Red Hat OpenStack Platform 17.1

为 overcloud 配置 IPv6 网络

将 overcloud 配置为使用 IPv6 网络

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 为 overcloud 配置 IPv6 网络

将 overcloud 配置为使用 IPv6 网络

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南提供有关使用 Red Hat OpenStack Platform director 的信息，它会创建一个使用 IPv6 进行端点的 overcloud。这包括 director 如何部署基于 IPv6 的 overcloud 和配置选项来达到此目的。

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 OVERCLOUD 的 IPV6 简介

Red Hat OpenStack Platform director 创建名为 overcloud 的云环境。默认情况下，overcloud 使用互联网协议版本 4 (IPv4) 来配置服务端点。但是，overcloud 还支持互联网协议版本 6 (IPv6)端点，这对支持 IPv6 基础架构的组织很有用。

此信息是对 使用 director 安装和管理 Red Hat OpenStack Platform 的补充。使用 director 安装和管理 Red Hat OpenStack Platform 中指定的相同要求也适用于本指南。根据需要实施这些要求。

1.1. IPV6 网络简介

IPv6 是互联网协议标准的最新版本。Internet Engineering Task Force (IETF)开发了 IPv6，作为防止当前通用 IPv4 标准的 IP 地址耗尽的方法。IPv6 与 IPv4 的各种区别，包括：

大 IP 地址范围

IPv6 范围大于 IPv4 范围。

更好的端到端连接

由于对网络地址转换的影响较低，IP 范围越大，可提供更好的端到端连接。

没有广播

IPv6 不支持传统的 IP 广播。相反，IPv6 使用多播以分级方式将数据包发送到适用的主机。

无状态地址自动配置(SLAAC)

IPv6 提供自动配置 IP 地址以及检测网络中重复地址的功能。这可减少对 DHCP 服务器进行分配地址的依赖。

IPv6 使用 128 位（代表使用组 16 位的 4 位十六进制）来定义地址，而 IPv4 只使用 32 位（代表使用组 8 位的十进制数字）。例如，IPv4 地址(192.168.0.1)的表示如下所示：

位	表示
11000000	192
10101000	168
00000000	0
00000001	1

对于 IPv6 地址(2001:db8:88ec:9fb3::1)，表示类似如下：

位	表示
0010 0000 0000 0001	2001
0000 1101 1011 1000	0db8
1000 1000 1110 1100	88ec

位	表示
1001 1111 1011 0011	9fb3
0000 0000 0000 0000	0000
0000 0000 0000 0000	0000
0000 0000 0000 0000	0000
0000 0000 0000 0001	0001

在代表 IPv6 地址时，每个位组中可以没有前面的零，并为每个 IP 地址省略一组零位组。在本例中，您可以使用 **db8** 来表示 **0db8** 位分组，省略三组的 **0000** 位组，这可以将 **2001:0db8:88ec:9fb3:0000:0000:0000:0001** 缩短为 **2001:db8:88ec:9fb3::1**。如需更多信息，请参阅 "RFC 5952: IPv6 Address Text Representation"

IPv6 中的子网划分

与 IPv4 类似，IPv6 地址使用位掩码将地址前缀定义为其网络。例如，如果您对示例 IP 地址 **2001:db8:88ec:9fb3::1/64** 包含一个 **/64** 位掩码，则位掩码充当定义前 64 位(**2001:db8:88ec:9fb**3)的前缀。剩余位 (**0000:0000:0000:0001**) 用于定义主机。

IPv6 也使用一些特殊的地址类型：

loopback

回环设备使用 IPv6 作为主机内部通信。此设备始终为 **::1/128**。

链接本地

链接本地地址是在特定网络段中有效的 IP 地址。IPv6 要求每个网络设备具有链路本地地址，并使用前缀 **fe80::/10**。但是，大多数时候，这些地址的前缀为 **fe80::/64**。

