UCS上RHOSP的部署指南，用于断开模式下移动工作负载

目录

简介

先决条件

- 要求
- 使用的组件

背景信息

了解RHOSP

- RHOSP架构
- Undercloud
  - Undercloud组件
- 过云
  - 控制器
  - 计算
  - 存储
- 本地存储库（REPO服务器）
- RHOSP网络
  - RHOSP物理连接
  - RHOSP逻辑连接

硬件参数调整

虚拟机监控程序 — KVM安装和网络创建

REPO和导向器VM创建

- 创建REPO和Director VM的前提条件
- REPO VM创建
- 导向器VM创建

本地脱机回购创建

RHOSP云部署

- 先决条件
- 更新输入文件
- 云下部署
- 超云部署
- 访问Horizon控制面板

RHOSP群集的运行状况检查

摘要

简介

本文档介绍在C220 M6 UCS服务器上部署RHOSP以支持Cisco VPC-DI的框架。

先决条件

要求

思科建议您具备Red Hat OpenStack Platform(RHOSP)知识并具备Red Hat Enterprise Linux(RHEL)方面的强大技能。 此外，还需要深入了解虚拟化和网络概念。

使用的组件

本文档不限于特定的软件和硬件版本。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

本指南概述了RHOSP与统一计算系统(UCS)基础设施的集成，强调可扩展性、可靠性和性能优化。

它详细介绍最佳实践，并使用基于脚本的自动化来部署OpenStack TripleO（包括Undercloud和Overcloud架构）。

通过使用本部署指南，组织可以实现为思科虚拟数据包核心 — 分布式实例(VPC-DI)移动虚拟网络功能(VNF)量身定制的强大高效的RHOSP云基础设施。

了解RHOSP

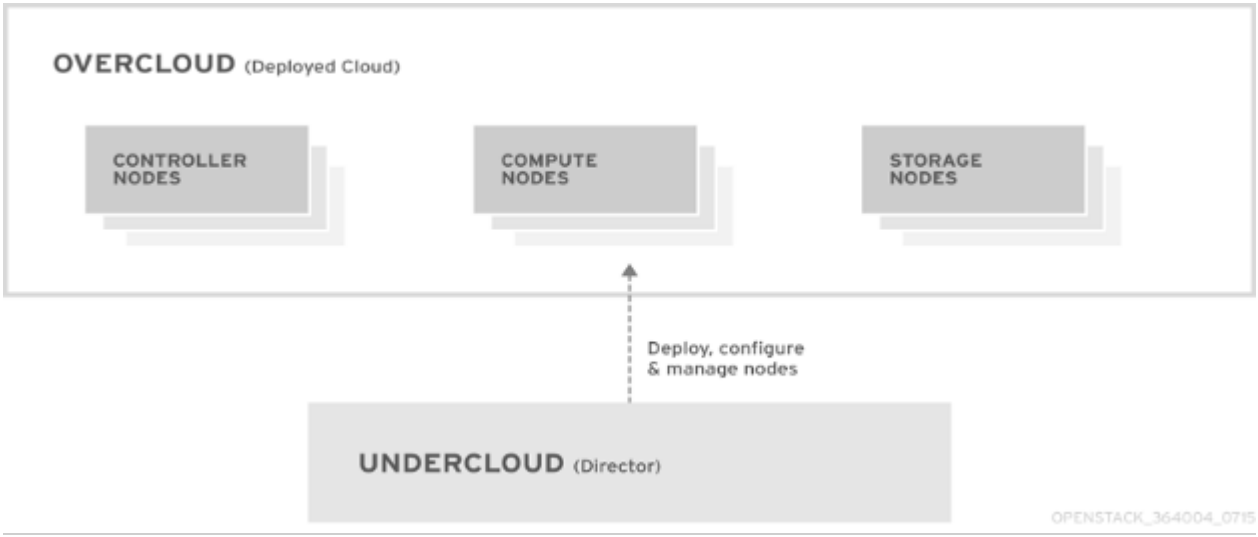
RHOSP是在开源OpenStack项目基础上构建的企业级私有云解决方案，由Red Hat集成和支持。它允许组织按需部署和管理虚拟机(VM)、网络和存储的基础设施即服务(IaaS)。

它提供高可用性(HA)、网络功能虚拟化和可定制部署等功能。

RHOSP架构

RHOSP主要基于OpenStack TripleO项目。Openstack使用Director作为工具集，用于安装和管理完整的RHOSP环境。

RHOSP旨在提供可扩展且灵活的云基础设施。其架构包括两个主要组件：Undercloud和Overcloud。



Undercloud

Undercloud是包含RHOSP导向器工具集的主要管理节点。它是单系统OpenStack安装，包括用于调配和管理构成OpenStack环境(Overcloud)的OpenStack节点的组件。

Undercloud组件

Undercloud使用OpenStack组件作为其基本工具集。每个组件在Undercloud上的独立容器中运行：

- OpenStack身份（密钥） — 为指挥交换机组件提供身份验证和授权
- OpenStack裸机（讽刺） — 管理裸机节点
- OpenStack网络(neutron)和Open vSwitch — 裸机节点的控制网络
- OpenStack协调（临时散热） — 在指挥交换机将重叠云映像写入磁盘后提供节点协调

过云

Overcloud是使用下层云创建的RHOSP环境。这包括根据客户要创建的OpenStack平台(OSP)环境定义的不同节点角色。

控制器

控制器节点为OpenStack环境提供管理、网络和HA。推荐的OpenStack环境在HA群集中包含三个控制器节点。

计算

计算节点为OpenStack环境提供计算资源。随着时间的推移，计算节点可以根据网络需求进行横向扩展/横向扩展。默认计算节点包含下列提到的组件：

i. OpenStack计算(NOVA)
ii. 基于内核的虚拟机(KVM)/快速仿真程序(QEMU)
iii. 打开vSwitch

存储

存储节点为OpenStack环境提供存储。



注意：在客户网络中部署Undercloud/Director和Offline Repository(REPO)有多种方法 — 可以直接部署在裸机节点上，也可以作为VM部署在KVM虚拟机监控程序之上。在当前部署指南中，Director UCS服务器托管KVM（虚拟机监控程序），以在顶部部署多个虚拟机。RHOSP Director节点和Offline-REPO节点部署为KVM虚拟机监控程序上的虚拟机。

本地存储库（REPO服务器）

Red Hat提供了一个称为reposync的实用程序，可用于从内容交付网络(CDN)下载软件包。 要从特定通道下载所有软件包，系统必须订阅该通道。如果系统未订阅所需的通道，则reposync无法在本地系统上下载和同步这些包。

在/etc/yum.repos.d/path中，存储库由以.repo扩展名结尾的文件进行配置。可以在同一文件中定义多个存储库。

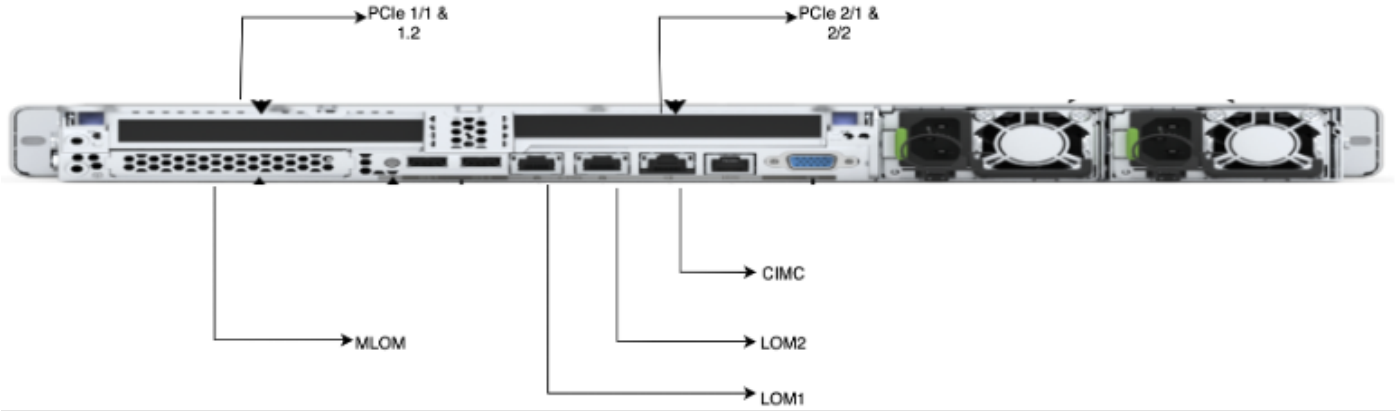
RHOSP网络

网络服务(neutron)是RHOSP的软件定义网络(SDN)组件。RHOSP网络服务管理进出虚拟机实例的内部和外部流量，并提供核心服务，例如路由、分段、DHCP和元数据。它为虚拟网络功能以及交换机、路由器、端口和防火墙的管理提供API。

RHOSP导向器将OpenStack服务映射到不同的隔离网络。传输每种流量类型的网络包括：思科集成管理控制器(CIMC)、调配、内部API、存储数据、存储管理、租户和外部(安全外壳(SSH)和操作、管理和维护(OAM))。

