红帽企业 Linux，CentOS，OpenStack 安装指南和

软呢帽

冰窖 (2014 年-06-20)

OpenStack® 系统包括你单独安装但这项工作在一起根据您的云计算需要的几个关键项目。这些项目包括计算、 标识服务、 网络、 图像服务、 数据块存储，对象存储、 遥测、 业务流程和数据库。可以单独安装任何这些项目并将它们配置单机或作为连接的实体。本指南说明你如何安装 OpenStack 红帽企业 Linux 和其衍生物通过座谈存储库使用通过 Fedora 20 以及可用的软件包。配置选项和示例的配置文件的解释都包括在内。

表的内容

数字列表

表的列表

前言

公约

OpenStack 文档使用几个排版约定。

通告

通知采取三种形式：

请注意

说明中的信息通常是在一个方便的提示或提醒的形式。

重要

重要通知中的信息是你必须意识到之前的东西。

警告

警告中的信息是至关重要的。警告提供有关风险的数据丢失或安全性问题的其他信息。

命令提示

前缀为 # 提示符下的命令都将由 root 用户执行。如果可用，这些例子也可以通过使用 **sudo** 命令的执行。

可以由任何用户包括根执行带有前缀 $ 提示符下的命令。

文档更改历史记录

此版本的指南 》 将替换和废弃所有以前的版本。下表介绍了最新的更改：

修订日期	更改的摘要
2014 年 4 月 16 日	• 更新为冰窖，返工联网安装程序使用 ML2 作为一个插件，添加数据库服务安装程序，提高了基本配置的新篇章。

2013 年 10 月 25 日	• 添加了初始的 Debian 支持。
2013 年 10 月 17 日	• 哈瓦那释放。
2013 年 10 月 16 日	• 添加对 SUSE Linux 企业的支持。
2013 年 10 月 8 日	• 完成重组为哈瓦那的。
2013 年 9 月 9 日	• 也为 openSUSE 的生成。
2013 年 8 月 1 日	• 修复到对象存储验证步骤。修复 bug 1207347 .
2013 年 7 月 25 日	• 添加创建煤渣用户和服务租客的除了。修复 bug 1205057 .
2013 年 5 月 8 日	• 更新书名的一致性。

修订日期	更改的摘要
2013 年 5 月 2 日	• 更新封面，并固定在附录中的小错误。

1。体系结构

表的内容

概述

OpenStack 项目是开源云计算平台，支持所有类型的云环境。该项目旨在为简单的实现，巨大的可扩展性和一组丰富的功能。来自世界各地的云计算专家项目作出贡献。

OpenStack 提供通过各种互补的服务基础设施作为服务 (*IaaS*) 解决方案。每个服务提供便利这种集成应用程序编程接口 (*API*)。下表提供 OpenStack 服务的列表：

表 1.1。OpenStack 服务

服务	项目名称	描述
仪表板	*地平线*	提供了一个基于 web 的自助服务门户与底层 OpenStack 服务，例如启动实例、 分配的 IP 地址和配置访问控件进行交互。
计算	*新星*	管理的生命周期的计算实例 OpenStack 的环境中。职责包括产卵、调度和退役的虚拟机上的需求。
网络	*中子*	可以作为一种服务对于其他 OpenStack 服务，例如 OpenStack 计算的网络连接。提供了一个 API 的用户来定义网络和附件到他们。有一个可插入的体系结构，支持许多流行的网络供应商和技术。
存储		

对象存储	*斯威夫特*	通过*基于 Rest* 的存储和检索任意非结构化的数据对象，基于 HTTP 的 API。它高度是容错功能以及其数据复制和建筑规模。其执行情况并不像与可挂载目录的文件服务器。
数据块存储	*煤渣*	提供持久性数据块存储到正在运行的实例。其可插拔驱动程序体系结构方便的创建和管理的块存储设备。
共享的服务		
身份服务	*基石*	提供其他 OpenStack 服务身份验证和授权服务。提供对所有 OpenStack 服务终结点的目录中。
图像服务	*看一眼*	存储和检索虚拟机磁盘映像。OpenStack 计算利用这期间实例资源调配。
遥测	*测云*	监视和米 OpenStack 云计费、标杆管理、可扩展性和统计的目的。
服务	**项目名称**	**描述**
高级服务		
业务流程	*热*	通过使用本机*热*模板格式或 AWS CloudFormation 模板格式，通过 OpenStack 本机 REST API 和 CloudFormation 兼容的查询 API 编排多个复合的云计算应用程序。
数据库服务	*宝库*	为这两个关系和非关系数据库引擎提供可扩展和可靠云数据库作为-一-服务功能。

本指南介绍了如何将这些服务的功能测试环境中部署，并通过例子，教你如何建立一个生产环境。

概念体系结构

启动虚拟机或实例涉及到多的几个服务之间的相互作用。下面的关系图提供了一个典型的 OpenStack 环境的概念架构。

图 1.1。概念体系结构

示例体系结构

OpenStack 是高度可配置以满足不同的需求，各种计算、 网络和存储选项。本指南使您可以选择您自己使用的基本服务和可选服务组合的 OpenStack 冒险。本指南使用以下示例体系结构：

- 三节点体系结构带 OpenStack 网络 （中子）。请参阅图 1.2，"Threenode 建筑与 OpenStack 联网 （中子）"[4]。

- 的基本控制器节点运行身份服务、 服务形象、 管理部分的计算和网络、 网络插件和仪表板。它还包括支持服务，如数据库、*消息代理*和*网络时间协议（NTP）*。

  （可选） 的控制器节点也运行部分的数据块存储、 对象存储、 数据库服务、 业务流程和遥测。这些组件提供的附加功能为您的环境。

- 的网络节点运行插件的联网、 第二层代理和调配和经营租客网络的几个第三层代理。第二层服务包括虚拟网络和隧道的资源调配。第 3 层服务包括路由、 *NAT* 和 *DHCP*。此节点还可以处理外部 （互联网） 连接为租客虚拟机或实例。

- 计算节点运行计算，经营租客虚拟机或实例的虚拟机管理程序部分。默认情况下计算使用 KVM 虚拟机监控程序。计算节点也运行网络插件和的经营租客网络、 实施安全组的第二层代理。您可以运行多个计算节点。

  （可选），计算节点也运行遥测代理。此组件为您的环境提供额外的功能。

请注意

当你实现这种体系结构，请跳过的部分称为"遗产联网 (nova 网络)"[66]在中章 7，"添加一种网络服务"[48]。若要使用可选服务，您可能需要安装额外的节点，在后续章节中所述。

图 1.2。三节点体系结构与 OpenStack 联网 （中子）

- 两个节点体系结构与旧版联网 (nova 网络)。请参阅图 1.3，"Twonode 建筑与旧式联网 (nova 网络)"[5]。

- 基本*控制器节点*运行身份服务、 图像服务、 管理部分的计算和启动一个简单的实例所需的仪表板。它还包括支持服务，如数据库、 消息代理和 NTP。

  （可选） 的控制器节点也运行部分的数据块存储、 对象存储、 数据库服务、 业务流程和遥测。这些组件提供的附加功能为您的环境。

- 基本*计算节点*运行计算，经营*租客的虚拟机*或实例的*虚拟机管理程序*部分。默认情况下，计算使用 *KVM* 作为*虚拟机管理程序*。计算还规定和租客网络的运作，并实现了*安全组*。您可以运行多个计算节点。

  （可选），计算节点也运行遥测代理。此组件为您的环境提供额外的功能。

请注意

当你实现这种体系结构，请跳过有一段叫"OpenStack 网络 （中子）"[48]在中章 7，"添加一种网络服务"[48]。若要使用可选服务，您可能需要安装额外的节点，在后续章节中所述。

图 1.3。两个节点体系结构与旧版联网 (nova 网络)

2。基本环境配置

表的内容

本章解释了如何配置在每个节点示例体系结构包括两个节点体系结构与旧版网络和三节点体系结构与 OpenStack 联网 （中子）.

请注意

虽然大多数环境包括 OpenStack 标识、 图像服务

计算，至少一个网络服务和仪表板，OpenStack 对象存储可以独立于其他大多数服务。如果您的用例只涉及对象存储，您可以跳到叫"对象存储的系统要求"部分 [83]。然而，仪表板将会工作，如果没有至少 OpenStack 图像服务和计算.

请注意

必须使用具有管理权限的帐户来配置每个节点。以 root 用户身份运行的命令或配置 sudo 实用程序。

在您开始之前

一个功能的环境，OpenStack 不需要大量的资源。我们建议您的环境满足或超过下列最低要求，可以支持几个最小 *CirrOS* 实例： • 控制器节点： 1 处理器、2GB 内存和 5 GB 存储空间

• 网络节点： 1 处理器、 512 MB 内存和 5 GB 存储空间

• 计算节点： 1 处理器、 2GB 内存和 10 GB 存储空间

要尽量减少混乱和为 OpenStack 提供更多的资源，我们推荐您的 Linux 发行版的最小安装。同时，我们强烈建议您安装您的发行版的 64 位版本上至少计算节点。如果您在计算节点上安装 32 位版本的您的发行版，尝试启动一个实例使用 64 位的图像将会失败。

请注意

每个节点上的单个磁盘分区为最基本的安装工作。但是，您应该考虑*逻辑卷管理器 (LVM)*可选服务，如数据块存储装置。

很多用户建立其上的*虚拟机 (Vm)*的测试环境。虚拟机的主要好处包括以下内容：

- 一台物理服务器可以支持多个节点，每个都有几乎任何网络接口的数量。
- 有能力采取定期在整个安装过程和"回滚"到一个工作配置出现问题时"拍镜头"。

然而，Vm 将降低您的实例，性能，特别是如果您的虚拟机管理程序和/或处理器缺少对嵌套的虚拟机的硬件加速支持。

请注意

如果你选择在虚拟机上安装，请确保您的虚拟机监控程序*外部网络*上允许*混杂模式*.

有关系统要求的详细信息，请参阅 OpenStack 操作指南.

网络

在您选择部署的架构的每个节点上安装操作系统之后，您必须配置的网络接口。我们建议您禁用任何自动化的网络管理工具，并手动编辑相应的配置文件为您的发行版。关于如何在您的发行版上配置网络的详细信息，请参阅文档。

要禁用网络管理器并启用网络服务：

-

```
# 服务网络管理器停止
# 服务网络启动
# chkconfig 网络管理器关闭
# chkconfig 网络上
```

RHEL 和衍生品包括 CentOS 和科学 Linux 在默认情况下启用具有限制性的*防火墙*。这在安装期间，某些步骤将失败，除非你改变或禁用防火墙。有关保护您的安装的进一步信息，请参阅 OpenStack 安全指南.

```
# 服务 firewalld 停止
# 服务 iptables 的开始
```

在 Fedora，firewalld 替换 iptables 作为默认的防火墙系统。虽然你可以成功地使用 firewalld，本指南引用 iptables 为与其他发行版的兼容性。

要禁用 firewalld 并启用 iptables：

-

```
# chkconfig firewalld 关闭
# chkconfig iptables 上
```

继续看网络配置为例 OpenStack 联网 （中子）或遗产联网 （新星网络）建筑。

OpenStack 联网 （中子）

带 OpenStack 网络 （中子） 的示例体系结构需要一个控制器节点、 一个网络节点和至少一个计算节点。控制器节点包含一个*管理网络*上的网络接口。网络节点都包含一个网络接口上的管理网络，在*实例隧道网络*上，一个，一个*外部网络*上。计算节点包含一个网络接口上的管理网络，另一个实例隧道网络上。

请注意

网络接口名称不同分布的差异。传统上，接口使用"eth"后跟一个序列号。要涵盖所有的变化，本指南只是指作为具有的最低数量的接口的第一个接口、 作为接口与中间的数，第二个接口和作为具有最高编号的接口的第三个接口。

图 2.1。三节点体系结构与 OpenStack 联网 （中子）

除非您打算使用在此的示例体系结构提供的确切配置，您必须修改此过程以匹配您的环境中的网络。此外，每个节点必须通过 IP 地址名称解析其他节点。例如，该*控制器*的名称必须解析到 10.0.0.11 上的控制器节点的管理接口的 IP 地址。

警告

重新配置网络接口将中断网络连接。我们建议使用这些程序的本地终端会话。

控制器节点

要配置网络：

- 为管理接口配置的第一个接口：

  IP 地址：10.0.0.11

  网络掩码： 255.255.255.0 （或 24）

默认网关： 10.0.0.1

若要配置名称解析：

1. 将该节点的主机名设置为*控制器*。

2. 编辑 /etc/主机文件来包含下列内容：

```
# 控制器
10.0.0.11 控制器

# 网络
10.0.0.21 网络

# compute1
10.0.0.31 计算 1
```

网络节点

要配置网络：

1. 作为管理接口配置的第一个接口：

   IP 地址：10.0.0.21

   网络掩码： 255.255.255.0 （或 24）

   默认网关： 10.0.0.1

2. 配置第二个接口作为实例隧道接口：

   IP 地址：10.0.1.21
   网络掩码： 255.255.255.0 （或 24）

3. 外部接口使用特殊的配置没有给它分配 IP 地址。作为外部接口配置的第三个接口：

   INTERFACE_NAME 替换为实际的接口名称。例如， *eth2* 或 *ens256*.

   - 编辑 /etc/sysconfig/network-scripts/ifcfg-*INTERFACE_NAME* 文件包含下列内容：

     请不要更改 HWADDR 和 UUID 的键。

```
设备 =INTERFACE_NAME
类型 = 以太网
启动 ="yes"
BOOTPROTO ="none"
```

4．重新启动网络：

```
# 服务网络重启
```

若要配置名称解析：

1．将该节点的主机名设置为网络。

2．编辑 /etc/主机文件来包含下列内容：

```
# 网络
10.0.0.21 网络

# 控制器
10.0.0.11 控制器

# compute1
10.0.0.31 计算 1
```

计算节点

要配置网络：

1.作为管理接口配置的第一个接口：

IP 地址：10.0.0.31

网络掩码： 255.255.255.0 （或 24）

默认网关： 10.0.0.1

请注意

额外的计算节点应使用 10.0.0.32、 10.0.0.33，等等。

2.作为实例隧道接口配置第二个接口：

IP 地址：10.0.1.31

网络掩码： 255.255.255.0 （或 24）

请注意

额外的计算节点应使用 10.0.1.32、 10.0.1.33，等等。

若要配置名称解析：

1．将该节点的主机名设置为 compute1.

