Red Hat OpenStack Platform 17.1

配置裸机置备服务

为裸机即服务(BMaaS)安装和配置裸机调配服务(ironic)

Last Updated: 2024-06-19

Red Hat OpenStack Platform 17.1 配置裸机置备服务

为裸机即服务(BMaaS)安装和配置裸机调配服务(ironic)

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

在 Red Hat OpenStack Platform 环境的 overcloud 中安装和配置裸机置备服务，以便为云用户置备和管理物理机。

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 裸机置备服务(IRONIC)功能

您可以使用裸机置备服务(ironic)组件来作为云用户的裸机实例置备和管理物理机器。要置备和管理裸机实例，裸机置备服务与 overcloud 中的以下 Red Hat OpenStack Platform (RHOSP)服务交互：

- 计算服务(nova)为虚拟机实例管理提供调度、租户配额和用户面向用户的 API。裸机置备服务为硬件管理提供管理 API。
- Identity 服务(keystone)提供请求身份验证，并帮助裸机置备服务定位其他 RHOSP 服务。
- 镜像服务(glance)管理磁盘和实例镜像和镜像元数据。
- 网络服务(neutron)提供 DHCP 和网络配置，并调配实例在引导时连接到的虚拟或物理网络。
- Object Storage 服务(swift)为某些驱动程序公开临时镜像 URL。

裸机置备服务组件

裸机置备服务由名为 **ironic-*** 的服务组成。以下服务是核心裸机置备服务：

裸机置备 API (ironic-api)

此服务向用户提供外部 REST API。API 通过远程过程调用(RPC)将应用程序请求发送到裸机置备编排器。

裸机置备编排器(ironic-conductor)

此服务使用驱动程序来执行以下裸机节点管理任务：

- 添加、编辑和删除裸机节点。
- 使用 IPMI、Redfish 或其他特定于供应商的协议打开和关闭裸机节点。
- 置备、部署和清理裸机节点。

裸机置备检查器(ironic-inspector)

此服务发现调度裸机实例所需的裸机节点的硬件属性，并为发现的以太网 MAC 创建裸机置备服务端口。

裸机置备数据库

此数据库跟踪硬件信息和状态。

消息队列

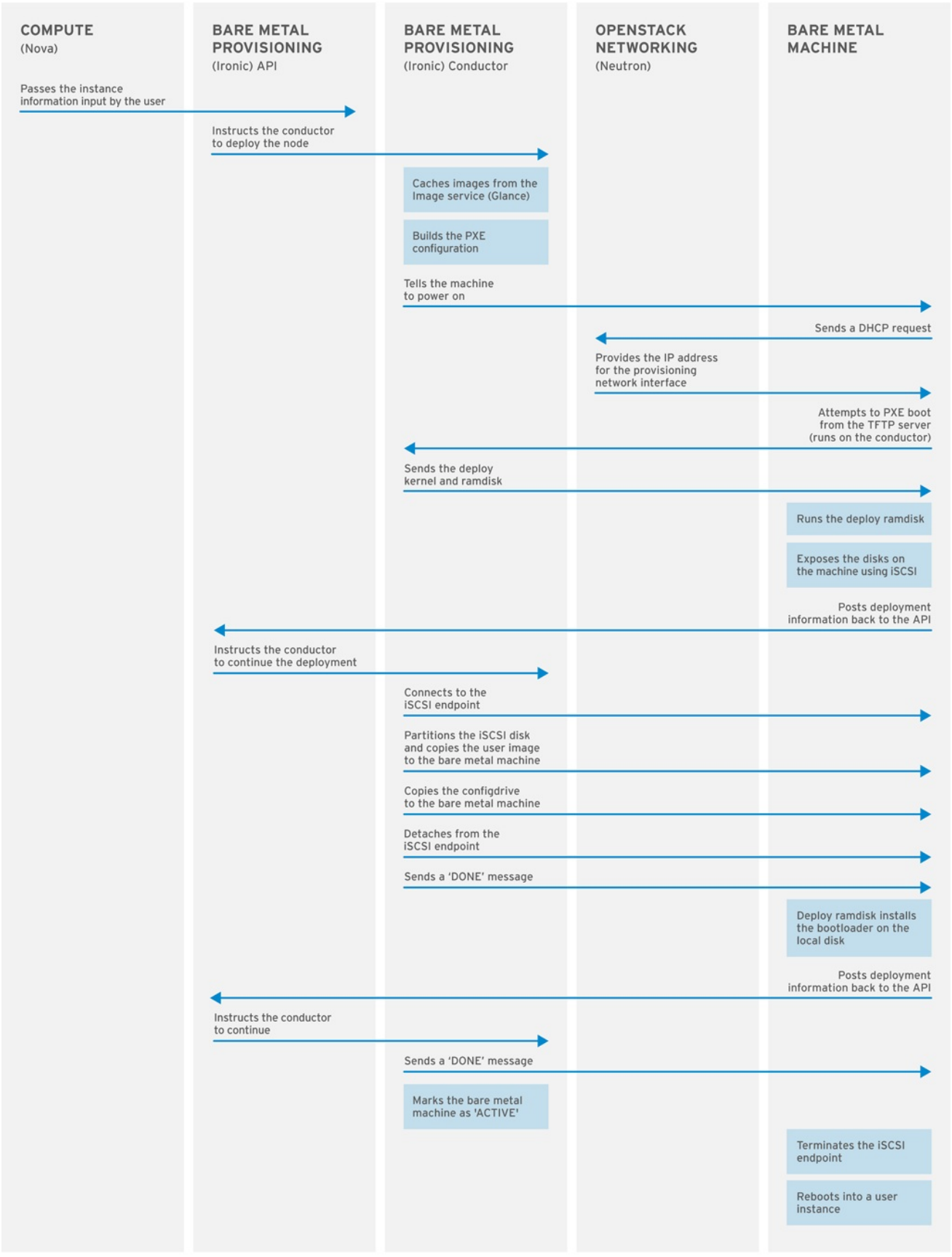
所有服务都使用此消息传递服务相互通信，包括在 **ironic-api** 和 **ironic-conductor** 之间实施 RPC。

裸机置备代理(ironic-python-agent)

此服务在临时 ramdisk 中运行，为 **ironic-conductor** 和 **ironic-inspector** 服务提供远程访问、带硬件控制和硬件内省。

置备裸机实例

裸机置备服务使用 iPXE 将物理计算机调配为裸机实例。下图显示了在云用户启动具有默认驱动程序的新裸机实例时，RHOSP 服务在置备过程中如何进行交互。



OPENSTACK_377593_1215

第 2 章 裸机置备的要求

为了提供可启动裸机实例的 overcloud，您的 Red Hat OpenStack Platform (RHOSP)环境必须具有所需的硬件和网络配置。

2.1. 硬件要求

您要提供给您的云用户使用的裸机的硬件要求取决于操作系统。有关 Red Hat Enterprise Linux 安装的硬件要求的详情，请查看 Red Hat Enterprise Linux 产品文档 。

您要提供给云用户用于置备的裸机机器都必须具有以下功能：

- 连接到裸机网络的 NIC。
- 一个电源管理接口，如 Redfish 或 IPMI，它连接到可从 **ironic-conductor** 服务访问的网络。默认情况下，**ironic-conductor** 在所有 Controller 节点上运行，除非您使用可组合角色并在其他位置运行 **ironic-conductor**。
- PXE 引导在裸机网络中。在部署中的所有其他 NIC 上禁用 PXE 引导。

2.2. 网络要求

裸机网络必须是专用网络，才能使裸机置备服务用于以下操作：

- overcloud 上裸机的调配和管理。
- 当节点被取消置备时，清理裸机节点。
- 对裸机的租户访问。

裸机网络提供 DHCP 和 PXE 引导功能来发现裸机系统。此网络必须在中继接口上使用原生 VLAN，以便裸机置备服务能够提供 PXE 引导和 DHCP 请求。

overcloud 中的裸机置备服务是为可信租户环境设计的，因为裸机可以直接访问 Red Hat OpenStack Platform (RHOSP)环境的 control plane 网络。因此，默认的裸机网络对 **ironic-conductor** 服务使用扁平网络。

默认扁平置备网络可能会给客户环境中引入安全问题，因为租户可能会干扰 control plane 网络。要防止这个风险，您可以为无法访问 control plane 的裸机置备服务配置自定义可组合裸机置备网络。

裸机网络必须取消标记以进行置备，还必须有权访问裸机置备 API。control plane 网络（也称为 director 置备网络）始终未标记。可以标记其他网络。

托管裸机置备服务的 Controller 节点必须有权访问裸机网络。

裸机机器配置为 PXE-boot 的 NIC 必须有权访问裸机网络。

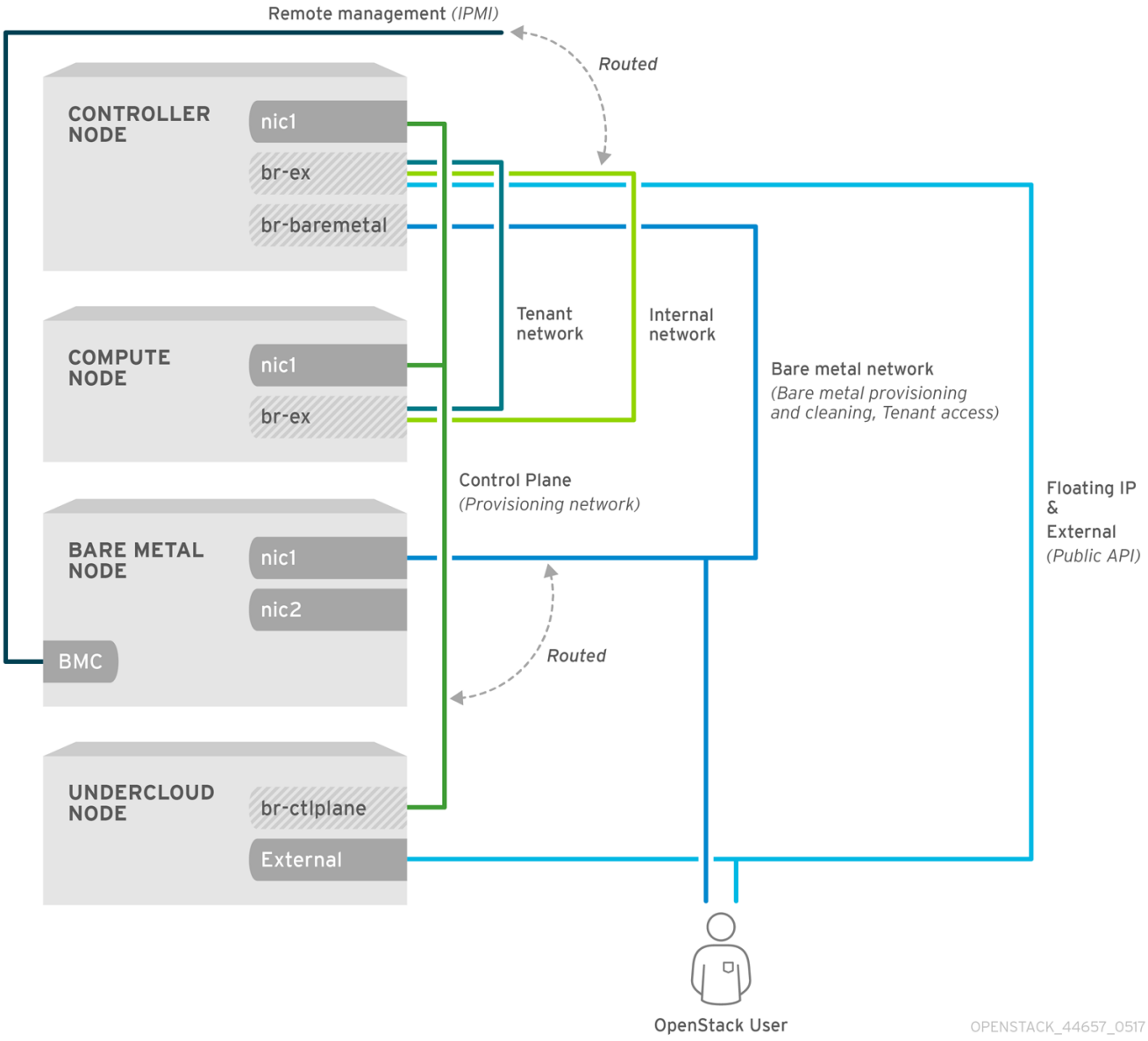
裸机网络由 OpenStack 操作器创建。云用户可以直接访问公共 OpenStack API 和裸机网络。对于默认的扁平裸机网络，云用户也具有对 control plane 的间接访问权限。

裸机置备服务使用裸机网络进行节点清理。

2.2.1. 默认裸机网络

在默认的裸机置备服务部署架构中，裸机网络与 control plane 网络分开。裸机网络是一个扁平网络，它也充当租户网络。此网络必须路由到 control plane 上的裸机置备服务，称为 director 置备网络。如果您定义了隔离的裸机网络，裸机节点无法 PXE 引导。

默认裸机网络架构图



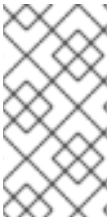
2.2.2. 自定义可组合裸机网络

当您在裸机置备服务部署架构中使用自定义可组合裸机网络时，裸机网络是一个自定义可组合网络，它无法访问 control plane。如果要限制对 control plane 的访问，请使用自定义可组合裸机网络。