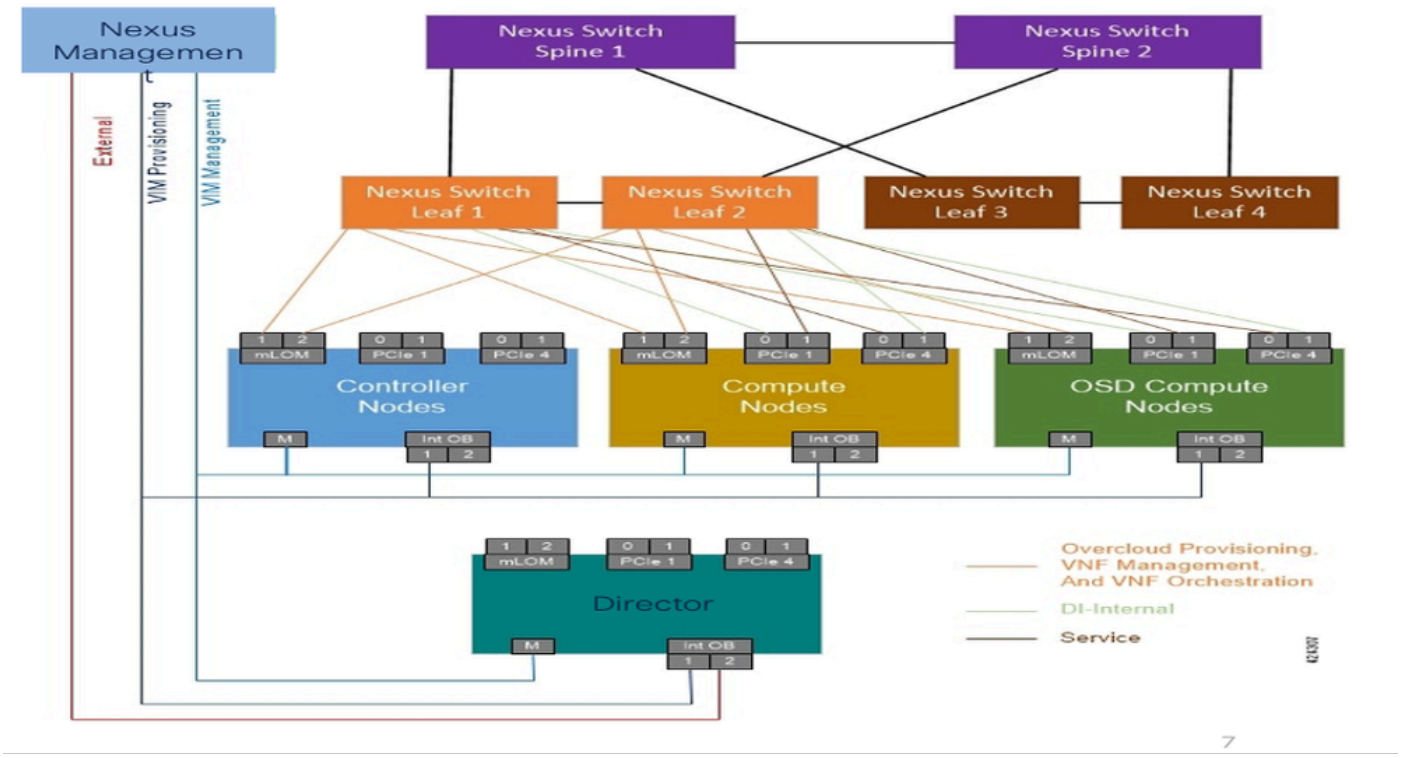
RHOSP物理连接

RHOSP部署使用Cisco UCS C220 M6服务器的不同物理端口来实现不同的连接目的。



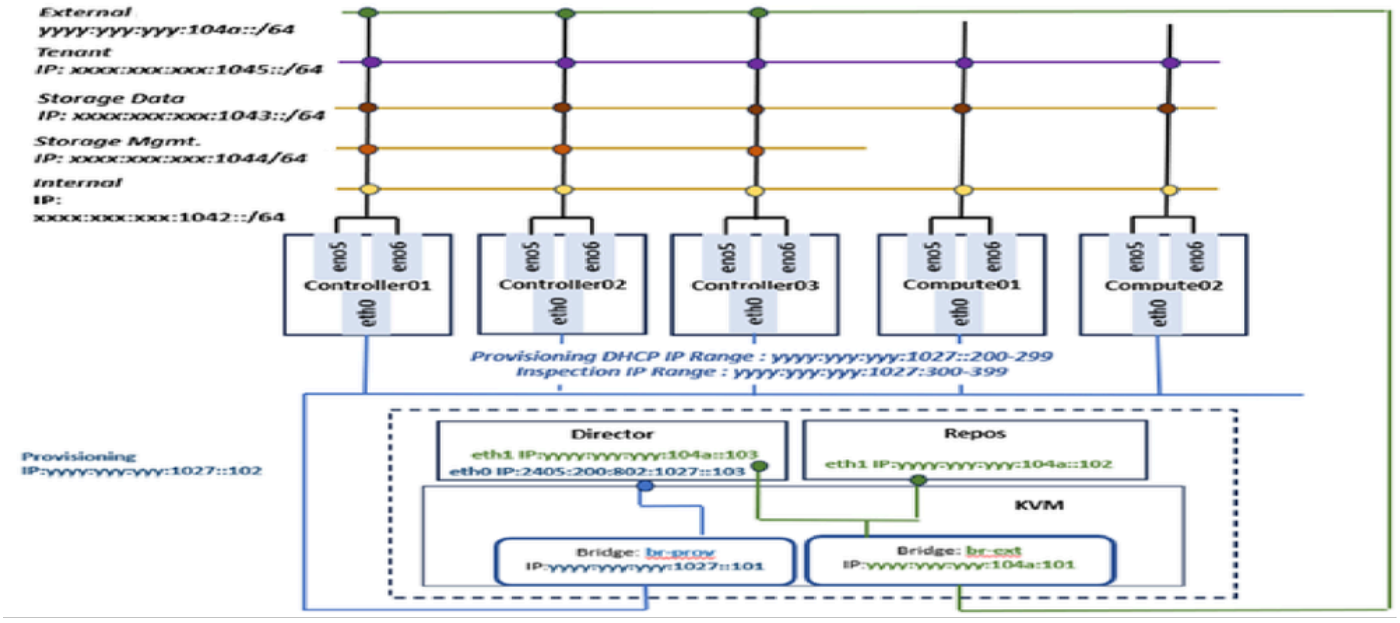
Serial Number	物理端口	详细信息
1.	CIMC	CIMC为服务器调配和管理提供带外连接。
2.	单根I/O虚拟化(SR-IOV)/外围组件快速互联(PCIe)	PCIe网络接口卡(NIC)用于DI内部网络的计算节点和VNF的服务网络。
3.	主板上的模块化LAN(MLOM)	MLOM端口配置为绑定。 osp_external、osp_internal、osp_tenant、osp_external、osp_storage_data、osp_storage_mgmt使用MLOM端口进行内部通信。
4.	主板上的LAN(LOM)	指挥交换机使用LOM1和LOM2端口，而计算和控制器仅使用LOM1端口。 LOM1用于在所有服务器上部署或调配Openstack。 LOM2用作指挥交换机上的OAM（外部网络）。

下图显示与服务器的物理连接。



RHOSP逻辑连接

RHOSP网络有多个子网，用于满足云中不同服务的需求。



- OSP_CIMC:

CIMC是智能编程管理接口(IPMI)，控制所有UCS服务器的管理。此CIMC网络在所有UCS服务器的独立CIMC端口上配置。

- OSP_Provisioning:

此网络负责在Overcloud部署期间对计算机和控制器服务器进行调配和预引导执行环境(PXE)引导管理，并负责获取DHCP IP。为了简单性和兼容性，调配网络在所有UCS服务器的LOM1端口上配置为本征VLAN。此调配网络负责在所有服务器上部署云。

由于导向器服务器上的虚拟化，需要在KVM上为导向器VM创建网桥网络，以便与其他服务器通信。

- OSP_内部：

内部API网络用于OpenStack服务（如neutron、nova、keystone等）之间的通信。

OSP_Internal网络在控制器和计算节点上的绑定MLOM端口上配置。

- OSP_租户：

默认情况下，租户网络在云项目中创建，用于VNF管理。在当前设置中，只为VNF部署创建一个Openstack项目。

OSP_Tenant网络在控制器和计算节点上的绑定MLOM端口上配置。

- OSP_External:

外部网络用于所有外部访问（如SSH）和API网络。

OSP_External网络配置在导向器节点的LOM2端口上，以及控制器和计算节点上的绑定MLOM端口上。

- OSP_Storage_Data:

