Red Hat OpenStack Platform 17.1

使用计算单元扩展部署

创建和配置多单元 overcloud 的指南

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 使用计算单元扩展部署

创建和配置多单元 overcloud 的指南

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南为云管理员提供了在大型 Red Hat OpenStack Platform (RHOSP) 部署中通过单元（cell）对 Compute 节点进行分组。

目录

使开源包含更多 3

对红**帽文档提供反馈** 4

第 1 章 多单元 OVERCLOUD 部署 5
1.1. 先决条件 5
1.2. 全局组件和服务 5
1.3. 特定于单元的组件和服务 6
1.4. 单元部署架构 6
1.5. MULTI-CELL 部署的注意事项 7

第 2 章 使用同一网络配置和部署多租户环境 9
2.1. 从 OVERCLOUD 堆栈 CONTROL PLANE 提取参数信息 9
2.2. 创建单元角色文件 9
2.3. 为 CELLCONTROLLER 角色设计主机 10
2.4. 使用相同的网络配置和部署每个单元堆栈 12
2.5. 后续步骤 13

第 3 章 使用路由网络配置和部署多同环境 14
3.1. 先决条件 14
3.2. 为单元网络路由准备 CONTROL PLANE 和默认单元 14
3.3. 从 OVERCLOUD 堆栈 CONTROL PLANE 提取参数信息 15
3.4. 为路由网络创建单元角色文件 16
3.5. 为单元角色设计主机 17
3.6. 使用路由网络配置和部署每个单元堆栈 19
3.7. 在部署后添加新单元子网 20
3.8. 后续步骤 21

第 4 章 在计算服务中创建和管理单元 22
4.1. 先决条件 22
4.2. 在 COMPUTE 服务中创建单元 22
4.3. 在单元中添加 COMPUTE 节点 23
4.4. 创建单元可用区 24
4.5. 从单元格删除 COMPUTE 节点 25
4.6. 删除单元 26

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 多单元 OVERCLOUD 部署

您可以使用单元将大型部署中的 Compute 节点划分为组，每个节点都有消息队列和包含实例信息的专用数据库。

默认情况下，director 为所有 Compute 节点使用一个单元（cell）安装 overcloud。此单元包含所有计算服务和数据库，以及所有实例和实例元数据。对于较大的部署，您可以使用多单元部署 overcloud，以适应大量 Compute 节点。在安装新的 overcloud 或之后的任何时间，您可以在环境中添加单元格。

在多单元部署中，每个单元格都运行特定于单元的计算服务和数据库的独立副本，并且仅为该单元中的实例存储实例元数据。全局信息和单元映射存储在全局控制器单元中，后者在一 个单元格失败时提供安全性和恢复。

小心

如果向现有 overcloud 添加单元格，默认单元格中的编排器也会执行超级编排器的角色。这会对部署中的单元和 overcloud 性能的单元通信产生负面影响。另外，如果您的默认单元离线，则超级编排器也会变为离线状态，这将停止整个 overcloud 部署。因此，要扩展现有的 overcloud，请不要将任何 Compute 节点添加到默认单元。取而代之，将 Compute 节点添加到您创建的新单元格中，允许默认单元充当超级编排器。

要创建多单元 overcloud，您必须执行以下任务：

1. 配置和部署 overcloud 以处理多个单元。
2. 在部署中创建并置备您需要的新单元。
3. 添加 Compute 节点到每个单元。
4. 将每个 Compute 单元添加到一个可用区。

1.1. 先决条件

- 您已部署了具有所需数量的 Controller 节点的基本 overcloud。

1.2. 全局组件和服务

无论计算单元的数量如何，每个 overcloud 都会在控制器单元格中部署以下组件。

compute API

为用户提供外部 REST API。

计算调度程序

决定在哪个 Compute 节点上分配实例。

放置服务

监控并分配计算资源到实例。

API 数据库

由计算 API 和计算调度程序服务用来跟踪实例的位置信息，并为构建但不调度的实例提供临时位置。在多单元部署中，此数据库还包含单元映射，用于为每个单元指定数据库连接。