唯一本地

唯一的本地地址用于本地通信。这些地址使用 **fc00::/7** 前缀。

多播

主机使用多播地址来加入多播组。这些地址使用 **ff00::/8** 前缀。例如，**FF02::1** 是网络中所有节点的多播组，**FF02::2** 则是所有路由的多播组。

全球广播

这些地址通常为公共 IP 地址保留。这些地址使用 **2000::/3** 前缀。

1.2. 在 RED HAT OPENSTACK PLATFORM 中使用 IPV6

Red Hat OpenStack Platform (RHOSP) director 将 OpenStack 服务映射到隔离的网络。这些网络包括：

- 内部 API
- 存储
- 存储管理
- 项目（租户）网络(Neutron VLAN 模式)

- 外部

有关这些网络流量类型的更多信息，请参阅使用 director 安装和管理 Red Hat OpenStack Platform 指南。

director 还提供了将这些网络使用 IPv6 通信的方法。这意味着所需的 OpenStack 服务、数据库和其他相关服务使用 IPv6 地址进行通信。这也适用于使用涉及多个 Controller 节点的高可用性解决方案的环境。这有助于组织将 RHOSP 与 IPv6 基础架构集成。

使用下表作为 RHOSP 网络支持 IPv6 的指南：

网络类型	支持的互联网协议(IP)	备注
内部 API	• IPv6 • IPv4	
存储	• IPv6 • IPv4	
存储管理	• IPv6 • IPv4	
项目网络	• Dual-stack (IPv4/v6) • IPv6 • IPv4	
项目网络端点	• 双堆栈(IPv4/v6) • IPv6 • IPv4	这指的是托管项目网络隧道的网络的 IP 地址，而不是项目网络本身。 用于网络端点的 IPv6 仅支持 VXLAN 和 Geneve。尚不支持通用路由封装(GRE)。
外部 - 公共 API （和 Horizon）	• IPv6 • IPv4	

网络类型	支持的互联网协议(IP)	备注
外部 - 浮动 IP	• 双堆栈(IPv4/v6) • IPv4	IPv6 使用全局广播地址(GUAs)而不是 NAT 和浮动 IP 地址。Networking (neutron)服务要求项目网络之间的 IPv6 寻址使用 GUAs，在项目网络上 GUAs 没有重叠，因此可以在没有 NAT 的情况下路由。 使用双堆栈(IPv4/v6)时，您可以使用浮动 IP 地址仅访问 IPv4 子网上的 IP 地址。
提供商网络	• 双堆栈(IPv4/v6) • IPv6 • IPv4	IPv6 支持取决于项目操作系统。
置备(PXE/DHCP)	• IPv6 • IPv4	
IPMI 或其他 BMC	• IPv6 • IPv4	RHOSP 通过 Provisioning 网络与基板管理控制器(BMC)接口通信。 如果 BMC 接口支持双堆栈 IPv4 或 IPv6，则不属于 RHOSP 的工具可以使用 IPv6 与 BMC 通信。
overcloud Provisioning 网络	IPv6	用于 overcloud 中的 ironic 的 Provisioning 网络。
Overcloud Cleaning 网络	无	用于清理机器的隔离网络，然后再准备好重复使用。

定义场景

本指南的场景是使用 IPv6 的隔离网络创建 overcloud。使用 heat 模板和环境文件来配置网络隔离。此情境还为这些 heat 模板和环境文件提供了特定的变体，以演示配置中的特定区别。

注意

在这种情况下，undercloud 仍然使用 IPv4 连接 PXE 启动、内省、部署和其他服务。

本指南使用类似于高级 overcloud 场景的场景。主要区别在于 Ceph Storage 节点的省略。

有关这种情况的更多信息，请参阅使用 director 安装和管理 Red Hat OpenStack Platform 指南。

重要

本指南使用 2001:DB8::/32 IPv6 前缀作为文档目的，如 RFC 3849 中定义的文档。确保您从您自己的网络中替换这些示例地址作为 IPv6 地址。