OSP_Storage网络用于与访问存储相关的所有操作。对于需要访问存储的CEPH服务和VNF之间的通信，这是必需的。它由控制器、计算节点和CEPH使用。

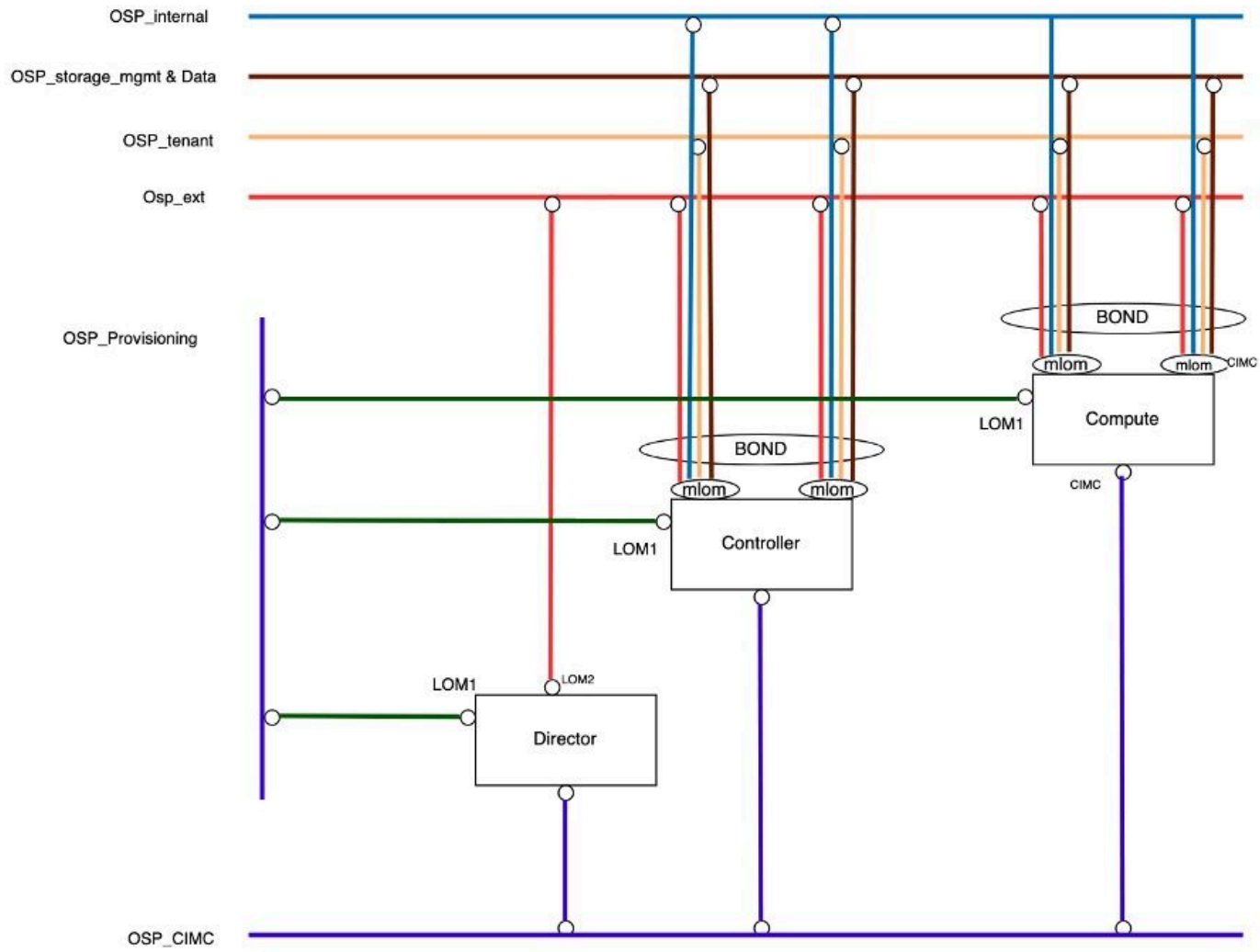
OSP_Storage_Data网络在控制器和计算节点上的绑定MLOM端口上配置。

- OSP_Storage_Management:

OpenStack对象存储使用此网络在存储群集中参与副本节点之间（在控制器计算节点之间形成）同步数据对象。

OSP_Storage_Mgmt网络在控制器和计算节点上的绑定MLOM端口上配置。

下图显示了云下逻辑网络如何连接到RHOSP集群中的每种类型的节点。



硬件参数调整

- 确保根据设计启用了IP并使用正确的管理地址进行配置。
- 确保管理员用户及其密码对于所有CIMC登录访问必须相同。
- 确保Bootstrap（启动）模式设置为统一可扩展固件接口(UEFI)。
- 确保使用推荐的参数设置BIOS设置。
- 将Boot-Order设置为：

a. LOM-PXE(eno1)
b. 引导硬盘驱动器(HDD)

- 设置LOM1(eno1)的mac地址。
- 确保所有UCS/CIMC服务器均使用最新推荐的固件版本进行升级。

虚拟机监控程序 — KVM安装和网络创建

在客户网络中部署Undercloud/Director和Offline REPO有多种方法。这些设备可以直接部署在裸机节点上，也可以作为KVM虚拟机监控程序上运行的VM进行部署。
在当前部署指南中，Director UCS服务器配置为托管KVM虚拟机监控程序，这有利于创建多个VM。RHOSP Director节点和Offline REPO节点部署为此KVM虚拟机监控程序上的VM。



注意：必须执行标准RHEL KVM安装步骤才能部署KVM虚拟机监控程序。

- 启动KVM后，更新主机名：
  hostnamectl set-hostname <hostname> —static

- 配置外部网桥和调配网桥并绑定接口。

br-prov :eth0

br-ext :eth1

这些网桥必须通过网络管理器文本用户界面(NMTUI)GUI创建。

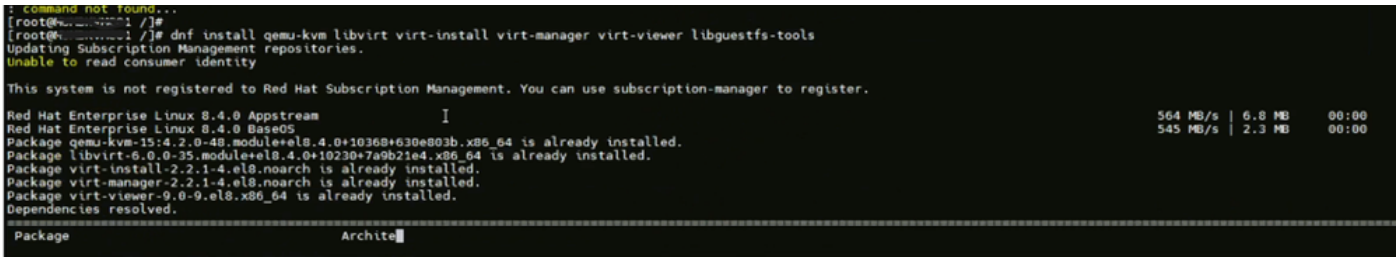
- 默认情况下，在KVM中为物理LOM1和LOM2端口创建eno1和eno2端口，使用CIMC日志记录中的MAC地址交叉验证这些端口。
- 创建网桥网络并通过映射正确的MAC地址在网络中添加从端口。
- 网桥网络创建后，确保可从KVM访问调配和外部网关。

REPO和导向器VM创建

创建REPO和Director VM的前提条件

```
# dnf install qemu-kvm libvirt virt-install virt-manager virt-viewer libguestfs-tools
```

- 在KVM中安装所需软件包，以创建REPO和Director VM。



安装libvirt和qemu软件包

- 在KVM上创建所需的目录。

mkdir /data # mkdir /data/offlineRepos # mkdir /data/isoImages # mkdir /data/qcow2Images # mkdir /data/images

- 将文件复制到目录。

```
# scp -r root@[remote-IP]:/root/rhel-8.4-x86_64-dvd.iso /data/isoImages/
# scp -r root@[remote-IP]:/root/offlineRepos/RHEL8.4 /data/offlineRepos/
# scp -r root@[remote-IP]:/etc/yum.repos.d/offlinedvd.repo /etc/yum.repos.d/
# scp -r root@[remote-IP]:/root/rhel-8.4-x86_64-kvm.qcow2 /data/qcow2Images/
# scp -r root@[remote-IP]:/root/OSREPO_RHEL_84.qcow2 /data/images/
# scp -r root@[remote-IP]:/root/OSREPO_DIRECTOR_84.qcow2 /data/images/
```

- 将ISO安装到/mnt/iso。

```
# mount -t iso9660 -o loop /data/isoImages/rhel-8.4-x86_64-dvd.iso /mnt/iso
```

- 在/etc/yum.repos.d路径下创建REPO文件。

```
# cat /etc/yum.repos.d/offlinedvd.repo

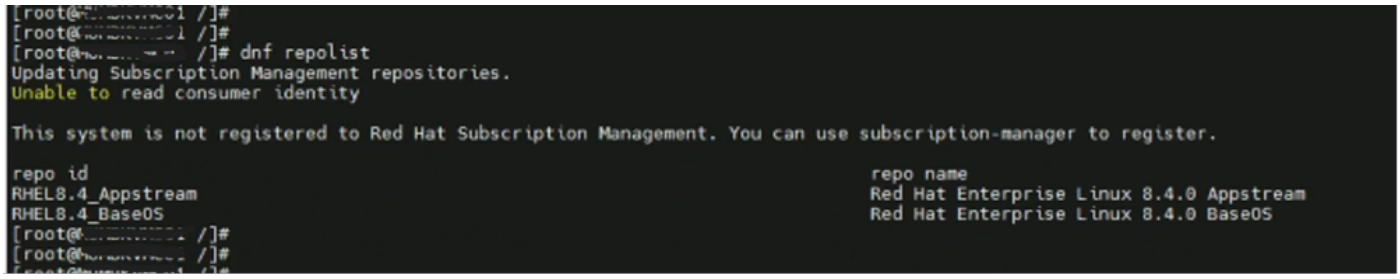
[RHEL8.4_Appstream]
name=Red Hat Enterprise Linux 8.4.0 Appstream
mediaid=None
```

```
metadata_expire=-1
gpgcheck=0
enabled=1
baseurl=file:///data/offlineRepos/RHEL8.4/AppStream/

[RHEL8.4_BaseOS]
name=Red Hat Enterprise Linux 8.4.0 BaseOS
mediaid=None
metadata_expire=-1
gpgcheck=0
enabled=1
baseurl=file:///data/offlineRepos/RHEL8.4/BaseOS/
```