cell0 数据库

专用数据库，用于有关无法调度的实例的信息。

超级编排器

此服务仅在多域部署中存在，以便协调全局服务和每个计算单元。此服务还会将失败的实例信息发送到 **cell0** 数据库。

1.3. 特定于单元的组件和服务

以下组件在每个计算单元中部署。

cell 数据库

包含有关实例的大部分信息。由全局 API、编排器和计算服务使用。

编排器

协调数据库查询和长时间运行的任务，使 Compute 节点无法直接访问数据库。

消息队列

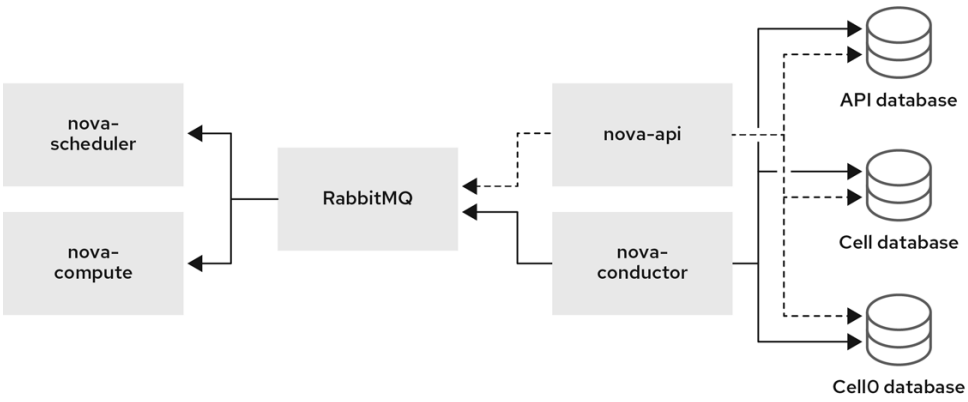
所有服务使用消息传递服务与单元格和全局服务相互通信。

1.4. 单元部署架构

director 安装的默认 overcloud 具有所有 Compute 节点的单一单元。您可以通过添加更多单元格来扩展 overcloud，如下架构图所示。

单单元部署架构

下图显示了默认单单元 overcloud 中基本结构和交互的示例。

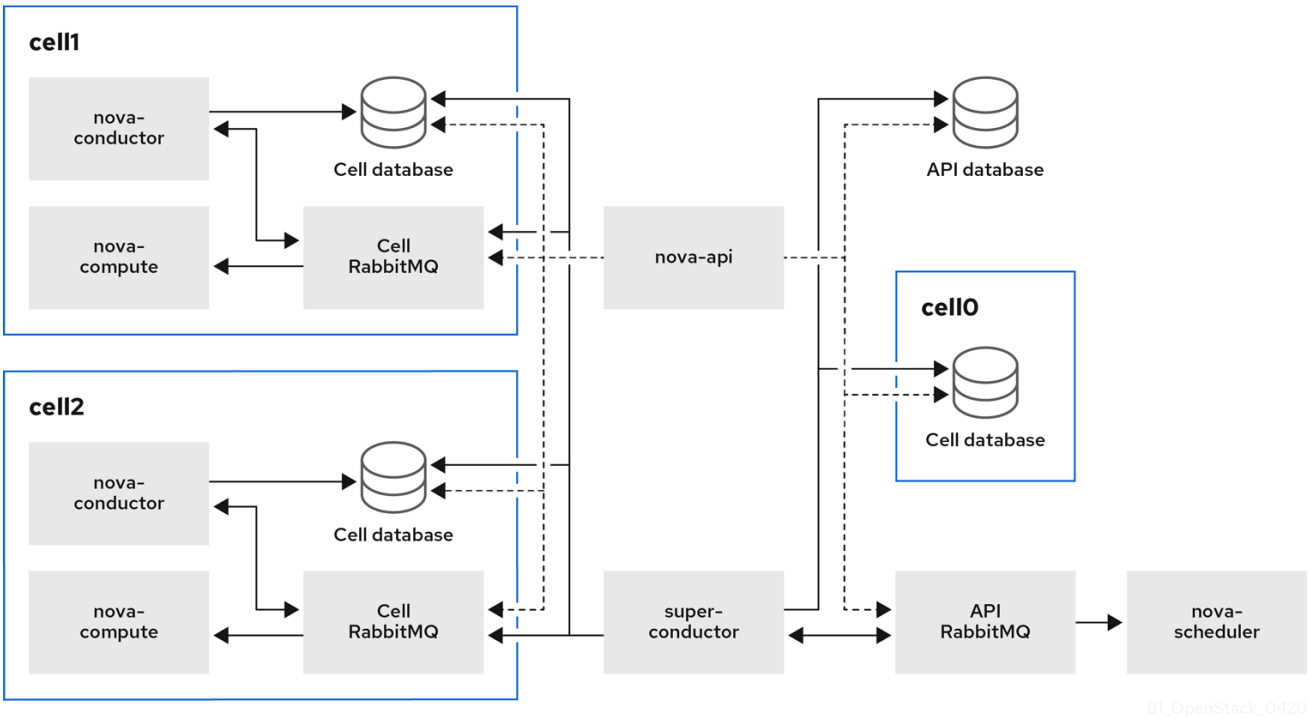


在本部署中，所有服务都配置为使用单个编排器在计算 API 和 Compute 节点之间进行通信，单个数据库存储所有实时实例数据。

在较小的部署中，此配置可能已经足够，但如果任何全局 API 服务或数据库失败，则整个计算部署无法发送或接收信息，无论高可用性配置是什么。

多cell 部署架构

下图显示了自定义多单元 overcloud 中基本结构和交互的示例。



在本部署中，计算节点划分为多个单元，每个单元都有自己的编排器、数据库和消息队列。全局服务使用超级编排器与每个单元通信，全局数据库仅包含整个 overcloud 所需的信息。

单元级服务无法访问全局服务。这种隔离在出现单元格失败时提供额外的安全性和故障安全功能。

重要

不要在第一个单元上运行任何计算服务，其名称为"default"。相反，请单独部署包含 Compute 节点的每个新单元。

1.5. MULTI-CELL 部署的注意事项

多站点部署中的最大 Compute 节点数量

跨所有单元格的 Compute 节点的最大数量为 500。

跨域实例迁移

不支持将实例从一个单元中的主机迁移到另一个单元的主机中。这个限制会影响以下操作：

- 冷迁移
- 实时迁移
- unshelve
- 调整大小
- 撤离

服务配额

计算服务配额在每个资源消耗点动态计算，而不是在数据库中静态计算。在多单元部署中，无法访问的单元无法实时提供使用信息，这可能会在单元再次访问时超过配额。
您可以使用放置服务和 API 数据库将配额计算配置为带失败或者无法访问的单元。