第 2 章 为 IPV6 配置 OVERCLOUD

以下章节提供了运行 **openstack overcloud deploy** 命令前所需的配置。这包括准备节点以进行调配、在 undercloud 上配置 IPv6 地址，以及创建网络环境文件来为 overcloud 定义 IPv6 参数。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 您的网络支持 IPv6 原生 VLAN 和 IPv4 原生 VLAN。

2.1. 在 UNDERCLOUD 上配置 IPV6 地址

undercloud 需要访问 overcloud 公共 API，该 API 位于外部网络上。要达到此目的，undercloud 主机在连接到外部网络的接口上需要一个 IPv6 地址。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 您的网络支持 IPv6 原生 VLAN 和 IPv4 原生 VLAN。
- undercloud 可用的 IPv6 地址。

原生 VLAN 或专用接口

如果 undercloud 使用原生 VLAN 或附加到外部网络的专用接口，请使用 **ip** 命令向接口添加 IPv6 地址。在本例中，专用接口是 **eth0** ：

```
$ sudo ip link set dev eth0 up; sudo ip addr add 2001:db8::1/64 dev eth0
```

中继 VLAN 接口

如果 undercloud 在与 control plane 网桥(**br-ctlplane**)相同的接口上使用中继的 VLAN 来访问外部网络，请创建一个新的 VLAN 接口，将它附加到 control plane，并将 IPv6 地址添加到 VLAN。在本例中，外部网络 VLAN ID 为 **100** ：

```
$ sudo ovs-vsctl add-port br-ctlplane vlan100 tag=100 -- set interface vlan100 type=internal
$ sudo ip l set dev vlan100 up; sudo ip addr add 2001:db8::1/64 dev vlan100
```

确认 IPv6 地址

使用 **ip** 命令确认添加 IPv6 地址：

```
$ ip addr
```

IPv6 地址会出现在所选接口上。

设置持久的 IPv6 地址

要使 IPv6 地址永久生效，请在 **/etc/sysconfig/network-scripts/** 中修改或创建适当的接口文件。在本例中，在 **ifcfg-eth0** 或 **ifcfg-vlan100** 中包含以下行：

```
IPV6INIT=yes
IPV6ADDR=2001:db8::1/64
```

如需更多信息，请参阅红帽客户门户网站 中如何为 IPv6 配置网络接口？

2.2. 为 IPV6 部署注册和检查节点

节点定义模板(**instackenv.json**)是一个 JSON 格式文件，其中包含注册节点的硬件和电源管理详情。例如：

```
{
    "nodes":[
        {
            "mac":[
                "bb:bb:bb:bb:bb:bb"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.205"
        },
        {
            "mac":[
                "cc:cc:cc:cc:cc:cc"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.206"
        },
        {
            "mac":[
                "dd:dd:dd:dd:dd:dd"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.207"
        },
        {
            "mac":[
                "ee:ee:ee:ee:ee:ee"
            ],
```

```
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.208"
    }
    {
        "mac":[
            "ff:ff:ff:ff:ff:ff"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.209"
    }
    {
        "mac":[
            "gg:gg:gg:gg:gg:gg"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.210"
    }
  ]
}
```

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 可用于 overcloud 部署的节点。

流程

1. 创建节点定义模板后，将文件保存到 stack 用户的主目录(**/home/stack/instackenv.json**)，然后将其导入到 director：

```
$ openstack overcloud node import ~/instackenv.json
```

   此命令导入模板，并将每个节点从模板注册到 director。

2. 将内核和 ramdisk 镜像分配给所有节点：

```
$ openstack overcloud node configure
```

现在，节点已在 director 中注册和配置。

验证步骤

- 注册节点后，检查每个节点的硬件属性：

```
$ openstack overcloud node introspect --all-manageable
```