第 3 章 使用裸机置备服务部署 OVERCLOUD

要使用裸机置备服务(ironic)部署 overcloud，您必须创建和配置裸机网络，并配置 overcloud 以启用裸机置备。

1. 创建裸机网络。您可以重复使用 Controller 节点上的 provisioning 网络接口来创建扁平网络，也可以创建自定义网络：
   - 配置默认的扁平网络
   - 配置自定义 IPv4 置备网络
   - 配置自定义 IPv6 置备网络
2. 配置 overcloud 以启用裸机置备：
   - 配置 overcloud 以启用裸机置备



注意

如果您使用 Open Virtual Network (OVN)，则裸机置备服务只支持 **ironic-overcloud.yaml** 文件中定义的 DHCP 代理，**neutron-dhcp-agent**。OVN 上的内置 DHCP 服务器无法置备裸机节点，或为置备网络提供 DHCP。要启用 iPXE 链加载，您必须在 dnsmasq 中设置 **--dhcp-match** 标签，这是 OVN DHCP 服务器不支持的。

前提条件

- 您的环境满足最低要求。如需更多信息，请参阅裸机置备的要求。

3.1. 配置默认的扁平网络

要使用默认的扁平裸机网络，您可以重复使用 Controller 节点上的 provisioning 网络接口来为裸机置备服务(ironic)创建桥接。

流程

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

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

3. 修改 **/home/stack/templates/nic-configs/controller.yaml** 文件，以重复利用 Controller 节点 **eth1** 上的 provisioning 网络接口，以便为裸机网络创建网桥：

```
network_config:
- type: ovs_bridge
  name: br-baremetal
  use_dhcp: false
  members:
    - type: interface
      name: eth1
  addresses:
    - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
```

注意

在创建裸机网络时，您无法通过重复使用 provisioning 网络来标记裸机网络。

4. 将 **br-baremetal** 添加到 **network-environment.yaml** 文件中的 **NeutronBridgeMappings** 参数：

```
parameter_defaults:
  NeutronBridgeMappings: datacentre:br-ex,baremetal:br-baremetal
```

5. 将 **baremetal** 添加到 **network-environment.yaml** 文件中的 **NeutronFlatNetworks** 参数指定的网络列表中：

```
parameter_defaults:
  NeutronBridgeMappings: datacentre:br-ex,baremetal:br-baremetal
  NeutronFlatNetworks: datacentre,baremetal
```

后续步骤

- 配置 overcloud 以启用裸机置备

3.2. 配置自定义 IPV4 置备网络

创建自定义 IPv4 调配网络，以通过 IPv4 来置备和部署 overcloud。

流程

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

```
$ source ~/stackrc
```

3. 将 **network_data.yaml** 文件复制到环境文件目录中：

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-
templates/network_data.yaml /home/stack/templates/network_data.yaml
```

4. 将 overcloud 置备的新网络添加到 **network_data.yaml** 文件中：

```
# custom network for overcloud provisioning
- name: OcProvisioning
  name_lower: oc_provisioning
  vip: true
  vlan: 205
  ip_subnet: '<ipv4_subnet_address>/<ipv4_mask>'
  allocation_pools: [{'start': '<ipv4_start_address>', 'end': '<ipv4_end_address>'}]
```

- 将 **<ipv4_subnet_address** > 替换为 IPv4 子网的 IPv4 地址。
- 将 **<ipv4_mask** > 替换为 IPv4 子网的 IPv4 网络掩码。

- 将 **<ipv4_start_address > 和 <ipv4_end_address** > 替换为您要用于地址分配的 IPv4 范围。

5. 在 **ServiceNetMap** 配置中配置 **IronicApiNetwork** 和 **IronicNetwork**，以使用新的 IPv4 置备网络：

```
ServiceNetMap:
  IronicApiNetwork: oc_provisioning
  IronicNetwork: oc_provisioning
```

6. 将新网络作为接口添加到本地 Controller NIC 配置文件：

```
network_config:
- type: vlan
  vlan_id:
    get_param: OcProvisioningNetworkVlanID
  addresses:
  - ip_netmask:
      get_param: OcProvisioningIpSubnet
```

7. 将 **roles_data.yaml** 文件复制到环境文件目录中：

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-
templates/roles_data.yaml /home/stack/templates/roles_data.yaml
```

8. 将控制器的新网络添加到 **roles_data.yaml** 文件中：

```
networks:
  ...
  OcProvisioning:
    subnet: oc_provisioning_subnet
```

9. 如果还没有存在，在您的 **roles_data.yaml** 文件的 **Ironic** 角色中包括 **IronicInspector** 服务：

```
ServicesDefault:
  OS::TripleO::Services::IronicInspector
```

后续步骤

- 配置 overcloud 以启用裸机置备

3.3. 配置自定义 IPV6 置备网络

创建自定义 IPv6 置备网络，以在 IPv6 上置备和部署 overcloud。

流程

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

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

3. 将 **network_data.yaml** 文件复制到环境文件目录中：

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-
templates/network_data.yaml /home/stack/templates/network_data.yaml
```

4. 将 overcloud 置备的新 IPv6 网络添加到您的 **network_data.yaml** 文件中：

```
# custom network for IPv6 overcloud provisioning
- name: OcProvisioningIPv6
  vip: true
  name_lower: oc_provisioning_ipv6
  vlan: 10
  ipv6: true
  ipv6_subnet: '<ipv6_subnet_address>/<ipv6_prefix>'
  ipv6_allocation_pools: [{'start': '<ipv6_start_address>', 'end': '<ipv6_end_address>'}]
  gateway_ipv6: '<ipv6_gw_address>'
```

   - 将 **<ipv6_subnet_address** > 替换为 IPv6 子网的 IPv6 地址。
   - 将 **<ipv6_prefix** > 替换为您的 IPv6 子网的 IPv6 网络前缀。
   - 将 **<ipv6_start_address > 和 <ipv6_end_address** > 替换为您要用于地址分配的 IPv6 范围。
   - 将 **<ipv6_gw_address** > 替换为网关的 IPv6 地址。

5. 在环境文件目录中创建新文件 **network_environment_overrides.yaml** ：

```
$ touch /home/stack/templates/network_environment_overrides.yaml
```

6. 在 **network_environment_overrides.yaml** 文件中配置 **IronicApiNetwork** 和 **IronicNetwork,** 以使用新的 IPv6 置备网络：

```
ServiceNetMap:
  IronicApiNetwork: oc_provisioning_ipv6
  IronicNetwork: oc_provisioning_ipv6
```

7. 将 **IronicIpVersion** 参数设置为 **6** ：

```
parameter_defaults:
  IronicIpVersion: 6
```

8. 启用 **RabbitIPv6**、**MysqlIPv6** 和 **RedisIPv6** 参数：

```
parameter_defaults:
  RabbitIPv6: True
  MysqlIPv6: True
  RedisIPv6: True
```

9. 将新网络作为接口添加到本地 Controller NIC 配置文件：

```
network_config:
- type: vlan
  vlan_id:
```

```
    get_param: OcProvisioningIPv6NetworkVlanID
  addresses:
  - ip_netmask:
      get_param: OcProvisioningIPv6IpSubnet
```

10. 将 **roles_data.yaml** 文件复制到环境文件目录中：

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml /home/stack/templates/roles_data.yaml
```

11. 将 Controller 角色的新网络添加到 **roles_data.yaml** 文件中：

```
networks:
  ...
  - OcProvisioningIPv6
```

12. 如果还没有存在，在您的 **roles_data.yaml** 文件的 **Ironic** 角色中包括 **IronicInspector** 服务：

```
ServicesDefault:
  OS::TripleO::Services::IronicInspector
```

后续步骤

- 配置 overcloud 以启用裸机置备

3.4. 配置 OVERCLOUD 以启用裸机置备

要在 overcloud 上启用裸机置备服务(ironic)，请使用 overcloud 部署中的 **/usr/share/openstack-tripleo-heat-templates/environments/services** 目录中的一个默认模板：

- 对于使用 OVS 的部署： **ironic.yaml**
- 对于使用 OVN 的部署：**ironic-overcloud.yaml**

您还可以创建本地环境文件来覆盖部署所需的默认配置。

如果要启用裸机置备检查器服务 **ironic-inspector** 服务，还必须在 overcloud 部署中包括 **/usr/share/openstack-tripleo-heat-templates/environments/services/ironic-inspector.yaml** 文件。

流程

1. 在本地目录中创建环境文件，为您的部署配置裸机置备服务，如 **ironic-overrides.yaml**。
2. 可选：配置在置备前以及置备间要在裸机上执行的清理类型：

```
parameter_defaults:
  IronicCleaningDiskErase: <cleaning_type>
```

将 **<cleaning_type** > 替换为以下值之一：

- **full**: （默认）执行完全清理。
- **元数据** ：仅清理分区表。这种类型的清理可显著加快清理过程。但是，由于部署在多租户环境中安全性较低，因此仅在可信租户环境中使用此选项。

3. 可选：在默认驱动程序中添加额外驱动程序：

```
parameter_defaults:
  IronicEnabledHardwareTypes: ipmi,idrac,ilo,[additional_driver_1],...,[additional_driver_n]
```

用您要启用的额外驱动程序替换 **[additional_driver_1]**，（可选）以及直到 **[additional_driver_n]**。

4. 要启用裸机内省，请在本地裸机置备服务环境文件中添加以下配置，**ironic-overrides.yaml** ：

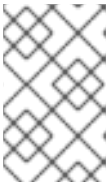
```
parameter_defaults:
  IronicInspectorSubnets:
    - ip_range: <ip_range>
  IPAImageURLs: '["http://<ip_address>:<port>/agent.kernel", "http://<ip_address>:
<port>/agent.ramdisk"]'
  IronicInspectorInterface: '<baremetal_interface>'
```

- 将 **<ip_range** > 替换为您的环境的 IP 范围，如 **192.168.0.100、192.168.0.120**。
- 将 **<ip_address>**:<port> 替换为托管 IPA 内核和 ramdisk 的 Web 服务器的 IP 地址和端口。要使用 undercloud 上使用的同一镜像，请将 IP 地址设置为 undercloud IP 地址，并将端口设为 **8088**。如果省略此参数，则必须在每个 Controller 节点上包括 alternatives。
- 将 **<baremetal_interface>** 替换为裸机网络接口，例如 **br-baremetal**。

5. 使用其他环境文件将新角色和自定义环境文件添加到堆栈中，并部署 overcloud：

```
(undercloud)$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -e /home/stack/templates/node-info.yaml \
  -r /home/stack/templates/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/services/<default_ironic_template> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic-inspector.yaml
\
  -e /home/stack/templates/network_environment_overrides.yaml \
  -n /home/stack/templates/network_data.yaml \
  -e /home/stack/templates/ironic-overrides.yaml
```

- 根据部署的网络服务机制驱动程序，将 **<default_ironic_template** > 替换为 **ironic.yaml** 或 **ironic-overcloud.yaml**。



注意

将环境文件传递给 **openstack overcloud deploy** 命令的顺序非常重要，因为后续文件中的配置具有优先权。因此，必须在任何网络配置文件后将在您的 overcloud 上启用和配置裸机置备的环境文件传递给命令。

3.5. 测试裸机置备服务

您可以使用 OpenStack Integration Test Suite 来验证 Red Hat OpenStack 部署。如需更多信息，请参阅使用 Red Hat OpenStack Platform Integration Test Suite 验证您的云 。

裸机置备服务的其他验证方法：

裸机置备服务的其他验证方法：

1. 将 shell 配置为以管理用户身份访问身份：

```
$ source ~/overcloudrc
```

2. 检查 **nova-compute** 服务是否在 Controller 节点上运行：

```
$ openstack compute service list -c Binary -c Host -c Status
```

3. 如果您更改了默认的 ironic 驱动程序，请确保启用了所需的驱动程序：

```
$ openstack baremetal driver list
```

4. 确保列出了 ironic 端点：

```
$ openstack catalog list
```

3.6. 其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 部署命令选项
- 为 overcloud 配置 IPv6 网络
- *Overcloud 参数指南*中的裸机 (ironic) 参数

第 4 章 部署后配置裸机置备服务

使用裸机置备服务(ironic)部署 overcloud 时，您必须为裸机工作负载准备 overcloud。要为裸机工作负载准备 overcloud，并使云用户能够创建裸机实例，请完成以下任务：

- 配置网络服务(neutron)，以与裸机置备服务集成。
- 配置节点清理。
- 创建裸机类别和资源类。
- 可选：创建裸机镜像。
- 添加物理机器作为裸机节点。
- 可选：配置 Redfish 虚拟介质引导。
- 可选：创建主机聚合以分隔物理和虚拟虚拟机调配。

4.1. 为裸机置备配置网络服务

您可以配置网络服务(neutron)来与裸机置备服务(ironic)集成。您可以使用以下方法之一配置裸机网络：

- 为裸机置备编排器服务 **ironic-conductor** 创建单一扁平裸机网络。此网络必须路由到 control plane 网络上的裸机置备服务。
- 创建自定义可组合网络，以便在 overcloud 中实施裸机置备服务。

4.1.1. 配置网络服务以与扁平网络上的裸机置备服务集成

您可以通过为裸机置备编排器服务 **ironic-conductor** 创建单一扁平裸机网络，将网络服务(neutron)配置为与裸机置备服务(ironic)集成。此网络必须路由到 control plane 网络上的裸机置备服务。

步骤

1. 以 **root** 用户身份登录托管网络服务(neutron)的节点。
2. 提供 overcloud 凭证文件：

```
# source ~/<credentials_file>
```

