升级到较新的次版本。

流程

1. 将 overcloud 订阅管理环境文件 rhsm.yaml 复制到 openstackclient ：

```
$ oc cp rhsm.yaml openstackclient:/home/cloud-admin/rhsm.yaml
```

2. 访问 openstackclient pod 的远程 shell：

```
$ oc rsh openstackclient
```

3. 打开 rhsm.yaml 文件，检查订阅管理配置是否包含 rhsm_release 参数。如果没有 rhsm_release 参数，请添加它并将其设置为 9.2 ：

```
parameter_defaults:
  RhsmVars:
    …
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
    rhsm_pool_ids: "1a85f9223e3d5e43013e3d6e8ff506fd"
    rhsm_method: "portal"
    rhsm_release: "9.2"
```

4. 保存 **rhsm.yaml** 文件。

5. 创建名为 **set_release.yaml** 的 **playbook**，其中包含将操作系统版本锁定到所有节点上的 **RHEL 9.2** 的任务：

```
- hosts: all
  gather_facts: false
  tasks:
    - name: set release to 9.2
      command: subscription-manager release --set=9.2
      become: true
```

6. 在 **openstackclient pod** 上运行 **set_release.yaml playbook:**

```
$ ansible-playbook -i /home/cloud-admin/ctlplane-ansible-inventory /home/cloud-
admin/set_release.yaml --limit Controller,Compute
```

使用 **--limit** 选项将内容应用到所有 **RHOSP** 节点。不要针对 **Red Hat Ceph Storage** 节点运行此 **playbook**，因为您可能具有这些节点的不同订阅。

注意

要手动将节点锁定到版本，请登录到节点并运行 **subscription-manager release** 命令：

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

7. 退出 **openstackclient pod** 的远程 **shell**：

```
$ exit
```

11.1.2. 更新 RHOSP 软件仓库

更新您的软件仓库以使用 **Red Hat OpenStack Platform (RHOSP) 17.1**。

流程

1. 打开 **rhsm.yaml** 文件，并将 **rhsm_repos** 参数更新为正确的存储库版本：

```
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - rhel-9-for-x86_64-baseos-eus-rpms
      - rhel-9-for-x86_64-appstream-eus-rpms
      - rhel-9-for-x86_64-highavailability-eus-rpms
      - openstack-17.1-for-rhel-9-x86_64-rpms
      - fast-datapath-for-rhel-9-x86_64-rpms
```

2. 保存 **rhsm.yaml** 文件。

3. 访问 **openstackclient pod** 的远程 **shell**：

```
$ oc rsh openstackclient
```

4. 创建名为 **update_rhosp_repos.yaml** 的 **playbook**，其中包含将存储库在所有节点上设置为 **RHOSP 17.1** 的任务：

```
- hosts: all
  gather_facts: false
  tasks:
    - name: change osp repos
      command: subscription-manager repos --enable=openstack-17.1-for-rhel-9-x86_64-rpms
      become: true
```

5. 在 **openstackclient pod** 上运行 **update_rhosp_repos.yaml playbook:**

```
$ ansible-playbook -i /home/cloud-admin/ctlplane-ansible-inventory /home/cloud-
admin/update_rhosp_repos.yaml --limit Controller,Compute
```

使用 **--limit** 选项将内容应用到所有 **RHOSP** 节点。不要针对 **Red Hat Ceph Storage** 节点运行此 **playbook**，因为它们使用不同的订阅。

6. 创建名为 **update_ceph_repos.yaml** 的 **playbook**，其中包含在所有 **Red Hat Ceph Storage** 节点上将存储库设置为 **RHOSP 17.1** 的任务：

```
- hosts: all
  gather_facts: false
  tasks:
```

```
    - name: change ceph repos
      command: subscription-manager repos --enable=openstack-17.1-deployment-tools-for-
rhel-9-x86_64-rpms
      become: true
```

7. **在 openstackclient pod 上运行 update_ceph_repos.yaml playbook:**

```
$ ansible-playbook -i /home/cloud-admin/ctlplane-ansible-inventory /home/cloud-
admin/update_ceph_repos.yaml --limit CephStorage
```

使用 --limit 选项将内容应用到 Red Hat Ceph Storage 节点。

8. **退出 openstackclient pod 的远程 shell：**

```
$ exit
```

11.1.3. 更新容器镜像准备文件

容器准备文件是包含 ContainerImagePrepare 参数的文件。您可以使用此文件定义用于获取 overcloud 的容器镜像的规则。

在更新环境前，请检查该文件以确保获取正确的镜像版本。

流程

1. **编辑容器准备文件。此文件的默认名称为 containers-prepare-parameter.yaml。**

2. **确保每个规则集的 tag 参数设置为 17.1 ：**

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

注意

如果您不想将特定的标签用于更新，如 17.1 或 17.1.1，请删除 tag 键-值对，并只指定 tag_from_label。tag_from_label 标签使用安装的 Red Hat OpenStack Platform (RHOSP)版本来确定标签的值，以作为更新过程的一部分。

3. 保存 containers-prepare-parameter.yaml 文件。

11.1.4. 在 overcloud 中禁用隔离

在更新 overcloud 之前，请确保禁用隔离。

如果在 Controller 节点更新过程中在您的环境中部署隔离，overcloud 可能会检测到某些节点被禁用并尝试隔离操作，这可能会导致意外的结果。

如果您在 overcloud 中启用了隔离功能，则必须在更新期间临时禁用隔离。

流程

1. 访问 openstackclient pod 的远程 shell：

```
$ oc rsh openstackclient
```

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

```
$ ssh <controller-0.ctlplane> "sudo pcs property set stonith-enabled=false"
```

- 将 <controller-0.ctlplane > 替换为 Controller 节点的名称。

3. 退出 openstackclient pod 的远程 shell：

```
$ exit
```

其它资源

- 使用 STONITH 隔离 Controller 节点

11.2. 为 DIRECTOR OPERATOR 运行 OVERCLOUD 更新准备

为准备 overcloud 以进行更新过程，请生成更新准备配置，该配置会创建更新的 ansible playbook 并为节点做好更新准备。

流程

1. 创建名为 **osconfiggenerator-update-prepare.yaml** 的 **OpenStackConfigGenerator** 资源：

```
$ cat <<EOF > osconfiggenerator-update-prepare.yaml
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackConfigGenerator
metadata:
  name: "update"
  namespace: openstack
spec:
  gitSecret: git-secret
  enableFencing: false
  heatEnvs:
    - lifecycle/update-prepare.yaml
  heatEnvConfigMap: heat-env-config-update
  tarballConfigMap: tripleo-tarball-config-update
EOF
```

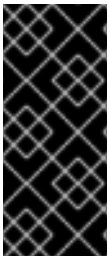
2. 应用配置：

```
$ oc apply -f osconfiggenerator-update-prepare.yaml
```

3. 等待更新准备过程完成。

11.3. 更新所有 OVERCLOUD 服务器上的 OVN-CONTROLLER 容器

如果您使用 Modular Layer 2 Open Virtual Network 机制驱动程序(ML2/OVN)部署 overcloud，请将 ovn-controller 容器更新至最新的 Red Hat OpenStack Platform (RHOSP) 17.1 版本。更新发生在每个运行 ovn-controller 容器的 overcloud 服务器上。



重要

以下流程在更新 Controller 节点上的 ovn-northd 服务前更新 Compute 节点上的 ovn-controller 容器。如果在此流程前意外更新 ovn-northd 服务，您可能无法访问虚拟机实例或创建新实例或虚拟网络。以下流程恢复连接。