- 检查重新轮询列表并确保映射了Appstream和Baseos REPO。

```
# dnf repolist
```



重新轮询器

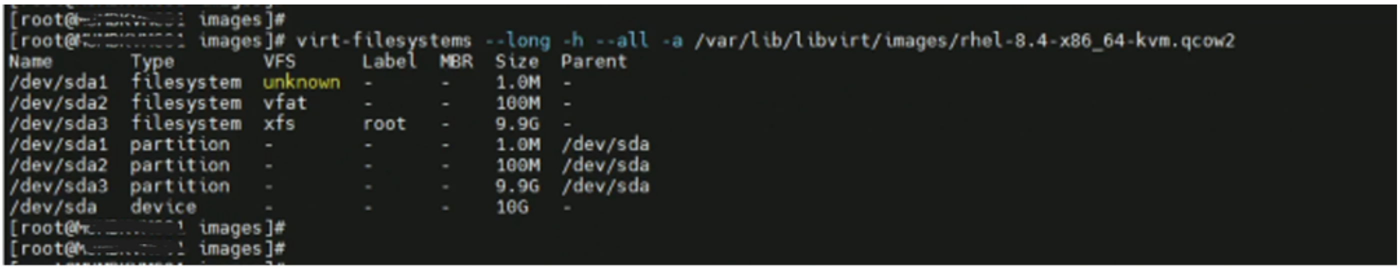
REPO VM创建

```
$ cd /var/lib/libvirt/images/
$ export LIBGUESTFS_BACKEND=direct
$ virt-customize -a /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 --root-password password:Cisco@12
```



RHEL映像自定义

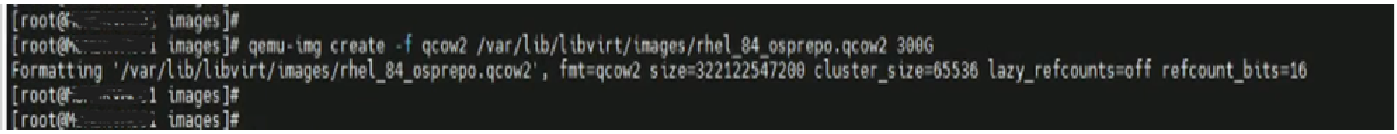
```
$ virt-filesystems --long -h --all -a /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2
```



文件系统映射

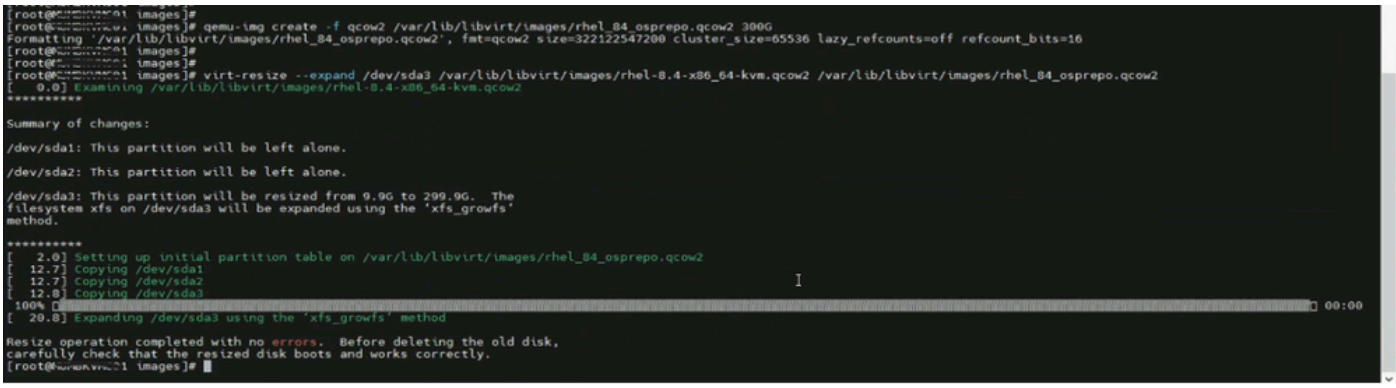
- 创建REPO服务器的映像。

```
$ qemu-img create -f qcow2 /var/lib/libvirt/images/rhel_84_osprepo.qcow2 500G
```



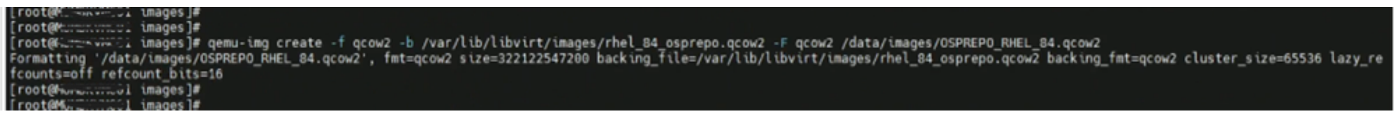
创建回购映像

```
$ virt-resize --expand /dev/sda3 /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 /var/lib/libvirt/ima
```



Repo安装

```
$ qemu-img create -f qcow2 -b /var/lib/libvirt/images/rhel_84_osprepo.qcow2 -F qcow2 /data/images/OSPRE
```



创建回购映像

```
$ guestfish -a /data/images/OSPREPO_RHEL_84.qcow2 -i ln-sf /dev/null /etc/systemd/system/cloud-init.ser
```

```
[root@[redacted] images]#
[root@[redacted] images]#
[root@[redacted] images]# guestfish -a /data/images/OSPREPO_RHEL_84.qcow2 -i ln-sf /dev/null /etc/systemd/system/cloud-init.service
[root@[redacted] images]#
[root@[redacted] images]#
[root@[redacted] images]#
```

Guestfish OSP Repo

```
$ osinfo-query os | grep rhel8
```

```
[root@[redacted] images]#
[root@[redacted] images]# osinfo-query os | grep rhel8

** (osinfo-query:49279): WARNING **: 17:12:33.856: Entity http://pcisig.com/pci/1B36/0100 referenced but not defined
 rhel8-unknown        | Red Hat Enterprise Linux 8 Unknown                | 8-unknown | http://redhat.com/rhel/8-unknown
 rhel8.0              | Red Hat Enterprise Linux 8.0                      | 8.0       | http://redhat.com/rhel/8.0
 rhel8.1              | Red Hat Enterprise Linux 8.1                      | 8.1       | http://redhat.com/rhel/8.1
 rhel8.2              | Red Hat Enterprise Linux 8.2                      | 8.2       | http://redhat.com/rhel/8.2
 rhel8.3              | Red Hat Enterprise Linux 8.3                      | 8.3       | http://redhat.com/rhel/8.3
 rhel8.4              | Red Hat Enterprise Linux 8.4                      | 8.4       | http://redhat.com/rhel/8.4
[root@[redacted] images]#
[root@[redacted] images]#
```

RHEL软件包

```
$ virt-install --cpu host --memory 32768 --vcpus 16 --os-variant rhel8.4 --disk path=/data/images/OSPRE
```

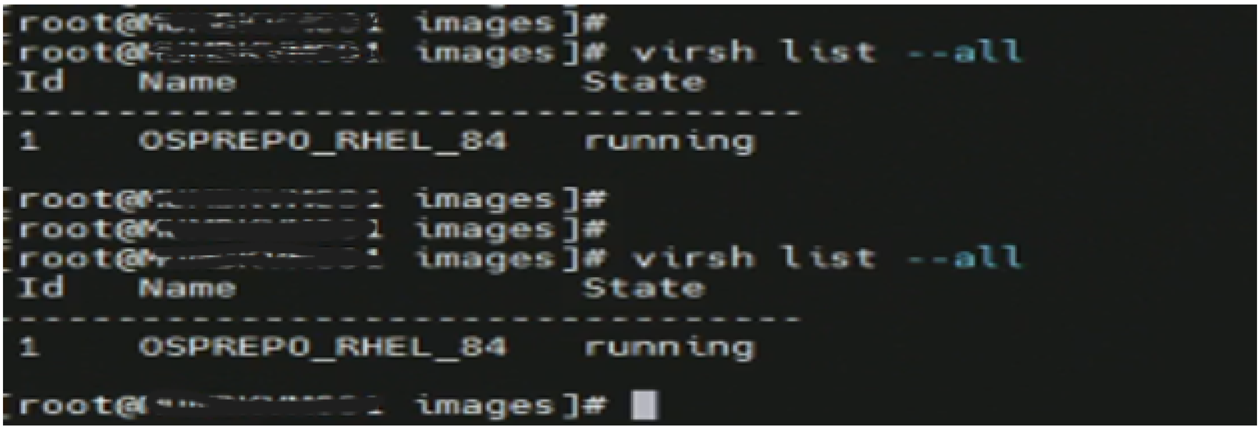
```
[root@[redacted] images]#
[root@[redacted] images]# virt-install --cpu host --memory 32768 --vcpus 16 --os-variant rhel8.4 --disk path=/data/images/OSPREPO_RHEL_84.qcow2,device=disk,bus=virtio,format=qcow2
--import --noautoconsole --vnc --network bridge:br-ext --name OSPREPO_RHEL_84

** (process:49296): WARNING **: 17:13:15.813: Entity http://pcisig.com/pci/1B36/0100 referenced but not defined

Starting install...
Domain creation completed.
[root@[redacted] images]#
[root@[redacted] images]#
```

Repo服务器安装

```
$ virsh list --all
```

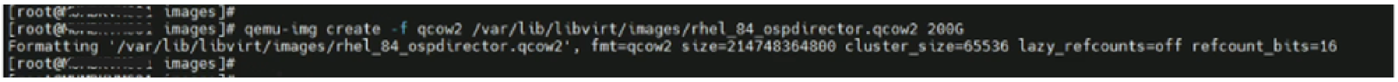


回购服务器状态

导向器VM创建

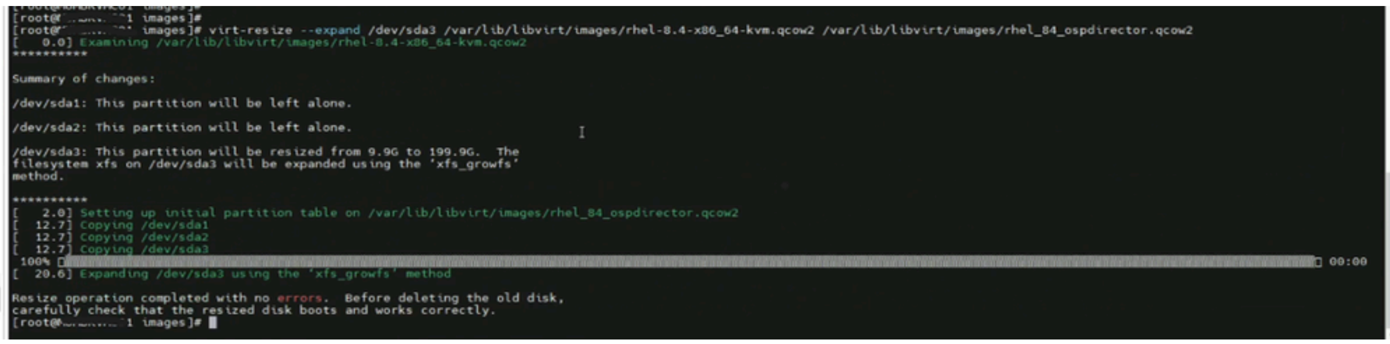
- 创建指挥交换机服务器的映像。

```
# qemu-img create -f qcow2 /var/lib/libvirt/images/rhel_84_ospdirector.qcow2 500G
```