2．编辑 /etc/主机文件来包含下列内容：

```
# compute1
10.0.0.31 计算 1

# 控制器
10.0.0.11 控制器

# 网络
10.0.0.21 网络
```

验证连接性

我们建议您在继续审议之前验证网络连接到互联网和节点之间。

1．从*控制器*节点，**坪**在互联网上的网站：

```
# ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
从 174.143.194.225 的 64 个字节： icmp_seq = 1 ttl = 54 时间 = 18.3 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 2 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 3 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 4 ttl = 54 时间 = 17.4 ms

— — openstack.org 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间
3022ms rtt min/平均/最大/mdev = 17.489/17.715/18.346/0.364 女士
```

2．从*控制器*节点，**坪**管理接口的*网络*节点上：

```
# ping-c 4 网络PING 网络 (10.0.0.21) 56(84) 字节
的数据。
64 字节从网络 (10.0.0.21)： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
64 字节从网络 (10.0.0.21)： icmp_seq = 2 ttl = 64 时间 = 0.202，远女士
```

```
64 字节从网络 （10.0.0.21）： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
64 字节从网络 （10.0.0.21）： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

—— 网络 ping 统计--4 数据包传输，4 收到，0%数据包丢失，时间 3000ms rtt
min/平均/最大/mdev = 0.202/0.217/0.263/0.030 女士
```

3．从*控制器*节点，**坪**管理界面的*计算*节点上：

```
# ping-c 4 compute1
PING compute1 (10.0.0.31) 56(84) 字节的数据。
从 compute1 的 64 字节 （10.0.0.31）： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
从 compute1 的 64 字节 （10.0.0.31）： icmp_seq = 2 ttl = 64 时间 = 0.202，远
女士
从 compute1 的 64 字节 （10.0.0.31）： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
从 compute1 的 64 字节 （10.0.0.31）： icmp_seq = 4 ttl = 64 时间 = 0.202，远
女士

—— 网络 ping 统计--4 数据包传输，4 收到，0%数据包丢失，时间 3000ms rtt
min/平均/最大/mdev = 0.202/0.217/0.263/0.030 女士
```

4．从*网络*节点，**坪**在互联网上的网站：

```
# ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
从 174.143.194.225 的 64 个字节： icmp_seq = 1 ttl = 54 时间 = 18.3 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 2 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 3 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 4 ttl = 54 时间 = 17.4 ms

— — openstack.org 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间
3022ms rtt min/平均/最大/mdev = 17.489/17.715/18.346/0.364 女士
```

5．从*网络*节点，**坪**管理接口*控制器*节点上：

```
# ping-c 4 控制器坪控制器 (10.0.0.11) 56(84) 字节的数
据。
64 字节从控制器 （10.0.0.11）： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
64 字节从控制器 （10.0.0.11）： icmp_seq = 2 ttl = 64 时间 = 0.202，远女士
64 字节从控制器 （10.0.0.11）： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
64 字节从控制器 （10.0.0.11）： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

—— 控制器坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间 3000ms rtt
min/平均/最大/mdev = 0.202/0.217/0.263/0.030 女士
```

6．从*网络*节点， **ping** 该实例隧道的*计算*节点上的接口：

```
# ping-c 4 10.0.1.31PING 10.0.1.31 (10.0.1.31)
56(84) 字节的数据。
64 字节从 10.0.1.31 （10.0.1.31）： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
64 字节从 10.0.1.31 （10.0.1.31）： icmp_seq = 2 ttl = 64 时间 = 0.202，远女士
64 字节从 10.0.1.31 （10.0.1.31）： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
64 字节从 10.0.1.31 （10.0.1.31）： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

---10.0.1.31 ping--4 传输的信息包，4 收到的统计，0%数据包丢失，时间
3000ms rtt min，avg，马克斯，.= 0.202/0.217/0.263/0.030 女士
```

7．从*计算*节点，**坪**在互联网上的网站：

```
# ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
从 174.143.194.225 的 64 个字节： icmp_seq = 1 ttl = 54 时间 = 18.3 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 2 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 3 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 4 ttl = 54 时间 = 17.4 ms

— — openstack.org 坪统计--
4 传输的数据包，，收到的 4 0%数据包丢失，时间 3022ms
rtt 最小/平均/最大/mdev = 17.489/17.715/18.346/0.364 女士
```

8．从*计算*节点，**坪**管理接口*控制器*节点上：

```
# ping-c 4 控制器坪控制器 (10.0.0.11) 56(84) 字节的数
据。
64 字节从控制器 （10.0.0.11）： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
64 字节从控制器 （10.0.0.11）： icmp_seq = 2 ttl = 64 时间 = 0.202，远女士
64 字节从控制器 （10.0.0.11）： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
64 字节从控制器 （10.0.0.11）： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

— — 控制器坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间 3000ms rtt
min/平均/最大/mdev = 0.202/0.217/0.263/0.030 女士
```

9．从*计算*节点， **ping** 该实例隧道接口的*网络*节点上：

```
# ping-c 4 10.0.1.21PING 10.0.1.21 (10.0.1.21)
56(84) 字节的数据。
64 字节从 10.0.1.21 （10.0.1.21）： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
64 字节从 10.0.1.21 （10.0.1.21）： icmp_seq = 2 ttl = 64 时间 = 0.202，远女士
64 字节从 10.0.1.21 （10.0.1.21）： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
```

```
64 字节从 10.0.1.21 （10.0.1.21）： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

---10.0.1.21 ping--4 传输的信息包，4 收到的统计，0%数据包丢失，时间
3000ms rtt min，avg，马克斯，.= 0.202/0.217/0.263/0.030 女士
```

旧版联网 （nova 网络）

与旧版联网 （nova 网络） 的示例体系结构需要一个控制器节点和至少一个计算节点。控制器节点包含一个*管理网络*上的网络接口。计算节点包含一个网络接口上的管理网络和*外部网络*上的一个人.

请注意

网络接口名称不同分布的差异。传统上，接口使用"eth"后跟一个序列号。要涵盖所有的变化，本指南只是指的第一个接口作为具有的最低数量的接口和人数最多的接口作为第二个接口。

图 2.2。两个节点体系结构与旧版联网 （nova 网络）

除非您打算使用在此的示例体系结构提供的确切配置，您必须修改此过程以匹配您的环境中的网络。此外，每个节点必须通过 IP 地址名称解析其他节点。例如，该*控制器*的名称必须解析到 10.0.0.11 上的控制器节点的管理接口的 IP 地址。

警告

重新配置网络接口将中断网络连接。我们建议使用这些程序的本地终端会话。

控制器节点

要配置网络：

- 为管理接口配置的第一个接口：

  IP 地址：10.0.0.11

  网络掩码： 255.255.255.0 （或 24）

  默认网关： 10.0.0.1

若要配置名称解析：

1. 将该节点的主机名设置为*控制器*。

2. 编辑 /etc/主机文件来包含下列内容：

```
# 控制器
10.0.0.11 控制器

# compute1
10.0.0.31 计算 1
```

计算节点

要配置网络：

1.作为管理接口配置的第一个接口：

  IP 地址：10.0.0.31

  网络掩码： 255.255.255.0 （或 24）

  默认网关： 10.0.0.1

请注意

额外的计算节点应使用 10.0.0.32、 10.0.0.33，等等。

2. 外部接口使用特殊的配置没有给它分配 IP 地址。将第二个接口配置为外部接口：

  INTERFACE_NAME 替换为实际的接口名称。例如， *eth1* 或 *ens224*.

- 编辑
  /etc/sysconfig/network-scripts/ifcfg-*INTERFACE_NAME*
  文件包含下列内容：

  请不要更改 HWADDR 和 UUID 的键。

```
设备 =INTERFACE_NAME
类型 = 以太网
启动 ="yes"
BOOTPROTO ="none"
```

3．重新启动网络：

```
# 服务网络重启
```

若要配置名称解析：

1．将该节点的主机名设置为 compute1．

2．编辑 /etc/主机文件来包含下列内容：

```
# compute1
10.0.0.31 计算 1

# 控制器
10.0.0.11 控制器
```

验证连接性

我们建议您在继续审议之前验证网络连接到互联网和节点之间。

1．从*控制器*节点，**坪**在互联网上的网站：

```
# ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
从 174.143.194.225 的 64 个字节： icmp_seq = 1 ttl = 54 时间 = 18.3 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 2 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 3 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 4 ttl = 54 时间 = 17.4 ms

— — openstack.org 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间
3022ms rtt min/平均/最大/mdev = 17.489/17.715/18.346/0.364 女士
```

2．从*控制器*节点，**坪**管理界面的*计算*节点上：

```
# ping-c 4 compute1 PING compute1 (10.0.0.31)
56(84) 字节的数据。
从 compute1 的 64 字节 (10.0.0.31)： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
从 compute1 的 64 字节 (10.0.0.31)： icmp_seq = 2 ttl = 64 时间 = 0.202，远
女士
从 compute1 的 64 字节 (10.0.0.31)： icmp_seq = 3 ttl = 64 时间 = 0.203 ms 64
字节从 compute1 (10.0.0.31)： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

--- compute1 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间 3000ms
rtt min/平均/最大/mdev = 0.202/0.217/0.263/0.030 女士
```

3．从*计算*节点，**坪**在互联网上的网站：

```
# ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
从 174.143.194.225 的 64 个字节： icmp_seq = 1 ttl = 54 时间 = 18.3 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 2 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 3 ttl = 54 时间 = 17.5 ms
从 174.143.194.225 的 64 个字节： icmp_seq = 4 ttl = 54 时间 = 17.4 ms

--- openstack.org 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间
3022ms rtt min/平均/最大/mdev = 17.489/17.715/18.346/0.364 女士
```

4．从*计算*节点，**坪**管理接口*控制器*节点上：

```
# ping-c 4 控制器坪控制器 (10.0.0.11) 56(84) 字节的数
据。
64 字节从控制器 (10.0.0.11)： icmp_seq = 1 ttl = 64 时间 = 0.263 ms
64 字节从控制器 (10.0.0.11)： icmp_seq = 2 ttl = 64 时间 = 0.202，远女士
64 字节从控制器 (10.0.0.11)： icmp_seq = 3 ttl = 64 时间 = 0.203 毫秒
64 字节从控制器 (10.0.0.11)： icmp_seq = 4 ttl = 64 时间 = 0.202，远女士

--- 控制器坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间 3000ms rtt
min/平均/最大/mdev = 0.202/0.217/0.263/0.030 女士
```

网络时间协议 (NTP)

若要在多台机器同步服务，您必须安装 *NTP*。本指南中的示例配置控制器节点作为参考服务器和任何附加的节点从控制器节点设置他们的时间。

在运行 OpenStack 服务每个系统上安装的 ntp 软件包：

```
# yum 安装 ntp
```

设置您的控制器节点上的 NTP 服务器，以便它接收数据通过修改 ntp.conf 文件并重新启动服务：

```
# ntpd 服务启动
# chkconfig ntpd 上
```

它建议您配置额外的节点来同步他们的时间从控制器节点而不是在您的 LAN。要这样做，安装 ntp 后台程序作为上面，然后编辑 /etc/ntp.conf 和更改要使用的控制器节点作为互联网的时间源的服务器指令。

密码

各种 OpenStack 服务和所需的软件，如数据库和消息传递服务器必须是受密码保护。配置服务时，您使用这些密码，然后再次访问该服务。您必须配置该服务时选择一个密码，以后记得要访问它时，使用相同的密码。（可选），您可以生成随机密码与 pwgen 程序。或者，反复一次创建一个密码，使用此命令的输出：

```
$ openssl 兰德-十六进制 10
```

本指南使用公约 》 *SERVICE_PASS* 是密码以访问服务服务 *SERVICE_DBPASS* 是服务服务用于访问数据库的数据库密码。

密码你需要在本指南中定义的完整列表如下：

表 2.1。密码

密码名称	**描述**
数据库密码 （没有使用的变量）	数据库根密码
密码名称	**描述**
KEYSTONE_DBPASS	标识服务数据库密码
DEMO_PASS	用户试用的密码
ADMIN_PASS	用户管理员密码
GLANCE_DBPASS	图像服务数据库密码
GLANCE_PASS	图像服务用户那一眼的密码
NOVA_DBPASS	计算服务数据库密码
NOVA_PASS	计算服务用户新星的密码
DASH_DBPASS	数据库密码的仪表板
CINDER_DBPASS	数据块存储服务数据库密码

CINDER_PASS	数据块存储服务用户煤渣的密码
NEUTRON_DBPASS	网络服务数据库密码
NEUTRON_PASS	网络服务用户中子的密码
HEAT_DBPASS	业务流程服务数据库密码
HEAT_PASS	业务流程服务用户热的密码
CEILOMETER_DBPASS	遥测服务数据库密码
CEILOMETER_PASS	遥测服务用户测云的密码
TROVE_DBPASS	数据库密码的数据库服务
TROVE_PASS	数据库服务用户宝库的密码

数据库

大多数 OpenStack 服务需要一个数据库来存储信息。这些示例使用控制器节点运行 MySQL 数据库。你必须在控制器节点上安装 MySQL 数据库。你必须在访问 MySQL 的任何其他节点上安装 MySQL Python 库。

控制器安装程序

在控制器节点，安装 MySQL 客户端和服务器包和 Python 库。

```
# yum 安装 mysql mysql 服务器 MySQL python
```

MySQL 配置需要一些修改，以与 OpenStack 工作。

- 编辑 /etc/my.cnf 文件：

  a. 第 [mysqld] 部分中，将绑定地址键设置为要启用访问由其他节点通过管理网络的控制器节点的管理 IP 地址：

  ```
  [mysqld].....
  绑定地址 = 10.0.0.11
  ```

  b. 在 [mysqld] 部分，设置以下键，使 InnoDB，UTF-8 字符集和 UTF-8，默认情况下的排序规则：

  ```
  [mysqld] 默认-存储-引擎 =
  innodb_file_per_table 排
  序规则服务器 innodb =
  utf8_general_ci init-连接
  = '集名称 utf8' 字符-集-服
  务器 = utf8
  ```

启动 MySQL 数据库服务器，并将其设置为在系统启动时自动启动：

```
# 服务 mysqld 启动
# chkconfig mysqld 上
```

最后，您应该为您的 MySQL 数据库设置根密码。设置数据库和表 OpenStack 程序提示您输入此密码如果它被设置。

您必须删除第一次启动数据库时创建的匿名用户。否则，数据库连接问题发生时您按照本指南中的说明。要执行此操作，请使用 **mysql_secure_installation** 命令。请注意，如果 **mysql_secure_installation** 失败，您可能需要使用 **mysql_install_db** 第一次：

```
# mysql_install_db
# mysql_secure_installation
```