重要

节点必须处于 **manageable** 状态。确保此进程完成运行。它可能需要 15 分钟来检查这些裸机节点。

2.3. 为 IPV6 部署标记节点

注册并检查节点的硬件后，将每个节点标记为特定的配置集。这些配置集标签将节点映射到类别，并将类别分配给部署角色。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。

流程

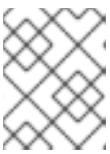
1. 检索节点列表来识别它们的 UUID：

```
$ ironic node-list
```

2. 在每个节点的 **properties/capabilities** 参数中添加 profile 选项。例如，要标记三个节点以使用控制器配置集和三个节点来使用 compute 配置集，请使用以下命令：

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 484587b2-b3b3-40d5-925b-a26a2fa3036f add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d010460b-38f2-4800-9cc4-d69f0d067efe add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d930e613-3e14-44b9-8240-4f3559801ea6 add
properties/capabilities='profile:compute,boot_option:local'
```

添加 **profile:control** 和 **profile:compute** 选项会将节点标记为每个对应的配置集。



注意

作为手动标记的替代选择，请使用自动配置集标记来根据基准测试数据标记大量节点。

2.4. 配置 IPV6 网络

默认情况下，overcloud 使用互联网协议版本 4 (IPv4) 来配置服务端点。但是，overcloud 还支持互联网协议版本 6 (IPv6)端点，这对支持 IPv6 基础架构的组织很有用。director 包含一组环境文件，可用于创建基于 IPv6 的 Overcloud。

有关在 Overcloud 中配置 IPv6 的更多信息，请参阅 overcloud 专用配置 IPv6 网络指南。

2.4.1. 配置可组合 IPv6 网络

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 您的网络支持 IPv6 原生 VLAN 和 IPv4 原生 VLAN。

流程

1. 复制默认 **network_data** 文件：

```
$ cp /usr/share/openstack-tripleo-heat-templates/network_data.yaml /home/stack/.
```

2. 编辑 **network_data.yaml** 文件的本地副本，并修改参数以满足您的 IPv6 网络要求。例如，External 网络包含以下默认网络详情：

```
- name: External
  vip: true
  name_lower: external
  vlan: 10
  ipv6: true
  ipv6_subnet: '2001:db8:fd00:1000::/64'
  ipv6_allocation_pools: [{'start': '2001:db8:fd00:1000::10', 'end':
'2001:db8:fd00:1000:ffff:ffff:ffff:fffe'}]
  gateway_ipv6: '2001:db8:fd00:1000::1'
```

   - **name** 是唯一的强制值，但您也可以使用 **name_lower** 规范化名称以提高可读性。例如，将 **InternalApi** 更改为 **internal_api**。
   - VIP ： **true** 在新网络上创建一个虚拟 IP 地址(VIP)，并将其余参数设置新网络的默认值。
   - **ipv6** 定义是否启用 IPv6。
   - **ipv6_subnet** 和 **ipv6_allocation_pools**，并且 **gateway_ip6** 为网络设置默认 IPv6 子网和 IP 范围。

3. 使用 **-n** 选项包括带有部署的自定义 **network_data** 文件。如果没有 **-n** 选项，部署命令将使用默认网络详细信息。

2.4.2. overcloud 中的 IPv6 网络隔离

默认情况下，overcloud 将服务分配给 provisioning 网络。但是，director 可以将 overcloud 网络流量划分为隔离的网络。这些网络在部署命令行中包含的文件中定义，默认名为 **network_data.yaml**。

当服务使用 IPv6 地址侦听网络时，您必须提供参数默认值，以指示该服务在 IPv6 网络上运行。每个服务所运行的网络都由文件 **network/service_net_map.yaml** 定义，可通过为各个 **ServiceNetMap** 条目声明参数默认值来覆盖。这些服务需要在环境文件中设置参数：