   - 将 **<credentials_file** > 替换为您的凭据文件的名称，如 **overcloudrc**。

3. 创建扁平网络来调配裸机实例：

```
# openstack network create \
  --provider-network-type flat \
  --provider-physical-network <provider_physical_network> \
  --share <network_name>
```

   - 将 < **provider_physical_network** > 替换为您在其上实现虚拟网络的物理网络名称，该网络使用 **network-environment.yaml** 文件中的 **NeutronBridgeMappings** 参数进行配置。
   - 将 **<network_name** > 替换为这个网络的名称。

4. 在扁平网络上创建子网：

```
# openstack subnet create \
  --network <network_name> \
  --subnet-range <network_cidr> \
  --ip-version 4 \
  --gateway <gateway_ip> \
  --allocation-pool start=<start_ip>,end=<end_ip> \
  --dhcp <subnet_name>
```

- 将 &lt **;network_name** > 替换为您在上一步中创建的 provisioning 网络的名称。
- 将 < **network_cidr** > 替换为子网代表的 IP 地址块的无类别域间路由(CIDR)表示。您在以 < **start_ip> 开始的范围内指定的 IP 地址块，并以 &** lt; **end_ip>** 结尾，必须在 < **network_cidr>** 指定的 IP 地址块内。
- 使用充当新子网的网关的路由器接口的 IP 地址或主机名替换 **<gateway_ip>**。此地址必须在 < **network_cidr** > 指定的 IP 地址块内，但在以 < **start_ip> 开头的范围指定的 IP 地址块之外，并以 &lt**; **end_ip** > 结尾。
- 将 < **start_ip** > 替换为指定浮动 IP 地址的新子网中 IP 地址范围的 IP 地址开始。
- 将 < **end_ip** > 替换为表示浮动 IP 地址从中分配 IP 地址范围的 IP 地址结束的 IP 地址。
- 将 **<subnet_name** > 替换为子网的名称。

5. 为网络和子网创建一个路由器，以确保网络服务服务元数据请求：

```
# openstack router create <router_name>
```

- 将 **<router_name** > 替换为路由器的名称。

6. 将子网附加到新路由器，以启用来自 **cloud-init** 的元数据请求以及要配置的节点：

```
# openstack router add subnet <router_name> <subnet>
```

- 将 **<router_name** > 替换为路由器的名称。
- 将 & **lt;subnet** > 替换为在第 4 步中创建的裸机子网的 ID 或名称。

4.1.2. 配置网络服务，以与自定义可组合网络上的裸机置备服务集成

您可以通过创建自定义可组合网络以在 overcloud 中实施裸机置备服务，将网络服务(neutron)配置为与裸机置备服务集成。

步骤

1. 登录 undercloud 主机。
2. 提供 overcloud 凭证文件：

```
$ source ~/<credentials_file>
```

- 将 **<credentials_file** > 替换为您的凭据文件的名称，如 **overcloudrc**。

3. 检索托管裸机置备服务的提供商网络的 UUID：

```
(overcloud)$ openstack network show <network_name> -f value -c id
```

- 将 < **network_name** > 替换为您要用于裸机实例置备网络的供应商网络的名称。

4. 打开本地环境文件，为您的部署配置裸机置备服务，如 **ironic-overrides.yaml**。

5. 配置网络，以用作裸机实例 provisioning 网络：

```
parameter_defaults:
  IronicProvisioningNetwork: <network_uuid>
```

- 将 **<network_uuid** > 替换为在第 3 步中检索到的提供商网络 UUID。

6. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

7. 要应用裸机实例置备网络配置，请将裸机置备环境文件与其他环境文件一起添加到堆栈中，并部署 overcloud：

```
(undercloud)$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -e /home/stack/templates/node-info.yaml \
  -r /home/stack/templates/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/services/<default_ironic_template> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic-inspector.yaml
\
  -e /home/stack/templates/network_environment_overrides.yaml \
  -n /home/stack/templates/network_data.yaml \
  -e /home/stack/templates/ironic-overrides.yaml
```

- 根据部署的网络服务机制驱动程序，将 **<default_ironic_template** > 替换为 **ironic.yaml** 或 **ironic-overcloud.yaml**。

4.2. 清理裸机节点

裸机置备服务会清理节点，以准备它们以进行置备。您可以使用以下方法之一清理裸机节点：

- 自动 ：您可以将 overcloud 配置为在取消置备节点时自动执行节点清理。
- 手动：您可以根据需要手动清理单个节点。

4.2.1. 配置自动节点清理

在注册节点后，自动裸机节点清理会在节点达到 **可用的** 置备状态之前运行。当节点被取消置备时，都会运行自动清理。

默认情况下，Bare Metal Provisioning 服务使用名为 **provisioning** 的网络进行节点清理。但是，网络名称在网络服务(neutron)中并不是唯一的，因此项目可能会创建具有相同名称的网络，这会导致与裸机置备服务冲突。要避免冲突，请使用网络 UUID 来配置节点清理网络。

流程

1. 登录 undercloud 主机。
2. 提供 overcloud 凭证文件：

```
$ source ~/<credentials_file>
```

   - 将 **<credentials_file** > 替换为您的凭据文件的名称，如 **overcloudrc**。

3. 检索托管裸机置备服务的提供商网络的 UUID：

```
(overcloud)$ openstack network show <network_name> -f value -c id
```

   - 将 < **network_name** > 替换为您要用于裸机节点清理网络的名称。

4. 打开本地环境文件，为您的部署配置裸机置备服务，如 **ironic-overrides.yaml**。
5. 配置网络，以用作节点清理网络：

```
parameter_defaults:
  IronicCleaningNetwork: <network_uuid>
```

   - 将 **<network_uuid** > 替换为在第 3 步中获取的提供商网络的 UUID。

6. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

7. 要应用节点清理网络配置，请将裸机置备环境文件与其他环境文件一起添加到堆栈中，并部署 overcloud：

```
(undercloud)$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -e /home/stack/templates/node-info.yaml \
  -r /home/stack/templates/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/services/<default_ironic_template> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic-inspector.yaml
\
  -e /home/stack/templates/network_environment_overrides.yaml \
  -n /home/stack/templates/network_data.yaml \
  -e /home/stack/templates/ironic-overrides.yaml
```

   - 根据部署的网络服务机制驱动程序，将 **<default_ironic_template** > 替换为 **ironic.yaml** 或 **ironic-overcloud.yaml**。

4.2.2. 手动清理节点

您可以根据需要手动清理特定的节点。节点清理有两种模式：

- 仅清理元数据：从节点上的所有磁盘中删除分区。仅清理元数据的模式比完全清理更快，但安全性较低，因为它只擦除分区表。仅在可信租户环境中使用此模式。

- full clean：使用 ATA 安全清除或已到期，从所有磁盘中删除所有数据。完全清理可能需要几小时才能完成。

流程

1. 提供 overcloud 凭证文件：

```
$ source ~/<credentials_file>
```

   - 将 **<credentials_file** > 替换为您的凭据文件的名称，如 **overcloudrc**。

2. 检查节点的当前状态：

```
$ openstack baremetal node show \
 -f value -c provision_state <node>
```

   - 将 **<node** > 替换为要清理的节点的名称或 UUID。

3. 如果节点没有处于 **manageable** 状态，然后将其设置为 **manageable** ：

```
$ openstack baremetal node manage <node>
```

4. 清理节点：

```
$ openstack baremetal node clean <node> \
  --clean-steps '[{"interface": "deploy", "step": "<clean_mode>"}]'
```

   - 将 **<node** > 替换为要清理的节点的名称或 UUID。
   - 将 **<clean_mode** > 替换为要在节点上执行的清理类型：
     - **erase_devices** ：执行完全清理。
     - **erase_devices_metadata** ：仅执行元数据清理。

5. 等待清理完成，然后检查节点的状态：
   - **可管理** ：清理成功，节点已准备好调配。
   - **清理失败** ：清理失败。检查 **last_error** 字段，以了解故障的原因。

4.3. 创建用于启动裸机实例的类别

您必须创建可供云用户用来请求裸机实例的类别。您可以使用资源类来指定哪些裸机节点应该用于使用特定类别启动的裸机实例。您可以使用标识节点上硬件资源的资源类标记裸机节点，例如 GPU。云用户可以选择具有 GPU 资源类的类别，以便为 vGPU 工作负载创建实例。计算调度程序使用资源类来识别适合实例的主机裸机节点。

流程

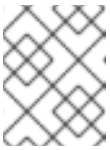
1. 提供 overcloud 凭证文件：

```
$ source ~/overcloudrc
```

2. 为裸机实例创建类别：

```
(overcloud)$ openstack flavor create --id auto \
 --ram <ram_size_mb> --disk <disk_size_gb> \
 --vcpus <no_vcpus> baremetal
```

- 将 **<ram_size_mb>** 替换为裸机节点的 RAM，以 MB 为单位。
- 将 **<disk_size_gb>** 替换为裸机节点中的磁盘大小（以 GB 为单位）。
- 将 **<no_vcpus>** 替换为裸机节点中的 CPU 数量。



注意

这些属性不可用于调度实例。但是，计算调度程序使用磁盘大小来确定根分区大小。

3. 检索节点列表来识别它们的 UUID：

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

4. 使用自定义裸机资源类标记每个裸机节点：

```
(overcloud)$ openstack baremetal node set \
 --resource-class baremetal.<CUSTOM> <node>
```

- 将 **<CUSTOM** > 替换为标识资源类目的的字符串。例如，设置为 **GPU** 以创建自定义 GPU 资源类，您可以使用它来标记您要为 GPU 工作负载指定的裸机节点。
- 将 **<node>** 替换为裸机节点的 ID。

5. 将裸机实例的类别与自定义资源类关联：

```
(overcloud)$ openstack flavor set \
 --property resources:CUSTOM_BAREMETAL_<CUSTOM>=1 \
 baremetal
```

要确定与裸机节点资源类对应的自定义资源类的名称，请将资源类转换为大写，将每个 punctuation 标记替换为下划线，并将前缀替换为 **CUSTOM_**。

注意

类别只能请求一个裸机资源类实例。

6. 设置以下类别属性，以防止计算调度程序使用裸机类别属性来调度实例：

```
(overcloud)$ openstack flavor set \
 --property resources:VCPU=0 \
 --property resources:MEMORY_MB=0 \
 --property resources:DISK_GB=0 baremetal
```

7. 验证新类别是否具有正确的值：

```
(overcloud)$ openstack flavor list
```

4.4. 创建用于启动裸机实例的镜像

包含裸机置备服务(ironic)的 overcloud 需要两组镜像：

- 部署镜像：部署镜像是 **agent.ramdisk** 和 **agent.kernel** 镜像，裸机置备代理(**ironic-python-agent**)需要通过网络引导 RAM 磁盘，并将 overcloud 节点的用户镜像复制到磁盘。作为 undercloud 安装的一部分安装部署镜像。如需更多信息，请参阅 获取 overcloud 节点的镜像。
- 用户镜像：云用户用来调配其裸机实例的镜像。用户镜像包括一个 **kernel** 镜像，一个 **ramdisk** 镜像和一个 **main** 镜像。主镜像是一个根分区，也可以是一个完整磁盘镜像：
  - 完整磁盘镜像：包含分区表和引导装载程序的镜像。
  - Root 分区镜像：仅包含操作系统的 root 分区。

兼容整个磁盘 RHEL 客户机镜像可以在不修改的情况下正常工作。要创建自己的自定义磁盘镜像，请参阅 创建和管理镜像 中的 创建 RHEL KVM 或 RHOSP 兼容 *镜像* 。

4.4.1. 将部署镜像上传到镜像服务

您必须将 director 安装的部署镜像上传到镜像服务。部署镜像由以下两个镜像组成：

- 内核镜像： **/tftpboot/agent.kernel**
- ramdisk 镜像： **/tftpboot/agent.ramdisk**

这些镜像安装在主目录中。有关如何安装部署镜像的更多信息，请参阅 获取 overcloud 节点的镜像。

流程

- 提取镜像并将其上传到镜像服务：

```
$ openstack image create \
  --container-format aki \
  --disk-format aki \
  --public \
  --file ./tftpboot/agent.kernel bm-deploy-kernel
$ openstack image create \
  --container-format ari \
  --disk-format ari \
  --public \
  --file ./tftpboot/agent.ramdisk bm-deploy-ramdisk
```

4.5. 将物理机器添加为裸机节点

使用以下方法之一注册裸机节点：

- 使用节点详情准备清单文件，将文件导入到裸机置备服务，并使节点可用。
- 将物理机注册为裸机节点，然后手动添加其硬件详情并为每个以太网 MAC 地址创建端口。您可以在具有 **overcloudrc** 文件的任何节点上执行这些步骤。

4.5.1. 使用清单文件注册裸机节点

您可以准备清单文件来注册定义每个裸机节点的详细信息的裸机节点。您可以将文件导入到裸机置备服务(ironic)，并使每个节点可用。

注意

有些驱动程序可能需要特定的配置。如需更多信息，请参阅 裸机驱动程序。

先决条件

- 包括裸机置备服务的 overcloud 部署。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

1. 创建一个清单文件来定义每个节点的详细信息，如 **overcloud-nodes.yaml**。
2. 对于每个节点，定义节点名称以及裸机驱动程序的地址和凭证。有关启用驱动程序的可用属性的详情，请参阅 裸机驱动程序。

```
nodes:
  - name: <node>
    driver: <driver>
    driver_info:
      <driver>_address: <ip>
      <driver>_username: <user>
      <driver>_password: <password>
      [<property>: <value>]
```