流程

1. **创建名为 osdeploy-ovn-update.yaml 的 OpenStackDeploy 自定义资源(CR)：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: ovn-update
spec:
  configVersion: <config_version>
  configGenerator: update
  mode: update
  advancedSettings:
    tags:
      - ovn
```

2. **应用更新的配置：**

```
$ oc apply -f osdeploy-ovn-update.yaml
```

3. **等待 ovn-controller 容器更新完成。**

11.4. 更新所有 CONTROLLER 节点

将所有 Controller 节点更新至最新的 Red Hat OpenStack Platform (RHOSP) 17.1 版本。

流程

1. **创建名为 osdeploy-controller-update.yaml 的 OpenStackDeploy 自定义资源(CR)：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: controller-update
spec:
  configVersion: <config_version>
```

```
  configGenerator: update
  mode: update
  advancedSettings:
    limit: Controller
```

2. **应用更新的配置：**

```
$ oc apply -f osdeploy-controller-update.yaml
```

3. **等待 Controller 节点更新完成。**

11.5. 更新所有 COMPUTE 节点

将所有 Compute 节点更新至最新的 Red Hat OpenStack Platform (RHOSP) 17.1 版本。要更新 Compute 节点，请使用 limit: Compute 选项创建一个 OpenStackDeploy 自定义资源(CR)，以仅将操作限制为 Compute 节点。

流程

1. **创建名为 osdeploy-compute-update.yaml 的 OpenStackDeploy CR：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: compute-update
spec:
  configVersion: <config_version>
  configGenerator: update
  mode: update
  advancedSettings:
    limit: Compute
```

2. **应用更新的配置：**

```
$ oc apply -f osdeploy-compute-update.yaml
```

3. **等待 Compute 节点更新完成。**

11.6. 更新所有 HCI COMPUTE 节点

将超融合基础架构(HCI) Compute 节点更新至最新的 Red Hat OpenStack Platform (RHOSP) 17.1 版本。要更新 HCI Compute 节点，请使用 limit: ComputeHCI 选项创建一个 OpenStackDeploy 自定义资源(CR)，将操作限制为 HCI 节点。您还必须使用 mode: external-update 和 tags: ["ceph"] 选项创建一个 OpenStackDeploy CR，以执行到容器化 Red Hat Ceph Storage 4 集群的更新。

流程

1. **创建名为 osdeploy-computehci-update.yaml 的 OpenStackDeploy CR：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: computehci-update
spec:
  configVersion: <config_version>
  configGenerator: update
  mode: update
  advancedSettings:
    limit: ComputeHCI
```

2. **应用更新的配置：**

```
$ oc apply -f osdeploy-computehci-update.yaml
```

3. **等待 ComputeHCI 节点更新完成。**

4. **创建名为 osdeploy-ceph-update.yaml 的 OpenStackDeploy CR：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: ceph-update
spec:
  configVersion: <config_version>
  configGenerator: update
  mode: external-update
  advancedSettings:
    tags:
      - ceph
```

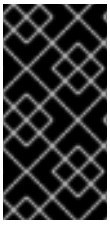
5. **应用更新的配置：**

```
$ oc apply -f osdeploy-ceph-update.yaml
```

6. 等待 Red Hat Ceph Storage 节点更新完成。

11.7. 更新所有 RED HAT CEPH STORAGE 节点

将 Red Hat Ceph Storage 节点更新至最新的 Red Hat OpenStack Platform (RHOSP) 17.1 版本。



重要

RHOSP 17.1 在 RHEL 9.2 上被支持。但是，映射到 CephStorage 角色的主机会更新到最新的主 RHEL 版本。如需更多信息，请参阅 Red Hat Ceph Storage: 支持的配置。

流程

1. 创建名为 **osdeploy-cephstorage-update.yaml** 的 **OpenStackDeploy** 自定义资源(CR)：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: cephstorage-update
spec:
  configVersion: <config_version>
  configGenerator: update
  mode: update
  advancedSettings:
    limit: CephStorage
```

2. 应用更新的配置：

```
$ oc apply -f osdeploy-cephstorage-update.yaml
```

3. 等待 Red Hat Ceph Storage 节点更新完成。

4. 创建名为 **osdeploy-ceph-update.yaml** 的 **OpenStackDeploy** CR：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: ceph-update
spec:
  configVersion: <config_version>
```

```
  configGenerator: update
  mode: external-update
  advancedSettings:
    tags:
      - ceph
```

5. **应用更新的配置：**

```
$ oc apply -f osdeploy-ceph-update.yaml
```

6. **等待 Red Hat Ceph Storage 节点更新完成。**

11.8. 更新 RED HAT CEPH STORAGE 集群

使用 cephadm Orchestrator 将 director 部署的 Red Hat Ceph Storage 集群更新至与 Red Hat OpenStack Platform (RHOSP) 17.1 兼容的最新版本。

流程

1. **访问 openstackclient pod 的远程 shell：**

```
$ oc rsh openstackclient
```

2. **登录到 Controller 节点：**

```
$ ssh <controller-0.ctlplane>
```

- **将 <controller-0.ctlplane > 替换为 Controller 节点的名称。**

3. **登录 cephadm shell：**

```
[cloud-admin@controller-0 ~]$ sudo cephadm shell
```

4. **使用 cephadm 升级 Red Hat Ceph Storage 集群。有关更多信息，请参阅 Red Hat Ceph Storage *6 升级指南中的使用 cephadm 升级 Red Hat Ceph Storage* 集群。**

5. 退出 openstackclient pod 的远程 shell：

```
$ exit
```

11.9. 执行在线数据库更新

有些 overcloud 组件需要在线更新或迁移其数据库表。在线数据库更新适用于以下组件：

- Block Storage 服务 (cinder)
- 计算服务(nova)

流程

1. 创建名为 osdeploy-online-migration.yaml 的 OpenStackDeploy 自定义资源(CR)：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackDeploy
metadata:
  name: online-migration
spec:
  configVersion: <config_version>
  configGenerator: update
  mode: external-update
  advancedSettings:
    tags:
      - online_upgrade
```

2. 应用更新的配置：

```
$ oc apply -f osdeploy-online-migration.yaml
```

11.10. 在 OVERCLOUD 中重新启用隔离

要更新到最新的 Red Hat OpenStack Platform (RHOSP) 17.1，您必须在 overcloud 中重新启用隔离。

流程

1. 访问 **openstackclient pod** 的远程 **shell**：

```
$ oc rsh openstackclient
```

2. 登录到 **Controller** 节点并运行 **Pacemaker** 命令启用隔离：

```
$ ssh <controller-0.ctlplane> "sudo pcs property set stonith-enabled=true"
```

- 将 **<controller-0.ctlplane >** 替换为 **Controller** 节点的名称。

3. 退出 **openstackclient pod** 的远程 **shell**：

```
$ exit
```

11.11. 重新引导 OVERCLOUD

执行小的 **Red Hat OpenStack Platform (RHOSP)**更新至最新的 **17.1** 版本后，重启您的 **overcloud**。重启会使用任何关联的内核、系统级和容器组件更新刷新节点。这些更新提供了性能和安全优势。计划停机时间来执行重启过程。

使用以下指导了解如何重新引导不同的节点类型：

- 如果重新引导一个角色中的所有节点，请单独重新引导每个节点。如果您同时重新引导角色中的所有节点，则重启操作过程中可能会发生服务停机时间。

- 按照以下顺序在节点上完成重启步骤：

  1. 第 11.11.1 节 “重新引导 Controller 和可组合节点”