```
parameter_defaults:
  # Enable IPv6 for Ceph.
  CephIPv6: True
  # Enable IPv6 for Corosync. This is required when Corosync is using an IPv6 IP in the cluster.
  CorosyncIPv6: True
  # Enable IPv6 for MongoDB. This is required when MongoDB is using an IPv6 IP.
  MongoDbIPv6: True
  # Enable various IPv6 features in Nova.
  NovaIPv6: True
  # Enable IPv6 environment for RabbitMQ.
  RabbitIPv6: True
  # Enable IPv6 environment for Memcached.
  MemcachedIPv6: True
  # Enable IPv6 environment for MySQL.
  MysqlIPv6: True
  # Enable IPv6 environment for Manila
  ManilaIPv6: True
  # Enable IPv6 environment for Redis.
  RedisIPv6: True
```

核心 heat 模板中的 **environments/network-isolation.j2.yaml** 文件是一个 Jinja2 文件，该文件在可组合网络文件中为每个 IPv6 网络定义所有端口和 VIP。当呈现时，它会在带有完整资源 registry 的同一位置生成一个 **network-isolation.yaml** 文件。

2.4.3. 配置 IPv6 隔离网络

默认的 heat 模板集合包含用于默认网络配置的基于 Jinja2 的环境文件。此文件是 **environments/network-environment.j2.yaml**。当使用我们的 **network_data** 文件呈现时，它会生成名为 **network-environment.yaml** 的标准 YAML 文件。此文件的某些部分可能需要覆盖。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 您的网络支持 IPv6 原生 VLAN 和 IPv4 原生 VLAN。

流程

- 创建一个自定义环境文件(**/home/stack/network-environment.yaml**)，详情如下：

  ```
  parameter_defaults:
    ControlPlaneDefaultRoute: 192.0.2.1
    ControlPlaneSubnetCidr: "24"
  ```

  parameter_defaults 部分包含对在 IPv4 上保留的某些服务的自定义。

2.4.4. IPv6 网络接口模板

overcloud 需要一组网络接口模板。director 包含一组基于 Jinja2 的 Heat 模板，它根据您的 **network_data** 文件呈现：

NIC 目录	描述	环境文件
single-nic-vlans	将 control plane 和 VLAN 附加到默认 Open vSwitch 网桥的单 NIC（**nic1**）。	**environments/net-single-nic-with-vlans-v6.j2.yaml**
single-nic-linux-bridge-vlans	带有 control plane 和 VLAN 附加到默认 Linux 网桥的单个 NIC（**nic1**）。	**environments/net-single-nic-linux-bridge-with-vlans-v6.yaml**
bond-with-vlans	附加到 **nic1** 的 control plane。默认 Open vSwitch 网桥带有绑定的 NIC 配置（**nic2** 和 **nic3**）并附加了 VLAN。	**environments/net-bond-with-vlans-v6.yaml**
multiple-nics	附加到 **nic1** 的 control plane。将每个后续 NIC 分配给 **network_data** 文件中定义的每个网络。默认情况下，Storage 到 **nic2**, Storage Management 到 **nic3**, Internal API 到 **nic4**, Tenant 到 **br-tenant** 网桥上的 **nic5**，External 到默认 Open vSwitch 网桥上的 **nic6**。	**environments/net-multiple-nics-v6.yaml**

2.5. 部署 IPV6 OVERCLOUD

若要部署使用 IPv6 网络的 overcloud，必须在部署命令中包括额外的参数。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。

流程

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml \
  -e /home/stack/templates/network-environment.yaml \
  --ntp-server pool.ntp.org \
  [ADDITIONAL OPTIONS]
```

以上命令使用以下选项：

- **--templates** - 从默认的 heat 模板集合创建 overcloud。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml** - 向 overcloud 部署添加额外的环境文件。在这种情况下，它是初始化 IPv6 的网络隔离配置的环境文件。

- **-e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml** - 向 overcloud 部署添加额外的环境文件。在这种情况下，它是初始化 IPv6 的网络隔离配置的环境文件。