   - 将 **<node** > 替换为节点的名称。
   - 将 **<driver** > 替换为以下裸机驱动程序之一：
     - **ipmi**
     - **Redfish**
     - **DRAC**
     - **irmc**
     - **ilo**
   - 将 **<ip** > 替换为 Bare Metal 控制器的 IP 地址。
   - 将 & **lt;user&** gt; 替换为您的用户名。
   - 将 & **lt;password&** gt; 替换为您的密码。
   - 可选：将 **<** property> 替换为您要配置的 driver 属性，并将 & **lt;value** > 替换为属性值。有关可用属性的详情，请参考 裸机驱动程序。
3. 定义节点属性和端口：

```
nodes:
```

```
- name: <node>
  ...
  properties:
    cpus: <cpu_count>
    cpu_arch: <cpu_arch>
    memory_mb: <memory>
    local_gb: <root_disk>
    root_device:
      serial: <serial>
  ports:
    - address: <mac_address>
```

- 将 **<cpu_count** > 替换为 CPU 数量。
- 将 **<cpu_arch** > 替换为 CPU 构架类型。
- 将 **<memory** > 替换为 MiB 的内存量。
- 将 **<root_disk** > 替换为 GiB 中的根磁盘大小。只有机器有多个磁盘时才需要。
- 将 **&lt**;serial> 替换为您要用于部署的磁盘的序列号。
- 将 **<mac_address** > 替换为用于 PXE 引导的 NIC 的 MAC 地址。

4. 获取 **overcloudrc** 文件：

```
$ source ~/overcloudrc
```

5. 将清单文件导入到裸机置备服务中：

```
$ openstack baremetal create overcloud-nodes.yaml
```

节点现在处于 **注册** 状态。

6. 指定部署内核并在每个节点上部署 ramdisk：

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

- 将 **<node>** 替换为节点的名称或 ID。
- 将 **<kernel_file** > 替换为 **.kernel** 镜像的路径，例如 **file:///var/lib/ironic/httpboot/agent.kernel**。
- 将 **<initramfs_file** > 替换为 **.initramfs** 镜像的路径，例如 **file:///var/lib/ironic/httpboot/agent.ramdisk**。

7. 可选：如果您配置了 IPMI 驱动程序，请为每个节点指定 IPMI 密码套件：

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

- 将 **<node>** 替换为节点的名称或 ID。
- 将 **<version** > 替换为节点上要使用的密码套件版本。设置为以下有效值之一：

- **3** - 节点使用带有 SHA1 密码套件的 AES-128。
- **17** - 节点使用带有 SHA256 密码套件的 AES-128。

8. 等待额外的网络接口端口配置数据填充网络服务(neutron)。这个过程至少需要 60 秒。
9. 将每个节点的置备状态设置为 **available** ：

```
$ openstack baremetal node manage <node>
$ openstack baremetal node provide <node>
```

如果启用了节点清理，裸机置备服务会清理节点。

10. 在每个节点上设置本地引导选项：

```
$ openstack baremetal node set <node> --property capabilities="boot_option:local"
```

11. 检查节点是否已注册：

```
$ openstack baremetal node list
```

注册节点及其状态之间可能会有延迟。

4.5.2. 手动注册裸机节点

将物理机注册为裸机节点，然后手动添加其硬件详情并为每个以太网 MAC 地址创建端口。您可以在具有 **overcloudrc** 文件的任何节点上执行这些步骤。

前提条件

- 包括裸机置备服务的 overcloud 部署。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。
- 必须使用 **IronicEnabledHardwareTypes** 参数启用新节点的驱动程序。有关支持的驱动程序的更多信息，请参阅 裸机驱动程序。

流程

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

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

3. 添加新节点：

```
$ openstack baremetal node create --driver <driver_name> --name <node_name>
```

- 将 **<driver_name** > 替换为驱动程序的名称，如 **ipmi**。
- 将 **<node_name** > 替换为新裸机节点的名称。

4. 注意在创建节点时分配给节点的 UUID。

5. 为每个注册的节点将引导选项设置为 **local** ：

```
$ openstack baremetal node set \
  --property capabilities="boot_option:local" <node>
```

将 **<node>** 替换为裸机节点的 UUID。

6. 指定部署内核并为节点驱动程序部署 ramdisk：

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

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

7. 更新节点属性以匹配节点上的硬件规格：

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

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

8. 可选：为每个节点指定 IPMI 密码套件：

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

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

9. 可选：指定每个节点的 IPMI 详情：

```
$ openstack baremetal node set <node> \
  --driver-info <property>=<value>
```

- 将 **<node>** 替换为裸机节点的 ID。
- 将 **<property >** 替换为您要配置的 IPMI 属性。有关可用属性的详情，请参考 智能平台管理接口(IPMI)电源管理驱动程序。
- 将 **<value >** 替换为属性值。

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

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

- 将 **<node>** 替换为裸机节点的 ID。
- 将 **<property > 和 <value >** 替换为您要用于部署的磁盘的详情，如 **root_device='{"size": "128"}'**
  RHOSP 支持以下属性：
  - **model**（字符串）：设备识别码。
  - **vendor**（字符串）：设备厂商。
  - **serial**（字符串）：磁盘序列号。
  - **hctl**（字符串）：SCSI 的 Host:Channel:Target:Lun。
  - **size**（整数）：设备的大小（以 GB 为单位）。
  - **wwn**（字符串）：唯一的存储 ID。
  - **wwn_with_extension**（字符串）：唯一存储 ID 附加厂商扩展名。
  - **wwn_vendor_extension**（字符串）：唯一厂商存储标识符。
  - **rotational**（布尔值）：旋转磁盘设备为 true (HDD)，否则为 false (SSD)。
  - **name** （字符串）：设备名称，例如： /dev/sdb1 仅对具有持久名称的设备使用此属性。

注意

如果您指定多个属性，该设备必须与所有这些属性匹配。

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

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

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

12. 验证节点的配置：

```
$ openstack baremetal node validate <node>
+------------+--------+--------------------------------------------+
| Interface  | Result | Reason                                     |
+------------+--------+--------------------------------------------+
```

```
| boot       | False  | Cannot validate image information for node  |
|            |        | a02178db-1550-4244-a2b7-d7035c743a9b        |
|            |        | because one or more parameters are missing  |
|            |        | from its instance_info. Missing are:        |
|            |        | ['ramdisk', 'kernel', 'image_source']       |
| console    | None   | not supported                               |
| deploy     | False  | Cannot validate image information for node  |
|            |        | a02178db-1550-4244-a2b7-d7035c743a9b        |
|            |        | because one or more parameters are missing  |
|            |        | from its instance_info. Missing are:        |
|            |        | ['ramdisk', 'kernel', 'image_source']       |
| inspect    | None   | not supported                               |
| management | True   |                                             |
| network    | True   |                                             |
| power      | True   |                                             |
| raid       | True   |                                             |
| storage    | True   |                                             |
+------------+--------+---------------------------------------------+
```

验证输出结果 **Result** 表示以下内容：

- **false** ：接口验证失败。如果提供的原因包括 **instance_info** 参数 **[\'ramdisk'、\'kernel' 和 \'image_source']**，这可能是因为计算服务在部署过程开始时填充这些缺少的参数，因此目前还没有设置它们。如果您使用整个磁盘镜像，则可能需要设置 **image_source** 来传递验证。
- **true** ：接口已通过验证。
- **None**: 接口不支持您的驱动。

4.5.3. 裸机节点置备状态

裸机节点在生命周期内转换多个置备状态。在节点上执行的 API 请求和编排器事件启动转换。调配状态有两种类别： "stable" 和 "in transition"。

使用下表了解节点可以处于的调配状态，以及用于将节点从一个调配状态转换到另一个调配状态的操作。

表 4.1. 置备状态

状态	类别	描述
注册	稳定	每个节点的初始状态。有关注册节点的详情，请参考 将物理机添加为裸机节点。
验证	过渡	Bare Metal Provisioning 服务使用节点注册过程中提供的 **driver_info** 配置验证它是否可以管理节点。

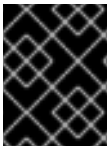
状态	类别	描述
可管理	稳定	当裸机置备服务验证了它可以管理节点时，节点将过渡到 manageable 状态。您可以使用以下命令将节点从 **manageable** 状态转换到以下状态之一： • **OpenStack baremetal node adopt → adopting → active** • **OpenStack baremetal 节点提供 → cleaning → available** • **OpenStack baremetal node clean → cleaning → available** • **OpenStack baremetal node inspect → check → manageable** 在转换到以下失败状态之一后，您必须将节点移到 **manageable** 状态： • **采用失败** • **清理失败** • **检查失败** 当您需要更新节点时，将节点移到 **manageable** 状态。
inspecting	过渡	裸机置备服务使用节点内省来更新硬件派生的节点属性，以反映硬件的当前状态。节点过渡到 **manageable** 以进行同步检查，**并检查等待** 异步检查。如果发生错误，节点将转换 **来检查失败**。
检查等待	过渡	指示异步检查正在进行的调配状态。如果节点检查成功，节点将过渡到 **manageable** 状态。
检查失败	稳定	指示节点检查失败的置备状态。您可以使用以下命令将节点从 **检查失败状态** 转换到以下状态之一： • **OpenStack baremetal node inspect → check → manageable** • **OpenStack baremetal node manage → manageable**
清理	过渡	处于 **清理** 状态的节点正在清理并重新编程到已知配置中。当节点处于 **清理** 状态时，根据网络管理，编排器执行以下任务： • 带外：编排器执行清理 **步骤**。 • In-band：编排器准备环境来引导 ramdisk 以运行带内清理步骤。准备任务包括构建 PXE 配置文件以及配置 DHCP。

状态	类别	描述
清理等待	过渡	处于 **clean 等待状态的节点** 将被清理，并被重新编程到已知配置中。这个状态与 **清理** 状态类似，但处于 **clean wait** 状态，编排器正在等待 ramdisk 引导或 **清理** 步骤完成。 您可以通过运行 **openstack baremetal node abort** 中断处于 **干净等待状态** 节点的清理过程。
可用	稳定	在成功预配置和清理节点后，它们会被移到 **available** 状态，并准备好置备。您可以使用以下命令将节点从 **available** 状态转换到以下状态之一： ● **openstack baremetal node deploy → deploying → active** ● **OpenStack baremetal node manage → manageable**
部署	过渡	处于 **deploying** 状态的节点是为工作负载准备的，这涉及执行以下任务： ● 为节点部署设置适当的 BIOS 选项。 ● 分区驱动器并创建文件系统。 ● 创建其他子系统可能需要的任何其他资源，如特定于节点的网络配置和配置驱动器分区。
wait call-back	过渡	处于 **wait call-back** 状态的节点是为工作负载准备的。这个状态与 **deploying** 状态类似，除了处于 **wait call-back** 状态时，编排器会在准备节点前等待任务完成。例如，在编排器可以准备节点前必须完成以下任务： ● ramdisk 已引导。 ● 已安装引导装载程序。 ● 该镜像写入磁盘。 您可以通过运行 **openstack baremetal node delete** 或 **openstack baremetal node undeploy** 中断处于 **wait call-back** 状态的节点部署。

状态	类别	描述
部署失败	稳定	指示节点部署失败的置备状态。您可以使用以下命令将节点从 **deploy failed** 状态转换到以下状态之一： • **openstack baremetal node deploy → deploying → active** • **OpenStack baremetal node rebuild → deploying → active** • **openstack baremetal node delete → deleting → cleaning → clean wait → cleaning → available** • **OpenStack baremetal node undeploy → deleting → cleaning → clean wait → cleaning → available**
active	稳定	处于 **active** 状态的节点有一个工作负载正在其上运行。裸机置备服务可能会定期收集带外传感器信息，包括电源状态。您可以使用以下命令将节点从 **active** 状态转换到以下状态之一： • **openstack baremetal node delete → deleting → available** • **OpenStack baremetal node undeploy → cleaning → available** • **OpenStack baremetal node rebuild → deploying → active** • **OpenStack baremetal node rescue → rescuing → rescue**
删除	过渡	当节点处于 **删除状态时**，裸机置备服务会解析活跃工作负载，并在节点部署或救援过程中删除它添加到节点的任何配置和资源。节点从 **删除** 状态快速过渡到 **清理** 状态，然后过渡到 **clean 等待状态**。
错误	稳定	如果节点删除失败，节点将移到 **错误状态**。您可以使用以下命令将节点从**错误**状态转换到以下状态之一： • **openstack baremetal node delete → deleting → available** • **OpenStack baremetal node undeploy → cleaning → available**
adopting	过渡	您可以使用 **openstack baremetal node adopt** 命令将具有现有工作负载的节点直接从 **manageable** 转换为 **active** 状态，而无需首先清理和部署节点。当节点处于 **采用** 状态时，裸机置备服务接管使用其现有工作负载管理节点。