如果已经设置了根数据库密码，按 **enter 键**，当系统提示您输入密码。此命令给出了大量的选项让你获得数据库的安装。除非你有很好的理由，否则对所有提示**是的**回应。

节点安装程序

在控制器节点以外的所有节点上安装 MySQL Python 库：

```
# yum 安装 MySQL python
```

OpenStack 包

分布可能释放 OpenStack 包作为其分布或通过其他方法的一部分，因为 OpenStack 和分布的释放时间是相互独立。

本节介绍了配置机器要安装最新的 OpenStack 软件包后，您必须完成的配置。

本指南中的示例使用 OpenStack 包从 RDO 资料库。这些软件包的工作红色帽子企业 Linux 6，CentOS 和 Fedora 20 的兼容版本。

安装 yum-插件-优先事项插件。此程序包允许将分配到配置的软件存储库的优先次序。此功能使用 RDO 发布包：

```
# yum 安装 yum-插件-优先事项
```

若要启用的率失真优化的存储库，请下载并安装 rdo-释放-冰窖包：

```
# yum 安装
http://repos.fedorapeople.org/repos/openstack/openstackicehouse/rdo-release-icehouse-3.noarch.rpm
```

战术包包括 GPG 密钥包签署和存储库的信息。这只应安装在红帽企业 Linux 和 CentOS，不顶软呢帽。安装最新的座谈释放包 （见 http://download.fedoraproject.org/pub/epel/6/x86_64/repoview/座谈-release.html）。例如：

```
# yum 安装
http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8。noarch.rpm
```

Openstack utils 软件包包含使安装和配置更简单的实用程序。这些程序都用在本指南。安装 openstackutils。这将验证您可以访问的率失真优化存储库：

```
# yum 安装 openstack utils
```

警告

Openstack 配置程序 openstack utils 软件包中的使用 crudini 配置文件进行操作。然而，crudini 版本 0.3 不支持多值的选项。请参阅 https://bugs.launchpad.net/openstack-manuals/ + bug/1269271。作为一项工作在附近，您必须手动设置多值的任何选项或新的值将覆盖以前的值而不是创建一个新的选项。

Openstack selinux 包包括 OpenStack 在 RHEL 和 CentOS 上安装期间配置 SELinux 所需的策略文件。OpenStack 在 Fedora 上安装期间，此步骤不是必需的。安装 openstack selinux：

```
# yum 安装 openstack selinux
```

升级你的系统软件包：

```
# 百胜餐饮升级
```

如果在升级包括一个新的内核包，重新启动系统以确保新的内核正在运行：

```
# 重新启动
```

邮件服务器

OpenStack 使用一个*消息代理*来协调行动和服务之间的状态信息。消息代理服务通常在控制器节点上运行。OpenStack 支持包括 RabbitMQ，Qpid 和 ZeroMQ 的几个消息代理。然而，大多数发行版的包 OpenStack 支持一个特定的消息代理。本指南包含支持的每个分布的消息代理。如果您希望实现一个不同的消息代理，咨询与它关联的文档。

- RabbitMQ
- Qpid
- ZeroMQ

要安装的消息代理服务

- 红色帽子企业 Linux （RHEL），CentOS，科学 Linux Fedora 使用 Qpid。

```
# yum 安装 qpid-cpp-服务器
```

若要配置消息代理服务

- 为了简化安装您的测试环境，我们建议您禁用身份验证。

  编辑 /etc/qpidd.conf 文件，并更改以下项：

```
auth = 否
```

请注意

对于生产环境，您应该启用身份验证。在确保安全的消息代理的详细信息，请参阅文档.

> 如果你决定为您的测试环境启用身份验证，您必须在配置文件中的每个 OpenStack 使用服务，消息代理配置的 qpid_username 和 qpid_password 键。

要完成安装

- 启动消息代理服务并将其配置为在系统引导时启动：

```
# 服务 qpidd 开始
# chkconfig qpidd 上
```

恭喜你，你现在准备好安装 OpenStack 服务 ！

3。配置标识服务

表的内容

身份服务概念

*标识服务*执行以下功能：

- 用户管理。跟踪用户和他们的权限。
- *服务目录*。可用服务目录中的提供的 API 终结点.

为了理解标识服务，你必须理解以下概念：

用户

人、 系统或服务使用 OpenStack 云服务的人的数字表示。标识服务验证传入的请求由自称打出电话的用户。用户有一个登录名和可能被分配令牌来访问资源。用户可以直接分配给某个特定的租户，表现得好像他们载于该租客。

凭据

众所周知，只能被证明了他们是谁的用户的数据。在标识服务，例子是： 用户名称和密码，用户名和 API 密钥或身份服务所提供的身份验证令牌。

身份验证

确认用户的身份的行为。标识服务确认通过验证一组的由用户提供的凭据来验证传入的请求。

这些凭据最初是一个用户名称和密码或用户名称和 API 密钥。在回应这些凭据，身份服务给用户，用户在后续请求中提供问题的身份验证令牌。

令牌

用来访问资源的文本的任意位元。

每个标记有一个范围，描述了哪些资源都可以访问它。一个标记可在任何时候撤销和有效期为有限的持续时间。

虽然身份服务支持基于令牌的身份验证在此版本中，目的是让其在将来支持附加议定书。其目的是为它是一个集成服务，首先是并不渴望成为一个正式的身份存储和管理解决方案。

租客

习惯于组或隔离资源和/或标识对象的容器。根据服务运营商，租客可能映射到客户、 帐户、 组织或项目。

服务

OpenStack 服务，如计算 (Nova)、 对象存储 (Swift) 或图像服务 (眼)。提供一个或多个终结点，通过它用户可以访问的资源和执行的操作。

终结点

网络访问的地址，通常由一个 URL，从哪里你访问的服务描述。如果使用模板的一个扩展，你可以创建一个终结点模板，代表所有的耗材服务的可用跨区域的模板。

作用

假定用户的个性，使他们对执行一组特定的操作。一个角色包括一组权利和特权。承担这一角色的用户将继承这些权利和特权。

在标识服务，向用户发出一个令牌包含列表中该用户具有的角色。被称为该用户的服务确定他们如何诠释的角色的用户，并且对哪个操作或资源的每个角色授予访问权限集。

下面的关系图显示了标识服务流程：

安装标识服务

1. 控制器节点，以及 pythonkeystoneclient （这是一个依赖项） 上安装 OpenStack 标识服务：

```
# yum 安装 openstack 基石 python keystoneclient
```

2. 标识服务使用一个数据库来存储信息。在配置文件中指定数据库的位置。在本指南中，我们使用一个 MySQL 数据库上的控制器节点的用户名的重点。用一个合适的密码数据库用户替换 *KEYSTONE_DBPASS* .

```
# openstack config--集 /etc/keystone/keystone.conf \ 数据库
连接 mysql://keystone:KEYSTONE_DBPASS@控制器/ 重点
```

3. 使用你以前设置要以 root 身份登录的密码。创建一个重点数据库用户：

```
$ mysql-u 根-p mysql >创建数据库的
基本原理 ；
mysql >授予所有权限上 keystone.* 到 'keystone'@'localhost' \ 确定由
'KEYSTONE_DBPASS';
mysql >授予所有权限上 keystone.* 到 'keystone'@'%' \ 确定由
'KEYSTONE_DBPASS';mysql >退出
```

4. 在身份服务创建的数据库表：

```
# 苏-s /bin/sh-c"梯形失真-管理 db_sync"重点
```

5. 定义授权令牌使用作为标识服务和其他 OpenStack 服务之间共享的秘密。使用 **openssl** 来生成一个随机的令牌并将其存储在配置文件中：

```
# ADMIN_TOKEN = $（openssl 兰德-十六进制 10）
# 回声 $ADMIN_TOKEN # openstack config--默认设置
/etc/keystone/keystone.conf \ admin_token $ADMIN_TOKEN
```

6．默认情况下，重点使用 PKI 令牌。创建签名的密钥和证书，并限制对所生成的数据的访问：

```
# 重点管理 pki_setup – – 重点用户重点 – – 重点集团重点# chown-R 重点： 重点
/etc/keystone/ssl
# chmod-R o 可 /etc/keystone/ssl
```

7．启动标识服务，使它能够在系统引导时启动：

```
# 服务 openstack 重点启动
# chkconfig openstack-基本原理
```

8．默认情况下，标识服务已过期，将标记存储在数据库中无限期。虽然有可能用于审计在生产环境中，过期令牌的积累将会大大增加数据库的大小，可能会降低服务性能，特别是在资源有限的测试环境中。我们推荐配置使用 cron 每小时清除过期的令牌定期任务。

- 运行以下命令，以清除过期令牌每一小时，将输出记录到 /var/log/keystone/keystone-tokenflush.log：

```
# （crontab-l-u 重点 2 > & 1 | grep-q token_flush） | |\ 回声 '
@hourly /usr/bin/keystone-manage token_flush >/var/日志/重点/
梯形 tokenflush.log 2 > & 1' >> /var/spool/cron/keystone
```

定义用户、 商户、 和角色

你安装的身份服务后，设置*用户*、*租户*和*角色*对进行身份验证。这些都用来允许访问服务和*终结点*下，一节中描述。

通常情况下，你可以表明一个用户和密码的身份服务进行身份验证。在这一点上，但是，你有不创建任何用户，所以你必须使用在前面的步骤中创建的授权令牌，请参阅的部分称为"安装身份服务"[24] 为进一步的细节。你可以将这– – os 令牌选项传递给**重点**命令或将 OS_SERVICE_TOKEN 环境变量设置。设置 OS_SERVICE_TOKEN，以及 OS_SERVICE_ENDPOINT 来指定在哪里的身份服务正在运行。 ***ADMIN_TOKEN*** 替换您的授权令牌。

```
$ 出口 OS_SERVICE_TOKEN =ADMIN_TOKEN
$ 出口 OS_SERVICE_ENDPOINT = http://controller:35357/v2.0
```

创建管理用户

请按照这些步骤来创建管理用户、 角色和租客。OpenStack 云与行政互动，您将使用此帐户。

默认情况下，标识服务创建一个特殊的 _member_ 角色。OpenStack 仪表板会自动与此角色的用户授予访问权限。你们会给管理员角色此角色的管理员用户权限。

请注意

您创建的任何角色必须映射到每个 OpenStack 服务中包含的 policy.json 文件中指定的角色。对于大多数服务的默认策略文件授予的管理访问权限的管理员角色。

1. 创建管理员用户：

```
$ 重点用户创建 – – 名称 = admin – – 通过 =ADMIN_PASS – – 电子邮件
=ADMIN_EMAIL
```

ADMIN_PASS 替换为一个安全的密码，替换与帐户关联的电子邮件地址为 *ADMIN_EMAIL* 。

2. 创建的管理员角色：

```
$ 重点角色创建 – – 名称 = 管理员
```

3. 创建的管理员租客：

```
$ 重点租客 – – 创建 – – 名称 = admin – – 描述 ="Admin 租客"
```

4. 你必须现在联系管理员用户、 管理员角色和 admin 租客在一起使用的用户角色添加选项：

```
$ 重点用户-角色添加 – – 用户 = admin – – 租客 = admin – – 角色 = 管理员
```

5. 链接的管理员用户，_member_ 的作用和 admin 租客：

```
$ 重点用户-角色添加 – – 用户 = admin – – 角色 = _member_ – – 租客 = 管
理员
```

创建一个正常的用户

按照这些步骤来创建一个普通用户和租客，并将它们链接到特殊 _member_ 的作用。OpenStack 云与日常非行政互动，您将使用此帐户。您也可以重复此过程来创建额外的云用户使用不同的用户名和密码。在创建这些用户时，跳过的租客创建步骤。

1．创建演示用户：

```
$ 重点用户创建 – – 名称 = 演示--通过 =DEMO_PASS – – 电子邮件 =DEMO_EMAIL
```

DEMO_PASS 替换为一个安全的密码，替换与帐户关联的电子邮件地址为 *DEMO_EMAIL* 。

2．创建演示租客：

```
$ 梯形租客 – – 创建 – – 名称 = 演示--描述 ="演示租客"
```

请注意

添加额外的用户时，不重复此步骤。

3.链接的演示用户，_member_ 的作用和演示租户：

```
$ 重点用户-角色添加 – – 用户 = 演示--角色 = _member_ – – 租客 = 演示
```

创建服务租客

OpenStack 服务还需要用户名、租客、和作用来访问其他 OpenStack 服务。在基本安装中，OpenStack 服务通常共享单个租户命名服务。

在安装和配置每个服务，您将创建额外用户名和角色下这个租客。

- 创建服务租户：

```
$ 梯形租客 – – 创建 – – 名称 = 服务 – – 描述 ="服务租客"
```

定义服务和终结点的 API

所以标识服务可以跟踪安装了哪些 OpenStack 服务和在哪里它们位于网络上，您必须注册每个服务在 OpenStack 安装中。若要注册一个服务，请运行以下命令：

- **重点服务创建**。描述服务。
- **重点创建终结点**。将 *API 端点*与该服务相关联。

您还必须注册标识服务本身。OS_SERVICE_TOKEN 环境变量，作为以前，一套用于身份验证。

1.为标识服务创建一个服务条目：

```
$ 重点服务创建 – – 名称 = 基石 – – 类型 = 身份。
```

```
— — 描述 ="OpenStack 身份"
+-------------+----------------------------------+
|属性 |值 |
+-------------+----------------------------------+
|描述 |OpenStack 身份 |
|id |15c11a23667e427e91bc31335b45f4bd |
|名称 |重点 |
|类型 |身份 |
+-------------+----------------------------------+
```

服务 ID 随机生成的是不同于此处所示。

2.指定 API 端点标识服务通过使用返回的服务 id。当你指定终结点时，您为公共 API、 内部 API 和 admin API 提供的 Url。在本指南中，使用控制器主机名称。请注意身份服务使用不同的端口用于管理 API。

```
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk ' / 标识 / {打印 $2}'）。
— — publicurl = http: / /控制器： 5000 /v2.0 \
— — internalurl = http: / /控制器： 5000 /v2.0 \
— — adminurl = http: / /控制器： 35357/v2.0
+-------------+----------------------------------+
|属性 |值 |
+-------------+----------------------------------+
|adminurl |http://controller:35357/v2.0 |
|id |11f9c625a3b94a3f8e66bf4e5de2679f |
|internalurl |http://controller:5000/v2.0 |
|publicurl |http://controller:5000/v2.0 |
|区域 |regionOne |
|service_id |15c11a23667e427e91bc31335b45f4bd
|+-------------+----------------------------------+
```