API 数据库

Compute API 数据库始终对所有单元进行全局设置，且每个单元都不能重复。

控制台代理

您必须为每个单元配置控制台代理，因为控制台令牌授权存储在单元数据库中。每个控制台代理服务器需要访问相应单元数据库的 **database.connection** 信息。

计算元数据 API

如果将相同的网络用于多个单元环境中的所有单元，您必须全局运行计算元数据 API，以便它在单元格之间进行桥接。当计算元数据 API 全局运行时，它需要访问 **api_database.connection** 信息。

如果您使用路由网络部署多个单元环境，您必须在每个单元中单独运行计算元数据 API，以提高性能和数据隔离。当计算元数据 API 在每个单元中运行时，**neutron-metadata-agent** 服务必须指向对应的 **nova-api-metadata** 服务。

您可以使用参数 **NovaLocalMetadataPerCell** 控制计算元数据 API 的运行位置。

第 2 章 使用同一网络配置和部署多租户环境

要配置 Red Hat OpenStack Platform (RHOSP)部署来使用相同的网络处理多个单元，您必须执行以下任务：

1. 从 overcloud 堆栈的 control plane 中提取参数信息。
2. 创建单元角色文件。您可以在单元格中为 **Compute** 节点使用默认的 Compute 角色，以及单元格节点的专用 **CellController** 角色。您还可以创建自定义角色，以便在多租户环境中使用，例如每个单元堆栈的自定义角色。有关创建自定义角色的更多信息，请参阅 可组合服务和自定义角色。
3. 为 **CellController** 角色指定主机。

注意

如果您为多单元环境创建了自定义角色，还必须为自定义角色指定主机。

4. 配置每个单元。
5. 部署每个单元堆栈。

2.1. 从 OVERCLOUD 堆栈 CONTROL PLANE 提取参数信息

在基本 overcloud 堆栈中，从名为 **default** 的第一个单元中提取参数信息。

流程

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

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

3. 将 overcloud 堆栈中的 **default** 单元中的单元格配置和密钥信息导出到多单元部署的一个新的通用环境文件：

```
(undercloud)$ openstack overcloud cell export --control-plane-stack overcloud \
 -f --output-file common/default_cell_export.yaml \
 --working-dir /home/stack/overcloud-deploy/overcloud/
```

此命令将 **EndpointMap**、**HostsEntry**、**AllNodesConfig**、**GlobalConfig** 参数以及密码信息导出到常见的环境文件。

提示

如果环境文件已存在，请输入带 **--force-overwrite** 或 **-f** 选项的命令。

2.2. 创建单元角色文件

当堆栈使用相同的网络且没有自定义角色时，您可以创建常见的单元角色文件，供所有单元堆栈使用。

流程

- 生成名为 **cell_roles_data.yaml** 的新角色数据文件，其中包含 **Compute** 和 **CellController** 角色：

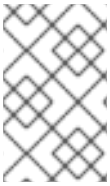
```
(undercloud)$ openstack overcloud roles generate \
  --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
  -o common/cell_roles_data.yaml Compute CellController
```

2.3. 为 CELLCONTROLLER 角色设计主机

要为 **CellController** 角色指定裸机节点，您必须使用资源类配置裸机节点，以标记 **CellController** 角色的节点。

提示

如果您为多个单元环境创建了自定义角色，您可以按照以下步骤为自定义角色配置资源类，方法是使用自定义角色的名称替换单元控制器名称。



注意

以下流程适用于尚未调配的新 overcloud 节点。要将资源类分配给已调配的现有 overcloud 节点，请缩减 overcloud 以取消置备节点，然后扩展 overcloud，以使用新的资源类分配来重新置备节点。有关更多信息，请参阅 扩展 overcloud 节点。

流程

1. 将 **CellController** 角色添加到节点定义模板： **node.json** 或 **node.yaml**，为 CellController 角色注册裸机节点。有关更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 注册节点。
2. 检查节点硬件：

```
(undercloud)$ openstack overcloud node introspect \
 --all-manageable --provide
```

   如需更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南中的创建 裸机节点硬件清单。

3. 检索节点列表来识别它们的 UUID：

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

4. 标记您要指定为单元控制器的每个裸机节点，并带有自定义单元控制器资源类：

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.CELL-CONTROLLER <node>
```

   - 将 **<node>** 替换为裸机节点的名称或 UUID。

5. 将 **CellController** 角色添加到节点定义文件 **overcloud-baremetal-deploy.yaml** 中，并定义您要分配给节点的任何预先节点放置、资源类、网络拓扑或其他属性：

```
- name: Controller
  count: 3
```

```
- name: Compute
  count: 3
  defaults:
    network_config:
      template: /home/stack/templates/nic-config/<cell_topology_file>
  instances:
  - hostname: cell1-compute-%index%
    name: computecell1
  - hostname: cell1-compute-%index%
    name: computecell2
  - hostname: cell1-compute-%index%
    name: computecell3
- name: CellController
  count: 1
  defaults:
    resource_class: baremetal.CELL-CONTROLLER
    network_config:
      template: /home/stack/templates/nic-config/<role_topology_file>
  instances:
  - hostname: cell1-cellcontroller-%index%
    name: cellcontroller
```

- 将 < **cell_topology_file** > 替换为用于单元堆栈的网络拓扑文件的名称，如 **compute.j2**。
- 将 **<role_topology_file** > 替换为用于 **CellController** 角色的网络拓扑文件的名称，如 **cell_controller_net_top.j2**。
  您可以重复使用现有网络拓扑，或为角色或单元创建新的自定义网络接口模板。如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 自定义网络接口模板。要使用默认网络定义设置，请不要在角色定义中包含 **network_config**。

有关您可以在节点定义文件中配置节点属性的属性的更多信息，请参阅 裸机节点置备属性。有关节点定义文件的示例，请参阅 节点定义文件 示例。

6. 为您的角色置备新节点：