指挥交换机映像创建

```
# virt-resize --expand /dev/sda3 /var/lib/libvirt/images/rhel-8.4-x86_64-kvm.qcow2 /var/lib/libvirt/imag
```



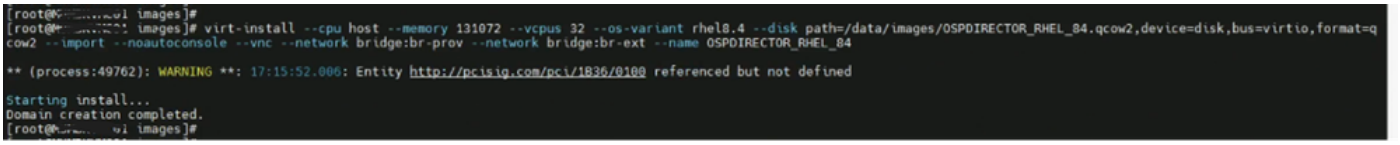
/dev/sda3的虚拟调整大小

```
# qemu-img create -f qcow2 -b /var/lib/libvirt/images/rhel_84_ospdirector.qcow2 -F qcow2 /data/images/O
```



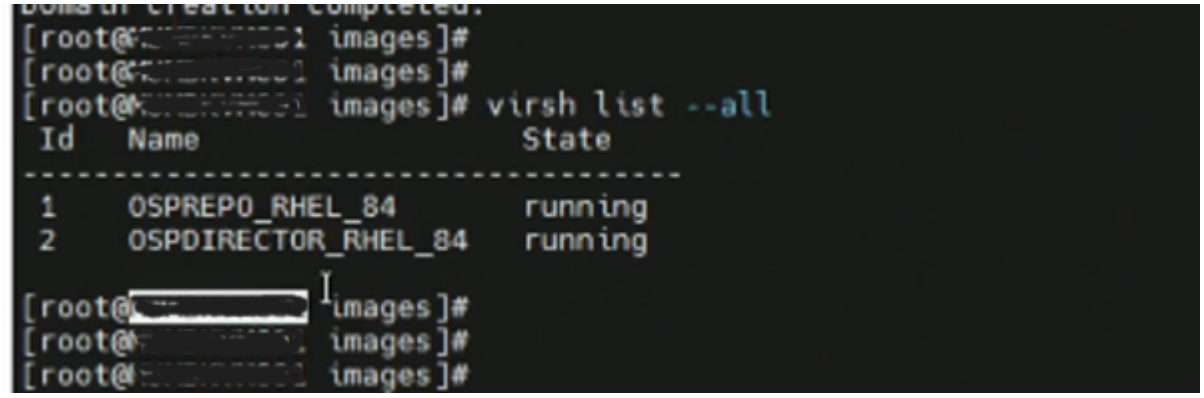
指挥交换机映像创建

```
# guestfish -a /data/images/OSPDIRECTOR_RHEL_84.qcow2 -i ln-sf /dev/null /etc/systemd/system/cloud-init
# virt-install --cpu host --memory 131072 --vcpus 32 --os-variant rhel8.4 --disk path=/data/images/OSPD
```

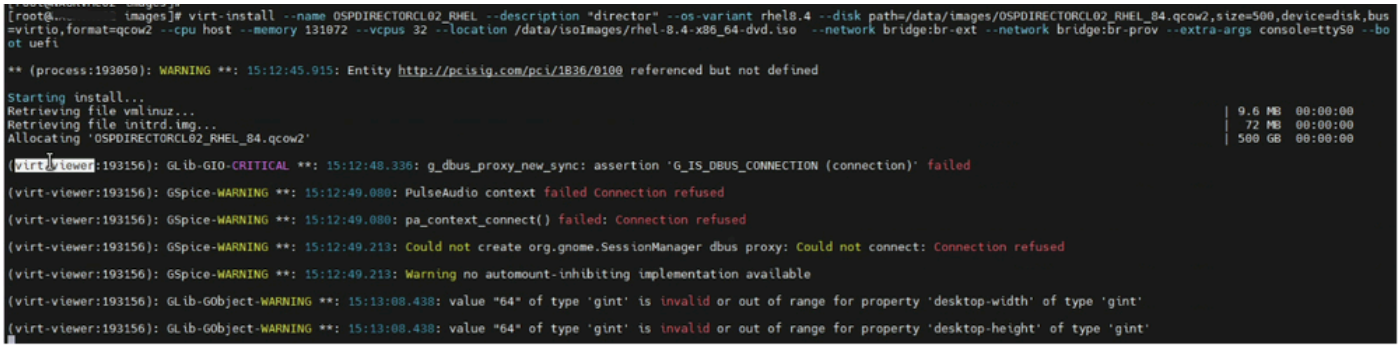


导向器VM创建

```
# virsh list --all
```



导向器VM创建

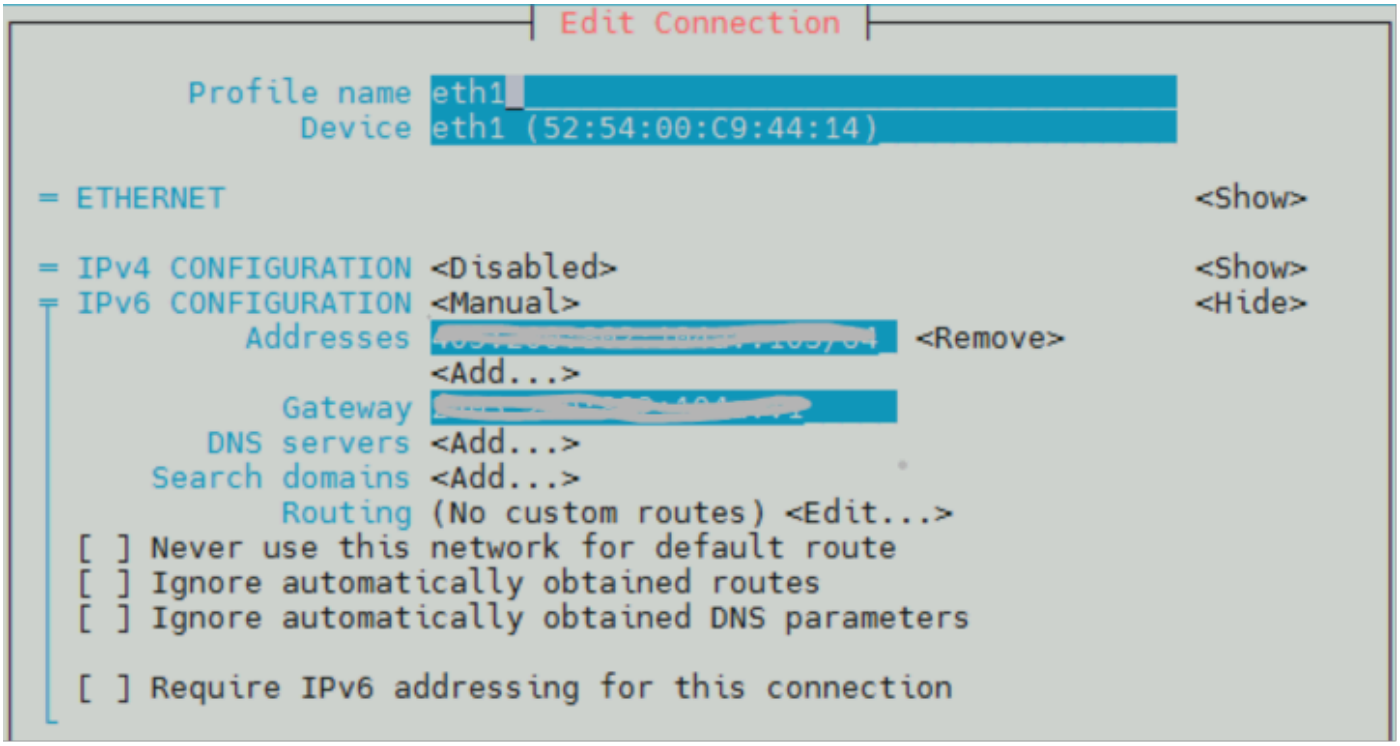


指挥交换机VM安装

- 打开新窗口并使用VM的控制台登录。

```
# virsh list -all
# virsh console <domain-id>
```

- 在导向器后VM部署，创建了两个eth接口eth0和eth1。
  - eth0用于OSP_provisioning网络，其中br-ctlplane网络在Undercloud部署期间创建。
  - eth1用于SSH访问的外部连接。因此，分配了OSP_external子网IP地址。



指挥交换机的网络配置

本地脱机回购创建

REPO服务器必须向Red Hat CDN注册，并且必须拥有部署所需的所有可用RHOSP 16.2软件包的存储库。必须使用代理将RHEL RPM软件包和RHOSP容器映像下载到REPO VM。