请注意

您将需要创建额外的终结点添加到您的 OpenStack 环境每个服务。本指南与安装的每个服务相关联的部分包括的特定服务的终结点创建步骤。

验证标识服务的安装

1. 要验证身份服务是否安装并正确配置，请清除中的 OS_SERVICE_TOKEN 和 OS_SERVICE_ENDPOINT 的环境变量的值：

```
$ 未设置的 OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

这些变量，它被用来引导的管理用户和注册身份服务，不再需要。

2. 您现在可以使用基于名称的普通用户的身份验证。

通过使用管理员用户和密码你选择了为该用户请求的身份验证令牌：

```
$ 重点 – – 操作系统用户名 = 管理员 – – os 密码 =ADMIN_PASS \
– – os-auth-url = http://controller:35357/v2.0 令牌获取
```

在响应，您会收到一个令牌搭配您的用户 id。这将验证身份服务正在运行的预期的终结点和与预期的凭据建立了您的用户帐户。

3. 验证授权行为与预期相同。要这样做，要求租客授权：

```
$ 重点 – – 操作系统用户名 = 管理员 – – os 密码 =ADMIN_PASS \ – – os-租客-名称 = admin – – os-auth-url = http://controller:35357/v2.0 \ 令牌获取
```

在响应，您会收到一个令牌，其中包含您指定的租户的 ID。这将验证您的用户帐户具有明确定义的作用给指定的租客，租客存在如预期。

4. 你也可以设置您的操作系统 – –-* 在您的环境，简化了命令行使用过程中的变量。设置管理员 openrc.sh 文件的管理员凭据与管理员终结点：

```
出口 OS_USERNAME =
admin 出口
OS_PASSWORD
=ADMIN_PASS 出口
OS_TENANT_NAME = 管理
员
出口 OS_AUTH_URL =
http://controller:35
357/v2.0
```

5. 源要在环境变量中读取此文件：

```
$ 源管理-openrc.sh
```

6. 验证已正确配置了您的管理员 openrc.sh 文件。运行相同的命令没有 --os-* 参数：

```
$ 重点标记得到
```

该命令返回一个令牌和 ID 的指定的租客。这将验证您已正确配置环境变量。

7. 验证您的管理员帐户具有执行行政命令的授权：

```
$ 重点用户列表
+----------------------------------+-------+---------+----------
---------+
|id |名称 |启用 |电子邮件 |
+----------------------------------+-------+---------+----------
---------+
|afea5bde3be9413dbd60e479fddf9228 |管理员 |True |admin@example.com
|
|32aca1f9a47540c29d6988091f76c934 |演示 |True |demo@example.com |
+----------------------------------+-------+---------+----------
---------+
$ 重点用户-角色-列表 -- 用户管理员 -- 租客管理员
+----------------------------------+----------
+----------------------------------+----------------------------------+
|id |名称 |user_id
|tenant_id |
+----------------------------------+----------
+----------------------------------+----------------------------------
+ |9fe2ff9ee4384b1894a90878d3e92bab |_member_
|afea5bde3be9413dbd60e479fddf9228 |e519b772cb43474582fa303da62559e5 |
|5d3b60b66f1f438b80eaae41a77b5951 |管理员
|afea5bde3be9413dbd60e479fddf9228 |e519b772cb43474582fa303da62559e5
|+----------------------------------+----------
+----------------------------------+----------------------------------+
```

既然**重点用户列表**命令的输出中的 id 匹配在**重点用户-角色-列表**命令中，user_id 和 admin 角色为相关的租客，列出该用户，这将验证您的用户帐户具有管理员角色，匹配标识服务的 policy.json 文件中使用的作用。

请注意

只要你定义您的凭据和身份服务端点通过命令行或环境变量，您可以从任何一台机器运行 OpenStack 客户端的所有命令。有关详细信息，请参阅章 4"安装和配置客户端，OpenStack"[30].

4。安装和配置客户端，OpenStack

表的内容

以下各节包含有关与 OpenStack 客户一起工作的信息。回忆：在前一节中，您使用的**重点**客户端。

您必须安装客户端工具来完成剩下的安装。

所以，你有类似的经历给您的用户在您的桌面上，而不是在服务器上配置客户端。

概述

你可以使用 OpenStack 命令行客户端运行简单的命令，使 API 调用。从命令行或脚本来自动执行任务中，您可以运行这些命令。如果您提供 OpenStack 凭据，您可以在任何计算机上运行这些命令。

在内部，每个客户端命令运行 cURL 命令，嵌入 API 发出的请求。OpenStack Api 是使用 HTTP 协议，包括方法、 Uri、 媒体类型和响应代码的 RESTful Api。

这些开放源代码 Python 客户端运行在 Linux 或 Mac OS X 系统上，很容易学习和使用。每个 OpenStack 服务都有其自己的命令行客户端。在一些客户端命令，你可以指定**调试**参数，显示该命令的基础 API 请求。这是很好地熟悉 OpenStack API 调用。

下表列出了与它的包的名称和描述每个 OpenStack 服务的命令行客户端。

表 4.1。OpenStack 服务和客户端

服务	客户端	包	描述
数据块存储	**煤渣**	python cinderclient	创建和管理卷。

计算	**新星**	python novaclient	创建和管理图像、 实例和口味。
数据库服务	**宝库**	python troveclient	创建和管理数据库。
身份	**基石**	python keystoneclient	创建和管理用户、 租户、 角色、 端点和凭据。
图像服务	**一眼**	python glanceclient	创建和管理图像。
服务	**客户端**	**包**	**描述**
网络	**中子**	python neutronclient	为客人服务器配置网络。此客户端以前被称为**量子**.
对象存储	**斯威夫特**	python swiftclient	收集统计数据、 列表项、 更新元数据和上传，下载，并删除文件存储的对象存储服务。访问对象存储安装专案处理。
业务流程	**热**	python heatclient	启动从模板堆栈、查看运行堆栈包括事件和资源的详细信息和更新和删除堆栈。
遥测	**测云**	pythonceilometerclient	创建和跨 OpenStack 收集的测量。

OpenStack **常见**客户端是在发展中。

安装 OpenStack 命令行客户端

为每个 OpenStack 客户端安装的必备软件和 Python 包。

安装必备软件

下表列出的软件，您需要要运行的命令行客户端，并根据需要提供的安装说明。

表 4.2。必备软件

前提条件	**描述**
Python 2.6 或更高版本	目前，客户端不支持 Python 3。

setuptools 包	在 Mac OS X 上的默认安装的。 许多 Linux 发行版都提供使 setuptools 易于安装的软件包。搜索为 setuptools 包管理器找到安装包。如果你不能找到一个，下载 setuptools 包直接从 http://pypi.python.org/pypi/setuptools . 安装 setuptools Microsoft Windows 上的推荐的方式是遵循提供的文件 setuptools 网站上。另一个选项是使用非官方的二进制安装程序由 Christoph Gohlke 维护 (http://www.lfd.uci.edu/~gohlke/pythonlibs/#setuptools).
pip 软件包	在 Linux、 Mac OS X 上或微软的 Windows 系统，使用 pip 上安装客户端。它是易于使用，可确保您获得最新版本的客户端从 Python 包指数，并使您可以更新或删除的软件包在稍后。 安装 pip 通过为您的系统的软件包管理器： **苹果 Mac。** # **这么 pip** **Microsoft Windows。** 确保在 PATH 环境变量中定义的 C:\Python27\Scripts 目录，并使用从 setuptools 包**这么**命令： C:\ >**这么 pip** 另一个选项是使用非官方的二进制安装程序提供的 Christoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#pip). **Ubuntu 12.04/14.04.** 打包的版本使您能够使用 dpkg 或 apt-get 安装 python novaclient：
前提条件	**描述**
	# **apt-get 来安装 python novaclient** **Ubuntu 和 Debian。** # **apt-get 来安装 python pip** **红帽企业 Linux，CentOS 或 Fedora。** 打包的版本中可用 RDO 使您可以使用 yum 来安装客户端，或者你可以安装 pip 和使用它来管理客户端安装：

	`# yum 安装 python pip`
	
	openSUSE 12.2 和早些时候。 A 打包的版本中打开可用生成服务使您可以使用 rpm 或 zypper 来安装客户端，或者你可以安装 pip 和使用它来管理客户端安装：
	
	`# zypper 安装 python pip`
	
	12.3 和后来的 openSUSE。 打包的版本使您能够使用 rpm 或 zypper 来安装客户端。请参阅有一段叫"安装客户端"[32]

安装客户端

当继此节中的说明，与客户端进行安装，比如**新星**的小写名称替换*项目*。为每个客户端重复。下列值是有效的：

- 测云-遥测 API
- 煤渣-块存储 API 和扩展
- 眼-图像服务 API
- 热-业务流程的 API
- 梯形-身份服务 API 和扩展
- 中子-网络 API
- 新星-计算 API 和扩展
- 斯威夫特-对象存储 API
- 宝库-数据库服务 API

下面的示例显示的命令与 *pip* 安装新星客户端.

```
# pip 安装 python novaclient
```

安装与 pip

使用 pip Linux、 Mac OS X 上或 Microsoft Windows 的系统上安装 OpenStack 客户。它是易于使用，确保你获得最新版本的客户端从 Python 软件包的索引。此外，pip 使您可以更新或删除程序包。

通过使用下面的命令来分别安装每个客户端：

- 为 Mac OS X 或 Linux：

```
# pip 安装 python 项目客户端
```

- 为微软视窗：

```
C:\ >pip 安装 python 项目客户端
```

从包安装

RDO 和 openSUSE 有可以安装没有 pip 的客户端软件包。

在红帽企业 Linux，CentOS 或顶软呢帽，使用 **yum** 来安装客户端从可用在 RDO 的打包版本：

```
# yum 安装 python 项目客户端
```

对于 openSUSE，使用 rpm 或 zypper 来安装客户端从开放构建服务中可用的打包版本：

```
# zypper 安装 python-项目
```

升级或删除客户端

若要升级的客户端，请添加--到 **pip 安装**命令升级选项：

```
# pip 安装--升级 python 项目客户端
```

要删除一个客户端，请运行 **pip 卸载**命令：

```
# pip 卸载 python 项目客户端
```

设置环境变量使用 OpenStack RC 文件

要为 OpenStack 命令行客户端设置所需的环境变量，您必须创建环境文件称为 OpenStack rc 文件或 openrc.sh 文件。此特定项目环境文件包含所有 OpenStack 服务都使用的凭据。

当你源的文件时，为你当前的 shell 设置环境变量。变量使 OpenStack 客户端命令与在云中运行 OpenStack 服务进行通信。

请注意

定义环境变量使用的环境文件不是在 Microsoft Windows 上常见的做法。环境变量的定义通常是在系统属性对话框的**高级**选项卡。

创建和源 OpenStack RC 文件

1. 在一个文本编辑器，创建一个名为*项目*文件-openrc.sh 文件并添加以下身份验证信息：

   下面的示例显示一个名为 admin，哪里该操作系统用户名也是 admin，和标识主机位于控制器项目的信息。

```
出口 OS_USERNAME =
admin 出口
OS_PASSWORD
=ADMIN_PASS 出口
OS_TENANT_NAME = 管理
员
出口 OS_AUTH_URL =
http://controller:35
357/v2.0
```

2. 任何的 shell 中要运行 OpenStack 命令，源各自项目的*项目*openrc.sh 文件。在此示例中，您源代码*管理*项目的 adminopenrc.sh 文件：

```
$ 源管理-openrc.sh
```

重写环境变量的值

OpenStack 客户端命令运行时，您可以在年底**帮助**命令的输出的各种客户端使用所列出的选项重写一些环境变量设置。例如，您可以重写中的 OS_PASSWORD 设置

项目-openrc.sh 文件通过，如下所示在**基石**的命令，指定一个密码：

```
$ 重点 – – os 密码密码服务-列表
```

其中*的密码*是您的密码。

创建 openrc.sh 文件

正如在解释的部分称为"创建和源 OpenStack RC 文件"[33]使用从凭据有一段叫"定义用户、商户、和角色"[25]和创建以下*项目*-openrc.sh 文件： • admin openrc.sh 为管理用户

对于普通用户 openrc.sh 的演示：

```
出口 OS_USERNAME = 演示出口 OS_PASSWORD
=DEMO_PASS 出口 OS_TENANT_NAME = 演示出
口 OS_AUTH_URL = http: / /控制器:
35357/v2.0
```

5。配置图像服务

表的内容

OpenStack 图像服务使用户能够发现、 登记，并检索虚拟机映像。也被称为一眼项目，图像服务提供一个 *REST* API 使您可以将虚拟机图像元数据的查询和检索的实际图像。您可以存储虚拟机映像可通过各种从简单的文件系统位置到像 OpenStack 对象存储的对象存储系统中的图像服务。

重要

为简单起见，本指南配置要使用的文件的后端的图像服务。这意味着图像上传到图像服务都存储在承载该服务的同一系统上的目录中。默认情况下，此目录是 /var/lib/看一眼/图像 /。

在你开始之前，确保系统具有足够的空间可用在此目录中存储虚拟机镜像和快照。在降到最低，几个 g 应该是空间的可供使用的概念证明部署中的图像服务。若要查看其他的后端的要求，请参阅*配置参考*.

图像服务概述

图像服务包括以下组件：

• 眼-api。接受图像 API 调用图像发现、 检索和存储为。

- 眼-注册表。存储、 处理，并检索有关图像的元数据。元数据包括项的大小和类型。

安全说明

注册表是一个私人的内部服务只打算使用由图像服务本身。不要向用户公开。

- 的数据库。存储图像的元数据。您可以根据自己的喜好来选择您的数据库。大多数部署使用 MySQL 或 SQlite。

- 存储图像文件的存储库。图像服务支持各种各样的存储库 （包括普通的文件系统，对象存储、 拉多斯块设备，HTTP 和亚马逊 S3。一些类型的存储库支持只有只读的用法.

定期运行的进程的图像服务以支持缓存的数量。复制服务确保一致性和可用性通过群集。其他定期的进程包括审计、 更新和收割者。

中所示如图 1.1，"概念性架构"[2]，图像服务是整体的 IaaS 图片的中心。它接受 API 请求为图像或图像元数据从最终用户或计算组件，可以将其磁盘文件存储在对象存储服务.

安装图像服务

OpenStack 图像服务充当虚拟磁盘图像注册表。用户可以添加新的图像或从现有的服务器为即时的存储图像的生成快照。使用快照回来并且作为模板来启动新的服务器。对象存储中或其他地方，您可以存储注册的图像。例如，您可以在简单的文件系统或外部 web 服务器中存储图像。

请注意

此过程假定您设置向您的凭据，如中所述的适当的环境变量有一段叫"验证身份服务安装"[27].

1. 控制器节点上安装图像服务：