```
(undercloud)$ openstack overcloud node provision \
[--stack <stack>] \
[--network-config \]
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 可选：将 **<stack>** 替换为置备裸机节点的堆栈的名称。默认为 **overcloud**。
- 可选：包含 **--network-config** 可选参数，为 **cli-overcloud-node-network-config.yaml** Ansible playbook 提供网络定义。如果您使用 **network_config** 属性在节点定义文件中未定义网络定义，则使用默认网络定义。
- 将 **<deployment_file>** 替换为用于部署命令生成的 heat 环境文件的名称，如 **/home/stack/templates/overcloud-baremetal-deployed.yaml**。

7. 在一个单独的终端中监控置备进度。当置备成功时，节点状态将从 **available** 变为 **active** ：

```
(undercloud)$ watch openstack baremetal node list
```

8. 如果您在没有 **--network-config** 选项运行 provisioning 命令，请在 **network-environment.yaml** 文件中配置 **<Role>NetworkConfigTemplate** 参数以指向 NIC 模板文件：

```
parameter_defaults:
  ComputeNetworkConfigTemplate: /home/stack/templates/nic-configs/compute.j2
  CellControllerNetworkConfigTemplate: /home/stack/templates/nic-
configs/<role_topology_file>
  ControllerNetworkConfigTemplate: /home/stack/templates/nic-configs/controller.j2
```

- 将 < **role_topology_file** > 替换为包含 **CellController** 角色的网络拓扑的文件的名称，如 **cell_controller_net_top.j2**。设置为 **compute.j2**，以使用默认的网络拓扑。

2.4. 使用相同的网络配置和部署每个单元堆栈

您必须配置每个单元堆栈，将单元标识为部署中的额外单元。

流程

1. 为新单元创建一个新目录：

```
(undercloud)$ mkdir cells
```

2. 为单元目录（**cells**）中每个额外单元创建一个新的环境文件，用于特定于单元的参数，例如 **/cells/cell1.yaml**。

3. 为每个环境文件添加以下参数，更新部署中每个单元的参数值：

```
parameter_defaults:
  # Disable network creation in order to use the `network_data.yaml` file from the overcloud
stack,
  # and create ports for the nodes in the separate stacks on the existing networks.
  ManageNetworks: false

  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # The DNS names for the VIPs for the cell
  CloudName: cell1.ooo.test
  CloudNameInternal: cell1.internalapi.ooo.test
  CloudNameStorage: cell1.storage.ooo.test
  CloudNameStorageManagement: cell1.storagemgmt.ooo.test
  CloudNameCtlplane: cell1.ctlplane.ooo.test
```

4. 使用其他环境文件将环境文件添加到堆栈中，并部署单元堆栈：

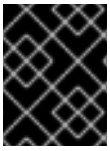
```
(undercloud)$ openstack overcloud deploy --templates \
 --stack cell1 \
 -e [your environment files] \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml \
 -r $HOME/common/cell_roles_data.yaml \
 -e $HOME/common/default_cell_export.yaml \
 -e $HOME/cells/cell1.yaml
```

为每个单元堆栈重复此步骤，直到部署所有单元堆栈。

2.5. 后续步骤

- 在计算服务中创建和管理单元

第 3 章 使用路由网络配置和部署多同环境



重要

本节中的内容在此发行版本中 *作为技术预览提供*，因此不受红帽完全支持。它只应用于测试，不应部署在生产环境中。如需更多信息，请参阅 技术预览。

要配置 Red Hat OpenStack (RHOSP)部署以处理带有路由网络的多个单元，您必须执行以下任务：

1. 为 overcloud 堆栈上的单元网络路由准备 control plane。
2. 从 overcloud 堆栈的 control plane 中提取参数信息。
3. 在单元堆栈上配置单元网络路由。
4. 为每个堆栈创建单元角色文件。您可以使用默认 **Compute** 角色作为单元内 Compute 节点的基础，专用 **CellController** 角色作为单元格节点的基础。您还可以创建自定义角色以用于您的多 cell 环境。有关创建自定义角色的更多信息，请参阅 可组合服务和自定义角色。
5. 为您创建的每个自定义角色指定一个主机。

注意

这个过程适用于带有单个 control plane 网络的环境。如果您的环境有多个 control plane 网络，如 spine leaf network 环境，那么您还必须为每个叶网络中的每个角色指定主机，以便您可以将节点标记到每个叶型中。如需更多信息，请参阅为叶节点设计角色。

6. 配置每个单元。
7. 部署每个单元堆栈。

3.1. 先决条件

- 为路由网络配置了 undercloud。如需更多信息，请参阅在 undercloud 中配置路由 spine-leaf。

3.2. 为单元网络路由准备 CONTROL PLANE 和默认单元

您必须在 overcloud 堆栈上配置路由，以便 overcloud 堆栈与单元通信。为达成此目标，创建一个网络数据文件，用于定义主堆栈中的所有网络和子网，并使用此文件部署 overcloud 堆栈和单元堆栈。