  2. 第 11.11.2 节 “重新引导 Ceph Storage (OSD) 集群”

  3. 第 11.11.3 节 “重新引导 Compute 节点”

11.11.1. 重新引导 Controller 和可组合节点

根据可组合角色重新引导 Controller 节点和独立节点，并排除 Compute 节点和 Ceph Storage 节点。

流程

1. 登录您要重新引导的节点。
2. 可选：如果节点使用 Pacemaker 资源，请停止集群：

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

3. 重新引导节点：

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

4. 稍等片刻，直到节点启动。

验证

1. 验证服务是否已启用。
   a. 如果该节点使用 Pacemaker 服务，请检查该节点是否已重新加入集群：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs status
```

   b. 如果该节点使用 Systemd 服务，请检查是否所有服务都已启用：

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

   c. 如果该节点使用容器化服务，则检查节点上的所有容器是否已激活：

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo podman ps
```

11.11.2. 重新引导 Ceph Storage (OSD) 集群

完成以下步骤以重新引导 Ceph Storage (OSD) 节点集群。

先决条件

- 在运行 ceph-mon 服务的 Ceph monitor 或 Controller 节点上，检查 Red Hat Ceph Storage 集群状态是否健康，pg 状态为 active+clean ：

```
$ sudo cephadm -- shell ceph status
```

如果 Ceph 集群处于健康状态，它会返回 HEALTH_OK 状态。

如果 Ceph 集群状态不健康，它将返回 HEALTH_WARN 或 HEALTH_ERR 的状态。有关故障排除指南，请参阅 Red Hat Ceph Storage 5 故障排除指南 或 Red Hat Ceph Storage 6 故障排除指南。

流程

1. 登录到运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点，并临时禁用 Ceph Storage 集群重新平衡：

```
$ sudo cephadm shell -- ceph osd set noout
$ sudo cephadm shell -- ceph osd set norebalance
```

注意

如果您有多堆栈或分布式计算节点(DCN)架构，您必须在设置 noout 和 norebalance 标志时指定 Ceph 集群名称。例如： sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring。

2. 选择第一个要重新引导的 Ceph Storage 节点并登录到该节点。

3. 重新引导节点：

```
$ sudo reboot
```

4. 稍等片刻，直到节点启动。

5. 登录节点并检查 Ceph 集群状态：

```
$ sudo cephadm -- shell ceph status
```

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

6. 注销节点，重新引导下一个节点，并检查其状态。重复此过程，直到您已重新引导所有 Ceph Storage 节点。

7. 完成后，登录到运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点，并启用 Ceph 集群重新平衡：

```
$ sudo cephadm shell -- ceph osd unset noout
$ sudo cephadm shell -- ceph osd unset norebalance
```

注意

如果您有多堆栈或分布式计算节点(DCN)架构，您必须在取消设置 noout 和 norebalance 标志时指定 Ceph 集群名称。例如： sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring

8. 执行最后的状态检查，确认集群报告 HEALTH_OK：

```
$ sudo cephadm shell ceph status
```

11.11.3. 重新引导 Compute 节点

为确保 Red Hat OpenStack Platform 环境中 实例的停机时间最少，迁移实例工作流 概述了从您要重新引导的 Compute 节点迁移实例的步骤。

迁移实例工作流

1. 决定是否在重新引导节点前将实例迁移到另一个 Compute 节点。

2. 选择并禁用您要重新引导的 Compute 节点，使其不置备新实例。

3. 将实例迁移到另一个 Compute 节点中。

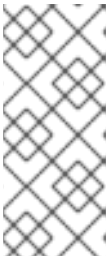
4. 重新引导空的 Compute 节点。

5. 启用空的 Compute 节点。

先决条件

- 重启 Compute 节点之前，必须决定是否在节点重启过程中将实例迁移到另一个 Compute 节点。

  查看在 Compute 节点之间迁移虚拟机实例时可能会遇到的迁移限制列表。如需更多信息，请参阅*为实例创建配置 Compute Service* 中的迁移限制。

  注意

  如果您有 Multi-RHEL 环境，并且希望将虚拟机从运行 RHEL 9.2 的 Compute 节点迁移到运行 RHEL 8.4 的 Compute 节点，则只支持冷迁移。有关冷迁移的更多信息，请参阅配置 *计算服务* 以进行实例创建中的冷迁移实例。

- 如果您无法迁移实例，则可设置以下核心模板参数以在 Compute 节点重启后控制实例的状态：

  NovaResumeGuestsStateOnHostBoot

  确定重新引导后是否将实例返回 Compute 节点上的相同状态。设为 False 时，实例保持关闭，必须手动启动。默认值为 False。

  NovaResumeGuestsShutdownTimeout

  重启前等待实例被关闭的时间（以秒为单位）。建议不要将此值设置为 0。默认值为 300。

  有关 overcloud 参数及其用法的更多信息，请参阅 Overcloud 参数。

流程

1. **以 stack 用户身份登录 undercloud。**

2. **检索 Compute 节点列表，以识别您要重新引导的节点的主机名：**

```
(undercloud)$ source ~/overcloudrc
(overcloud)$ openstack compute service list
```

识别您要重新引导的 Compute 节点的主机名。

3. **在您要重新引导的 Compute 节点上禁用 Compute 服务：**

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

- **将 <hostname > 替换为 Compute 节点的主机名。**

4. **列出 Compute 节点上的所有实例：**

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

5. **可选： 要将实例迁移到另一个 Compute 节点，请完成以下步骤：**

   a. **如果您决定将实例迁移至另一个 Compute 节点，则使用以下命令之一：**

   - **要将实例迁移到其他主机，请运行以下命令：**

```
(overcloud) $ openstack server migrate <instance_id> --live <target_host> --wait
```

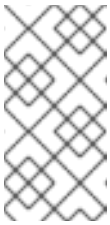
     - **将 <instance_id& gt; 替换为您的实例 ID。**
     - **将 <target_host > 替换为您要将实例迁移到的主机。**

- 让 **nova-scheduler** 自动选择目标主机：

```
(overcloud) $ nova live-migration <instance_id>
```

- 一次性实时迁移所有实例：

```
$ nova host-evacuate-live <hostname>
```



注意

nova 命令可能会引发一些弃用警告，这些警告信息可以被安全忽略。

b. 稍等片刻，直至迁移完成。

c. 确认迁移成功完成：

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

d. 继续迁移实例，直到 **Compute** 节点上没有保留任何实例。

6. 登录到 **Compute** 节点并重启节点：

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

7. 稍等片刻，直到节点启动。

8. 重新启用 **Compute** 节点：

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

9. 确认是否已启用 **Compute** 节点：

-

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

11.11.4. 在 overcloud 更新后验证 RHOSP

更新 Red Hat OpenStack Platform (RHOSP)环境后，使用 tripleo-validations playbook 验证 overcloud。

有关验证的更多信息， *请参阅使用 director 安装和管理 Red Hat OpenStack Platform 中的使用验证框架*。

流程

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

```
$ source ~/stackrc
```

3. 运行验证：

```
$ validation run -i ~/overcloud-deploy/<stack>/tripleo-ansible-inventory.yaml --group post-update
```

- 将 <stack> 替换为堆栈的名称。

验证

1. 要查看验证报告的结果， *请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 中的 查看验证历史记录。

注意

忽略报告 No host 与唯一返回错误匹配的 FAILED 验证。此错误意味着您没有与验证主机组匹配的主机，这可能是预期的。FAILED 验证不会阻止您使用更新的 RHOSP 环境。但是，FAILED 验证可能会指示您的环境出现问题。