```
# yum 安装 openstack 眼 python glanceclient
```

2. 图像服务将有关图像的信息存储在数据库中。本指南中的示例使用 MySQL 数据库所使用的其他 OpenStack 服务。

   配置数据库的位置。图像服务提供的 glanceapi 和眼注册表的服务，每个都有它自己的配置文件。您必须更新整个这一节这两个配置文件。*GLANCE_DBPASS* 替换图像服务数据库密码。

```
# openstack config--集的 /etc/glance/glance-api.conf 数据库 \
连接 mysql://glance：GLANCE_DBPASS@控制器/glance # openstack
```

```
config--集的 /etc/glance/glance-registry.conf 数据库 \ 连接
mysql://glance：GLANCE_DBPASS@控制器/glance
```

3．图像将服务配置为使用消息代理：

```
# openstack config--默认设置
/etc/glance/glance-api.conf \ rpc_backend qpid #
openstack config--默认设置
/etc/glance/glance-api.conf \ qpid_hostname 控制器
```

4．使用你创建以 root 身份登录并创建一眼的数据库用户的密码：

```
$ mysql-u 根-p mysql >创建数据库
的一瞥 ；
mysql >授予所有权限上 glance.* 到 'glance'@'localhost' \ 确定由
'GLANCE_DBPASS';
mysql >授予所有权限上 glance.* 到 'glance'@'%' \
由标识 'GLANCE_DBPASS';
```

5．图像服务创建的数据库表：

```
# 苏-s /bin/sh-c"眼-管理 db_sync"看一眼
```

6．创建一眼用户图像服务可以使用身份服务进行身份验证。选择一个密码，并指定一眼用户的电子邮件地址。使用服务租客，并给用户管理员角色：

```
$ 重点用户创建 – – 名称 = 一眼 – – 通过 =GLANCE_PASS \
– – 电子邮件 =glance@example.com
$ 重点用户-角色添加 – – 用户 = 一眼 – – 租客 = 服务 – – 角色 = 管理员
```

7．图像将服务配置为使用标识服务进行身份验证。

运行下面的命令并将 *GLANCE_PASS* 替换为眼用户身份服务中所选的密码：

```
# openstack config--集的 /etc/glance/glance-api.conf
keystone_authtoken \ auth_uri http://控制器： 5000 # openstack
config--集的 /etc/glance/glance-api.conf keystone_authtoken \
auth_host 控制器# openstack config--设置的
/etc/glance/glance-api.conf keystone_authtoken \ auth_port 35357
# openstack config--集的 /etc/glance/glance-api.conf
keystone_authtoken \ auth_protocol http # openstack config--集的
/etc/glance/glance-api.conf keystone_authtoken \
admin_tenant_name 服务# openstack config--集的
/etc/glance/glance-api.conf keystone_authtoken \ admin_user 一眼
# openstack config--设置的 /etc/glance/glance-api.conf
keystone_authtoken \ admin_password GLANCE_PASS # openstack
config--集的 /etc/glance/glance-api.conf paste_部署 \ 风味重点#
```

```
openstack config--集的 /etc/glance/glance-registry.conf
keystone_authtoken \ auth_uri http://控制器： 5000 # openstack
config--设置的 /etc/glance/glance-registry.conf
keystone_authtoken \ auth_host 控制器# openstack config--集的
/etc/glance/glance-registry.conf keystone_authtoken \ auth_port
35357 # openstack config--集的 /etc/glance/glance-registry.conf
keystone_authtoken \ auth_protocol http # openstack config--集的
/etc/glance/glance-registry.conf keystone_authtoken \
admin_tenant_name 服务# openstack config--设置的
/etc/glance/glance-registry.conf keystone_authtoken \ admin_user
一眼# openstack config--集的 /etc/glance/glance-registry.conf
keystone_authtoken \ admin_password GLANCE_PASS# openstack
config--设置的 /etc/glance/glance-registry.conf paste_deploy \ 风
味基石
```

8．寄存器的身份形象服务服务，以便其他 OpenStack 服务可以找到它。注册服务并创建终结点：

```
$ 重点服务创建 — — 名称 = 一眼 — — 类型 = 图像。
```

```
— — 描述 ="OpenStack 图像服务"
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk '/ 图像 / {打印 $2}'）。
— — publicurl = http: / /控制器： 9292 \
— — internalurl = http: / /控制器： 9292 \
— — adminurl = http: / /控制器： 9292
```

9．启动眼 api 和眼注册表服务并将它们配置为在系统引导时启动：

```
# 服务 openstack-眼-api 的开始
# 服务 openstack-眼-注册表启动
# chkconfig openstack 眼 api 上
# chkconfig openstack 眼注册表上
```

验证图像服务的安装

若要测试图像服务的安装，请下载众所周知，OpenStack 与工作的至少一个虚拟机映像。例如，CirrOS 是一个小小的测试图像，通常用于测试 OpenStack 部署 （CirrOS 下载）。这个步行通过使用 64 位 CirrOS QCOW2 图像。

有关如何下载和生成的图像的详细信息，请参阅 *OpenStack 虚拟机映像指南*。有关如何管理映像的信息，请参阅 *OpenStack 用户指南*.

1．将映像下载到使用 **wget** 或**卷曲**专用目录：

```
$ mkdir/tmp/图像
$ cd /tmp/图像 / $ wget
http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64disk.im
g
```

2．将图像上传到图像服务：

```
$ 一眼图像创建 — — 名称 =IMAGELABEL — — 磁盘格式 =文件格式 \
— — 容器格式 =CONTAINERFORMAT — — 是公众 =ACCESSVALUE < IMAGEFILE
```

地点：

IMAGELABEL	任意的标签。该用户是指图像的名称。
文件格式	指定的图像文件的格式。有效格式包括 qcow2 原料，vhd vmdk，vdi、 iso、 aki，ari，、阿美族。

您可以验证使用**文件**命令的格式：

```
$ 文件 cirros-0.3.2-x86_64-disk.img
cirros-0.3.2-x86_64-disk.img: QEMU QCOW 图像 (v2)
41126400 字节
```

CONTAINERFORMAT 指定的容器格式。有效格式包括： 裸，ovf aki、 ari、 急性心肌梗死。

指定裸以指示的图像文件不是在一种文件格式，其中包含有关该虚拟机的元数据。

虽然这一领域目前需要，但不会实际使用

由任何 OpenStack 服务和对系统行为没有影响。因为不在任何地方使用的值，它是安全始终指定裸作为容器格式。

ACCESSVALUE 指定图像的访问：

• 真正的所有用户都可以查看和使用的图像。

• false-只有管理员可以查看和使用该图像。

IMAGEFILE 指定您的下载的图像文件的名称。

例如：

```
$ 源管理-openrc.sh
$ 一眼图像创建 — — 名称"cirros-0.3.2-x86_64"— — 磁盘格式 qcow2 \ — — 容器格式裸露 — — 是公共真实 — — 进展 <
cirros-0.3.2x86_64-disk.img+------------------+----------------------------------+
|属性 |值 |
+------------------+----------------------------------+
|校验和 |64d7c1cd2b6f60c92c14662941cb7913 |
|container_format |裸 |
|created_at |2014-04-08T18:59:18 |
|删除 |虚假 |
|deleted_at |没有人 |
|disk_format |qcow2 |
|id |acafc7c0-40aa-4026-9673-b879898e1fc2 |
|is_public |True |
|min_disk |0 |
|min_ram |0 |
|名称 |cirros-0.3.2-x86_64 |
```

```
|所有者 |efa984b0a914450e9a47788ad330699d |
|保护 |虚假 |
|大小 |13167616 |
|状态 |活动 |
|updated_at |2014-01-08T18:59:18 |
+------------------+--------------------------------------+
```

请注意

因为返回的图像 ID 动态生成的您的部署将生成一个不同的 ID 比在此示例中所示。

3.确认，图片上传和显示它的属性：

```
$ 一眼图像列表

+--------------------------------------+--------------------
+-------------+-----------------+----------+--------+
|ID |名称 |磁盘格式
|容器格式 |大小 |状态 |
+--------------------------------------+--------------------
+-------------+-----------------+----------+--------+
|acafc7c0-40aa-4026-9673-b879898e1fc2 |cirros-0.3.2-x86_64 |qcow2
|裸 |13167616 |活动 |
+--------------------------------------+--------------------
+-------------+-----------------+----------+--------+
```

4.你现在可以删除的本地下载的图像，因为它是存储并可通过图像服务。

```
$ rm-r/tmp/图像
```

另外上，传到图像服务可以完成而无需使用本地磁盘空间来存储文件，通过使用 — — *副本从*参数。

例如：

```
$ 一眼图像创建 — — 名称 ="cirros-0.3.2-x86_64"— — 磁盘格式 = qcow2。
— — 容器格式 = 裸 — — 是公众 = true \ — — 副本从
http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64disk.i
mg+------------------+--------------------------------------+
|属性 |值 |
+------------------+--------------------------------------+
|校验和 |64d7c1cd2b6f60c92c14662941cb7913 |
```

```
|container_format |裸 |
|created_at |2014-04-08T06:13:18 |
|删除 |虚假 |
|disk_format |qcow2 |
|id |3cce1e32-0971-4958-9719-1f92064d4f54 |
|is_public |True |
|min_disk |0 |
|min_ram |0 |
|名称 |cirros-0.3.2-x86_64 |
|所有者 |efa984b0a914450e9a47788ad330699d |
|保护 |虚假 |
|大小 |13167616 |
|状态 |活动 |
|updated_at |2014-04-08T06:13:20 |
+------------------+--------------------------------------+
```

6。配置计算服务

表的内容

计算服务

计算服务是一个云计算织物控制器，它是 IaaS 系统的主要组成部分。使用它来承载和管理云计算系统。在 Python 中实现的主要模块。

计算与身份验证、 对于图像，图像服务和为用户和管理界面的仪表板的身份服务交互。对图像的访问是有限的按项目和用户 ；配额是有限的每个项目 （例如，实例数目）。计算服务在标准硬件上，水平缩放和下载启动实例作为所需的图像。

计算服务由以下功能区域和其底层组件组成：

API

- nova api 服务。接受并响应最终用户计算 API 调用。支持 OpenStack 计算 API、 亚马逊 EC2 API 和特权用户一个特殊的管理 API 来执行管理操作。此外，启动大部分业务流程活动，如运行一个实例，并强制执行一些策略。

- 新星-api-元数据服务。接受来自实例元数据请求。当你运行在多主机模式与 nova 网络安装时一般只使用 novaapi 元数据服务。有关详细信息，请参阅元数据服务*云管理员指南 》*中。

  在 Debian 系统上，它列入 nova api 的一揽子计划，并可以通过 debconf 选定。

计算核心

- 新星计算过程。一个工人守护进程创建和终止虚拟机实例通过虚拟机管理程序的 Api。例如，XenAPI 的 XenServer/XCP，libvirt 为 KVM 或 QEMU，VMwareAPI 为 VMware，等等。其中它这样做的过程是相当复杂，但基本是简单： 接受从队列的操作和执行一系列的系统命令，像发射一个 KVM 实例，他们进行同时更新数据库中的状态。

- 新星调度程序进程。从概念上讲最简单代码段在计算中。需要一个虚拟机实例请求从队列，并确定它应该在哪个计算服务器主机上运行。
- 新星导体模块。介导新星计算和数据库之间的交互。旨在消除对 novacompute 所作的云数据库的直接访问。Nova 导体模块横向缩放。然而，不部署它在新星计算运行任何节点上。更多的信息，请参阅新新星服务： 新星导体.

虚拟机的联网

- 新星网络工作者守护进程。类似于新星-计算，它接受网络从队列的任务和执行操作的网络，例如设置桥接接口或改变 iptables 规则的任务。此功能将被迁移到 OpenStack 联网，这是一个单独的 OpenStack 服务。

- nova dhcpbridge 脚本。跟踪 IP 地址租约和通过使用 dnsmasq dhcp 脚本工具将它们记录在数据库中。此功能是正在迁移到 OpenStack 联网。OpenStack 联网提供了不同的脚本。

控制台界面

- 新星 consoleauth 守护进程。授权用户控制台代理提供的标记。请参阅 nova novncproxy 和新星 xvpnvcproxy。该服务必须运行，以便控制台代理工作。任一类型的很多代理可以针对单个 novaconsoleauth 服务在群集配置中运行。有关信息，请参阅关于 novaconsoleauth.

- 新星 novncproxy 守护进程。提供用于访问正在运行的实例通过 VNC 连接的代理服务器。支持客户端基于浏览器的 novnc.

- 新星 xvpnvncproxy 守护进程。通过 VNC 连接访问正在运行的实例的代理。支持 Java 的客户端，专为 OpenStack。

- 新星 cert 守护进程。管理 x 509 证书。

图像管理 （EC2 的情况）

- 新星 objectstore 守护进程。用于注册的图像的图像服务提供一个 S3 接口。主要用于必须支持 euca2ools 的安装。Euca2ools 工具跟在 *S3 的语言*中，nova objectstore 和新星 objectstore S3 请求转化为图像服务请求。

- euca2ools 客户端。一套用于管理云资源的命令行解释器命令。虽然不是 OpenStack 模块，您可以配置 nova api 来支持此 EC2 接口。有关详细信息，请参阅桉树 3.4 文件。

命令行客户端和其他接口

- 新星客户端。使用户能够作为租户管理员或最终用户提交命令。

- 新星-管理客户端。启用云管理员提交命令。

其他组件

- 队列。中心位置的守护进程之间传递消息。通常用 RabbitMQ，实现，但可以是任何 AMQP 消息队列，如 Apacheqpid 或零 MQ.

- SQL 数据库。存储云基础架构的大多数生成时间和运行时状态。包括可供使用，在使用、可用的网络和项目中的实例的实例类型。从理论上讲，OpenStack 计算可以支持 SQLAlchemy 支持，任何数据库，但广泛使用的唯一数据库很 SQLite3 数据库 （只适合测试和开发的工作）、 MySQL 和 PostgreSQL。

计算服务与交互，其他 OpenStack 服务： 身份服务进行身份验证，对于图像，图像服务和为一个 web 界面的 OpenStack 仪表板。

安装计算控制器服务

计算是服务，使您能够启动虚拟机实例的集合。您可以配置这些服务，以在单独的节点或相同的节点上运行。在本指南中，大多数的服务节点上运行控制器和专用的计算节点上运行的服务的启动虚拟机。此部分显示你如何安装和配置上的控制器节点的这些服务。

1. 安装所需的控制器节点的计算软件包。

```
# yum 安装 openstack-新星-api openstack 新星证书 openstack
novaconductor \ openstack-新星-控制台 openstack-新星
-novncproxy openstack novascheduler \ python novaclient
```

2. 计算在数据库中存储的信息。在本指南中，我们使用一个 MySQL 数据库上的控制器节点。使用的数据库的位置和凭据配置计算。将在稍后的步骤中创建的数据库的密码替换 *NOVA_DBPASS* .