流程

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

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

3. 为通用堆栈配置创建新目录：

```
(undercloud)$ mkdir common
```

4. 将默认 **network_data_subnets_routed.yaml** 文件复制到您的 **common** 目录中，为您的 overcloud 堆栈添加可组合网络：

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-
templates/network_data_subnets_routed.yaml
~/common/network_data_routed_multi_cell.yaml
```

有关可组合网络的更多信息，请参阅 *Director 安装和使用指南中的可组合网络*。

5. 更新您网络的 **/common/network_data_routed_multi_cell.yaml** 中的配置，并更新单元子网名称以方便识别，例如，将 **internal_api_leaf1** 改为 **internal_api_cell1**。

6. 确保每个角色的 NIC 模板中的接口包括 **<network_name>InterfaceRoutes**，例如：

```
-
  type: vlan
  vlan_id:
    get_param: InternalApiNetworkVlanID
  addresses:
  -
    ip_netmask:
      get_param: InternalApiIpSubnet
  routes:
    get_param: InternalApiInterfaceRoutes
```

7. 使用其他环境文件，将 **network_data_routed_multi_cell.yaml** 文件添加到 overcloud 栈，并部署 overcloud：

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack overcloud \
 -e [your environment files]
 -n /home/stack/common/network_data_routed_multi_cell.yaml \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml
```

3.3. 从 OVERCLOUD 堆栈 CONTROL PLANE 提取参数信息

在基本 overcloud 堆栈中，从名为 **default** 的第一个单元中提取参数信息。

流程

1. 以 **stack** 用户的身份登录 undercloud。

2. Source **stackrc** 文件：

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

3. 将 overcloud 堆栈中的 **default** 单元中的单元格配置和密钥信息导出到多单元部署的一个新的通用环境文件：

```
(undercloud)$ openstack overcloud cell export --control-plane-stack overcloud \
 -f --output-file common/default_cell_export.yaml \
 --working-dir /home/stack/overcloud-deploy/overcloud/
```

-

此命令将 **EndpointMap**、**HostsEntry**、**AllNodesConfig**、**GlobalConfig** 参数以及密码信息导出到常见的环境文件。

提示

如果环境文件已存在，请输入带 **--force-overwrite** 或 **-f** 选项的命令。

3.4. 为路由网络创建单元角色文件

当每个堆栈使用不同的网络时，为每个单元堆栈创建一个单元角色文件，其中包含自定义单元角色。

注意

您必须为每个自定义角色创建一个类别。如需更多信息，请参阅为单元角色设计主机。

流程

1. 生成包含 **CellController** 角色的新角色数据文件，以及您在单元堆栈所需的其他角色。以下示例生成角色数据文件 **cell1_roles_data.yaml**，其中包括角色 **CellController** 和 **Compute** ：

```
(undercloud)$ openstack overcloud roles generate \
  --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
  -o cell1/cell1_roles_data.yaml \
  Compute:ComputeCell1 \
  CellController:CellControllerCell1
```

2. 将 **HostnameFormatDefault** 添加到新单元角色文件中的每个角色定义中：

```
- name: ComputeCell1
  ...
  HostnameFormatDefault: '%stackname%-compute-cell1-%index%'
  ServicesDefault:
  ...
  networks:
  ...
- name: CellControllerCell1
  ...
  HostnameFormatDefault: '%stackname%-cellcontrol-cell1-%index%'
  ServicesDefault:
  ...
  networks:
  ...
```

3. 将网络服务(neutron) DHCP 和元数据代理添加到 **ComputeCell1** 和 **CellControllerCell1** 角色（如果它们尚不存在）：

```
- name: ComputeCell1
  ...
  HostnameFormatDefault: '%stackname%-compute-cell1-%index%'
  ServicesDefault:
  - OS::TripleO::Services::NeutronDhcpAgent
  - OS::TripleO::Services::NeutronMetadataAgent
  ...
```

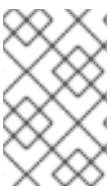
```
  networks:
  ...
- name: CellControllerCell1
  ...
  HostnameFormatDefault: '%stackname%-cellcontrol-cell1-%index%'
  ServicesDefault:
  - OS::TripleO::Services::NeutronDhcpAgent
  - OS::TripleO::Services::NeutronMetadataAgent
  ...
  networks:
  ...
```

4. 将您在 **network_data_routed_multi_cell.yaml** 中配置的子网添加到 **ComputeCell1** 和 **CellControllerCell1** 角色中：

```
- name: ComputeCell1
  ...
  networks:
    InternalApi:
      subnet: internal_api_subnet_cell1
    Tenant:
      subnet: tenant_subnet_cell1
    Storage:
      subnet: storage_subnet_cell1
...
- name: CellControllerCell1
  ...
  networks:
    External:
      subnet: external_subnet
    InternalApi:
      subnet: internal_api_subnet_cell1
    Storage:
      subnet: storage_subnet_cell1
    StorageMgmt:
      subnet: storage_mgmt_subnet_cell1
    Tenant:
      subnet: tenant_subnet_cell1
```

3.5. 为单元角色设计主机

要为单元角色指定裸机节点，您必须使用资源类配置裸机节点，以标记单元角色的节点。执行以下步骤为 **cellcontrollercell1** 角色创建裸机资源类。通过用自定义角色的名称替换单元控制器名称，为每个自定义角色重复此步骤。