第 12 章 使用 DIRECTOR OPERATOR 为公共端点部署 TLS

使用 TLS 部署 overcloud，为 director Operator (OSPdO)创建公共端点 IP 或 DNS 名称。

先决条件

- 您已在正常运行的 Red Hat OpenShift Container Platform (RHOCP)集群上安装了 OSPdO。
- 您已在工作站上安装了 oc 命令行工具。
- 您已创建了证书颁发机构、密钥和证书。如需更多信息，请参阅在 overcloud 公共端点中启用 SSL/TLS。

12.1. TLS 用于公共端点 IP 地址

要引用公共端点 IP 地址，请通过创建 ConfigMap 资源来存储 CA 证书，然后在 OpenStackControlPlane 资源中引用该 ConfigMap 资源，将 CA 证书添加到 openstackclient pod。

流程

1. 创建 ConfigMap 资源以存储 CA 证书：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cacerts
  namespace: openstack
data:
 local_CA: |
    -----BEGIN CERTIFICATE-----
    …
   -----END CERTIFICATE-----
  another_CA: |
    -----BEGIN CERTIFICATE-----
    …
   -----END CERTIFICATE-----
```

2. 创建 OpenStackControlPlane 资源并引用 ConfigMap 资源：

```
apiVersion: osp-director.openstack.org/v1beta2
kind: OpenStackControlPlane
metadata:
  name: <overcloud>
  namespace: openstack
spec:
  caConfigMap: cacerts
```

- **将 <overcloud > 替换为 overcloud control plane 的名称。**

3. **在 ~/custom_environment_files 目录中创建一个名为 tls-certs.yaml 的文件，它使用 SSLCertificate、SSLIntermediateCertificate、SSLKey 和 CAMap 参数为部署指定生成的证书。**

4. **更新 heatEnvConfigMap 以添加 tls-certs.yaml 文件：**

```
$ oc create configmap -n openstack heat-env-config --from-file=~/custom_environment_files/
--dry-run=client -o yaml | oc apply -f -
```

5. **创建 OpenStackConfigGenerator 资源，并添加所需的 heatEnvs 配置文件来为公共端点 IP 配置 TLS：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackConfigGenerator
…
spec:
  …
  heatEnvs:
    - ssl/tls-endpoints-public-ip.yaml
    - ssl/enable-tls.yaml
  …
  heatEnvConfigMap: heat-env-config
  tarballConfigMap: tripleo-tarball-config
```

6. **使用 OpenStackConfigGenerator 生成 Ansible playbook，并应用 overcloud 配置。如需更多信息，请参阅使用 director Operator 配置和部署 overcloud。**

12.2. TLS 用于公共端点 DNS 名称

要引用公共端点 DNS 名称，请通过创建 ConfigMap 资源来存储 CA 证书，然后将您的 CA 证书添加到 openstackclient pod 中，然后在 OpenStackControlPlane 资源中引用该 ConfigMap 资源。

流程

流程

1. **创建 ConfigMap 资源以存储 CA 证书：**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cacerts
  namespace: openstack
data:
 local_CA: |
    -----BEGIN CERTIFICATE-----
    …
   -----END CERTIFICATE-----
  another_CA: |
    -----BEGIN CERTIFICATE-----
    …
   -----END CERTIFICATE-----
```

2. **创建 OpenStackControlPlane 资源并引用 ConfigMap 资源：**

```
apiVersion: osp-director.openstack.org/v1beta2
kind: OpenStackControlPlane
metadata:
  name: <overcloud>
  namespace: openstack
spec:
  caConfigMap: cacerts
```

- **将 <overcloud > 替换为 overcloud control plane 的名称。**

3. **在 ~/custom_environment_files 目录中创建一个名为 tls-certs.yaml 的文件，它使用 SSLCertificate、SSLIntermediateCertificate、SSLKey 和 CAMap 参数为部署指定生成的证书。**

4. **更新 heatEnvConfigMap 以添加 tls-certs.yaml 文件：**

```
$ oc create configmap -n openstack heat-env-config --from-file=~/custom_environment_files/
--dry-run=client -o yaml | oc apply -f -
```

5. **创建 OpenStackConfigGenerator 资源并添加所需的 heatEnvs 配置文件，以便为公共端点 DNS 名称配置 TLS：**

```
apiVersion: osp-director.openstack.org/v1beta1
```

```
kind: OpenStackConfigGenerator
…
spec:
  …
  heatEnvs:
    - ssl/tls-endpoints-public-dns.yaml
    - ssl/enable-tls.yaml
  …
  heatEnvConfigMap: heat-env-config
  tarballConfigMap: tripleo-tarball-config
```

6. **使用 OpenStackConfigGenerator 生成 Ansible playbook，并应用 overcloud 配置。如需更多信息，请参阅使用 director Operator 配置和部署 overcloud。**

第 13 章 使用 DIRECTOR OPERATOR 更改服务帐户密码

Red Hat OpenStack Platform (RHOSP)服务和它们使用的数据库通过其身份服务(keystone)凭证进行身份验证。Identity 服务在初始 RHOSP 部署过程过程中生成这些 RHOSP 密码。您可能需要定期更新密码以进行威胁缓解或安全合规。您可以使用 director Operator (OSPdO)的原生工具在部署 RHOSP 环境后更改许多生成的密码。

13.1. 使用 DIRECTOR OPERATOR 轮转 OVERCLOUD 服务帐户密码

您可以轮转与部署 Red Hat OpenStack Platform (RHOSP)环境的 director Operator (OSPdO)使用的 overcloud 服务帐户密码。

流程

1. 创建当前 tripleo-passwords secret 的备份：

```
$ oc get secret tripleo-passwords -n openstack -o yaml > tripleo-passwords_backup.yaml
```

2. 创建名为 tripleo-overcloud-passwords_preserve_list 的纯文本文件，以指定以下服务的密码不应轮转：

```
parameter_defaults
BarbicanSimpleCryptoKek
KeystoneCredential0
KeystoneCredential1
KeystoneFernetKey0
KeystoneFernetKey1
KeystoneFernetKeys
CephClientKey
CephClusterFSID
CephManilaClientKey
CephRgwKey
HeatAuthEncryptionKey
MysqlClustercheckPassword
MysqlMariabackupPassword
PacemakerRemoteAuthkey
PcsdPassword
```

如果有要保留密码的其他服务，您可以在此列表中添加其他服务。

3. 创建一个密码参数文件 tripleo-overcloud-passwords.yaml，它列出了不应修改的密码：

```
$ oc get secret tripleo-passwords -n openstack \
-o jsonpath='{.data.tripleo-overcloud-passwords\.yaml}' \
| base64 -d | grep -f ./tripleo-overcloud-passwords_preserve_list > tripleo-overcloud-passwords.yaml
```

4. **验证 tripleo-overcloud-passwords.yaml 文件包含您不想轮转的密码。**

5. **更新 tripleo-password secret：**

```
$ oc create secret generic tripleo-passwords -n openstack \
--from-file=./tripleo-overcloud-passwords.yaml \
--dry-run=client -o yaml | oc apply -f -
```

6. **创建 Ansible playbook 以使用 OpenStackConfigGenerator CRD 配置 overcloud。如需更多信息，请参阅使用 OpenStackConfigGenerator CRD 为 overcloud 配置创建 Ansible playbook。**

7. **应用更新的配置。如需更多信息，请参阅使用 director Operator 应用 overcloud 配置。**

验证

将机密中的新 NovaPassword 与 Controller 节点上现在安装的内容进行比较。

1. **从更新的 secret 获取密码：**