```
# openstack config--集 /etc/nova/nova.conf \
数据库连接 mysql://nova:NOVA_DBPASS@控制器/nova
```

3. 设置这些配置项来配置计算使用 Qpid 消息代理：

```
# openstack config--集 /etc/nova/nova.conf \
默认 rpc_backend qpid # openstack config--设置
/etc/nova/nova.conf 默认 qpid_hostname 控制器
```

4. 设置的 my_ip，vncserver_listen 和 vncserver_proxyclient_address 配置选项的管理接口的控制器节点的 IP 地址：

```
# openstack config--集的 /etc/nova/nova.conf 默认 my_ip 10.0.0.11 #
openstack config--集的 /etc/nova/nova.conf 默认 vncserver_listen
10.
0.0.11
# openstack config--集的 /etc/nova/nova.conf 默认
vncserver_proxyclient_address 10.0.0.11
```

5. 使用密码您之前创建，以 root 身份登录。创建一个新数据库用户：

```
$ mysql-u 根-p mysql >新星创建
数据库 ；
mysql >授予所有权限上 nova.* 到 'nova'@'localhost' \ 确定由 'NOVA_DBPASS';
mysql >授予所有权限上 nova.* 到 'nova'@'%' \
由标识 'NOVA_DBPASS';
```

6．创建的计算服务表：

```
# 苏-s /bin/sh-c"新星-db 同步管理"新星
```

7．创建计算的新星用户使用身份服务进行身份验证。使用服务租客，并给用户管理员角色：

```
$ 重点用户创建 — — 名称 = 新星 — — 通过 =NOVA_PASS — — 电子邮件 =nova@example。 com $重点用户-角色添加 — — 用户 = 新星 — — 租客 = 服务 — — 角色 = 管理员
```

8．配置计算与运行在控制器上的身份服务使用这些凭据。*NOVA_PASS* 替换为你计算密码。

```
# openstack config--集的 /etc/nova/nova.conf 默认 auth_strategy 梯形# openstack config--设置的 /etc/nova/nova.conf keystone_authtoken auth_uri http://控制器： 5000 # openstack config--设置 /etc/nova/nova.conf keystone_authtoken auth_host 控制器# openstack config--设置的 /etc/nova/nova.conf keystone_authtoken auth_protocol http # openstack config--设置 /etc/nova/nova.conf keystone_authtoken auth_port
35357 # openstack config--设置的 /etc/nova/nova.conf keystone_authtoken admin_user nova # openstack config--/etc/nova/nova.conf keystone_authtoken admin_tenant_name 服务设置为# openstack config--设置 /etc/nova/nova.conf keystone_authtoken admin_password NOVA_PASS
```

9．你必须计算注册与标识服务，以便其他 OpenStack 服务可以找到它。注册服务和指定的终结点：

```
$ 重点服务创建 — — 名称 = 新星 — — 类型 = 计算。
— — 描述 ="OpenStack 计算"
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk '/ 计算 / {打印 $2}'）。
— — publicurl = http: / /控制器： 8774 /v2/%\ (tenant_id\) s。
— — internalurl = http: / /控制器： 8774 /v2/%\ (tenant_id\) s。
— — adminurl = http: / /控制器： 8774 /v2/%\ (tenant_id\) s
```

10．开始计算服务，并将它们配置为在系统引导时启动：

```
# 服务 openstack-新星-api 的开始
# 服务 openstack 新星证书开始
# 服务 openstack-新星-consoleauth 开始
# 服务 openstack-新星-调度程序的开始
```

```
# 服务 openstack-新星-导体的开始
# 服务 openstack-新星-novncproxy 开始
# chkconfig openstack nova api 上
# chkconfig openstack 新星证书上
# chkconfig openstack nova consoleauth 上
# chkconfig openstack 新星-调度器上
# chkconfig openstack 新星导体上
# chkconfig openstack nova novncproxy 上
```

11.为了验证您的配置，请列出可用的图像：

```
$ nova 图像列表
+--------------------------------------+---------------------+--
------
+--------+
|ID |名称 |状态 |
服务器 |
+--------------------------------------+---------------------+--
------
+--------+
|acafc7c0-40aa-4026-9673-b879898e1fc2 |cirros-0.3.2-x86_64 |活动 |
     |
+--------------------------------------+---------------------+--
------
+--------+
```

配置一个计算节点

你配置上的控制器节点的计算服务后，您必须作为计算节点配置另一个系统。计算节点接收来自控制器节点和主机的虚拟机实例的请求。您可以在单个节点上，运行的所有服务，但本指南中的示例使用独立的系统。这使得它易于规模水平通过添加额外的计算节点，这一节中的说明。

计算服务依赖于虚拟机监控程序运行虚拟机实例。OpenStack 可以使用各种虚拟机监控程序，但本指南使用 KVM。

1．安装计算软件包：

```
# yum 安装 openstack-新星-计算
```

2．编辑的 /etc/nova/nova.conf 配置文件：

```
# openstack config--集的 /etc/nova/nova.conf 数据库连接 mysql： / /
新星：NOVA_DBPASS@controller/nova # openstack config--集的
/etc/nova/nova.conf 默认 auth_strategy 梯形# openstack config--设
置的 /etc/nova/nova.conf keystone_authtoken auth_uri http://控制
器： 5000 # openstack config--设置 /etc/nova/nova.conf
keystone_authtoken auth_host 控制器# openstack config--设置的
/etc/nova/nova.conf keystone_authtoken auth_protocol http #
openstack config--设置 /etc/nova/nova.conf keystone_authtoken
auth_port
35357 # openstack config--设置的 /etc/nova/nova.conf
keystone_authtoken admin_user nova # openstack
config--/etc/nova/nova.conf keystone_authtoken
admin_tenant_name 服务设置为# openstack config--设置
/etc/nova/nova.conf keystone_authtoken admin_password NOVA_PASS
```

3．计算将服务配置为使用 Qpid 消息代理通过设置这些配置键：

```
# openstack config--集 /etc/nova/nova.conf \
默认 rpc_backend qpid # openstack config--设置
/etc/nova/nova.conf 默认 qpid_hostname 控制器
```

4．配置计算来提供对实例的远程控制台访问。

```
# openstack config--集的 /etc/nova/nova.conf 默认 my_ip 10.0.0.31
# 设置的 /etc/nova/nova.conf 默认 vnc_enabled 真 openstack-config-- #
openstack config--集的 /etc/nova/nova.conf 默认 vncserver_listen 0.0。
0.0
# openstack config--设置 /etc/nova/nova.conf 默认
vncserver_proxyclient_address 10.0.0.31 # openstack
config--集 /etc/nova/nova.conf \
默认 novncproxy_base_url http://控制器： 6080 /vnc_auto.html
```

5．指定的主机的运行图像服务。

```
# openstack config--集的 /etc/nova/nova.conf 默认
glance_host 控制器
```

6．您必须确定您的系统的处理器和/或虚拟机管理程序是否支持硬件加速的虚拟机。

运行以下命令：

```
$ egrep-c '(vmx|svm)' / proc/cpuinfo
```

如果此命令返回的值*一个或更多*，您的系统支持硬件加速，通常不需要额外配置。

如果此命令返回值为*零*，则您的系统不支持硬件加速，您必须配置 libvirt 使用 QEMU 而不是 KVM。

- 运行下面的命令：

```
# openstack config--集的 /etc/nova/nova.conf libvirt virt_type qemu
```

7.开始计算服务并将其配置为在系统引导时启动：

```
# 服务 libvirtd 开始
# 服务 messagebus 开始
# chkconfig libvirtd 上
# chkconfig messagebus 上
# 服务 openstack-新星-计算开始
# chkconfig openstack 新星计算上
```

7。添加网络服务

表的内容

配置网络中 OpenStack 可以是一个令人困惑的经验。本指南提供 OpenStack 联网 （中子） 和遗留的网络 （新星网络） 服务的分步说明。如果您不确定使用哪一个，我们建议您尝试 OpenStack 网络因为它提供了相当多的功能和灵活性，包括*插件程序*的各种新兴产品支持*虚拟网络*。请参阅 *OpenStack 云管理员指南*的详细信息的网络一章。

OpenStack 联网 （中子）

网络的概念

OpenStack 联网 （中子） 为虚拟管理的所有网络方面
网络基础设施 (VNI) 和访问层方面的物理网络基础设施 (PNI) OpenStack 环境中。OpenStack 联网允许租户要创建先进虚拟网络拓扑结构包括*防火墙*、*负载平衡器*和*虚拟专用网络 (Vpn)*等服务.

网络提供以下的对象抽象概念： 网络、 子网和路由器。每个人都有模仿其物理对应的功能： 网络包含子网和路由器路由不同的子网和网络之间的通信。

任何给定的网络设置有至少一个外部网络。这一网络，不同于其他网络，不是仅仅是一个几乎已定义的网络。相反，它表示成一片外 OpenStack 安装可以访问外部网络的视图。网络外部网络上的 IP 地址都可以访问物理上在外部网络上的任何人。因为此网络只是代表一片外部网络，此网络上禁用 DHCP。

除了外部网络，任何网络设置有一个或更多的内部网络。这些软件定义网络直接连接到虚拟机。只有在任何给定的内部网络，Vm 或那些在子网通过类似路由器的接口连接，可以访问虚拟机直接连接到该网络。

为外部网络访问 Vm，反之亦然，需要在网络之间的路由器。每个路由器已连接到网络的一个网关和多个接口连接到子网。类似于一个物理的路由器，子网可以

访问上都连接到同一路由器上，其他子网的机器，机器可以访问外部网络通过路由器的网关。

此外，您可以分配到端口的内部网络外部网络上的 IP 地址。每当东西连接到一个子网，该连接被称为一个端口。
您可以将外部网络的 IP 地址与端口到虚拟机相关联。这种方式，外部网络上的实体可以访问虚拟机。

网络还支持*安全组*。安全组使管理员能够在群体中定义防火墙规则。VM 可以属于一个或多个安全组，并联网应用中这些安全组来阻止或允许该 vm 的端口、端口范围或通信类型的规则。

每个插件网络使用有它自己的概念。虽然对经营不重要
网络，了解这些概念可以帮助您设置网络。所有
网络安装使用核心插件和插件的安全组 （或只是没有 Op 安全组插件）。此外，防火墙作为服务 （FWaaS） 和负载的平衡-作为-服务 （LBaaS） 的插件都可用。

模块化的图层 2 （ML2） 插件

配置控制器节点

系统必备组件

你配置 OpenStack 联网 （中子） 之前，你必须创建一个数据库，包括用户和服务身份服务凭据。

1. 连接到数据库，请以 root 用户身份创建中子数据库中，并授予给它相应的访问权限：

   用一个合适的密码替换 *NEUTRON_DBPASS* 。

   ```
   $ mysql -u 根 -p mysql >创建数据库中
   子 ；
   mysql >授予所有权限上 neutron.* 到 'neutron'@'localhost' \ 确定由
   'NEUTRON_DBPASS';
   mysql >授予所有权限上 neutron.* 到 'neutron'@'%' \
   由标识 'NEUTRON_DBPASS';
   ```

2. 创建身份服务凭据用于网络：

   a. 创建子用户：

替换一个合适的密码和 *neutron@example.com* 与一个合适的电子邮件地址为 *NEUTRON_PASS* 。

```
$ 重点用户创建 – – 中子的名字 – – 通过 NEUTRON_PASS -电子邮件 neutron@example.com
```

b. 中子用户链接到服务租客和管理员角色：

```
$ 用户-角色添加 – – 重点用户中子 – – 租客服务--角色管理
```

c. 创建的子服务：

```
$ 重点服务创建 – – 中子的名字 – – 类型网络--描述 "OpenStack 联网"
```

d. 创建服务终结点：

```
$ 重点创建终结点。
– – 服务 id $（重点服务列表 | awk '/ 网络 / {打印 $2}'）
 \
– – publicurl http://控制器： 9696 \
– – adminurl http://控制器： 9696 \
– – internalurl http://控制器： 9696
```

要安装的网络组件

- `# yum 安装 openstack-中子-ml2 python neutronclient openstack-中子`若要

配置的网络服务器组件

网络服务器组件配置包括数据库，身份验证机制，消息代理，拓扑变化通告程序，和插件。

1. 配置网络要使用的数据库：

用一个合适的密码替换 *NEUTRON_DBPASS* 。

```
# openstack config--集的 /etc/neutron/neutron.conf 数据库连接 \
mysql://neutron：NEUTRON_DBPASS@控制器/neutron
```

2. 配置网络要使用的身份验证的身份服务：

NEUTRON_PASS 替换为中标识服务的中子用户选择的密码。

```
# openstack config--集 /etc/neutron/neutron.conf 默认 \
auth_strategy 重点# openstack config--集的
```

```
/etc/neutron/neutron.conf keystone_authtoken \ auth_uri http://
控制器： 5000 # openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ auth_host 控制器# openstack config--设置的
/etc/neutron/neutron.conf keystone_authtoken \ auth_protocol
http # openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ auth_port 35357 # openstack config--集的
/etc/neutron/neutron.conf keystone_authtoken \
admin_tenant_name 服务# openstack config--设置的
/etc/neutron/neutron.conf keystone_authtoken \ admin_user 中子
# openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ admin_密码 NEUTRON_PASS
```

3．配置联网使用的消息代理：

```
# openstack config--默认设置
/etc/neutron/neutron.conf \ rpc_backend
neutron.openstack.common.rpc.impl_qpid # openstack
config--默认设置 /etc/neutron/neutron.conf \
qpid_hostname 控制器
```

4．配置网络，关于网络拓扑更改通知计算：

```
# openstack config--集 /etc/neutron/neutron.conf 默认 \
notify_nova_on_port_status_changes True # openstack config--
默认设置 /etc/neutron/neutron.conf \
notify_nova_on_port_data_changes True
# openstack config--集 /etc/neutron/neutron.conf 默认。
nova_url http://控制器： 8774 /v 2 # openstack config--集
/etc/neutron/neutron.conf 默认 \ nova_admin_username nova #
openstack config--集 /etc/neutron/neutron.conf 默认 \
nova_admin_tenant_id $(基石租客列表 | awk '/ 服务 / {打印 $2}') #
openstack config--集 /etc/neutron/neutron.conf 默认 \
nova_admin_password NOVA_PASS # openstack config--集
/etc/neutron/neutron.conf 默认 \ nova_admin_auth_url http://控制
器： 35357/v2.0
```

5．配置联网使用的模块化图层 2 （ML2） 插件和相关服务：