状态	类别	描述
rescuing	过渡	处于 **rescuing** 状态的节点已准备好执行以下救援操作： • 为节点部署设置适当的 BIOS 选项。 • 创建其他子系统可能需要的任何其他资源，如特定于节点的网络配置。
rescue wait	过渡	处于 **rescue wait** 状态的节点正在救援。这个状态与 **rescuing** 状态类似，除了处于 **rescue** 等待状态外，编排器正在等待 ramdisk 引导，或者执行需要在该节点上运行 in-band 的 rescue 部分，如为名为 rescue 的用户设置密码。 您可以通过运行 **openstack baremetal node abort** 中断处于 **rescue wait** 状态的节点的救援操作。
rescue 失败	稳定	指示节点救援失败的置备状态。您可以使用以下命令将节点从 **rescue failed** 状态转换到以下状态之一： • **OpenStack baremetal node rescue → rescuing → rescue** • **OpenStack baremetal node unrescue → unrescuing → active** • **openstack baremetal node delete → deleting → available**
rescue	稳定	处于 **rescue** 状态的节点正在运行救援 ramdisk。裸机置备服务可能会定期收集带外传感器信息，包括电源状态。您可以使用以下命令将节点从 **rescue** 状态转换到以下状态之一： • **OpenStack baremetal node unrescue → unrescuing → active** • **openstack baremetal node delete → deleting → available**
unrescuing	过渡	处于 **unrescuing** 状态的节点已准备好从 **rescue** 状态过渡到 **active** 状态。
unrescue 失败	稳定	指示节点 unrescue 操作失败的置备状态。您可以使用以下命令将节点从 **unrescue failed** 状态转换到以下状态之一： • **OpenStack baremetal node rescue → rescuing → rescue** • **OpenStack baremetal node unrescue → unrescuing → active** • **openstack baremetal node delete → deleting → available**

4.6. 配置 REDFISH 虚拟介质引导



重要

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

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

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

4.6.1. 使用 Redfish 虚拟介质引导部署裸机服务器

重要

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

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

前提条件

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

流程

1. 将 Bare Metal 服务 (ironic) 引导接口设置为 **redfish-virtual-media**：

```
$ openstack baremetal node set --boot-interface redfish-virtual-media $NODE_NAME
```

   将 **$NODE_NAME** 替换为节点的名称。

2. 对于 UEFI 节点，将引导模式设置为 **uefi**：

```
$ openstack baremetal node set --property capabilities="boot_mode:uefi" $NODE_NAME
```

   将 **$NODE_NAME** 替换为节点的名称。

注意

对于 BIOS 节点，请不要完成此步骤。

3. 对于 UEFI 节点，定义 EFI 系统分区 (ESP) 镜像：

```
$ openstack baremetal node set --driver-info bootloader=$ESP $NODE_NAME
```

将 **$ESP** 替换为 glance 镜像 UUID 或 ESP 镜像的 URL，并将 **$NODE_NAME** 替换为节点的名称。

注意

对于 BIOS 节点，请不要完成此步骤。

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

```
$ openstack baremetal port create --pxe-enabled True --node $UUID $MAC_ADDRESS
```

将 **$UUID** 替换为裸机节点的 UUID，并将 **$MAC_ADDRESS** 替换为裸机节点上 NIC 的 MAC 地址。

5. 创建新的裸机服务器：

```
$ openstack server create \
    --flavor baremetal \
    --image $IMAGE \
    --network $NETWORK \
    test_instance
```

将 **$IMAGE** 和 **$NETWORK** 替换为您要使用的镜像和网络的名称。

第 5 章 管理裸机节点

部署包含裸机置备服务(ironic)的 overcloud 后，您可以在注册的裸机节点上置备物理计算机，并在 overcloud 中启动裸机实例。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

5.1. 启动裸机实例

您可以从命令行或 OpenStack 控制面板启动实例。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

5.1.1. 使用命令行界面启动实例

您可以使用 OpenStack 客户端 CLI 创建裸机实例。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

1. 将 shell 配置为以管理用户访问 Identity 服务(keystone)：

```
$ source ~/overcloudrc
```

2. 创建裸机实例：

```
$ openstack server create \
 --nic net-id=<network_uuid> \
 --flavor baremetal \
 --image <image_uuid> \
 myBareMetalInstance
```

   - 将 **<network_uuid>** 替换为您创建的网络的唯一标识符，以用于裸机置备服务。
   - 将 **<image_uuid>** 替换为包含实例需要的软件配置集的镜像的唯一标识符。

3. 检查实例的状态：

```
$ openstack server list --name myBareMetalInstance
```

5.1.2. 使用仪表板启动实例

使用仪表板图形用户界面部署裸机实例。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

1. 登录到位于 **http[s]://*DASHBOARD_IP*/dashboard** 的仪表板。
2. 点 **Project > Compute > Instances**
3. 单击 **Launch Instance**。
   - 在 **Details** 选项卡中，指定 **Instance Name**，然后选择 **1** 作为 **Count**。
   - 在 **Source** 选项卡中，从 **Select Boot Source** 中选择一个**镜像**，然后点 **+(** 加)符号来选择操作系统磁盘镜像。您选择要移到 **Allocated** 的镜像。
   - 在 **Flavor** 选项卡中，选择 **baremetal**。
   - 在**网络**选项卡中，使用 **+**（加）和 **-**（减）按钮，将所需的网络从 **Available** 移到 **Allocated**。确保为裸机置备服务创建的共享网络已在此处选择。
   - 如果要将实例分配给安全组，在 **Security Groups** 选项卡中，使用箭头将组移到 **Allocated**。
4. 单击 **Launch Instance**。

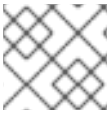
5.2. 在裸机置备服务中配置端口组

注意

本发行版本中，裸机节点的端口组 **功能作为技术预览提供**，因此不受红帽完全支持。它应该只用于测试，不应在生产环境中部署。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

端口组(bond)提供了一种将多个网络接口聚合到一个 'bonded' 接口的方法。端口组配置始终优先于单个端口配置。

如果端口组具有物理网络，则该端口组中的所有端口必须具有相同的物理网络。裸机置备服务使用 **configdrive** 支持配置实例中端口组。



注意

裸机置备服务 API 版本 1.26 支持端口组配置。.Prerequisites

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

5.2.1. 在交换机上配置端口组

要在裸机部署中配置端口组，您必须在交换机上手动配置端口组。您必须确保交换机上的模式和属性与裸机上的模式和属性对应，因为命名在交换机上有所不同。

注意

如果您需要使用 iPXE 引导部署，则无法将端口组用于置备和清理。

使用端口组回退时，端口组中的所有端口在连接失败时都可以回退到单独的交换机端口。根据交换机是否支持端口组回退，您可以使用 **--support-standalone-ports** 和 **--unsupport-standalone-ports** 选项。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

5.2.2. 在裸机置备服务中配置端口组

创建一个端口组，将多个网络接口聚合到一个 *绑定接口* 中。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

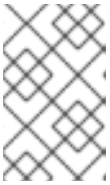
1. 通过指定它所属的节点、名称、地址、模式、属性以及它是否支持到独立端口的节点来创建端口组。

```
# openstack baremetal port group create --node NODE_UUID --name NAME --address
MAC_ADDRESS --mode MODE  --property miimon=100 --property
xmit_hash_policy="layer2+3" --support-standalone-ports
```

   您还可以使用 **openstack baremetal port group set** 命令来更新端口组。

   如果没有指定地址，部署的实例端口组地址与 OpenStack Networking 端口相同。如果没有附加 neutron 端口，则端口组配置会失败。

   在接口附加过程中，端口组的优先级高于端口，因此会首先使用它们。目前，无法在接口附加请求中指定端口组或端口。没有任何端口的端口组将被忽略。



   注意

   您必须在镜像或生成 **configdrive** 中手动配置端口组，并将其添加到节点的 **instance_info** 中。确保您有 **cloud-init** 版本 0.7.7 或更高版本，以使端口组配置正常工作。

2. 将端口与端口组关联：

   - 在创建端口期间：

```
# openstack baremetal port create --node NODE_UUID --address MAC_ADDRESS --
port-group test
```

   - 在端口更新过程中：

```
# openstack baremetal port set PORT_UUID --port-group PORT_GROUP_UUID
```

3. 通过提供具有 **cloud-init** 或支持绑定的镜像来引导实例。
   要检查端口组是否已正确配置，请运行以下命令：

```
# cat /proc/net/bonding/bondX
```

   这里，**X** 是 **cloud-init** 为每个配置的端口组自动生成的数字，从 **0** 开始，并为每个配置的端口组递增一个。

5.3. 确定主机到 IP 地址映射

使用以下命令，确定将哪些 IP 地址分配给哪个主机和裸机节点。使用这些命令，您可以在不直接访问主机的情况下从 undercloud 查看主机到 IP 映射。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

1. 运行以下命令以显示每个主机的 IP 地址：

```
(undercloud) [stack@host01 ~]$ openstack stack output show overcloud HostsEntry --max-width 80

+--------------+------------------------------------------------------------+
| Field        | Value                                                      |
+--------------+------------------------------------------------------------+
| description  | The content that should be appended to your /etc/hosts if you |
|              | want to get                                                |
|              | hostname-based access to the deployed nodes (useful for    |
|              | testing without                                            |
|              | setting up a DNS).                                         |
|              |                                                            |
| output_key   | HostsEntry                                                 |
| output_value | 172.17.0.10 overcloud-controller-0.localdomain overcloud-  |
|              | controller-0                                               |
|              | 10.8.145.18 overcloud-controller-0.external.localdomain    |
|              | overcloud-controller-0.external                            |
|              | 172.17.0.10 overcloud-controller-0.internalapi.localdomain |
|              | overcloud-controller-0.internalapi                         |
|              | 172.18.0.15 overcloud-controller-0.storage.localdomain     |
|              | overcloud-controller-0.storage                             |
|              | 172.21.2.12 overcloud-controller-0.storagemgmt.localdomain |
|              | overcloud-controller-0.storagemgmt                         |
|              | 172.16.0.15 overcloud-controller-0.tenant.localdomain      |
|              | overcloud-controller-0.tenant                              |
|              | 10.8.146.13 overcloud-controller-0.management.localdomain  |
|              | overcloud-controller-0.management                          |
|              | 10.8.146.13 overcloud-controller-0.ctlplane.localdomain    |
|              | overcloud-controller-0.ctlplane                            |
|              |                                                            |
|              | 172.17.0.21 overcloud-compute-0.localdomain overcloud-     |
```

```
|             | compute-0                                                  |
|             | 10.8.146.12 overcloud-compute-0.external.localdomain       |
|             | overcloud-compute-0.external                               |
|             | 172.17.0.21 overcloud-compute-0.internalapi.localdomain    |
|             | overcloud-compute-0.internalapi                            |
|             | 172.18.0.20 overcloud-compute-0.storage.localdomain        |
|             | overcloud-compute-0.storage                                |
|             | 10.8.146.12 overcloud-compute-0.storagemgmt.localdomain    |
|             | overcloud-compute-0.storagemgmt                            |
|             | 172.16.0.16 overcloud-compute-0.tenant.localdomain overcloud- |
|             | compute-0.tenant                                           |
|             | 10.8.146.12 overcloud-compute-0.management.localdomain     |
|             | overcloud-compute-0.management                             |
|             | 10.8.146.12 overcloud-compute-0.ctlplane.localdomain       |
|             | overcloud-compute-0.ctlplane                               |
|             |                                                            |
|             |                                                            |
|             |                                                            |
|             |                                                            |
|             | 10.8.145.16  overcloud.localdomain                         |
|             | 10.8.146.7  overcloud.ctlplane.localdomain                 |
|             | 172.17.0.19  overcloud.internalapi.localdomain             |
|             | 172.18.0.19  overcloud.storage.localdomain                 |
|             | 172.21.2.16  overcloud.storagemgmt.localdomain             |
+-------------+------------------------------------------------------------+
```

2. 要过滤特定主机，请运行以下命令：

```
(undercloud) [stack@host01 ~]$ openstack stack output show overcloud HostsEntry -c
output_value -f value | grep overcloud-controller-0

172.17.0.12 overcloud-controller-0.localdomain overcloud-controller-0
10.8.145.18 overcloud-controller-0.external.localdomain overcloud-controller-0.external
172.17.0.12 overcloud-controller-0.internalapi.localdomain overcloud-controller-0.internalapi
172.18.0.12 overcloud-controller-0.storage.localdomain overcloud-controller-0.storage
172.21.2.13 overcloud-controller-0.storagemgmt.localdomain overcloud-controller-
0.storagemgmt
172.16.0.19 overcloud-controller-0.tenant.localdomain overcloud-controller-0.tenant
10.8.146.13 overcloud-controller-0.management.localdomain overcloud-controller-
0.management
10.8.146.13 overcloud-controller-0.ctlplane.localdomain overcloud-controller-0.ctlplane
```

3. 要将主机映射到裸机节点，请运行以下命令：

```
(undercloud) [stack@host01 ~]$ openstack baremetal node list --fields uuid name
instance_info -f json
[
  {
    "UUID": "c0d2568e-1825-4d34-96ec-f08bbf0ba7ae",
    "Instance Info": {
      "root_gb": "40",
      "display_name": "overcloud-compute-0",
      "image_source": "24a33990-e65a-4235-9620-9243bcff67a2",
      "capabilities": "{\"boot_option\": \"local\"}",
      "memory_mb": "4096",
```

```
      "vcpus": "1",
      "local_gb": "557",
      "configdrive": "******",
      "swap_mb": "0",
      "nova_host_id": "host01.lab.local"
    },
    "Name": "host2"
  },
  {
    "UUID": "8c3faec8-bc05-401c-8956-99c40cdea97d",
    "Instance Info": {
      "root_gb": "40",
      "display_name": "overcloud-controller-0",
      "image_source": "24a33990-e65a-4235-9620-9243bcff67a2",
      "capabilities": "{\"boot_option\": \"local\"}",
      "memory_mb": "4096",
      "vcpus": "1",
      "local_gb": "557",
      "configdrive": "******",
      "swap_mb": "0",
      "nova_host_id": "host01.lab.local"
    },
    "Name": "host3"
  }
]
```