RHOSP云部署

RHOSP 16.2通过自动化部署在客户网络中。Ansible脚本用于自动化Undercloud和Overcloud部署。

先决条件

开始实际云部署之前要观察的步骤：

1.确保KVM节点与OSP_EXT网络上的本地回购VM和指挥交换机VM的连接。

2.确保将所有可能的脚本和自动化脚本上传到指定文件夹中的KVM主机。

3.创建名称为“cisco”和“automation”的文件夹，并放置可触发脚本的标签。

```
# cd /home
# mkdir cisco
# cd /home/cisco
# mkdir automation
# cd /home/cisco/automation
```

tarball将包含三个文件夹目录结构，命名为：

- 脚本：包含用于裸机节点配置的shell脚本。
- rpms:包括英特尔以太网连接(ICE)驱动程序、英特尔驱动程序和Red Hat Package Manager(RPM)软件包。
- ansible:包含输入vars文件、ansible yaml文件和jinja-templates。

4.安装sshpass包。sshpass是一个命令行实用程序，用于以非交互方式为ssh提供密码。它主要用于手动输入密码不可行的脚本或自动化场景。

- 从internet/现有服务器下载sshpass包sshpass.tar.gz。
- 安装GNU编译器集合(GCC)软件包。

```
# yum install gcc
```

- 安装make程序包。

```
# yum install make
```

- 解压缩sshpass软件包并安装SSH pass。

```
# tar -xvzf sshpass.tar.gz
# cd sshpass-1.10/
# ./configure
# sudo make install
# sshpass -V
```

5.指挥交换机安装过程需要非根用户执行命令。必须在具有sudo访问权限的指挥交换机VM中创建“堆栈”用户。

```
# useradd stack
# passwd stack

Disable password requirements for the ‘stack’ user when using sudo.
# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
# chmod 0440 /etc/sudoers.d/stack
```

6.将rootCA.crt文件从REPO服务器复制到指定路径的指挥交换机VM和KVM。此外，更新信任列表中的回购VM证书。

```
# /etc/pki/ca-trust/source/anchors
# update ca-trust
```

7.更新Director VM和/etc/hosts文件中的KVM中的本地REPO服务器主机名详细信息。

8.在KVM和Director VM上，安装其他软件包（如python、ansible等）以执行ansible自动化脚本。

```
# dnf install python3 python3-devel ansible httpd -y
# update-alternatives --set python /usr/bin/python3
```

9.必须能够从Director的调配网络访问CIMC子网，才能在云部署期间启用调配。如果需要，请为同一路由添加静态路由。

```
# ip -6 route add <CIMC Subnet> via <Provisioning Subnet>
```

10.在KVM和Director VM中，在/ansible文件夹下创建主机文件，并根据需要添加堆栈特定详细信息。

```
[ospd]
# <PODNAME> ansible_host=<OSPD IP> ansible_ssh_user=stack ansible_ssh_pass='<STACKPASSWD>' ansible_ssh_

<podname> - Stack Name of the Cloud.
<OSPD IP> - Baremetal OSPD Node IP Address
<STACKPASSWD> - OSPD Node password for ‘stack’ user
```

11.确保所有可能的手册和输入文件必须保存在导向器VM的/home/stack文件夹下方。

更新输入文件

有一个输入变量文件，其中包含客户网络特定详细信息，必须针对云部署做好准备。

路径：/home/cisco/automation/ansible/podvars

文件名：<stack-name>_vars.yml

根据站点特定IP计划/低层设计文档更新突出显示的参数。



注意：虚拟IP地址仅用于表示目的。

<#root>

```
# ############################
# XR21 Specific Variables
# ############################

# ==========================
# Common Variables
# ==========================

# UCS hardware type: 'm4/m5/m6'
hardware: m6

# Platform type: 'epc/pcrf'
platform: epc

# RHEL version
rhel: { version: 84, tag: 8.4 }

# Openstack version
osp: { version: 16, major: 2 }

# Container version
container: { tag: 16.2, tools: 3.0 }

# Overcloud stack name

stack_name: '



    '
```

```
# OSPD full hostname

fqdn_hostname: '



     .epdg.ap.hamb.a6.cloud.com’




# OSPD host login
ospd_host: { ip:

'2405:XXXX:089:1054::11'

, username: 'stack', password: '*******' }

# OSPD cimc login
ospd_cimc: { ip:

'2405:XXXX:089:1054::11'

, username: 'admin', password: '********' }

# CIMC username and pasword must be same across all Overcloud nodes
cimc: { username: 'admin', password: '********', ip_pool: '2405:XXXX:089:1055::/64' }

# Undercloud-Overcloud provision
internal_network: {
   ip_type: 'v6',
   local_interface: 'eth0',

   local_ip: ‘2405:XXXX:089:1041::103',



   undercloud_public_host: ‘2405:XXXX:089:1041::105',



   undercloud_admin_host: ‘2405:XXXX:089:1041::104',
```

```
   cidr: ‘2405:XXXX:089:1041::/64',

   dhcp_start: ‘2405:XXXX:089:1041::200',

   dhcp_end: ‘2405:XXXX:089:1041::299',

   gateway: ‘2405:XXXX:089:1041::199',

#   nexthop: ‘2405:XXXX:089:1041::1',

   inspection_iprange_start: ‘2405:XXXX:089:1041::300',

   inspection_iprange_end: ‘2405:XXXX:089:1041::399',

}
```

```
# DNS
dns_ips: [ '2405:YYYY:a10:f100::1' ]
dns_search_domains: [ 'cloud.com' ]

# NTP
ntp_ips: [ '2405:YYYY:801:700::afa', '2405:YYYY:801:700::afb' ]

# Deployment type: 'offline/online'
repos: { rhel: 'offline', container: 'offline' }

# Offline details if repos is 'offline'
offline: {
    environment: 'v01_00',
    deliverymedia: '/home/stack/deliverymedia/'
}

# Satellite details if repos is 'online'
satellite: {
    fqdn_name: 'rh-satellite2.mitg-bxb300.cisco.com',
    ip: '10.XX.XX.XX',
    org: 'MITG',
    user: 'admin',
    password: '*******',
```

```
    environment: 'production',
    activation_key: 'ak-rhel{{rhel.version}}-osp{{osp.version}}{{osp.major}}',
    repos_file: 'rhel{{rhel.version}}osp{{osp.version}}{{osp.major}}.yaml'
}

# Offline container registry details
offline_registry: {
    ip: '2405:XXXX:089:1055::100',
    name: '
```

.

',

```
    port: '5000',
    container_tag: '16.2.6',
    user: 'ciscoadmin',
    password: '******'
}

# Custom cloud domain details
domain_name: {
    domain: '
```

',

```
    cloudshortname: 'nl'
}

# Container images namespace
container_namespace: 'mitg-{{satellite.environment}}-cv-rhel{{rhel.version}}-osp{{osp.version}}{{osp.ma

# List of cimc IPs

ctrl_cimc_ip:
```

- 2405:XXXX:YYYY:1036::12

```
  - 2405:XXXX:YYYY:1036::13

  - 2405:XXXX:YYYY:1036::14

osdc_cimc_ip:

cmpt_cimc_ip:

  - 2405:XXXX:YYYY:1037::17

  - 2405:XXXX:YYYY:1038::18

  - 2405:XXXX:YYYY:1038::19

  - 2405:XXXX:YYYY:1038::20

mgmt_cimc_ip:

  - 2405:XXXX:YYYY:1051::15

  - 2405:XXXX:YYYY:1051::16

# ============================
# Hardware Specific Variables
```

```
# ============================

# Isolcpu for cpu pinning
isolcpus: { osdc: '4-31,36-63', cmpt: '2-31,34-63', mgmt: '2-31,34-63' }

# Hugepages in 1G Pages
hugepages: { osdc: 428, cmpt: 448, mgmt: 448 }

# Reserved host memory in MB
reserved_host_memory: { osdc: 84000, cmpt: 64000, mgmt: 64000 }

# Number of VFs per SR-IOV port
sriov_vfs_per_port: 16

# List of SR-IOV ports
sriov_port_list: [ens1f0, ens1f1, ens9f0, ens9f1]

# List of OVS bonding interface
ovs_bond_interface: [eno5, eno6]

# Physical networks
physical_network: [phys_pcie1_0, phys_pcie1_1, phys_pcie2_0, phys_pcie2_1]

# Boot disk size
boot_disk_mb_size: { ctrl: 761985, osdc: 761985, cmpt: 761985, mgmt: 1524925 }

# Boot disk PD slot number
boot_disk_pd_slot: { ctrl: [1,2], osdc: [1,2], cmpt: [1,2], mgmt: [1,2] }

# Boot disk VD slot number
boot_disk_vd_slot: { ctrl: 237, osdc: 235, cmpt: 239, mgmt: 239 }

# Storage backend 'swift' or 'ceph'
storage_backend: 'swift'

# Storage disk size
storage_disk_mb_size: { swift: 761985, ceph: 914573, journal: 0 }

# Storage disk PD slot number
storage_disk_pd_slot: { swift: [6,7], ceph: [3,4,5,6], journal: [0] }

# Storage disk VD slot number
storage_disk_vd_slot: { swift: [238,239], ceph: [236,237,238,239], journal: [0] }

# Firmware version 'yes' or 'no' ???
firmware: { check: 'no', bios_version: '4.2.3c', cimc_version: '4.2(3e)' }

# ============================
# OSP Specific Variables
# ============================


# Timezone for overcloud nodes



timezone: 'Asia/Kolkata'
```

```
# Overcloud node count to deploy



node_count: { ctrl: 3, osdc: 0, cmpt: 11, mgmt: 2 }



local_network: {
   ip_type: 'v6',
   tenant_vlan_id: 1045,
   tenant_net_cidr: '240f:ppp:rr:1045::/64',
   tenant_alloc_pools_start: '240f:ppp:rr:1045::10',
   tenant_alloc_pools_end: '240f:ppp:rr:1045:ffff:ffff:ffff:fffe',

   storage_vlan_id: 1043,
   storage_net_cidr: '240f:ppp:rr:1043::/64',
   storage_alloc_pools_start: '240f:ppp:rr:1043::10',
   storage_alloc_pools_end: '240f:ppp:rr:1043:ffff:ffff:ffff:fffe',

   storage_mgmt_vlan_id: 1044,
   storage_mgmt_net_cidr: '240f:ppp:rr:1044::/64',
   storage_mgmt_alloc_pools_start: '240f:ppp:rr:1044::10',
   storage_mgmt_alloc_pools_end: '240f:ppp:rr:1044:ffff:ffff:ffff:fffe',

   internal_api_vlan_id: 1042,
   internal_api_net_cidr: '240f:ppp:rr:1042::/64',
   internal_api_alloc_pools_start: '240f:ppp:rr:1042::10',
   internal_api_alloc_pools_end: '240f:ppp:rr:1042:ffff:ffff:ffff:fffe'
}

# External VLAN and IP configs
external_network: {

   ip_type: 'v6',



   vlan_id: 1046,



   default_route: '2405:XXXX:YYYY:1055::1',



   network_cidr: '2405:XXXX:YYYY:1055::/64',



   alloc_pool_start: '2405:XXXX:YYYY:1055::100',



   alloc_pool_end: '2405:XXXX:YYYY:1055::200',
```

```
   horizon_ip: '2405:XXXX:YYYY:1055::107'


}

# Neutron mechanism driver 'ovs' or 'ovn'
neutron: {
   driver: 'ovs',
   dvr: false,
   datacenter_vlan_start: 1050,
   datacenter_vlan_end: 1070
}


# ============================
# OS Specific Variables
# ============================

# RHEL kernel version

kernelversion: '4.18.0-305.88.1.el8_4.x86_64'



# E810 ICE driver
ice_driver: { check: 'yes', version: 1.12.6, intel_aux_version: 1.0.1 }

# ENIC and FNIC verison
nic_version: { enic: '2.3.0.53', fnic: '1.6.0.53' }

# IPMI watchdog timer config
watchdog: { action: enabled, version: '2.0.31-3.el8.x86_64', timer: 250 }

# StorCLI raid management
storcliver: '007.2612.0000.0000-1.noarch'

# ============================
# Platform Specific Variables
# ============================

# Buffer pool size based on platform type
innodb_buffer_pool_size: 1610
# ##################################
# END - XR21 Specific Variables
# ##################################
```