```
# openstack config--默认设置
/etc/neutron/neutron.conf \ core_plugin ml2 #
openstack config--默认设置
/etc/neutron/neutron.conf \ service_plugins 路由器
```

请注意

我们建议添加详细真实的 [默认] 节中 = / etc/neutron/neutron.conf 来帮助解决。

若要配置模块化图层 2 （ML2） 插件

ML2 插件使用开放 vSwitch （OVS） 机制 （代理） 建立虚拟的网络架构为实例。然而，控制器节点不需要 OVS 代理或服务因为它无法处理实例的网络流量。

- 运行下面的命令：

```
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \ type_drivers gre
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
tenant_network_types gre # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
mechanism_drivers openvswitch # openstack config--设置的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre \
tunnel_id_ranges 1: 1000 # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini 基本 \
firewall_driver neutron.agent.linux.iptables_firewall。
OVSHybridIptablesFirewallDriver # openstack config--基本设
置的 /etc/neutron/plugins/ml2/ml2_conf.ini \
enable_security_group True
```

若要配置计算要使用网络

默认情况下，大多数发行版配置计算使用旧版网络。您必须重新配置计算机来管理网络通过联网。

- 运行下面的命令：

NEUTRON_PASS 替换为中标识服务的中子用户选择的密码。

```
# openstack config--集 /etc/nova/nova.conf 默认 \ network_api_class
nova.network.neutronv2.api.API # openstack config--集
/etc/nova/nova.conf 默认 \ neutron_url http://控制器： 9696 #
openstack config--集 /etc/nova/nova.conf 默认 \
neutron_auth_strategy 重点# openstack config--集
/etc/nova/nova.conf 默认 \ neutron_admin_tenant_name 服务#
openstack config--集 /etc/nova/nova.conf 默认 \
neutron_admin_username 中子# openstack config--集
/etc/nova/nova.conf 默认 \ neutron_admin_password NEUTRON_PASS #
```

```
openstack config--集 /etc/nova/nova.conf 默认 \
neutron_admin_auth_url http://控制器: 35357/v2.0 # openstack
config--集 /etc/nova/nova.conf 默认 \ linuxnet_interface_driver
nova.network.linux_net.LinuxOVSInterfaceDriver # openstack
config--默认设置 /etc/nova/nova.conf \ firewall_driver
nova.virt.firewall.NoopFirewallDriver # openstack config--默认设置
/etc/nova/nova.conf \ security_group_api 中子
```

请注意

默认情况下，计算使用内部防火墙服务。由于网络包括防火墙服务，您必须通过使用 nova.virt.firewall.NoopFirewallDriver 防火墙驱动程序禁用计算防火墙服务。

要完成安装

1. 网络服务初始化脚本期望指向与你选择的插件关联的配置文件的符号链接 /etc/中子/plugin.ini.使用 ML2，例如，符号链接必须指向/等/中子/插件/ml2/ml2_conf.ini。如果这个符号链接不存在，请创建它使用下面的命令：

```
# ln-s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

2. 重新启动计算服务：

```
# 服务 openstack-新星-api 重新启动
# 服务 openstack-新星-调度程序重新启动
# 服务 openstack-新星-导体重新启动
```

3. 启动网络服务并将其配置为在系统引导时启动：

```
# 在服务中子服务器启动
# chkconfig 中子服务器上
```

配置网络节点

系统必备组件

您在配置 OpenStack 网络之前，您必须启用某些内核网络功能。
1. 编辑 /etc/sysctl.conf，包含下列内容：

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2．实施的改变：

```
# sysctl-p
```

要安装的网络组件

•

若要配置的网络常见组件

```
# yum 安装 openstack 中子 openstack-中子-ml2 \ openstack-中子-openvswitch
```

联网通用组件配置包括身份验证机制、 消息代理和插件。

1．配置网络要使用的身份验证的身份服务：

NEUTRON_PASS 替换为中标识服务的中子用户选择的密码。

```
# openstack config--集 /etc/neutron/neutron.conf 默认 \
auth_strategy 重点# openstack config--集的
/etc/neutron/neutron.conf keystone_authtoken \ auth_uri http://
控制器： 5000 # openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ auth_host 控制器# openstack config--设置的
/etc/neutron/neutron.conf keystone_authtoken \ auth_protocol
http # openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ auth_port 35357 # openstack config--集的
/etc/neutron/neutron.conf keystone_authtoken \
admin_tenant_name 服务# openstack config--设置的
/etc/neutron/neutron.conf keystone_authtoken \ admin_user 中子
# openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ admin_密码 NEUTRON_PASS
```

2．配置联网使用的消息代理：

```
# openstack config--默认设置
/etc/neutron/neutron.conf \ rpc_backend
neutron.openstack.common.rpc.impl_qpid # openstack
config--默认设置 /etc/neutron/neutron.conf \
qpid_hostname 控制器
```

3．配置联网使用的模块化图层 2 （ML2） 插件和相关服务：

```
# openstack config--默认设置
/etc/neutron/neutron.conf \ core_plugin ml2 #
```

```
openstack config--默认设置
/etc/neutron/neutron.conf \ service_plugins 路由器
```

请注意

我们建议添加详细真实的 [默认] 节中 = / etc/neutron/neutron.conf 来帮助解决。

若要配置层-3 （L3） 代理

*图层-3 （L3） 代理*提供路由服务例如虚拟网络。

- 运行以下命令：

```
# openstack config--默认设置 /etc/neutron/l3_agent.ini \
interface_driver
neutron.agent.linux.interface.OVSInterfaceDriver #
openstack config--默认设置 /etc/neutron/l3_agent.ini \
use_namespaces True
```

请注意

我们建议添加详细真实的 [默认] 节中 = / etc/neutron/l3_agent.ini 来帮助解决。

要配置的 DHCP 代理

*DHCP 代理*提供例如虚拟网络的 *DHCP* 服务。

- 运行以下命令：

```
# openstack config--集 /etc/neutron/dhcp_agent.ini 默认 \
interface_driver
neutron.agent.linux.interface.OVSInterfaceDriver #
openstack config--默认设置 /etc/neutron/dhcp_agent.ini \
dhcp_driver neutron.agent.linux.dhcp.Dnsmasq # openstack
config--集 /etc/neutron/dhcp_agent.ini 默认 \
use_namespaces True
```

请注意

我们建议添加详细真实的 [默认] 节中 = / etc/neutron/dhcp_agent.ini 来帮助解决。

若要配置元数据代理

*元数据代理*提供配置信息，如对实例的远程访问的凭据。

1.运行下面的命令：

NEUTRON_PASS 替换为中标识服务的中子用户选择的密码。替换一个合适的秘密 *METADATA_SECRET* 元数据代理。

```
# openstack config--默认设置
/etc/neutron/metadata_agent.ini \ auth_url http://控制器:
5000 /v 2.0 # openstack config--默认设置
/etc/neutron/metadata_agent.ini \ auth_region regionOne
# openstack config--集 /etc/neutron/metadata_agent.ini 默
认 \ admin_tenant_name 服务# openstack config--集
/etc/neutron/metadata_agent.ini 默认 \ admin_user 中子#
openstack config--集 /etc/neutron/metadata_agent.ini 默认
\ admin_password NEUTRON_PASS # openstack config--集
/etc/neutron/metadata_agent.ini 默认 \ nova_metadata_ip 控
制器# openstack config--集 /etc/neutron/metadata_agent.ini
默认 \ metadata_proxy_shared_secret METADATA_SECRET
```

请注意

我们建议添加详细真实的 [默认] 节中 = /etc/neutron/metadata_agent.ini 来帮助解决。

2. 注

*控制器*节点上执行以下两个步骤。

3. 在*控制器*节点上，配置计算要使用元数据服务：

METADATA_SECRET 替换为元数据代理选择的秘密。

```
# openstack config--默认设置
/etc/nova/nova.conf \
service_neutron_metadata_proxy true #
openstack config--默认设置
/etc/nova/nova.conf \
neutron_metadata_proxy_shared_secret
METADATA_SECRET
```

4. 的*控制器*节点上重新启动计算 *API* 服务：

```
# 服务 openstack-新星-api 重新启动
```

若要配置模块化图层 2 （ML2） 插件

ML2 插件使用开放 vSwitch （OVS） 机制 （代理） 来构建虚拟网络框架的实例。

- 运行以下命令：

  替换在您的网络节点上的实例隧道网络接口的 IP 地址为 *INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS* 。本指南使用 10.0.1.21 的网络节点上的实例隧道网络接口的 IP 地址。

```
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \ type_drivers gre
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
tenant_network_types gre # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
mechanism_drivers openvswitch # openstack config--设置的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre \
tunnel_id_ranges 1: 1000 # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ovs \ local_ip
INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS # openstack config--
集的 /etc/neutron/plugins/ml2/ml2_conf.ini ovs \ tunnel_type
gre
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ovs \
enable_tunneling True # openstack config--基本设置的
/etc/neutron/plugins/ml2/ml2_conf.ini \ firewall_driver
neutron.agent.linux.iptables_firewall。
OVSHybridIptablesFirewallDriver # openstack config--基本设
置的 /etc/neutron/plugins/ml2/ml2_conf.ini \
enable_security_group True
```

若要配置打开 vSwitch （OVS） 服务

OVS 服务提供为基础的虚拟网络架构实例。桥 br int 一体化处理内 OVS 内部实例网络通信。外部桥 br-前处理在 OVS 内的外部实例网络通信。外部的桥需要物理的外部网络接口提供实例访问外部网络上的端口。从本质上说，此端口的桥梁在您的环境中的虚拟和物理的外部网络。

1. 启动的 OVS 服务并将其配置为在系统引导时启动：

```
# 服务 openvswitch 开始
```

```
# chkconfig openvswitch 上
```

2．添加集成桥：

```
# ovs vsctl 添加 br br int
```

3．添加外部桥：

```
# ovs vsctl 添加 br br-前
```

4．将端口添加到连接到物理的外部网络接口的外部桥：

INTERFACE_NAME 替换为实际的接口名称。例如， *eth2* 或 *ens256*.

```
# ovs vsctl 添加端口 br 前 INTERFACE_NAME
```

请注意

根据您网络接口驱动程序，您可能需要禁用*通用接收卸载 (GRO)*来实现适用于您的实例和外部网络之间的吞吐量。

若要测试您的环境时暂时禁用 GRO 外部网络接口上：

```
# ethtool-K INTERFACE_NAME gro 关闭
```

要完成安装

1．网络服务初始化脚本期望指向与你选择的插件关联的配置文件的符号链接 /etc/中子/plugin.ini.使用 ML2 插件，例如，符号链接必须指向 /etc/中子/plugins/ml2/ml2_conf.ini。如果这个符号链接不存在，请创建它使用下面的命令：

```
# ln-s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

由于包装错误，打开 vSwitch 代理初始化脚本显式打开 vSwitch 插件配置文件，而不是指向 ML2 插件配置文件的符号链接 /etc/ neutron/plugin.ini 看起来。运行下面的命令来解决此问题：

```
# cp /etc/init.d/neutron-openvswitch-agent
/etc/init.d/neutronopenvswitch-agent.orig # sed-i
's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g'
/etc/ init.d/中子-openvswitch-代理
```

2．启动网络服务并将它们配置为在系统引导时启动：

```
# 在服务中子-openvswitch-代理启动
# 在服务中子-l3-代理启动
```

```
# 在服务中子-dhcp-代理启动
# 在服务中子-元数据-代理启动
# chkconfig 中子-openvswitch-代理上
# chkconfig 中子-l3-代理上
# chkconfig 中子 dhcp 代理上
# chkconfig 中子元数据代理上
```

配置计算节点

系统必备组件

您在配置 OpenStack 网络之前，您必须启用某些内核网络功能。

1. 编辑 /etc/sysctl.conf，包含下列内容：

```
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2. 实施的改变：

```
# sysctl-p
```

要安装的网络组件

•

若要配置的网络常见组件

```
# yum 安装 openstack-中子-ml2 openstack-中子-openvswitch
```

联网通用组件配置包括身份验证机制、 消息代理和插件。

1. 配置网络要使用的身份验证的身份服务：

NEUTRON_PASS 替换为中标识服务的中子用户选择的密码。

```
# openstack config--集 /etc/neutron/neutron.conf 默认 \
auth_strategy 重点# openstack config--集的
/etc/neutron/neutron.conf keystone_authtoken \ auth_uri http://
控制器： 5000 # openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ auth_host 控制器# openstack config--设置的
/etc/neutron/neutron.conf keystone_authtoken \ auth_protocol
http # openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ auth_port 35357 # openstack config--集的
/etc/neutron/neutron.conf keystone_authtoken \
```

```
admin_tenant_name 服务# openstack config--设置的
/etc/neutron/neutron.conf keystone_authtoken \ admin_user 中子
# openstack config--集的 /etc/neutron/neutron.conf
keystone_authtoken \ admin_密码 NEUTRON_PASS
```

2. 配置联网使用的消息代理：

```
# openstack config--默认设置
/etc/neutron/neutron.conf \ rpc_backend
neutron.openstack.common.rpc.impl_qpid # openstack
config--默认设置 /etc/neutron/neutron.conf \
qpid_hostname 控制器
```

3. 配置联网使用的模块化图层 2 （ML2） 插件和相关服务：

```
# openstack config--默认设置
/etc/neutron/neutron.conf \ core_plugin ml2 #
openstack config--默认设置
/etc/neutron/neutron.conf \ service_plugins 路由器
```

请注意

我们建议添加详细真实的 [默认] 节中 = / etc/neutron/neutron.conf 来帮助解决。

若要配置模块化图层 2 （ML2） 插件

ML2 插件使用开放 vSwitch (OVS) 机制 （代理） 建立虚拟的网络架构为实例。

- 运行以下命令：

  替换上你计算节点的实例隧道网络接口的 IP 地址为 *INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS* 。本指南使用 10.0.1.31 上的第一个计算节点的实例隧道网络接口的 IP 地址。

```
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \ type_drivers gre
# openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
tenant_network_types gre # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
mechanism_drivers openvswitch # openstack config--设置的
/etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre \
```

```
tunnel_id_ranges 1: 1000 # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ovs \ local_ip
INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS # openstack config--
集的 /etc/neutron/plugins/ml2/ml2_conf.ini ovs \ tunnel_type
gre # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini ovs \
enable_tunneling True # openstack config--集的
/etc/neutron/plugins/ml2/ml2_conf.ini 基本 \
firewall_driver neutron.agent.linux.iptables_firewall。
OVSHybridIptablesFirewallDriver # openstack config--基本设
置的 /etc/neutron/plugins/ml2/ml2_conf.ini \
enable_security_group True
```