5.4. 附加和分离虚拟网络接口

裸机置备服务有一个 API，可用于管理虚拟网络接口之间的映射。例如，OpenStack 网络服务和物理接口(NIC)中的接口。您可以为每个裸机置备节点配置这些接口，将虚拟网络接口(VIF)设置为物理网络接口(PIF)映射逻辑。要配置接口，请使用 **openstack baremetal node vif*** 命令。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

1. 列出当前连接到裸机节点的 VIF ID：

```
$ openstack baremetal node vif list baremetal-0
+--------------------------------------+
| ID                                   |
+--------------------------------------+
| 4475bc5a-6f6e-466d-bcb6-6c2dce0fba16 |
+--------------------------------------+
```

2. 附加 VIF 后，裸机调配服务使用物理端口的实际 MAC 地址更新 OpenStack 网络服务中的虚拟端口。检查此端口地址：

```
$ openstack port show 4475bc5a-6f6e-466d-bcb6-6c2dce0fba16 -c mac_address -c fixed_ips
+-------------+-------------------------------------------------------------------------------+
| Field       | Value                                                                         |
```

```
+-------------+-------------------------------------------------------------------------+
| fixed_ips   | ip_address='192.168.24.9', subnet_id='1d11c677-5946-4733-87c3-
23a9e06077aa' |
| mac_address | 00:2d:28:2f:8d:95                                                       |
+-------------+-------------------------------------------------------------------------+
```

3. 在创建了 **baremetal-0** 节点的网络中创建一个新端口：

```
$ openstack port create --network baremetal --fixed-ip ip-address=192.168.24.24 baremetal-
0-extra
```

4. 从实例中删除端口：

```
$ openstack server remove port overcloud-baremetal-0 4475bc5a-6f6e-466d-bcb6-
6c2dce0fba16
```

5. 检查列表中是否不再存在 IP 地址：

```
$ openstack server list
```

6. 检查是否有 VIFs 附加到节点：

```
$ openstack baremetal node vif list baremetal-0
$ openstack port list
```

7. 添加新创建的端口：

```
$ openstack server add port overcloud-baremetal-0 baremetal-0-extra
```

8. 验证新 IP 地址是否显示新端口：

```
$ openstack server list
+--------------------------------------+------------------------+--------+-----------------------+----------
-------------------+---------+
| ID                                   | Name                   | Status | Networks              | Image
| Flavor  |
+--------------------------------------+------------------------+--------+-----------------------+----------
-------------------+---------+
| 53095a64-1646-4dd1-bbf3-b51cbcc38789 | overcloud-controller-2  | ACTIVE |
ctlplane=192.168.24.7  | overcloud-hardened-uefi-full | control |
| 3a1bc89c-5d0d-44c7-a569-f2a3b4c73d65 | overcloud-controller-0  | ACTIVE |
ctlplane=192.168.24.8  | overcloud-hardened-uefi-full | control |
| 6b01531a-f55d-40e9-b3a2-6d02be0b915b | overcloud-controller-1  | ACTIVE |
ctlplane=192.168.24.16 | overcloud-hardened-uefi-full | control |
| c61cc52b-cc48-4903-a971-073c60f53091 | overcloud-novacompute-0overcloud-baremetal-
0 | ACTIVE | ctlplane=192.168.24.24 | overcloud-hardened-uefi-full | compute |
+--------------------------------------+------------------------+--------+-----------------------+----------
-------------------+---------+
```

9. 检查 VIF ID 是否为新端口的 UUID：

```
$ openstack baremetal node vif list baremetal-0
+--------------------------------------+
```

```
| ID                                   |
+--------------------------------------+
| 6181c089-7e33-4f1c-b8fe-2523ff431ffc |
+--------------------------------------+
```

10. 检查 OpenStack 网络端口 MAC 地址是否已更新，并与裸机置备服务端口之一匹配：

```
$ openstack port show 6181c089-7e33-4f1c-b8fe-2523ff431ffc -c mac_address -c fixed_ips
+-------------+-------------------------------------------------------------------------------+
| Field       | Value                                                                         |
+-------------+-------------------------------------------------------------------------------+
| fixed_ips   | ip_address='192.168.24.24', subnet_id='1d11c677-5946-4733-87c3-
23a9e06077aa' |
| mac_address | 00:2d:28:2f:8d:95                                                             |
+-------------+-------------------------------------------------------------------------------+
```

11. 重启裸机节点，以便它识别新的 IP 地址：

```
$ openstack server reboot overcloud-baremetal-0
```

分离或附加接口后，裸机操作系统会删除、添加或修改已更改的网络接口。当您替换端口时，DHCP 请求会获取新的 IP 地址，但这可能需要一些时间，因为旧的 DHCP 租期仍然有效。要立即启动这些更改，请重启裸机主机。

5.5. 为裸机置备服务配置通知

您可以配置裸机置备服务(ironic)，以显示服务内发生的不同事件的通知。外部服务将这些通知用于计费目的、监控数据存储和其他目的。要为裸机置备服务启用通知，您必须在 **ironic.conf** 配置文件中设置以下选项。

先决条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

- **[DEFAULT]** 部分中的 **notification_level** 选项决定了发送通知的最低优先级级别。您可以将此选项的值设置为 **debug**、**info**、**warning**、**error** 或 **critical**。如果将选项设为 **warning**，则发送带有优先级级别 **warning**, **error**, 或 **critical** 的所有通知，但不发送优先级为 **debug** 或 **info** 的通知。如果没有设置这个选项，则不会发送通知。每个可用通知的优先级级别记录在以下。

- **[oslo_messaging_notifications]** 部分中的 **transport_url** 选项决定发送通知时使用的消息总线。如果没有设置，则使用 RPC 的默认传输。

所有通知都会在消息总线中的 **ironic_versioned_notifications** 主题上发出。通常，遍历消息总线的每种消息类型都与描述消息内容的主题相关联。

5.6. 配置自动电源故障恢复

Bare Metal Provisioning 服务(ironic)有一个字符串字段 错误，用于记录节点的电源、清理和救援中止失败。

表 5.1. Ironic 节点故障

Fulting	Description
电源失败	该节点处于维护模式，因为电源同步失败超过最大重试次数。
清理失败	由于清理操作失败，节点处于维护模式。
rescue abort 失败	该节点处于维护模式，因为 rescue 中止过程中清理操作失败。
none	不存在故障。

编排器定期检查此字段的值。如果编排器检测到 **电源故障** 状态，并可成功恢复到节点的电源，则会从维护模式中删除该节点，并恢复到操作。

注意

如果 Operator 手动将节点置于维护模式，则编排器不会自动从维护模式中删除该节点。

默认间隔为 300 秒，但您可以使用 hieradata 为 director 配置此间隔。

前提条件

- 一个成功的 overcloud 部署，其中包括裸机置备服务。如需更多信息，请参阅使用 裸机置备服务部署 overcloud。

流程

- 包含以下 hieradata 来配置自定义恢复间隔：

```
ironic::conductor::power_failure_recovery_interval
```

  要禁用自动电源故障恢复，请将值设为 **0**。

5.7. 内省 OVERCLOUD 节点

执行 overcloud 节点内省，以识别并存储 director 中节点的规格。

流程

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

```
$ source ~/overcloudrc
```

3. 运行内省命令：

```
$ openstack baremetal introspection start [--wait] <NODENAME>
```

将 <NODENAME> 替换为您要检查的节点的名称或 UUID。

4. 检查内省状态：

```
$ openstack baremetal introspection status <NODENAME>
```

将 <NODENAME> 替换为节点的名称或 UUID。

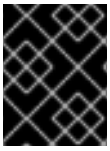
后续步骤

- 提取内省数据：

```
$ openstack baremetal introspection data save <NODE-UUID>
```

将 <NODENAME> 替换为节点的名称或 UUID。

第 6 章 配置裸机节点，以启用从可引导卷创建裸机实例



重要

此功能在 Red Hat OpenStack Platform 17.1 中已弃用。RHOSP 17.1 中提供了程序错误修正和支持，但不会进行新的功能增强。

您可以在块存储服务(cinder)中创建卷，并将这些卷连接到使用裸机置备服务(ironic)创建的裸机实例。

要让您的云用户从可引导卷创建裸机实例，请完成以下任务：

1. 配置每个裸机节点，以启用从可引导卷启动裸机实例。
2. 在引导磁盘中配置 iSCSI 内核参数。

6.1. 先决条件

- 裸机置备服务(ironic)通过 iSCSI 接口将裸机节点连接到块存储卷。因此，overcloud 必须使用块存储服务(cinder)的 iSCSI 后端进行部署。要为块存储服务启用 iSCSI 后端，请将 **CinderEnableIscsiBackend** 参数设置为 **true** 并部署 overcloud。

注意

您不能将 Block Storage 卷引导功能与 Red Hat Ceph Storage 后端一起使用。

6.2. 配置节点以从可引导卷创建裸机实例

您必须配置每个裸机节点，使其能够从可引导卷启动裸机实例。

流程

1. 提供 overcloud 凭证文件：

```
$ source ~/<credentials_file>
```

   - 将 **<credentials_file>** 替换为您的凭据文件的名称，如 **overcloudrc**。

2. 为每个裸机节点将 **iscsi_boot** 功能设置为 **true** ：

```
$ openstack baremetal node set --property capabilities=iscsi_boot:true <node_uuid>
```

   - 将 **<node_uuid>** 替换为裸机节点的 ID。

3. 为每个裸机节点将 **storage-interface** 设置为 **cinder** ：

```
$ openstack baremetal node set --storage-interface cinder <node_uuid>
```

4. 为节点创建 iSCSI 连接器：

```
$ openstack baremetal volume connector create --node <node_uuid> \
 --type iqn --connector-id <connector_id>
```

- 将 **<connector_id>** 替换为每个节点的唯一 ID，例如 **iqn.2010-10.org.openstack.node<NUM>**，其中 **<NUM>** 是每个节点的递增号。

6.3. 在引导磁盘中配置 ISCSI 内核参数

您必须配置实例镜像，以便在内核中启用 iSCSI 引导。

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. 从 Red Hat Enterprise Linux 产品软件下载页，以 QCOW2 格式下载 Red Hat Enterprise Linux KVM 镜像。
4. 将镜像复制到 undercloud 上的 **/home/stack/** 目录中。
5. 挂载 QCOW2 镜像，并以 **root** 用户身份访问它：
   a. 加载 **nbd** 内核模块：

```
$ sudo modprobe nbd
```

   b. 以 **/dev/nbd0** 身份连接 QCOW 镜像：

```
$ sudo qemu-nbd --connect=/dev/nbd0 <image>
```

   c. 检查 NBD 上的分区：

```
$ sudo fdisk /dev/nbd0 -l
```

   新的 Red Hat Enterprise Linux QCOW2 镜像仅包含一个分区，它通常在 NBD 中命名为 **/dev/nbd0p1**。

   d. 为镜像创建挂载点：

```
$ mkdir /tmp/mountpoint
```

   e. 挂载镜像：

```
$ sudo mount /dev/nbd0p1 /tmp/mountpoint/
```

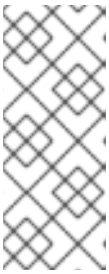
   f. 挂载您的 **dev** 目录，以便镜像能够访问主机上的设备信息：

```
$ sudo mount -o bind /dev /tmp/mountpoint/dev
```

   g. 将根目录改为挂载点：

```
$ sudo chroot /tmp/mountpoint /bin/bash
```

6. 在镜像中配置 iSCSI：



注意

此步骤中的一些命令可能会报告以下错误：

```
lscpu: cannot open /proc/cpuinfo: No such file or directory
```

此错误不重要，您可以忽略错误。

a. 将 **resolv.conf** 文件移到临时位置：

```
# mv /etc/resolv.conf /etc/resolv.conf.bak
```

b. 创建一个临时 **resolv.conf** 文件来解析 Red Hat Content Delivery Network 的 DNS 请求。这个示例使用 **8.8.8.8** 作为名称服务器：

```
# echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

c. 将挂载的镜像注册到 Red Hat Content Delivery Network 中：

```
# subscription-manager register
```

当命令提示您时，输入您的用户名和密码。

d. 附加包含 Red Hat Enterprise Linux 的订阅：

```
# subscription-manager list --all --available
# subscription-manager attach --pool <POOLID>
```

将 **<POOLID** > 替换为订阅的池 ID。

e. 禁用默认软件仓库：

```
# subscription-manager repos --disable "*"
```

f. 启用 Red Hat Enterprise Linux 软件仓库：

- Red Hat Enterprise Linux 7:

```
# subscription-manager repos --enable "rhel-7-server-rpms"
```

- Red Hat Enterprise Linux 8:

```
# subscription-manager repos --enable "rhel-8-for-x86_64-baseos-eus-rpms"
```

g. 安装 **iscsi-initiator-utils** 软件包：

```
# yum install -y iscsi-initiator-utils
```

h. 取消挂载的镜像：

```
# subscription-manager unregister
```

i. 恢复原始 **resolv.conf** 文件：

```
# mv /etc/resolv.conf.bak /etc/resolv.conf
```

j. 检查挂载的镜像中的内核版本：

```
# rpm -qa kernel
```

例如，如果输出为 **kernel-3.10.0-1062.el7.x86_64**，则内核版本为 **3.10.0-1062.el7.x86_64**。请注意下一步这个内核版本。

注意

新的 Red Hat Enterprise Linux QCOW2 镜像只有一个内核版本。如果安装了多个内核版本，请使用最新的内核版本。

k. 在 initramfs 镜像中添加 **network** 和 **iscsi** dracut 模块：