```
$ oc get secret tripleo-passwords -n openstack -o jsonpath='{.data.tripleo-overcloud-passwords\.yaml}' | base64 -d | grep NovaPassword
```

输出示例：

```
NovaPassword: hp4xpt7t2p79ktqjjnxpqwbp6
```

2. **检索在 Controller 节点上运行的 Compute 服务(nova)的密码：**

   a. **访问 openstackclient 远程 shell：**

```
$ oc rsh openstackclient -n openstack
```

b.

确保您位于主目录中：

```
$ cd
```

c.

检索 Compute 服务密码：

```
$ ansible -i /home/cloud-admin/ctlplane-ansible-inventory Controller -b -a "grep
^connection /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf"
```

输出示例：

```
172.22.0.120 | CHANGED | rc=0 >>
connection=mysql+pymysql://nova_api:hp4xpt7t2p79ktqjjnxpqwbp6@172.17.0.10/nova_api
?read_default_file=/etc/my.cnf.d/tripleo.cnf&read_default_group=tripleo
connection=mysql+pymysql://nova:hp4xpt7t2p79ktqjjnxpqwbp6@172.17.0.10/nova?
read_default_file=/etc/my.cnf.d/tripleo.cnf&read_default_group=tripleo
```

第 14 章 使用 DIRECTOR OPERATOR 部署带有 SPINE-LEAF 配置的节点

使用 spine-leaf 网络架构部署节点，在您的环境中复制广泛的网络拓扑。当前限制只允许一个 provisioning 网络用于 Metal3。

14.1. 创建或更新 OPENSTACKNETCONFIG 自定义资源以定义所有子网

定义 OpenStackNetConfig 自定义资源(CR)并指定 overcloud 网络的子网。Red Hat OpenStack Platform (RHOSP) director Operator (OSPdO)然后呈现配置并创建或更新网络拓扑。

先决条件

- 您已在正常运行的 Red Hat OpenShift Container Platform (RHOCP)集群上安装了 OSPdO。
- 您已在工作站上安装了 oc 命令行工具。

流程

1. 创建名为 openstacknetconfig.yaml 的配置文件：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackNetConfig
metadata:
  name: openstacknetconfig
spec:
  attachConfigurations:
    br-osp:
      nodeNetworkConfigurationPolicy:
        nodeSelector:
          node-role.kubernetes.io/worker: ""
        desiredState:
          interfaces:
          - bridge:
              options:
                stp:
                  enabled: false
              port:
              - name: enp7s0
            description: Linux bridge with enp7s0 as a port
            name: br-osp
            state: up
            type: linux-bridge
            mtu: 1500
```

```
  br-ex:
    nodeNetworkConfigurationPolicy:
      nodeSelector:
        node-role.kubernetes.io/worker: ""
      desiredState:
        interfaces:
        - bridge:
            options:
              stp:
                enabled: false
            port:
            - name: enp6s0
          description: Linux bridge with enp6s0 as a port
          name: br-ex
          state: up
          type: linux-bridge
          mtu: 1500
# optional DnsServers list
dnsServers:
- 192.168.25.1
# optional DnsSearchDomains list
dnsSearchDomains:
- osptest.test.metalkube.org
- some.other.domain
# DomainName of the OSP environment
domainName: osptest.test.metalkube.org
networks:
- name: Control
  nameLower: ctlplane
  subnets:
  - name: ctlplane
    ipv4:
      allocationEnd: 192.168.25.250
      allocationStart: 192.168.25.100
      cidr: 192.168.25.0/24
      gateway: 192.168.25.1
    attachConfiguration: br-osp
- name: InternalApi
  nameLower: internal_api
  mtu: 1350
  subnets:
  - name: internal_api
    ipv4:
      allocationEnd: 172.17.0.250
      allocationStart: 172.17.0.10
      cidr: 172.17.0.0/24
      routes:
      - destination: 172.17.1.0/24
        nexthop: 172.17.0.1
      - destination: 172.17.2.0/24
        nexthop: 172.17.0.1
    vlan: 20
    attachConfiguration: br-osp
  - name: internal_api_leaf1
    ipv4:
      allocationEnd: 172.17.1.250
```

```
      allocationStart: 172.17.1.10
      cidr: 172.17.1.0/24
      routes:
      - destination: 172.17.0.0/24
        nexthop: 172.17.1.1
      - destination: 172.17.2.0/24
        nexthop: 172.17.1.1
    vlan: 21
    attachConfiguration: br-osp
  - name: internal_api_leaf2
    ipv4:
      allocationEnd: 172.17.2.250
      allocationStart: 172.17.2.10
      cidr: 172.17.2.0/24
      routes:
      - destination: 172.17.1.0/24
        nexthop: 172.17.2.1
      - destination: 172.17.0.0/24
        nexthop: 172.17.2.1
    vlan: 22
    attachConfiguration: br-osp
- name: External
  nameLower: external
  subnets:
  - name: external
    ipv4:
      allocationEnd: 10.0.0.250
      allocationStart: 10.0.0.10
      cidr: 10.0.0.0/24
      gateway: 10.0.0.1
    attachConfiguration: br-ex
- name: Storage
  nameLower: storage
  mtu: 1350
  subnets:
  - name: storage
    ipv4:
      allocationEnd: 172.18.0.250
      allocationStart: 172.18.0.10
      cidr: 172.18.0.0/24
      routes:
      - destination: 172.18.1.0/24
        nexthop: 172.18.0.1
      - destination: 172.18.2.0/24
        nexthop: 172.18.0.1
    vlan: 30
    attachConfiguration: br-osp
  - name: storage_leaf1
    ipv4:
      allocationEnd: 172.18.1.250
      allocationStart: 172.18.1.10
      cidr: 172.18.1.0/24
      routes:
      - destination: 172.18.0.0/24
        nexthop: 172.18.1.1
      - destination: 172.18.2.0/24
```

```
        nexthop: 172.18.1.1
    vlan: 31
    attachConfiguration: br-osp
  - name: storage_leaf2
    ipv4:
      allocationEnd: 172.18.2.250
      allocationStart: 172.18.2.10
      cidr: 172.18.2.0/24
      routes:
      - destination: 172.18.0.0/24
        nexthop: 172.18.2.1
      - destination: 172.18.1.0/24
        nexthop: 172.18.2.1
    vlan: 32
    attachConfiguration: br-osp
- name: StorageMgmt
  nameLower: storage_mgmt
  mtu: 1350
  subnets:
  - name: storage_mgmt
    ipv4:
      allocationEnd: 172.19.0.250
      allocationStart: 172.19.0.10
      cidr: 172.19.0.0/24
      routes:
      - destination: 172.19.1.0/24
        nexthop: 172.19.0.1
      - destination: 172.19.2.0/24
        nexthop: 172.19.0.1
    vlan: 40
    attachConfiguration: br-osp
  - name: storage_mgmt_leaf1
    ipv4:
      allocationEnd: 172.19.1.250
      allocationStart: 172.19.1.10
      cidr: 172.19.1.0/24
      routes:
      - destination: 172.19.0.0/24
        nexthop: 172.19.1.1
      - destination: 172.19.2.0/24
        nexthop: 172.19.1.1
    vlan: 41
    attachConfiguration: br-osp
  - name: storage_mgmt_leaf2
    ipv4:
      allocationEnd: 172.19.2.250
      allocationStart: 172.19.2.10
      cidr: 172.19.2.0/24
      routes:
      - destination: 172.19.0.0/24
        nexthop: 172.19.2.1
      - destination: 172.19.1.0/24
        nexthop: 172.19.2.1
    vlan: 42
    attachConfiguration: br-osp
- name: Tenant
```

```
    nameLower: tenant
    vip: False
    mtu: 1350
    subnets:
    - name: tenant
      ipv4:
        allocationEnd: 172.20.0.250
        allocationStart: 172.20.0.10
        cidr: 172.20.0.0/24
        routes:
        - destination: 172.20.1.0/24
          nexthop: 172.20.0.1
        - destination: 172.20.2.0/24
          nexthop: 172.20.0.1
      vlan: 50
      attachConfiguration: br-osp
    - name: tenant_leaf1
      ipv4:
        allocationEnd: 172.20.1.250
        allocationStart: 172.20.1.10
        cidr: 172.20.1.0/24
        routes:
        - destination: 172.20.0.0/24
          nexthop: 172.20.1.1
        - destination: 172.20.2.0/24
          nexthop: 172.20.1.1
      vlan: 51
      attachConfiguration: br-osp
    - name: tenant_leaf2
      ipv4:
        allocationEnd: 172.20.2.250
        allocationStart: 172.20.2.10
        cidr: 172.20.2.0/24
        routes:
        - destination: 172.20.0.0/24
          nexthop: 172.20.2.1
        - destination: 172.20.1.0/24
          nexthop: 172.20.2.1
      vlan: 52
      attachConfiguration: br-osp
```