- **-e /home/stack/network-environment.yaml** - 向 overcloud 部署添加额外的环境文件。在本例中，它包含与 IPv6 相关的覆盖。
  确保 **network_data.yaml** 文件包含设置 **ipv6: true**。以前的 Red Hat OpenStack director 版本包含两个路由：一个用于外部网络上的 IPv6 （默认），另一个用于 Control Plane 上的 IPv4。要使用这两个默认路由，请确保 **roles_data.yaml** 文件中的 Controller 定义在 **default_route_networks** 参数中包含这两个网络。例如，**default_route_networks: ['External', 'ControlPlane']**。

- **--ntp-server pool.ntp.org** - 设置 NTP 服务器。

overcloud 创建过程开始，director 置备 overcloud 节点。完成此过程需要一些时间。要查看 overcloud 创建的状态，请以 **stack** 用户身份打开一个单独的终端并运行：

```
$ source ~/stackrc
$ heat stack-list --show-nested
```

访问 overcloud

director 会生成脚本来配置和帮助验证 director 主机与 overcloud 的交互。director 将此文件 (**overcloudrc**)保存在 **stack** 用户的主目录中。运行以下命令来使用此文件：

```
$ source ~/overcloudrc
```

这会加载必要的环境变量，以便从 director 主机 CLI 与 overcloud 交互。要返回与 director 主机交互，请运行以下命令：

```
$ source ~/stackrc
```

第 3 章 部署后 IPV6 操作

使用 IPv6 网络部署 overcloud 后，您必须执行一些额外的配置。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 成功部署 overcloud。如需更多信息，请参阅使用 CLI 工具创建基本 overcloud。

3.1. 在 OVERCLOUD 上创建 IPV6 项目网络

overcloud 需要基于 IPv6 的项目网络用于实例。提供 **overcloudrc** 文件，并在 **neutron** 中创建初始项目网络。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。
- 成功部署 overcloud。如需更多信息，请参阅使用 CLI 工具创建基本 overcloud。
- 您的网络支持 IPv6 原生 VLAN 和 IPv4 原生 VLAN。

流程

1. 提供 overcloud 凭证文件：

```
$ source ~/overcloudrc
```

2. 创建网络和子网：

```
$ openstack network create default --external --provider-physical-network datacentre --provider-network-type vlan --provider-segment 101

$ openstack subnet create default --subnet-range 2001:db8:fd00:6000::/64 --ipv6-address-mode slaac --ipv6-ra-mode slaac --ip-version 6 --network default
```

这会创建一个名为 **default** 的基本 **neutron** 网络。

验证步骤

- 验证网络是否已成功创建：

```
$ openstack network list
$ openstack subnet list
```

3.2. 在 OVERCLOUD 上创建 IPV6 公共网络

在将节点接口配置为使用外部网络后，您必须在 overcloud 上创建此网络以启用网络访问。

先决条件

- 成功安装 undercloud。如需更多信息，请参阅安装 director。

- 成功部署 overcloud。如需更多信息，请参阅使用 CLI 工具创建基本 overcloud。
- 您的网络支持 IPv6 原生 VLAN 和 IPv4 原生 VLAN。

流程

1. 创建外部网络和子网：

```
$ openstack network create public --external --provider-physical-network datacentre --
provider-network-type vlan --provider-segment 100

$ openstack subnet create public --network public --subnet-range 2001:db8:0:2::/64 --ip-
version 6 --gateway 2001:db8::1 --allocation-pool
start=2001:db8:0:2::2,end=2001:db8:0:2::ffff --ipv6-address-mode slaac --ipv6-ra-mode slaac
```

   此命令创建一个名为 **public** 的网络，它为我们的实例提供了超过 65000 IPv6 地址的分配池。

2. 创建路由器，将实例流量路由到外部网络。

```
$ openstack router create public-router
$ openstack router set public-router --external-gateway public
```