```
# dracut --force --add "network iscsi" /boot/initramfs-<KERNELVERSION>.img
<KERNELVERSION>
```

将 **<KERNELVERSION** > 替换为从 **rpm -qa 内核** 获取的版本号。以下示例使用 **3.10.0-1062.el7.x86_64** 作为内核版本：

```
# dracut --force --add "network iscsi" /boot/initramfs-3.10.0-1062.el7.x86_64.img 3.10.0-
1062.el7.x86_64
```

l. 从挂载的镜像退出回您的主机操作系统：

```
# exit
```

7. 卸载镜像：

a. 从临时挂载点卸载 **dev** 目录：

```
$ sudo umount /tmp/mountpoint/dev
```

b. 从挂载点卸载镜像：

```
$ sudo umount /tmp/mountpoint
```

c. 从 **/dev/nbd0/** 断开 QCOW2 镜像：

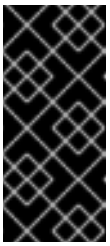
```
$ sudo qemu-nbd --disconnect /dev/nbd0
```

8. 在镜像上重建 **grub** 菜单配置：

a. 安装 **libguestfs-tools** 软件包：

```
$ sudo yum -y install libguestfs-tools
```

-



重要

如果在 undercloud 上安装 **libguestfs-tools** 软件包，请禁用 **iscsid.socket**，以避免在 undercloud 上与 **tripleo_iscsid** 服务冲突：

```
$ sudo systemctl disable --now iscsid.socket
```

b. 将 **libguestfs** 后端设置为直接使用 QEMU：

```
$ export LIBGUESTFS_BACKEND=direct
```

c. 更新镜像上的 grub 配置，并在引导磁盘上设置 **rd.iscsi.firmware=1** 内核参数：

```
$ guestfish -a /tmp/images/{{ dib_image }} -m /dev/sda3 sh "mount /dev/sda2 /boot/efi &&
rm /boot/grub2/grubenv && /sbin/grub2-mkconfig -o /boot/grub2/grub.cfg && cp
/boot/grub2/grub.cfg /boot/efi/EFI/redhat/grub.cfg && grubby --update-kernel=ALL --
args=\"rd.iscsi.firmware=1\" && cp /boot/grub2/grubenv /boot/efi/EFI/redhat/grubenv &&
echo Success"
```

9. 将启用了 iSCSI 的镜像上传到镜像服务(glance)：

```
$ openstack image create --disk-format qcow2 --container-format bare \
 --file <image> <image_name>
```

- 将 **<image** > 替换为启用了 iSCSI 的镜像的名称，如 **rhel-server-7.7-x86_64-kvm.qcow2**。
- 将 ** 替换为用于引用镜像的名称，如 **rhel-server-7.7-iscsi**。

6.4. 从可引导卷创建裸机实例

要验证裸机节点是否可以托管从可引导卷创建的裸机实例，请创建可引导卷并启动实例。

流程

1. 提供 overcloud 凭证文件：

```
$ source ~/<credentials_file>
```

- 将 **<credentials_file** > 替换为您的凭据文件的名称，如 **overcloudrc**。

2. 从启用了 iSCSI 的实例镜像创建卷：

```
$ openstack volume create --size 10 --image  --bootable myBootableVolume
```

- 将 ** 替换为要写入卷的镜像的名称或 ID，如 **rhel-server-7.7-iscsi**。

3. 创建使用引导卷的裸机实例：

```
$ openstack server create --flavor baremetal --volume myBootableVolume --key default
myBareMetalInstance
```

第 7 章 对裸机置备服务进行故障排除

诊断包含裸机置备服务(ironic)的环境中的问题。

7.1. PXE 引导错误

使用以下故障排除过程来评估您可能会在 PXE 引导中遇到的问题。

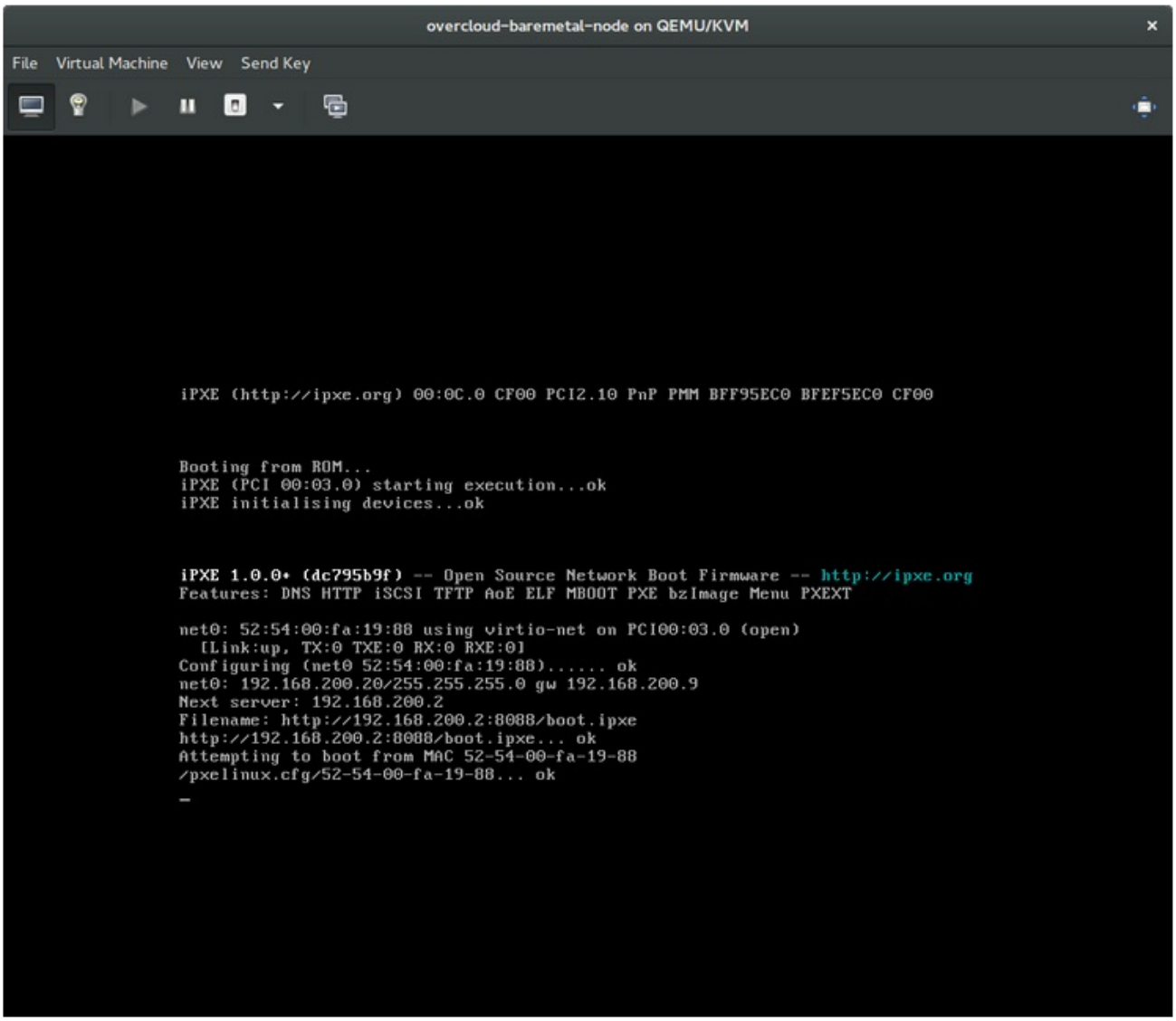
Permission Denied 错误

如果裸机节点的控制台返回 **Permission Denied** 错误，请确保已将适当的 SELinux 上下文应用到 **/httpboot** 和 **/tftpboot** 目录：

```
# semanage fcontext -a -t httpd_sys_content_t "/httpboot(/.*)?"
# restorecon -r -v /httpboot
# semanage fcontext -a -t tftpdir_t "/tftpboot(/.*)?"
# restorecon -r -v /tftpboot
```

引导过程冻结在 /pxelinux.cfg/XX-XX-XX-XX-XX-XX-XX

在节点的控制台中，如果看起来像您收到 IP 地址，但进程停止，您可能会在 **ironic.conf** 文件中使用错误的 PXE 引导条目。



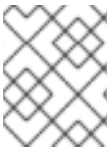
```
$ grep ^pxe_config_template ironic.conf
pxe_config_template=$pybasedir/drivers/modules/ipxe_config.template
```

默认模板是 **pxe_config.template**，因此可以轻松地省略 *i*，并意外输入 **ipxe_config.template**。

7.2. 裸机节点引导后登录错误

当使用您在配置过程中设置的 root 密码时，无法登录到节点，表示您没有引导到部署的镜像。您可以登录到 **deploy-kernel/deploy-ramdisk** 镜像，系统还没有加载正确的镜像。

要解决这个问题，请验证 Compute 或 Bare Metal Provisioning 服务节点上的 **/httpboot/pxelinux.cfg/MAC_ADDRESS** 中的 PXE 引导配置文件，并确保此文件中列出的所有 IP 地址是否与 Bare Metal 网络上的 IP 地址对应。



注意

Bare Metal Provisioning 服务节点使用的唯一网络是裸机网络。如果其中一个端点不在网络上，则端点无法作为引导过程的一部分访问裸机置备服务节点。

例如，您的文件中的 kernel 行如下：

```
kernel http://192.168.200.2:8088/5a6cdbe3-2c90-4a90-b3c6-85b449b30512/deploy_kernel selinux=0
disk=cciss/c0d0,sda,hda,vda iscsi_target_iqn=iqn.2008-10.org.openstack:5a6cdbe3-2c90-4a90-b3c6-
85b449b30512 deployment_id=5a6cdbe3-2c90-4a90-b3c6-85b449b30512
deployment_key=VWDYDVVEFCQJNOSTO9R67HKUXUGP77CK
ironic_api_url=http://192.168.200.2:6385 troubleshoot=0 text nofb nomodeset vga=normal
boot_option=netboot ip=${ip}:${next-server}:${gateway}:${netmask} BOOTIF=${mac}  ipa-api-
url=http://192.168.200.2:6385 ipa-driver-name=ipmi boot_mode=bios initrd=deploy_ramdisk
coreos.configdrive=0 || goto deploy
```

上例的 **kernel** 行中的值	对应信息
http://192.168.200.2:8088	**/etc/ironic/ironic.conf** 文件中的参数 **http_url**。此 IP 地址必须位于 Bare Metal 网络中。
5a6cdbe3-2c90-4a90-b3c6-85b449b30512	**ironic node-list** 中的 baremetal 节点的 UUID。
deploy_kernel	这是镜像服务中的部署内核镜像，它被复制为 **/httpboot/<NODE_UUID>/deploy_kernel**。
http://192.168.200.2:6385	**/etc/ironic/ironic.conf** 文件中的参数 **api_url**。此 IP 地址必须位于 Bare Metal 网络中。
ipmi	此节点的裸机置备服务使用的 IPMI 驱动程序。
deploy_ramdisk	这是镜像服务中的部署 ramdisk 镜像，其复制为 **/httpboot/<NODE_UUID>/deploy_ramdisk**。

上例的 kernel 行中的值	对应信息

如果值没有与 **/httpboot/pxelinux.cfg/MAC_ADDRESS** 和 **ironic.conf** 文件对应：

1. 更新 **ironic.conf** 文件中的值
2. 重启裸机置备服务
3. 重新部署裸机实例

7.3. 部署节点上的引导磁盘错误

对于某些硬件，您可能会遇到部署的节点在连续引导操作中无法从磁盘引导的问题。这通常是因为 BMC 不符合 director 在节点上请求的持久性引导设置。相反，节点从 PXE 目标引导。

在这种情况下，您必须在节点的 BIOS 中更新引导顺序。将 HDD 设置为第一个引导设备，然后将 PXE 设置为后续选项，以便节点默认从磁盘引导，但可以根据需要从网络启动。

注意

这个错误主要适用于使用 LegacyBIOS 固件的节点。

7.4. 裸机置备服务没有收到正确的主机名

如果裸机置备服务没有收到正确的主机名，这意味着 **cloud-init** 失败。要解决此问题，请将裸机子网连接到 OpenStack 网络服务中的路由器。此配置可以正确地将请求路由到 meta-data 代理。

7.5. 执行裸机置备服务命令时无效的 OPENSTACK IDENTITY 服务凭证

如果您无法验证身份服务，请检查 **ironic.conf** 文件中的 **identity_uri** 参数，并确保从 **keystone** AdminURL 中删除 **/v2.0**。例如，将 **identity_uri** 设置为 **http://IP:PORT**。

7.6. 硬件扩展

节点注册详情不正确可能会导致注册的硬件出现问题。确保正确输入属性名称和值。当您输入属性名称错误时，系统会向节点详情中添加属性，但忽略它们。

使用 **openstack baremetal node set** 命令更新节点详情。例如，将节点注册到 2 GB 的内存大小更新为 2 GB：

```
$ openstack baremetal node set --property memory_mb=2048 NODE_UUID
```

7.7. IDRAC 问题故障排除

Redfish 管理界面无法设置引导设备

当您将 **idrac-redfish** 管理界面与某些 iDRAC 固件版本搭配使用，并尝试在使用 UEFI 引导的裸机服务器上设置引导设备，S iDRAC 会返回以下错误：