2. **创建内部 API 网络：**

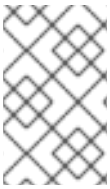
```
$ oc create -f openstacknetconfig.yaml -n openstack
```

3. **验证 OpenStackNetConfig 资源的资源和子资源是否已创建：**

```
$ oc get openstacknetconfig/openstacknetconfig -n openstack
$ oc get openstacknetattachment -n openstack
$ oc get openstacknet -n openstack
```

14.2. 将叶网络的角色添加到您的部署中

要将叶网络的角色添加到您的部署中，请更新 **roles_data.yaml** 配置文件。如果叶型网络角色具有不同的 **NIC** 配置，您可以为每个角色创建 **Ansible NIC** 模板，以配置 **spine-leaf** 网络，注册 **NIC** 模板并创建 **ConfigMap** 自定义资源。



注意

您必须使用 **roles_data.yaml** 作为文件名。

流程

1. 更新 **roles_data.yaml** 文件：

```
...
##############################################################################
####
# Role: ComputeLeaf1                                                         #
##############################################################################
####
- name: ComputeLeaf1
  description: |
    Basic ComputeLeaf1 Node role
  # Create external Neutron bridge (unset if using ML2/OVS without DVR)
  tags:
    - compute
    - external_bridge
  networks:
    InternalApi:
      subnet: internal_api_leaf1
    Tenant:
      subnet: tenant_leaf1
    Storage:
      subnet: storage_leaf1
  HostnameFormatDefault: '%stackname%-novacompute-leaf1-%index%'
...
##############################################################################
####
# Role: ComputeLeaf2                                                         #
##############################################################################
####
- name: ComputeLeaf2
  description: |
    Basic ComputeLeaf1 Node role
  # Create external Neutron bridge (unset if using ML2/OVS without DVR)
  tags:
    - compute
    - external_bridge
  networks:
```

```
    InternalApi:
      subnet: internal_api_leaf2
    Tenant:
      subnet: tenant_leaf2
    Storage:
      subnet: storage_leaf2
  HostnameFormatDefault: '%stackname%-novacompute-leaf2-%index%'
...
```

2. 为每个 Compute 角色创建一个 NIC 模板。如需 Ansible NIC 模板示例，请参阅 https://github.com/openstack/tripleo-ansible/tree/stable/wallaby/tripleo_ansible/roles/tripleo_network_config/templates。

3. 将新节点的 NIC 模板添加到环境文件中：

```
parameter_defaults:
  ComputeNetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
  ComputeLeaf1NetworkConfigTemplate: 'multiple_nics_vlans_dvr.j2'
  ComputeLeaf2NetworkConfigTemplate: 'multiple_nics_compute_leaf_2_vlans_dvr.j2'
```

4. 在 ~/custom_environment_files 目录中，将 roles_data.yaml 文件、环境文件和 NIC 模板归档到 tarball 中：

```
$ tar -cvzf custom-spine-leaf-config.tar.gz *.yaml
```

5. 创建 tripleo-tarball-config ConfigMap 资源：

```
$ oc create configmap tripleo-tarball-config --from-file=custom-spine-leaf-config.tar.gz -n openstack
```

14.3. 使用多个路由网络部署 OVERCLOUD

要使用多组路由网络部署 overcloud，请为 spine-leaf 网络创建 control plane 和 Compute 节点，然后呈现并应用 Ansible playbook。要创建 control plane，请指定 Controller 节点的资源。要从裸机机器为 leafs 创建 Compute 节点，请在 OpenStackBaremetalSet 自定义资源中包含资源规格。

流程

1. 在工作站上创建一个名为 openstack-controller.yaml 的文件。包含 Controller 节点的资源规格。以下示例显示了由三个 Controller 节点组成的 control plane 规格：

```
apiVersion: osp-director.openstack.org/v1beta2
```

```
kind: OpenStackControlPlane
metadata:
  name: overcloud
  namespace: openstack
spec:
  gitSecret: git-secret
  openStackClientImageURL: registry.redhat.io/rhosp-rhel9/openstack-tripleoclient:17.1
  openStackClientNetworks:
    - ctlplane
    - external
    - internal_api
    - internal_api_leaf1 # optionally the openstackclient can also be connected to subnets
  openStackClientStorageClass: host-nfs-storageclass
  passwordSecret: userpassword
  domainName: ostest.test.metalkube.org
  virtualMachineRoles:
    Controller:
      roleName: Controller
      roleCount: 1
      networks:
        - ctlplane
        - internal_api
        - external
        - tenant
        - storage
        - storage_mgmt
      cores: 6
      memory: 20
      rootDisk:
        diskSize: 50
        baseImageVolumeName: openstack-base-img
        storageClass: host-nfs-storageclass
        storageAccessMode: ReadWriteMany
        storageVolumeMode: Filesystem
  enableFencing: False
```

2. **创建 control plane：**

```
$ oc create -f openstack-controller.yaml -n openstack
```

3. **等待 Red Hat OpenShift Container Platform (RHOCP)创建与 OpenStackControlPlane 资源相关的资源。**

4. **在您的工作站上为每个 Compute leaf 创建一个文件，如 openstack-computeleaf1.yaml。包括 leaf 的 Compute 节点的资源规格。以下示例显示了一个 Compute leaf 的规格，它包含一个 Compute 节点：**

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackBaremetalSet
```

```
metadata:
  name: computeleaf1
  namespace: openstack
spec:
  # How many nodes to provision
  count: 1
  # The image to install on the provisioned nodes
  baseImageUrl: http://<source_host>/rhel-9.2-x86_64-kvm.qcow2
  # The secret containing the SSH pub key to place on the provisioned nodes
  deploymentSSHSecret: osp-controlplane-ssh-keys
  # The interface on the nodes that will be assigned an IP from the mgmtCidr
  ctlplaneInterface: enp7s0
  # Networks to associate with this host
  networks:
        - ctlplane
        - internal_api_leaf1
        - external
        - tenant_leaf1
        - storage_leaf1
  roleName: ComputeLeaf1
  passwordSecret: userpassword
```

5. **为每个叶叶创建 Compute 节点：**