注意

以下流程适用于尚未调配的新 overcloud 节点。要将资源类分配给已调配的现有 overcloud 节点，请缩减 overcloud 以取消置备节点，然后扩展 overcloud，以使用新的资源类分配来重新置备节点。有关更多信息，请参阅 扩展 overcloud 节点。

流程

1. 将 **cellcontrollercell1** 角色添加到节点定义模板： **node.json** 或 **node.yaml**，为 cellcontrollercell1 角色注册裸机节点。有关更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 注册节点。

2. 检查节点硬件：

```
(undercloud)$ openstack overcloud node introspect \
 --all-manageable --provide
```

   如需更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南中的创建 裸机节点硬件清单。

3. 检索节点列表来识别它们的 UUID：

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

4. 标记您要指定为单元控制器的每个裸机节点，并带有自定义单元控制器资源类：

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.CELL-CONTROLLER <node>
```

   - 将 **<node>** 替换为裸机节点的名称或 UUID。

5. 将 **cellcontrollercell1** 角色添加到节点定义文件 **overcloud-baremetal-deploy.yaml** 中，并定义您要分配给节点的任何预先节点放置、资源类、网络拓扑或其他属性：

```
- name: cellcontrollercell1
  count: 1
  defaults:
    resource_class: baremetal.CELL1-CONTROLLER
    network_config:
      template: /home/stack/templates/nic-config/<role_topology_file>
  instances:
  - hostname: cell1-cellcontroller-%index%
    name: cell1controller
```

   - 将 **<role_topology_file** > 替换为用于 **cellcontrollercell1** 角色的网络拓扑文件的名称，如 **cell1_controller_net_top.j2**。您可以重复使用现有网络拓扑，或为角色或单元创建新的自定义网络接口模板。如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 自定义网络接口模板。要使用默认网络定义设置，请不要在角色定义中包含 **network_config**。
     有关您可以在节点定义文件中配置节点属性的属性的更多信息，请参阅 裸机节点置备属性。
     有关节点定义文件的示例，请参阅 节点定义文件 示例。

6. 为您的角色置备新节点：

```
(undercloud)$ openstack overcloud node provision \
[--stack <stack>] \
[--network-config \]
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

   - 可选：将 **<stack>** 替换为置备裸机节点的堆栈的名称。默认为 **overcloud**。

- 可选：包含 **--network-config** 可选参数，为 **cli-overcloud-node-network-config.yaml** Ansible playbook 提供网络定义。如果您使用 **network_config** 属性在节点定义文件中未定义网络定义，则使用默认网络定义。

- 将 **<deployment_file>** 替换为用于部署命令生成的 heat 环境文件的名称，如 **/home/stack/templates/overcloud-baremetal-deployed.yaml**。

7. 在一个单独的终端中监控置备进度。当置备成功时，节点状态将从 **available** 变为 **active** ：

```
(undercloud)$ watch openstack baremetal node list
```

8. 如果您在没有 **--network-config** 选项运行 provisioning 命令，请在 **network-environment.yaml** 文件中配置 **<Role>NetworkConfigTemplate** 参数以指向 NIC 模板文件：

```
parameter_defaults:
  ComputeNetworkConfigTemplate: /home/stack/templates/nic-configs/compute.j2
  CellControllerCell1NetworkConfigTemplate: /home/stack/templates/nic-configs/<role_topology_file>
  ControllerNetworkConfigTemplate: /home/stack/templates/nic-configs/controller.j2
```

- 将 **<role_topology_file** > 替换为包含 **cellcontrollercell1** 角色的网络拓扑的文件的名称，如 **cell1_controller_net_top.j2**。设置为 **controller.j2** 以使用默认网络拓扑。

3.6. 使用路由网络配置和部署每个单元堆栈

执行以下步骤配置一个单元堆栈 **cell1**。对您要部署的每个额外单元堆栈重复这个过程，直到部署所有单元堆栈为止。

流程

1. 为单元目录中的额外单元格创建一个新的环境文件，以用于特定于单元的参数，例如：**/home/stack/cell1/cell1.yaml**。

2. 在环境文件中添加以下参数：

```
resource_registry:
  OS::TripleO::CellControllerCell1::Net::SoftwareConfig: /home/stack/templates/nic-configs/cellcontroller.yaml
  OS::TripleO::ComputeCell1::Net::SoftwareConfig: /home/stack/templates/nic-configs/compute.yaml

parameter_defaults:
  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # Enable local metadata API for each cell
  NovaLocalMetadataPerCell: True

  #Disable network creation in order to use the `network_data.yaml` file from the overcloud stack,
  # and create ports for the nodes in the separate stacks on the existing networks.
  ManageNetworks: false

  # Specify that this is an additional cell
  NovaAdditionalCell: True
```

```
# The DNS names for the VIPs for the cell
CloudDomain: redhat.local
CloudName: cell1.redhat.local
CloudNameInternal: cell1.internalapi.redhat.local
CloudNameStorage: cell1.storage.redhat.local
CloudNameStorageManagement: cell1.storagemgmt.redhat.local
CloudNameCtlplane: cell1.ctlplane.redhat.local
```

3. 要在每个单元格而不是全局 Controller 中运行 Compute 元数据 API，请在单元环境文件中添加以下参数：

```
parameter_defaults:
  NovaLocalMetadataPerCell: True
```