```
Unable to Process the request because the value entered for the
parameter Continuous is not supported by the implementation.
```

如果您遇到这个问题，请将节点上的 **driver-info** 中的 **force_persistent_boot_device** 参数设置为 **Never** ：

```
openstack baremetal node set --driver-info force_persistent_boot_device=Never ${node_uuid}
```

关闭时超时

在关闭和超时时，一些服务器可能会太慢。默认重试计数为 **6**，这会导致 30 秒超时。要将超时时间增加到 90 秒，请在 undercloud hieradata 覆盖文件中将 **ironic::agent::rpc_response_timeout** 值设置为 **18**，并重新运行 **openstack undercloud install** 命令：

```
ironic::agent::rpc_response_timeout: 18
```

供应商透传超时

当 iDRAC 无法执行厂商透传命令时，这些命令需要很长时间且超时：

```
openstack baremetal node passthru call --http-method GET \
aed58dca-1b25-409a-a32f-3a817d59e1e0 list_unfinished_jobs
Timed out waiting for a reply to message ID 547ce7995342418c99ef1ea4a0054572 (HTTP 500)
```

要增加消息传递的超时时间，请增加 undercloud hieradata 覆盖文件中的 **ironic::default::rpc_response_timeout** 参数的值，并重新运行 **openstack undercloud install** 命令：

```
ironic::default::rpc_response_timeout: 600
```

7.8. 配置服务器控制台

overcloud 节点的控制台输出并不总是发送到服务器控制台。如果要在服务器控制台中查看此输出，您必须将 overcloud 配置为为您的硬件使用正确的控制台。使用以下方法之一执行此配置：

- 修改每个 overcloud 角色的 **KernelArgs** heat 参数。
- 自定义 director 用来置备 overcloud 节点的 **overcloud-hardened-uefi-full.qcow2** 镜像。

前提条件

- 成功安装 undercloud。如需更多信息，请参阅使用 director 安装和管理 Red Hat OpenStack Platform 指南。
- overcloud 节点已准备好进行部署。

在部署过程中使用 heat 修改 KernelArgs

1. 以 **stack** 用户身份登录 undercloud 主机。

2. 查找 **stackrc** 凭证文件：

```
$ source stackrc
```

3. 使用以下内容创建环境文件 **overcloud-console.yaml** ：

```
parameter_defaults:
  <role>Parameters:
    KernelArgs: "console=<console-name>"
```

将 &**lt;role** > 替换为您要配置的 overcloud 角色的名称，并将 < **console-name** > 替换为您要使用的控制台的 ID。例如，使用以下代码片段，将默认角色中的所有 overcloud 节点配置为使用 **tty0** ：

```
parameter_defaults:
  ControllerParameters:
    KernelArgs: "console=tty0"
  ComputeParameters:
    KernelArgs: "console=tty0"
  BlockStorageParameters:
    KernelArgs: "console=tty0"
  ObjectStorageParameters:
    KernelArgs: "console=tty0"
  CephStorageParameters:
    KernelArgs: "console=tty0"
```

4. 使用 **-e** 选项，将 **overcloud-console-tty0.yaml** 文件包含在部署命令中。

修改 overcloud-hardened-uefi-full.qcow2 镜像

1. 以 **stack** 用户身份登录 undercloud 主机。

2. 查找 **stackrc** 凭证文件：

```
$ source stackrc
```

3. 修改 **overcloud-hardened-uefi-full.qcow2** 镜像中的内核参数，为您的硬件设置正确的控制台。例如，将控制台设置为 **tty1** ：

```
$ virt-customize --selinux-relabel -a overcloud-hardened-uefi-full.qcow2 --run-command
'grubby --update-kernel=ALL --args="console=tty1"'
```

4. 将镜像导入到 director：

```
$ openstack overcloud image upload --image-path overcloud-hardened-uefi-full.qcow2
```

5. 部署 overcloud。

验证

1. 从 undercloud 登录 overcloud 节点：

```
$ ssh tripleo-admin@<IP-address>
```

将 **<IP-address** > 替换为 overcloud 节点的 IP 地址。

2. 检查 **/proc/cmdline** 文件的内容，并确保将 **console=** 参数设置为您要使用的控制台的值：

```
[tripleo-admin@controller-0 ~]$ cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos2)/boot/vmlinuz-4.18.0-193.29.1.el8_2.x86_64
root=UUID=0ec3dea5-f293-4729-b676-5d38a611ce81 ro console=tty0
console=ttyS0,115200n8 no_timer_check crashkernel=auto rhgb quiet
```

第 8 章 裸机驱动程序

您可以将裸机节点配置为使用裸机置备服务中启用的驱动程序之一。每个驱动程序都包含一个调配方法和电源管理类型。有些驱动程序需要额外的配置。本节中描述的每个驱动程序都使用 PXE 进行置备。驱动程序通过其电源管理类型列出。

您可以通过在 **ironic.yaml** 文件中配置 **IronicEnabledHardwareTypes** 参数来添加驱动程序。默认情况下启用 **ipmi** 和 **redfish**。

有关支持的插件和驱动程序的完整列表，请参阅 Red Hat OpenStack Platform 中的组件、插件和驱动程序支持。

8.1. 智能平台管理接口(IPMI)电源管理驱动程序

IPMI 是一个提供远程功能的接口，包括电源管理和服务器监控。要使用此电源管理类型，所有裸机置备服务节点都需要一个连接到共享裸机网络的 IPMI。IPMI 电源管理器驱动程序使用 **ipmitool** 程序远程管理硬件。您可以使用以下 **driver_info** 属性为节点配置 IPMI 电源管理器驱动程序：

表 8.1. IPMI driver_info 属性

属性	Description	等效的 ipmitool 选项
ipmi_address	（必需）节点的 IP 地址或主机名。	**-H**
ipmi_username	IPMI 用户名。	**-U**
ipmi_password	IPMI 密码。密码被写入一个临时文件。您可以使用 **-f** 选项将文件名传递给 **ipmitool**。	**-f**
ipmi_hex_kg_key	IPMIv2 身份验证的十六进制 Kg 密钥。	**-y**
ipmi_port	远程 IPMI RMCP 端口。	**-p**
ipmi_priv_level	IPMI 特权级别。设置为以下有效值之一： • **ADMINISTRATOR**（默认） • **回调** • **OPERATOR** • **USER**	**-L**
ipmi_protocol_version	IPMI 协议的版本。设置为以下有效值之一： • **1.5** 用于 lan • **2.0** 用于 lanplus（默认）	**-I**

属性	Description	等效的 ipmitool 选项
ipmi_bridging	桥接类型。与嵌套机箱管理控制器(CMC)一起使用。设置为以下有效值之一： • **单个** • **dual** • **no**（默认）	不适用
ipmi_target_channel	网桥请求的目的地频道。仅在 **ipmi_bridging** 设置为 **single** 或 **dual** 时才需要。	**-b**
ipmi_target_address	网桥请求的目的地地址。仅在 **ipmi_bridging** 设置为 **single** 或 **dual** 时才需要。	**-t**
ipmi_transit_channel	桥接请求的传输频道。仅在 **ipmi_bridging** 设置为 **dual** 时才需要。	**-B**
ipmi_transit_address	网桥请求的传输地址。仅在 **ipmi_bridging** 设置为 **dual** 时才需要。	**-T**
ipmi_local_address	网桥请求的本地 IPMB 地址。仅在 **ipmi_bridging** 设置为 **single** 或 **dual** 时才使用。	**-m**
ipmi_force_boot_device	设置为 **true** 以指定 Bare Metal Provisioning 服务是否应在每次服务器打开时将引导设备指定为 BMC。BMC 无法记住所选引导设备跨电源周期。默认禁用此选项。	不适用
ipmi_disable_boot_timeout	设置为 **false**，不发送原始 IPMI 命令，以禁用节点上引导的 60 秒超时。	不适用
ipmi_cipher_suite	要在节点上使用的 IPMI 密码套件版本。设置为以下有效值之一： • **3** 用于带有 SHA1 的 AES-128 • **17** 对于带有 SHA256 的 AES-128	不适用

8.2. REDFISH

由分布式管理任务组(DMTF)开发的 IT 基础架构的标准 RESTful API。您可以使用以下 **driver_info** 属性将 Bare Metal Provisioning serive (ironic)连接到 Redfish：

表 8.2. Redfish driver_info 属性

属性	Description
redfish_address	（必需） Redfish 控制器的 IP 地址。地址必须包含 URL 的授权部分。如果您没有包含方案，则默认为 **https**。
redfish_system_id	Redfish 驱动程序与之交互的系统资源的规范路径。该路径必须在与 **redfish_address** 属性相同的颁发机构中包含根服务、版本和系统的唯一路径。例如：**/redfish/v1/Systems/CX34R87**.只有在目标 BMC 管理多个资源时，才需要此属性。
redfish_username	Redfish 用户名。
redfish_password	Redfish 密码。
redfish_verify_ca	布尔值、到 CA_BUNDLE 文件的路径，或者包含可信 CA 证书的目录。如果将此值设置为 **True**，则驱动程序会验证主机证书。如果将此值设置为 **False**，则驱动程序忽略验证 SSL 证书。如果将此值设置为路径，则驱动程序将使用指定的证书或目录中的其中一个证书。默认值为 **True**。
redfish_auth_type	Redfish HTTP 客户端验证方法。设置为以下有效值之一： • **基本的** • **会话** （推荐） • **auto** （默认）- 在可用时使用 **会话** 验证方法，并在 **会话** 方法不可用时使用 **基本身份验证** 方法。

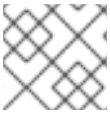
8.3. DELL REMOTE ACCESS CONTROLLER (DRAC)

DRAC 是一个提供远程功能的接口，包括电源管理和服务器监控。要使用此电源管理类型，所有裸机置备服务节点都需要一个连接到共享裸机置备网络的 DRAC。启用 **idrac** 驱动程序，并在节点的 **driver_info** 中设置以下信息：

- **drac_address** - DRAC NIC 的 IP 地址。
- **drac_username** - DRAC 用户名。
- **drac_password** - DRAC 密码。
- 可选： **drac_port** - 用于 WS-Management 端点的端口。默认值为端口 **443**。
- 可选： **drac_path** - 用于 WS-Management 端点的路径。默认路径为 **/wsman**。
- 可选： **drac_protocol** - 用于 WS-Management 端点的协议。有效值： **http**、**https**.默认协议是 **https**。

8.4. 集成的远程管理控制器(IRMC)

Fujitsu 的 iRMC 是一个提供远程管理功能的接口，包括电源管理和服务器监控。要在裸机置备服务节点上使用此电源管理类型，该节点需要一个连接到共享 Bare Metal 网络的 iRMC 接口。



注意

要使用 iRMC 驱动程序，需要 iRMC S4 或更高版本。

您可以使用以下 **driver_info** 属性为节点配置 iRMC 驱动程序：

表 8.3. iRMC driver_info 属性

属性	描述
irmc_address	iRMC 接口 NIC 的 IP 地址。
irmc_username	iRMC 用户名。
irmc_password	iRMC 密码。
irmc_snmp_version	设置为 **v3**。如果在 RHOSP 环境中启用了 FIPS 安全性，则需要此项。
irmc_snmp_user	设置为目标裸机节点上运行的 iRMC 固件的 SNMPv3 基于用户的安全模型 (USM)用户名。必须为每个裸机节点设置。SNMP 用户名不能是数字的字符串 (0-9)。 如果在 RHOSP 环境中启用了 FIPS 安全性，则需要此项。
irmc_snmp_auth_password	设置为 SNMPv3 用户名的 SNMPv3 消息身份验证密钥。SNMP 密码的最小长度必须为 8 个字符。 如果在 RHOSP 环境中启用了 FIPS 安全性，则需要此项。
irmc_snmp_priv_password	设置为 SNMPv3 用户名的 SNMPv3 消息隐私密钥。SNMP 密码的最小长度必须为 8 个字符。 如果在 RHOSP 环境中启用了 FIPS 安全性，则需要此项。
irmc_snmp_auth_proto	根据在 Fujitsu 服务器上运行的 iRMC 固件版本，设置为以下值之一： ● 早于 "iRMC S6": **sha** ● "iRMC S6": **sha256**,**sha384**, 或 **sha512** 如果在 RHOSP 环境中启用了 FIPS 安全性，则需要此项。
irmc_snmp_priv_proto	将 设置为 **aes**。如果在 RHOSP 环境中启用了 FIPS 安全性，则需要此项。

要使用 IPMI 设置引导模式或 SCCI 来获取传感器数据，您必须完成以下步骤：

1. 在 **ironic.conf** 文件中启用 sensor 方法：

```
$ openstack-config --set /etc/ironic/ironic.conf \
  irmc sensor_method <method>
```

- 将 &**lt;method&** gt; 替换为 **scci** 或 **ipmitool**。

2. 如果启用了 SCCI，请安装 **python-scciclient** 软件包：

```
# dnf install python-scciclient
```

3. 重启裸机编排器服务：

```
# systemctl restart openstack-ironic-conductor.service
```

8.5. INTEGRATED LIGHTS-OUT (ILO)

iLO 是惠普提供的一个远程电源功能的接口，这些功能包括电源管理和服务器监控。要使用此电源管理类型，所有裸机节点都需要一个连接到共享裸机网络的 iLO 接口。启用 **ilo** 驱动程序，并在节点的 **driver_info** 中设置以下信息：

- **ilo_address** - iLO 接口 NIC 的 IP 地址。
- **ilo_username** - iLO 用户名。
- **ilo_password** - iLO 密码。