云下部署

Undercloud使用ansible脚本分七步部署。所有步骤都必须从在此情况下充当跳转主机的KVM主机执行。

步骤	标记	描述	攻略YAML

步骤1:	检查文件	在Openstack Platform Director(OSPD)上验证所需的攻略、脚本和RPM。	osp16_published_playbook
Step 2.	genpodvar	生成与硬件、RHEL等相关的POD特定变量文件，如common_vars.yml（硬件、软件、网络详细信息）、hw_m6_vars.yaml（CPU、内存、Hugepages、磁盘、NIC等）、rhel_84_vars.yml（RHEL、内核、ICE驱动程序、NIC版本）、pf_esc_vars.yml(弹性服务控制器(ESC)详细信息)、osp_16_vars.yml（RLlllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll OSP版本、时区、IP类型、VLAN ID、IP、中子详细信息）。	osp16_generate_pod_spec
第 3 步：	preuc部署	配置完全限定域名(FQDN)、网络时间协议(NTP)，并更新指挥交换机节点上的所有数据包。	osp16_pre_undercloud_de
第 4 步：	firstreboot	在之前的配置和软件包安装后，执行Undercloud导向器节点的第一次重新启动。	osp16_undercloud_deploy.
第 5 步：	ucdeploy	在控制器上安装Undercloud堆栈	osp16_undercloud_tuning.
第六步：	cstate	在控制器节点上配置BIOS CPU C状态设置。	osp16_cstate.yml
第 7 步：	seconddreboot	在BIOS更改后，在Undercloud director上再次重新启动。	不适用

名为osp16_auto_undercloud_deploy.yml的文件是可在单个迭代中运行的主要可读手册，但建议使用不同标签分步执行手册，以便在出现任何部署问题时轻松进行故障排除。

<#root>

```
#

cd /home/stack/ansible/

# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=

TAG
```

For Ex -

#

```
ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=checkfiles
# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=genpodvars
# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=preucdeploy
# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=firstreboot
# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=ucdeploy
# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=cstate
# ansible-playbook -i hosts osp16_auto_undercloud_deploy.yml -e podname=<> --tags=secondreboot
```

Note :-

```
 Deployment Logs would be generated in “/home/stack/autologs” in Director-VM.

Post-Checks for Verification of Undercloud Deployment.
“stackrc” & “undercloud.conf” file must be generated in /home/stack folder.

# sudo podman ps -a
# source stackrc
# openstack stack list
# openstack stack show <stack-name> --fit
# openstack server list
# openstack network list
# openstack subnet list
```

超云部署

Overcloud部署至少包含三个高可用性模式控制器和一个计算。Overcloud使用ansible脚本在17个步骤中部署。所有步骤都必须从充当跳转主机的Director-VM执行。

步骤	标记	描述	攻略YAML
步骤1:	genpodvar	为与硬件、RHEL等相关的Overcloud生成特定于POD的变量文件，如common_vars.yml（硬件、软件、网络详细信息）、hw_m6_vars.yaml（CPU、内存、Hugepages、磁盘、NIC等）、rhel_84_vars.yml（RHEL、内核、ICE驱动程序、NIC版本）、	osp16_generate_pod_specific_vars.yml

		pf_esc_vars.yml（ESC详细信息）、osp_16_vars.yml版本(OSP)时区、IP类型、VLAN ID、IP、中子详细信息)。	
Step 2.	geninstack	从上一步中创建的/var/common_vars.yml生成Instackenv JSON文件。 指挥交换机需要手动创建的节点定义模板。此文件instackenv.json使用JSON格式，包含节点的所有硬件和电源管理详细信息。此步骤还会在生成文件之前验证UCS服务器上的硬件配置。	osp16_generate_instackenv.yml
第 3 步：	cimcvd	参考common_vars.yml、hw_m6_vars.yaml和rhel_84_vars.yml，在每个服务器上配置CIMC设置和虚拟磁盘(VD)。	osp16_cimc_vd_configure.yml
第 4 步：	preocdeploy	此步骤执行部署Overcloud的所有必备条件。它设置FQDN、NTP并更新所有包，将映像推送到路径以进行部署。	osp16_pre_overcloud_deploy.yml
第 5 步：	导入节点	在此步骤中，会内省服务器CPU、内存、网卡以及网络交换机上的接口和端口。内省在所有控制器和计算机的相连网络交换机上执行。	osp16_import_ironic_nodes.yml
第六步：	gentemplates	为控制器和计算生成自定义模板文件。在自定义模板中，为在其上运行的所有服务定义控制器和计算角色。它还通过应用证书、路由等执行系统强化。	osp16_generate_custom_templates.yml
第 7 步：	ocdeploy	在此步骤中，完成Openstack Overcloud部署。运行由Red Hat提供的deploy.sh，以进行RHOSP部署。	osp16_overcloud_deploy.yml
步骤	geninventory	在此步骤中，生成供ansible使用的库	osp16_build_inventory_v3.py

8		存yml文件，其中预配IP、IPMI(CIMC)IP和凭证与控制器一起存储和映射，并计算自动化以登录系统并执行进一步的步骤。	
步骤9	离线回购	在/etc/yum.repo.d/offline.repo文件中配置Overcloud Offline REPO指向通过外部网络的REPO服务器。	osp16_config_offline_repo.yml
步骤10	围栏	在所有控制器节点上使用“Shoot The Other Node In The Head”（HA群集中的一种围栏技术）(STONITH)配置围栏。	osp16_config_fencing.yml
步骤11	raidcache	为所有控制器和计算机配置Raid缓存设置还配置SWIFT存储设置。	osp16_raid_cache_tuning.yml
步骤12	dnfupdate	对所有节点上的所有程序包运行DNF更新。	dnf_update_all_packages.yml
步骤13	setiplink	在此步骤中，为演进分组数据网关(EPDG)内部和数据流量启用SR-IOV端口的信任模式控制。对SR-IOV端口的支持在neutron中提供，允许VM通过SR-IOV虚拟功能访问网络。	osp16_setIpLink.yml
步骤14	监视程序	在此步骤中，控制器节点上的IPMI设置配置为通过带外连接在所有服务器上执行管理任务。	osp16_config_ipmi_watchdog.yml
步骤15	冰河	将用于外围组件互联(PCI)卡的英特尔E810 ICE驱动程序更新为EPDG版1.12.6，以便将英特尔NIC端口用作SR-IOV。	osp16_ice_driver_install.yml
步骤16	reboot	执行上述步骤后，重新启动所有重叠云节点。	osp16_reboot_overcloud_hosts.yml
步骤	verifyrhosp	检验RHOSP部署配置和运行状况。	osp16_rhosp_verify.yml

17			

对于Overcloud节点的调配，Undercloud使用“overcloud-hroughted-uefi-full.qcow2”。 因此，开始超云部署之前，映像必须存储在云下/导向器中的指定路径中。