```
$ oc create -f openstack-computeleaf1.yaml -n openstack
```

6. **使用 OpenStackConfigGenerator 生成 Ansible playbook，并应用 overcloud 配置。如需更多信息，请参阅使用 director Operator 配置和部署 overcloud。**

验证

1. **查看 control plane 的资源：**

```
$ oc get openstackcontrolplane/overcloud -n openstack
```

2. **查看 OpenStackVMSet 资源，以验证 control plane 虚拟机(VM)集的创建：**

```
$ oc get openstackvmsets -n openstack
```

3. **查看 VM 资源以验证 OpenShift Virtualization 中创建 control plane 虚拟机：**

```
$ oc get virtualmachines -n openstack
```

4. 测试对 **openstackclient pod** 远程 **shell** 的访问：

```
$ oc rsh -n openstack openstackclient
```

5. 查看每个 **Compute leaf** 的资源：

```
$ oc get openstackbaremetalset/computeleaf1 -n openstack
```

6. 查看 **RHOCP** 管理的裸机机器，以验证 **Compute** 节点的创建：

```
$ oc get baremetalhosts -n openshift-machine-api
```

第 15 章 备份和恢复 DIRECTOR OPERATOR 部署的 OVERCLOUD

Red Hat OpenStack Platform (RHOSP) director Operator (OSPdO)提供自定义资源定义(CRD)来备份和恢复部署。您不必手动导出和导入多个配置。OSPdO 知道哪些自定义资源(CR)，包括 ConfigMap 和 Secret CR，它需要创建完整的备份，因为它了解所有资源的状态。因此，OSPdO 不会备份处于不完整或错误状态的任何配置。

要备份和恢复 OSPdO 部署，您可以创建一个 OpenStackBackupRequest CR 来启动备份的创建或恢复。您的 OpenStackBackupRequest CR 创建 OpenStackBackup CR，用于存储指定命名空间的自定义资源(CR)、ConfigMap 和 Secret 配置。

15.1. 备份 DIRECTOR OPERATOR

要创建备份，您必须为命名空间创建 OpenStackBackupRequest 自定义资源(CR)。当 OpenStackBackup Request 对象以 save 模式创建时，会创建 OpenStackBackup CR。

流程

1. 在您的工作站上创建一个名为 openstack_backup.yaml 的文件。

2. 在 openstack_backup.yaml 文件中添加以下配置，以创建 OpenStackBackupRequest 自定义资源(CR)：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackBackupRequest
metadata:
  name: openstackbackupsave
  namespace: openstack
spec:
  mode: save (1)
  additionalConfigMaps: [] (2)
  additionalSecrets: [] (3)
```

(1) 将 模式设置为 save 以请求创建 OpenStackBackup CR。

(2) 可选：包含您手动创建的任何 ConfigMap 资源。

(3)

可选：包含您手动创建的任何 Secret 资源。

注意

OSPdO 尝试包括与命名空间中 OSPdO CR 关联的所有 ConfigMap 和 Secret 对象，如 OpenStackControlPlane 和 OpenStackBaremetalSet。您不需要在附加列表中包括它们。

3. 保存 openstack_backup.yaml 文件。

4. 创建 OpenStackBackupRequest CR：

```
$ oc create -f openstack_backup.yaml -n openstack
```

5. 监控 OpenStackBackupRequest CR 的创建状态：

```
$ oc get openstackbackuprequest openstackbackupsave -n openstack
```

- Quiescing 状态表示 OSPdO 正在等待 CR 进入其完成状态。CR 数量可能会影响完成创建备份所需的时间。

```
NAME                 OPERATION   SOURCE   STATUS      COMPLETION TIMESTAMP
openstackbackupsave  save                 Quiescing
```

如果状态保留在 Quiescing 状态的时间超过预期，您可以调查 OSPdO 日志以检查进度：

```
$ oc logs <operator_pod> -c manager -f
2022-01-11T18:26:15.180Z        INFO    controllers.OpenStackBackupRequest
Quiesce for save for OpenStackBackupRequest openstackbackupsave is waiting for:
[OpenStackBaremetalSet: compute, OpenStackControlPlane: overcloud,
OpenStackVMSet: controller]
```

  - 将 <operator_pod > 替换为 Operator pod 的名称。

- **Saved 状态表示 OpenStackBackup CR 已创建。**

```
NAME                    OPERATION   SOURCE   STATUS   COMPLETION TIMESTAMP
openstackbackupsave     save                 Saved    2022-01-11T19:12:58Z
```

- **Error 状态表示备份无法创建。查看请求内容以查找错误：**

```
$ oc get openstackbackuprequest openstackbackupsave -o yaml -n openstack
```

6. **查看 OpenStackBackup 资源以确认它存在：**

```
$ oc get openstackbackup -n openstack
NAME                              AGE
openstackbackupsave-1641928378    6m7s
```

15.2. 从备份中恢复 DIRECTOR OPERATOR

当您请求恢复备份时，Red Hat OpenStack Platform (RHOSP) director Operator (OSPdO)会获取指定 OpenStackBackup 资源的内容，并尝试将它们应用到命名空间中存在的所有现有自定义资源 (CR)、ConfigMap 和 Secret 资源。OSPdO 覆盖命名空间中任何现有资源，并为命名空间中未找到的资源创建新资源。

流程

1. **列出可用的备份：**

```
$ oc get osbackup
```

2. **检查特定备份的详情：**

```
$ oc get backup <name> -o yaml
```

- **将 <name> 替换为您要检查的备份名称。**

3. **在您的工作站上创建一个名为 openstack_restore.yaml 的文件。**

4.

在 openstack_restore.yaml 文件中添加以下配置，以创建 OpenStackBackupRequest 自定义资源(CR)：

```
apiVersion: osp-director.openstack.org/v1beta1
kind: OpenStackBackupRequest
metadata:
  name: openstackbackuprestore
  namespace: openstack
spec:
  mode: <mode>
  restoreSource: <restore_source>
```

- 将 <mode> 替换为以下选项之一：
  - 恢复 ：从现有的 OpenStackBackup 请求恢复。
  - cleanRestore ：在从现有的 OpenStackBackup 恢复和创建新资源之前，完全擦除命名空间中的现有 OSPdO 资源。
- 将 <restore_source > 替换为要恢复的 OpenStackBackup 的 ID，例如 openstackbackupsave-1641928378。

5. 保存 openstack_restore.yaml 文件。

6. 创建 OpenStackBackupRequest CR：

```
$ oc create -f openstack_restore.yaml -n openstack
```

7. 监控 OpenStackBackupRequest CR 的创建状态：

```
$ oc get openstackbackuprequest openstackbackuprestore -n openstack
```

- Loading 状态表示 OpenStackBackup 中的所有资源都对集群应用。

```
NAME                    OPERATION  SOURCE                      STATUS     COMPLETION TIMESTAMP
openstackbackuprestore  restore    openstackbackupsave-1641928378  Loading
```

- **Reconciling 状态表示所有资源都已加载，OSPdO 已协调以尝试置备所有资源。**

```
NAME                    OPERATION  SOURCE                        STATUS       COMPLETION
TIMESTAMP
openstackbackuprestore  restore    openstackbackupsave-1641928378  Reconciling
```

- **Restored 状态表示 OpenStackBackup CR 已恢复。**

```
NAME                    OPERATION  SOURCE                        STATUS     COMPLETION
TIMESTAMP
openstackbackuprestore  restore    openstackbackupsave-1641928378  Restored
2022-01-12T13:48:57Z
```

- **Error 状态表示恢复失败。查看请求内容以查找错误：**

```
$ oc get openstackbackuprequest openstackbackuprestore -o yaml -n openstack
```