4. 在单元环境文件中添加单元的虚拟 IP 地址(VIP)信息：

```
parameter_defaults:
  ...
  VipSubnetMap:
    InternalApi: internal_api_cell1
    Storage: storage_cell1
    StorageMgmt: storage_mgmt_cell1
    External: external_subnet
```

这会在与 L2 网络段关联的子网中创建虚拟 IP 地址，这个单元格 Controller 节点已连接到这个地址。

5. 使用其他环境文件将环境文件添加到堆栈中，并部署单元堆栈：

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack cell1 \
 -e [your environment files] \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml \
 -r /home/stack/cell1/cell1_roles_data.yaml \
 -n /home/stack/common/network_data_spine_leaf.yaml \
 -e /home/stack/common/default_cell_export.yaml \
 -e /home/stack/cell1/cell1.yaml
```

3.7. 在部署后添加新单元子网

要在部署多代环境后向 overcloud 堆栈添加新单元子网，您必须更新 **NetworkDeploymentActions** 的值，使其包含 **'UPDATE'**。

流程

1. 将以下配置添加到 overcloud 堆栈的环境文件，以使用新的单元子网更新网络配置：

```
parameter_defaults:
  NetworkDeploymentActions: ['CREATE','UPDATE']
```

2. 将新单元子网的配置添加到 **/common/network_data_routed_multi_cell.yaml** 中。

3. 部署 overcloud 堆栈：

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack overcloud \
 -n /home/stack/common/network_data_routed_multi_cell.yaml \
 -e [your environment files]
```

4. 可选：将 **NetworkDeploymentActions** 重置为下一个部署的默认值：

```
parameter_defaults:
  NetworkDeploymentActions: ['CREATE']
```

3.8. 后续步骤

- 在计算服务中创建和管理单元

第 4 章 在计算服务中创建和管理单元

使用单元堆栈部署 overcloud 后，您必须在计算服务中创建单元。要在计算服务中创建单元格，您可以为全局 API 数据库中的单元和消息队列映射创建条目。然后，您可以通过在其中一个 Controller 节点上运行单元发现将 Compute 节点添加到单元格中。

要创建单元，您必须执行以下任务：

1. 使用 **nova-manage** 实用程序在全局 API 数据库中创建单元和消息队列映射记录。
2. 添加 Compute 节点到每个单元。
3. 为每个单元创建一个可用性区域。
4. 将各个单元格中的所有计算节点添加到该单元的可用性区域。

4.1. 先决条件

- 您已使用多个单元配置并部署了 overcloud。

4.2. 在 COMPUTE 服务中创建单元

使用新单元堆栈部署 overcloud 后，您必须在计算服务内创建单元。要在计算服务中创建单元格，您可以为全局 API 数据库中的单元和消息队列映射创建条目。注意：您必须对创建和启动的每个单元重复这个过程。您可以自动执行 Ansible playbook 中的步骤。如需 Ansible playbook 的示例，请参阅 OpenStack 社区文档中的 创建单元和发现 Compute 节点 部分。社区文档按原样提供，不受正式支持。

流程

1. 获取 control plane 和单元控制器的 IP 地址：

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r .ctlplane_ip)
$ CELL_CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/cell1/config-download/cell1/tripleo-ansible-inventory.yaml --host <cell_controller_node> | jq -r .ctlplane_ip)
```

   - 将 **<controller_node>** 替换为 Controller 节点的名称，例如 **controller-0**。
   - 将 **<cell_controller_node>** 替换为单元 Controller 节点的名称，如 **cell1-controller-0**。

2. 在所有 Controller 节点中添加单元信息。此信息用于从 undercloud 连接到单元端点。以下示例使用前缀 **cell1** 来只指定单元系统并排除控制器系统：

```
(undercloud)$ CELL_INTERNALAPI_INFO=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 egrep cell1.*\.internalapi /etc/hosts)
(undercloud)$ ansible -i /home/stack/overcloud-deploy/overcloud/config-download/overcloud/tripleo-ansible-inventory.yaml \
 Controller -b -m lineinfile -a "dest=/etc/hosts line=\"$CELL_INTERNALAPI_INFO\""
```

3. 从 **transport_url** 参数获取控制器单元的消息队列端点，以及来自 **database.connection** 参数的控制器单元的数据库连接：

```
(undercloud)$ CELL_TRANSPORT_URL=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 sudo crudini --get /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf \
 DEFAULT transport_url)
(undercloud)$ CELL_MYSQL_VIP=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 sudo crudini --get /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf \
 database connection | awk -F[@/] '{print $4}')
```

4. 登录到其中一个全局 Controller 节点并创建单元：

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman \
 exec -i -u root nova_api \
 nova-manage cell_v2 create_cell --name cell1 \
 --database_connection "{scheme}://{username}:{password}@$CELL_MYSQL_VIP/nova?
{query}" \
 --transport-url "$CELL_TRANSPORT_URL"
```

5. 检查单元是否已创建并出现在单元列表中：

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_cells --verbose
```

6. 重启 Controller 节点上的 Compute 服务：

```
$ ansible -i /usr/bin/tripleo-ansible-inventory Controller -b -a \
"systemctl restart tripleo_nova_api tripleo_nova_conductor tripleo_nova_scheduler"
```

7. 检查单元控制器服务是否已置备：