从远程站点复制Overcloud qcow2文件。

<#root>

```
# su - stack
# cd /home/stack
# mkdir deliverymedia
# cd deliverymedia


### Copy overcloud-hardened-uefi-full.qcow2 to deliverymedia ###


# scp overcloud-hardened-uefi-full.qcow2 stack@[Director-IP]:/home/stack/deliverymedia
[stack@[stack@ Undercloud ~]$ cd /home/stack/ansible/
[stack@[stack@ Undercloud ansible]$ ansible-playbook osp16_auto_overcloud_deploy.yml -e podname=POD_NAM


For Ex –


# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=genpodvars
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=geninstack
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=cimcvd
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=preocdeploy
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=importnodes
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=gentemplates
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=ocdeploy


#### Push & Update the rootCA.pem in all the Controllers & Computes ####


# for node in $(nova list | grep -i active | awk '{print $12}' | awk -F "=" '{print $2}' ); do scp -o S


#### Append the Director Entry in "/etc/hosts" file ########


# for node in $(nova list | grep -i active | awk '{print $12}' | awk -F "=" '{print $2}' ); do ssh -o S

# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=geninventory
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=offlinerepo
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=fencing


### In case of Fencing Failures, please check the reachability of CIMC Subnet from Controllers ######
```

```
## If CIMC Subnet is not pinging, Do add the static Route ###


# ip -6 route add <CIMC Subnet> via <Provisioning Subnet>
Ex: ip -6 route add 2405:XXXX:YYY:9999::/64 via 2405:XXXX:YYY:9999:1

# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=raidcache
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=dnfupdate
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=setiplink
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=watchdog
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=icedriver
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=reboot
# ansible-playbook -i hosts osp16_auto_overcloud_deploy.yml  -e podname=<> --tags=verifyrhosp
```

要监控部署日志，请使用最新的日志文件。

```
# tail -F </home/stack/autologs/osp16_auto_overcloud_deploy_*.log>
```

确保所有17个步骤均已通过。

检查失败=计数必须为00。

日志= /home/stack/autologs/osp16_rhosp_verify.yml _20200703T042257.log

#==========

#步骤 |标记 |说明 |攻略

#==========

step1 | genpodvar |生成POD特定变量文件 | osp16_generate_pod_specific_vars.yml -e podname=

step2 | geninstack |生成Instackenv JSON文件 | osp16_generate_instackenv.yml -e podname=

step3 | cimcvd |配置CIMC VD | osp16_cimc_vd_configure.yml

#第4步 | preocdeploy |配置Pre Overcloud部署 | osp16_pre_overcloud_deploy.yml

#第5步 |导入节点 | Import Openstack Baremetal Ironic Nodes | osp16_import_ironic_nodes.yml

#第6步 | gentemplates |生成自定义模板 | osp16_generate_custom_templates.yml

#第7步 | ocdeploy | Openstack Overcloud部署 | osp16_overcloud_deploy.yml

step8 | geninventory |生成资产文件 | osp16_build_inventory_v3.py —ipmipass

step9 | offlinerepo |从脱机TAR文件配置Overcloud脱机回购 | osp16_config_offline_repo.yml

step10 |围栏 |在重新启动之前部署MOP — 配置围栏 | osp16_config_fencing.yml

step11 | raidcache |在重新启动前部署MOP - Raid缓存和PR调整 | osp16_raid_cache_tuning.yml

step12 | dnfupdate |在重新启动之前部署MOP - Dnf更新包 | dnf_update_all_packages.yml

step13 | setiplink |在重新启动前部署MOP — 将VF IP链路信任设置为ON | osp16_setIpLink.yml

step14 |监视程序 |在重新启动前部署MOP — 配置IPMI监视程序 | osp16_config_ipmi_watchdog.yml

step15 | icedriver |在重新启动之前部署MOP — 更新E810 ICE驱动程序 | osp16_ice_driver_install.yml

step16 |重新启动 |重新引导所有Overcloud节点 | osp16_reboot_overcloud_hosts.yml

#第17步 | verifyrhosp |验证RHOSP部署配置和运行状况 | osp16_rhosp_verify.yml -e podname=

#==========

所有主机均可访问

==========

检查完成=> 17

通过的检查数=> 17

检查失败=> 00

==========

总体状态=>通过!!

==========

成功部署Overcloud后，确保Horizon控制面板可访问。

访问Horizon控制面板

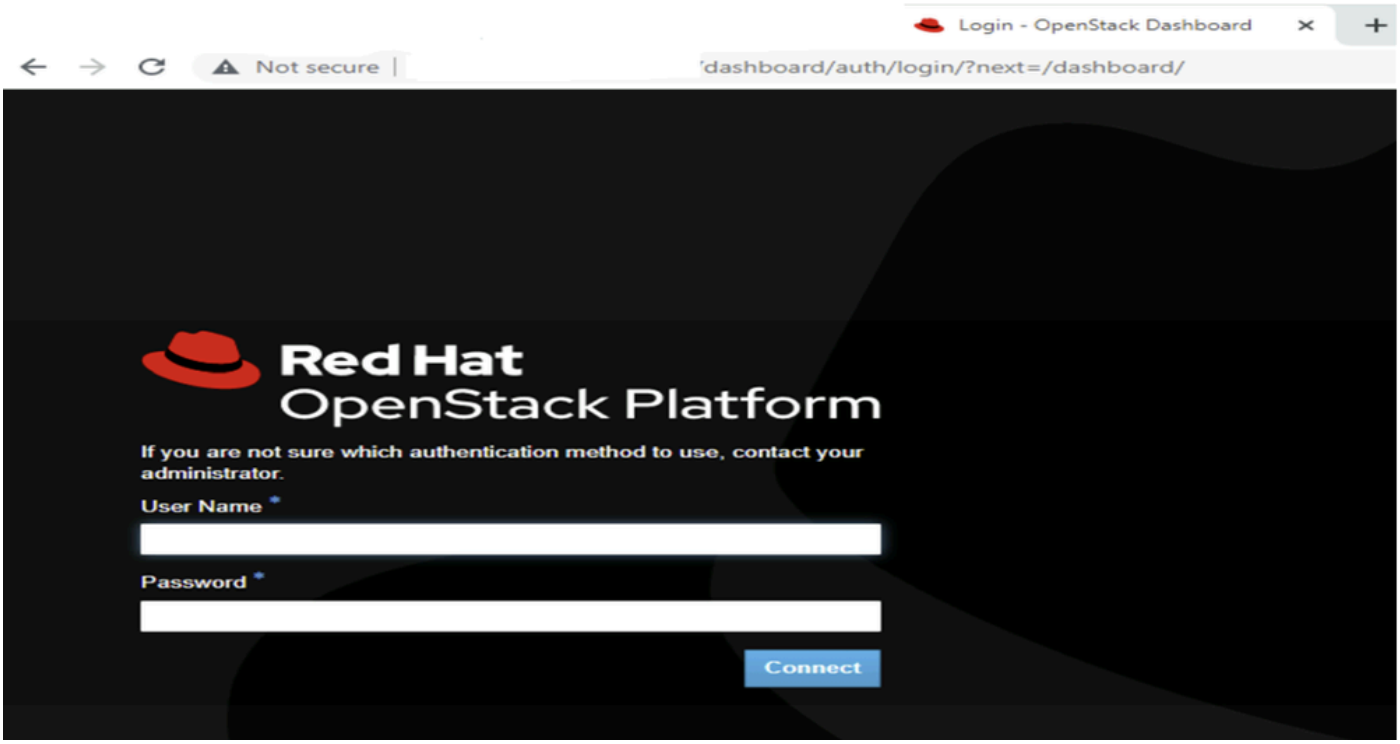
对于水平仪表板URL，请使用“overcloudcr”中的“OS_AUTH URL”。

```
[stack@[redacted]-ospd ~]$ cat [redacted]rc
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="}  /^OS_/ {print $1}' ); do unset $key ; done
export NOVA_VERSION=1.1
export COMPUTE_API_VERSION=1.1
export OS_USERNAME=admin
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_NO_CACHE=True
export OS_CLOUDNAME=[redacted]
export no_proxy=',[redacted],[redacted]
export PYTHONWARNINGS='ignore:Certificate has no, ignore:A true SSLContext object is not available'
export OS_AUTH_TYPE=password
export OS_PASSWORD=openstack
export OS_AUTH_URL='http://[redacted]:5000'
export OS_IDENTITY_API_VERSION=3
export OS_COMPUTE_API_VERSION=2.latest
export OS_IMAGE_API_VERSION=2
export OS_VOLUME_API_VERSION=3
export OS_REGION_NAME=regionOne

# Add OS_CLOUDNAME to PS1
if [ -z "${CLOUDPROMPT_ENABLED:-}" ]; then
    export PS1=${PS1:-""}
    export PS1=\${OS_CLOUDNAME:+"(\$OS_CLOUDNAME)"}\ $PS1
    export CLOUDPROMPT_ENABLED=1
fi
```

overcloudr文件创建

Horizon控制面板：



RHOSP群集的运行状况检查

<#root>

```
### Check OpenStack Services Status ###
```

```
# openstack compute service list
# openstack network agent list
# openstack volume service list
# openstack orchestration service list
# openstack identity service list
# openstack endpoint list
# openstack server list
# openstack image list
```

摘要

RHOSP 16.2部署指南提供全面的分步说明，说明如何使用Red Hat提供的经验证的工具和方法部署可扩展且生产就绪的OpenStack云环境。本指南专为系统管理员和云架构师量身定制，重点介绍如何使用基于TripleO(OpenStack on OpenStack)的OpenStack Director部署RHOSP 16.2。

该指南涵盖所有关键部署阶段，包括：

- 基础设施规划和先决条件
- 环境准备和网络配置
- Undercloud安装和配置
- 超云部署和部署后步骤
- 高可用性、存储和服务扩展选项

本指南对于寻求具备生态系统集成和Red Hat支持的可靠的企业级云平台的团队至关重要。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。