第 16 章 使用 DIRECTOR OPERATOR 更改虚拟机上的资源

要更改 OpenStackVMSet 自定义资源(CR)的 CPU、RAM 和磁盘资源，请使用 OpenStackControlPlane CRD。

16.1. 更改 OPENSTACKVMSET CR 的 CPU 或 RAM

您可以使用 OpenStackControlPlane CRD 更改 OpenStackVMSet 自定义资源(CR)的 CPU 或 RAM。

流程

1. 将 Controller virtualMachineRole 内核数改为 8:

```
$ oc patch -n openstack osctlplane overcloud --type='json' -p='[{"op": "add", "path": "/spec/virtualMachineRoles/controller/cores", "value": 8 }]'
```

2. 将 Controller virtualMachineRole RAM 大小更改为 22GB：

```
$ oc patch -n openstack osctlplane overcloud --type='json' -p='[{"op": "add", "path": "/spec/virtualMachineRoles/controller/memory", "value": 22 }]'
```

3. 验证 virtualMachineRole 资源：

```
$ oc get osvmset
NAME         CORES   RAM   DESIRED   READY   STATUS        REASON
controller   8       22    1         1       Provisioned   All requested VirtualMachines have been provisioned
```

4. 从虚拟机内部执行正常关闭。逐一关闭每个更新的虚拟机。

5. 打开虚拟机：

```
$ `virtctl start <VM>` to power on the virtual machine.
```

- 将 <VM > 替换为您的虚拟机的名称。

16.2. 在 OPENSTACKVMSET CR 中添加额外的磁盘

您可以通过编辑 **additionalDisks** 属性来使用 **OpenStackControlPlane CRD** 为虚拟机添加其他磁盘。

流程

1. 在 **OpenStackControlPlane** 对象中添加或更新 **additionalDisks** 参数：

```
spec:
  ...
  virtualMachineRoles:
    Controller:
      ...
      additionalDisks:
      - baseImageVolumeName: openstack-base-img
        dedicatedIOThread: false
        diskSize: 10
        name: "data-disk1"
        storageAccessMode: ReadWriteMany
        storageClass: host-nfs-storageclass
        storageVolumeMode: Filesystem
```

2. 应用补丁：

```
$ oc patch -n openstack osctlplane overcloud --patch-file controller_add_data_disk1.yaml
```

3. 验证 **virtualMachineRole** 资源：

```
$ oc get osvmset controller -o json | jq .spec.additionalDisks
[
  {
    "baseImageVolumeName": "openstack-base-img",
    "dedicatedIOThread": false,
    "diskSize": 10,
    "name": "data-disk1",
    "storageAccessMode": "ReadWriteMany",
    "storageClass": "host-nfs-storageclass",
    "storageVolumeMode": "Filesystem"
  }
]
```

4. 从虚拟机内部执行正常关闭。逐一关闭每个更新的虚拟机。

5. **打开虚拟机：**

```
$ `virtctl start <VM>` to power on the virtual machine.
```

- **将 <VM > 替换为您的虚拟机的名称。**

第 17 章 AIRGAPPED 环境

air-gapped 环境通过物理地将其与其他网络和系统隔离来确保安全性。您可以在 air-gapped 环境中安装 director Operator，以确保安全性并提供某些法规要求。

17.1. 先决条件

- 一个正常运行的 Red Hat Openshift Container Platform (RHOCP)集群，版本 4.12 或更高版本。集群必须包含 provisioning 网络以及以下 Operator：
  - 一个 裸机集群 Operator。必须启用 baremetal 集群 Operator。如需有关 baremetal 集群 Operator 的更多信息，请参阅 Bare-metal 集群 Operator。
  - OpenShift Virtualization Operator。有关安装 OpenShift Virtualization Operator 的更多信息，请参阅使用 Web 控制台安装 OpenShift Virtualization。
  - SR-IOV Network Operator。
- 您有一个断开连接的 registry，它遵循 docker v2 模式。如需更多信息，请参阅为断开连接的安装镜像镜像。
- 您可以访问 Satellite 服务器或用于注册 overcloud 节点的任何其他存储库，并安装软件包。
- oc 命令行工具已安装在您的工作站上。
- 您可以访问本地 git 存储库来存储部署工件。
- 您已在工作站上安装了 podman 和 skopeo 命令行工具。

17.2. 配置 AIRGAPPED 环境

要配置 airgapped 环境，您必须有权访问 registry.redhat.io 和 airgapped 环境的 registry。有关如何访问这两个 registry 的更多信息，请参阅将 目录内容镜像到 airgapped registry。

流程

1. **创建 openstack 命名空间：**

```
$ oc new-project openstack
```

2. **创建索引镜像并将其推送到 registry：**

```
$ podman login registry.redhat.io
$ podman login your.registry.local
$ BUNDLE_IMG="registry.redhat.io/rhosp-rhel8/osp-director-operator-
bundle@sha256:c19099ac3340d364307a43e0ae2be949a588fefe8fcb17663049342e7587f055
"
```

注意

您可以从中获取最新的捆绑包镜像： 认证的容器镜像**。搜索 osp-director-operator-bundle。**

3. **根据 Operator 索引镜像镜像相关的镜像：**

```
$ oc adm catalog mirror ${INDEX_IMG} your.registry.local --insecure --index-filter-by-
os='Linux/x86_64'
```

4. **当镜像（mirror）完成后，会在当前目录中生成一个 manifests 目录，名为 manifests-osp-director-operator-index-<random_number>。将创建的 ImageContentSourcePolicy 应用到集群：**

```
$ os apply -f manifests-osp-director-operator-index-
<random_number>/imageContentSourcePolicy.yaml
```

- **将 <random_number > 替换为随机生成的数字。**

5. **创建名为 osp-director-operator.yaml 的文件，并包含以下 YAML 内容来配置安装 director Operator 所需的三个资源：**

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: osp-director-operator-index
```

```
  namespace: openstack
spec:
  sourceType: grpc
  image: your.registry.local/osp-director-operator-index:1.3.x-y
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: "osp-director-operator-group"
  namespace: openstack
spec:
  targetNamespaces:
  - openstack
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: osp-director-operator-subscription
  namespace: openstack
spec:
  config:
    env:
    - name: WATCH_NAMESPACE
      value: openstack,openshift-machine-api,openshift-sriov-network-operator
  source: osp-director-operator-index
  sourceNamespace: openstack
  name: osp-director-operator
```

6. **在 openstack 命名空间中创建新资源：**

```
$ oc apply -f osp-director-operator.yaml
```

7. **将所需的 overcloud 镜像复制到存储库：**

```
$ for i in $(podman search --limit 1000 "registry.redhat.io/rhosp-rhel9/openstack" --format="{{
.Name }}" | awk '{print $1 ":" "17.1.0"}' | awk -F "/" '{print $2 "/" $3}'); do skopeo copy --all
docker://registry.redhat.io/$i docker://your.registry.local/$i;done
```

注意

如果使用 Red Hat Satellite 作为本地 registry，请参阅 为容器镜像准备 Satellite 服务器。

8. **现在，您可以继续安装 和准备 director Operator。**

验证

1. **确认您已成功安装了 director Operator：**

```
$ oc get operators
NAME                                 AGE
osp-director-operator.openstack      5m
```

其它资源

- 使用 CLI 从 OperatorHub 安装。
- 镜像用于断开连接的集群的 Operator 目录。
- 将目录内容镜像到 airgapped registry。
- 为容器镜像准备 Satellite 服务器。
- 从私有 registry 获取容器镜像