```
(overcloud)$ openstack compute service list -c Binary -c Host -c Status -c State
+----------------+------------------------+---------+-------+
| Binary         | Host                   | Status  | State |
+----------------+------------------------+---------+-------+
| nova-conductor | controller-0.ostest    | enabled | up    |
| nova-scheduler | controller-0.ostest    | enabled | up    |
| nova-conductor | cellcontroller-0.ostest | enabled | up    |
| nova-compute   | compute-0.ostest       | enabled | up    |
| nova-compute   | compute-1.ostest       | enabled | up    |
+----------------+------------------------+---------+-------+
```

4.3. 在单元中添加 COMPUTE 节点

在其中一个 Controller 节点上运行单元主机发现功能，发现 Compute 节点，并使用 node-to-cell 映射更新 API 数据库。

流程

1. 以 **stack** 用户身份登录 undercloud。
2. 为单元获取 control plane 的 IP 地址：

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)
```

- 将 **<controller_node>** 替换为 Controller 节点的名称，例如 **controller-0**。

3. 公开并分配 Compute 主机到单元：

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 discover_hosts --by-service --verbose
```

4. 验证 Compute 主机是否已分配给这个单元：

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 list_hosts
```

4.4. 创建单元可用区

您必须为每个单元创建一个可用区(AZ)，以确保在该单元内在 Compute 节点上创建的实例仅迁移到同一单元单元中的其他 Compute 节点。不支持在单元格之间迁移实例。

创建单元 AZ 后，您需要将单元中的所有计算节点添加到单元 AZ。默认单元必须在与计算单元不同的可用区中。

流程

1. 获取 **overcloudrc** 文件：

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

2. 为单元创建 AZ：

```
(overcloud)# openstack aggregate create \
 --zone <availability_zone> \
 <aggregate_name>
```

- 将 **<availability_zone>** 替换为您要分配给可用区的名称。
- 将 **<aggregate_name>** 替换为您要分配给主机聚合的名称。

3. 可选：在可用区中添加元数据：

```
(overcloud)# openstack aggregate set --property <key=value> \
  <aggregate_name>
```

- 将 **<key=value>** 替换为您的原始键值对。您可以根据需要添加任意数量的键值属性。
- 将 **<aggregate_name>** 替换为可用区主机聚合的名称。

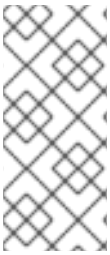
4. 检索分配给该单元的 Compute 节点列表：

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 list_hosts
```

5. 将分配给单元格的 Compute 节点添加到单元可用区：

```
(overcloud)# openstack aggregate add host <aggregate_name> \
 <host_name>
```

- 将 **<aggregate_name>** 替换为可用区主机聚合的名称，以将 Compute 节点添加到其中。
- 将 **<host_name>** 替换为添加到可用区的 Compute 节点的名称。



注意

- 您不能使用 **OS::TripleO::Services::NovaAZConfig** 参数在部署期间自动创建 AZ，因为此阶段尚未创建单元格。
- 不支持在单元格之间迁移实例。要将实例移动到不同的单元格，您必须将其从旧单元中删除，并在新单元格中重新创建。

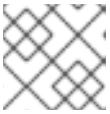
有关主机聚合和可用区的更多信息，请参阅 创建和管理主机聚合。

4.5. 从单元格删除 COMPUTE 节点

要从单元格中删除 Compute 节点，您必须从单元格中删除所有实例，并从放置数据库中删除主机名。

流程

1. 从单元格中的 Compute 节点删除所有实例。



注意

不支持在单元格之间迁移实例。您必须删除实例并在另一个单元中创建它们。

2. 在其中一个全局 Controller 上，从单元中删除所有 Compute 节点：

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_hosts

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 delete_host --cell_uuid <uuid> --host <compute>
```

- 将 **<controller_node** > 替换为 Controller 节点的名称，如 **controller-0**。

3. 从放置服务中删除单元格的资源供应商，以确保在稍后将具有相同主机名的 Compute 节点添加到另一个单元时可以使用主机名：

```
(undercloud)$ source ~/overcloudrc

(overcloud)$ openstack resource provider list
```

```
+--------------------------------------+--------------------------------------+------------+
| uuid                                 | name                                 | generation |
+--------------------------------------+--------------------------------------+------------+
| 9cd04a8b-5e6c-428e-a643-397c9bebcc16 | computecell1-novacompute-0.site1.test |
11 |
+--------------------------------------+--------------------------------------+------------+

(overcloud)$ openstack resource provider  \
 delete 9cd04a8b-5e6c-428e-a643-397c9bebcc16
```

4.6. 删除单元

要删除单元格，您必须首先从单元中删除所有实例和 Compute 节点，如从单元中删除计算节点所述。然后，您将删除单元本身和单元堆栈。

流程

1. 在其中一个全局控制器上，删除单元：

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_cells

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 delete_cell --cell_uuid <uuid>
```

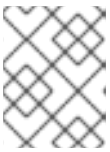
   - 将 **<controller_node>** 替换为 Controller 节点的名称，如 **controller-0**。

2. 启动临时 Heat 进程并导出 heat 环境：

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-
deploy/cell1/heat-launcher --restore-db
(undercloud)$ export OS_CLOUD=heat
```

3. 从 overcloud 删除单元堆栈：

```
$ openstack stack delete <stack name> --wait --yes
```



注意

如果您为控制器和计算单元部署了独立的单元堆栈，请先删除计算单元堆栈，然后删除控制器单元堆栈。

4. 当单元堆栈删除完成后，从 undercloud 中删除临时 Heat 进程：

```
(undercloud)$ openstack tripleo launch heat --kill
```