若要配置打开 vSwitch （OVS） 服务

OVS 服务提供为基础的虚拟网络架构实例。桥 br int 一体化处理内 OVS 内部实例网络通信。

1．启动的 OVS 服务并将其配置为在系统引导时启动：

```
# 服务 openvswitch 开始
# chkconfig openvswitch 上
```

2．添加集成桥：

```
# ovs vsctl 添加 br br int
```

若要配置计算要使用网络

默认情况下，大多数发行版配置计算使用旧版网络。您必须重新配置计算机来管理网络通过联网。

- 运行以下命令：

  NEUTRON_PASS 替换为中标识服务的中子用户选择的密码。

```
# openstack config--集 /etc/nova/nova.conf 默认 \ network_api_class
nova.network.neutronv2.api.API # openstack config--集
/etc/nova/nova.conf 默认 \ neutron_url http://控制器： 9696 #
openstack config--集 /etc/nova/nova.conf 默认 \
neutron_auth_strategy 重点# openstack config--集
/etc/nova/nova.conf 默认 \ neutron_admin_tenant_name 服务#
openstack config--集 /etc/nova/nova.conf 默认 \
neutron_admin_username 中子# openstack config--集
```

```
/etc/nova/nova.conf 默认 \ neutron_admin_password NEUTRON_PASS #
openstack config--集 /etc/nova/nova.conf 默认 \
neutron_admin_auth_url http://控制器: 35357/v2.0 # openstack
config--集 /etc/nova/nova.conf 默认 \ linuxnet_interface_driver
nova.network.linux_net.LinuxOVSInterfaceDriver # openstack
config--默认设置 /etc/nova/nova.conf \ firewall_driver
nova.virt.firewall.NoopFirewallDriver # openstack config--默认设置
/etc/nova/nova.conf \
security_group_api 中子
```

请注意

默认情况下，计算使用内部防火墙服务。由于网络包括防火墙服务，您必须通过使用 nova.virt.firewall.NoopFirewallDriver 防火墙驱动程序禁用计算防火墙服务。

要完成安装

1. 网络服务初始化脚本期望指向与你选择的插件关联的配置文件的符号链接 /etc/中子/plugin.ini.使用 ML2 插件，例如，符号链接必须指向 /etc/中子/plugins/ml2/ml2_conf.ini。如果这个符号链接不存在，请创建它使用下面的命令：

```
# ln-s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

由于包装错误，打开 vSwitch 代理初始化脚本显式打开 vSwitch 插件配置文件，而不是指向 ML2 插件配置文件的符号链接 /etc/ neutron/plugin.ini 看起来。运行下面的命令来解决此问题：

```
# cp /etc/init.d/neutron-openvswitch-agent
/etc/init.d/neutronopenvswitch-agent.orig # sed-i
's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g'
/etc/ init.d/中子-openvswitch-代理
```

2. 重新启动计算服务：

```
# 服务 openstack-新星-计算重新启动
```

3. 启动打开 vSwitch (OVS) 代理并将其配置为在系统引导时启动：

```
# 在服务中子-openvswitch-代理启动
# chkconfig 中子-openvswitch-代理上
```

创建初始网络

在启动之前您的第一个实例，您必须创建该实例将连接到，包括必要的虚拟网络基础设施外部网络和租客网络。请参阅图 7.1，"初始网络"[61]。在创建之后这种基础设施，我们建议你验证连接和解决任何问题之前进一步。

外部网络

外部网络通常会提供互联网接入，为您的实例。默认情况下，这一网络只允许互联网访问*从*实例使用*网络地址转换 (NAT)*。您可以启用互联网访问*到*的各个实例使用一个*浮动 IP 地址*和合适*的安全组*的规则。Admin 租客拥有这一网络，因为它为多个租户提供外部网络访问。您还必须启用共享，以允许访问到那些的租户。

请注意

控制器节点上执行这些命令。

若要创建外部网络

1. 源 admin 租客凭据：

```
$ 源管理-openrc.sh
```

2. 创建网络：

```
$ 中子网创建 ext-网 — — 共享 — — 路由器：外部 = True 创建一个新的网络：
+---------------------------+--------------------------------------+
|字段 |值 |
+---------------------------+--------------------------------------+
|admin_state_up |True |
|id |893aebb9-1c1e-48be-8908-6b947f3237b3 |
|名称 |ext 网 |
|提供者： network_type |gre 考试 |
|提供者： physical_network | |
|提供者： segmentation_id |1 |
|路由器：外部 |True |
|共享 |True |
```

```
|状态 |活动 |
|子网 | |
|tenant_id |54cd044c64d5408b83f843d63624e0d8 |
+--------------------------+--------------------------------------+
```

就像一个物理的网络，一个虚拟的网络需要分配给它*的子网*。相同的子网和*网关*与物理网络相关联的外部网络共享连接到的网络节点上的外部接口。您应该指定独家片的这个子网的*路由器*和浮动 IP 地址，防止外部网络上的其他设备的干扰。

替换你想要为浮动 IP 地址分配的范围的第一个和最后一个 IP 地址为 *FLOATING_IP_START* 和 *FLOATING_IP_END* 。*EXTERNAL_NETWORK_CIDR* 替换与物理网络相关联的子网。替换 *EXTERNAL_NETWORK_GATEWAY* 与网关通常与物理网络中，关联 ".1" 的 IP 地址。因为实例做不直接连接到外部网络和浮动 IP 地址需要手动分配，则应禁用 *DHCP* 这个子网上。

在外部网络上创建一个子网

- 创建子网：

```
$ 中子子网创建 ext 网--ext-子网名称。
— — 分配池开始 =FLOATING_IP_START，结束 =FLOATING_IP_END \
—— 禁用 dhcp —— 网关 EXTERNAL_NETWORK_GATEWAY EXTERNAL_NETWORK_CIDR
```

例如，使用 203.0.113.0/24，利用浮动 IP 地址范围 203.0.113.101 至 203.0.113.200：

```
$ 中子子网创建 ext 网--ext-子网名称。
— — 分配池开始 = 203.0.113.101，结束 = 203.0.113.200。
— — 禁用 dhcp — — 网关 203.0.113.1 203.0.113.0/24 创建新的子网：
+-------------------
+-----------------------------------------------------+
|字段 |价值
 |
+-------------------
+-----------------------------------------------------+
|allocation_pools |{"开始": "203.0.113.101", "结束": "203.0.113.200"}
 |
|cidr |203.0.113.0/24
 |
|dns_nameservers |
 |
```

```
|enable_dhcp |虚假
 |
|gateway_ip |203.0.113.1
 |
|host_routes |
 |
|id |9159f0dc-2b63-41cf-bd7a-289309da1391
 |
|ip_version |4
 |
|ipv6_address_mode |
 |
|ipv6_ra_mode |
 |
|名称 |ext 子网
 |
|network_id |893aebb9-1c1e-48be-8908-6b947f3237b3
| |tenant_id |54cd044c64d5408b83f843d63624e0d8
 |
+-------------------
+-----------------------------------------------------+
```

租客网络

租客网络提供了内部网络的访问实例。该体系结构隔离这种类型的网络从其他租户。演示租客拥有这一网络，因为它只提供了网络的访问内它的实例。

请注意

控制器节点上执行这些命令。

若要创建的租客网络

1．源演示租客凭据：

```
$ 源演示-openrc.sh
```

2．创建网络：

```
$ 中子网创建演示-网创建一个新的网络：
+---------------+--------------------------------------+
```

```
|字段 |值 |
+----------------+--------------------------------------+
|admin_state_up |True |
|id |ac108952-6096-4243-adf4-bb6615b3de28 |
|名称 |演示网 |
|共享 |虚假 |
|状态 |活动 |
|子网 | |
|tenant_id |cdef0071a0194d19ac6bb63802dc9bae |
+----------------+--------------------------------------+
```

像外部网络，您的租客网络还需要一个附加到它的子网。因为该体系结构分离株租客网络，您可以指定任何有效的子网。*TENANT_NETWORK_CIDR* 替换为你想要与租客网络关联的子网。*TENANT_NETWORK_GATEWAY* 替换为你想要与此网络，通常关联的网关".1" 的 IP 地址。默认情况下，此子网将使用 DHCP，所以您的实例可以获得 IP 地址。

租客网络上创建子网

- 创建子网：

```
$ 中子子网创建演示网--演示-子网名称。
— — 网关 TENANT_NETWORK_GATEWAY TENANT_NETWORK_CIDR
```

使用 192.168.1.0/24 的示例：

```
$ 中子子网创建演示网--演示-子网名称。
— — 网关为 192.168.1.1 192.168.1.0/24 创建新的子
网：
+-------------------
+-----------------------------------------------------+
|字段 |价值
 |
+-------------------
+-----------------------------------------------------+
|allocation_pools |{"开始": "192.168.1.2", "结束": "192.168.1.254"}
 |
|cidr |192.168.1.0/24
 |
|dns_nameservers |
```

```
 |
|enable_dhcp |真实
 |
|gateway_ip |192.168.1.1
 |
|host_routes |
 |
|id |69d38773-794a-4e49-b887-6de6734e792d
 |
|ip_version |4
 |
|ipv6_address_mode |
 |
|ipv6_ra_mode |
 |
|名称 |演示子网
 |
|network_id |ac108952-6096-4243-adf4-bb6615b3de28
 |
|tenant_id |cdef0071a0194d19ac6bb63802dc9bae
 |
+-------------------
+-----------------------------------------------------+
```

一个虚拟的路由器之间两个或更多的虚拟网络传递的网络流量。每个路由器，需要一个或更多*的接口*和/或提供对特定网络访问的网关。在这种情况下，您将创建一个路由器，将你的租客和外部网络附加到它。

租客网络上创建一个路由器和附加外部及租客网络到它的

1．创建路由器：

```
$ 中子路由器创建演示路由器创建一个新的路由器：
+-----------------------+---------------------------------------+
|字段 |值 |
+-----------------------+---------------------------------------+
|admin_state_up |True |
|external_gateway_info | |
```

```
|id |635660ae-a254-4feb-8993-295aa9ec6418 |
|名称 |演示-路由器 |
|状态 |活动 |
|tenant_id |cdef0071a0194d19ac6bb63802dc9bae |
+----------------------+--------------------------------------+
```

2．附加到演示租客子网的路由器：

```
$ 中子子网路由器接口添加演示-路由器演示-添加的接口路由器
demorouter 到
b1a894fd-aee8-475c-9262-4342afdc1b58。
```

3．附加到外部网络的路由器通过将它设置为网关：

```
$ 中子路由器-网关-设置演示-路由器 ext-网
网关设置为路由器演示-路由器
```

验证连接性

我们建议您验证网络连接和在进一步行动之前解决任何问题。以下的外部网络的子网示例中使用 203.0.113.0/24，租客路由器网关应在浮动 IP 地址范围内，203.0.113.101 占据的最低的 IP 地址。如果您的外部物理网络和虚拟网络配置正确，您应该 **ping** 这个 IP 地址从你外在的物理网络上的任何主机。

请注意

如果你正在构建你 OpenStack 节点作为虚拟机，则必须配置虚拟机监控程序，外部网络上允许混杂模式。

要验证网络连接，

- Ping 的租客路由器网关：

```
$ ping-c 4 203.0.113.101PING 203.0.113.101 (203.0.113.101)
56(84) 字节的数据。
64 字节从 203.0.113.101: icmp_req = 1 ttl = 64 时间 = 0.619 女士
64 字节从 203.0.113.101: icmp_req = 2 ttl = 64 时间 = 0.189 ms
64 字节从 203.0.113.101: icmp_req = 3 ttl = 64 时间 = 0.165 ms
64 字节从 203.0.113.101: icmp_req = 4 ttl = 64 时间 = 0.216 ms

---203.0.113.101 ping--4 传输的信息包，4 收到的统计，0%数据包丢失，rtt
时间 2999ms 分钟，avg，马克斯，.= 0.165/0.297/0.619/0.187 女士
```

旧版联网 (nova 网络)

配置控制器节点

遗产联网主要涉及到计算节点。但是，您必须配置的控制器节点使用它。

若要配置旧版网络

1. 运行下面的命令：

```
# openstack config--默认设置
/etc/nova/nova.conf \
network_api_class
nova.network.api.API # openstack
config--默认设置
/etc/nova/nova.conf \
security_group_api nova
```

2. 重新启动计算服务：

```
# 服务 openstack-新星-api 重新启动
# 服务 openstack-新星-调度程序重新启动
# 服务 openstack-新星-导体重新启动
```

配置计算节点

```
# yum 安装 openstack-新星-网络 openstack-新星-api
```

本节介绍部署的一个简单的*平面网络*，提供给您的实例通过 *DHCP* 的 IP 地址。如果您的环境包括多个计算节点，*多主机*功能通过传播跨计算节点的网络功能提供了冗余。

要安装旧版网络组件

•

若要配置旧版网络

1. 运行下面的命令：

   替换 *INTERFACE_NAME* 用于外部网络的实际的接口名称。例如，*eth1* 或 *ens224*.

```
# openstack config--集 /etc/nova/nova.conf 默认 \
network_api_class nova.network.api.API #
```

```
openstack config--集 /etc/nova/nova.conf 默认 \
security_group_api nova # openstack config--集
/etc/nova/nova.conf 默认 \ network_manager
nova.network.manager.FlatDHCPManager #
openstack config--集 /etc/nova/nova.conf 默认 \
firewall_driver
nova.virt.libvirt.firewall.IptablesFirewallDri
ver # openstack config--集 /etc/nova/nova.conf 默
认 \ network_size 254 # openstack config--集
/etc/nova/nova.conf 默认 \
allow_same_net_traffic False # openstack
config--集 /etc/nova/nova.conf 默认 \ multi_host
True # openstack config--集 /etc/nova/nova.conf
默认 \ send_arp_for_ha True # openstack-config--
设置 /etc/nova/nova.conf 默认 \
share_dhcp_address True # openstack config--集
/etc/nova/nova.conf 默认 \ force_dhcp_release
True # openstack config--集 /etc/nova/nova.conf
默认 \ flat_network_bridge br100 # openstack
config--集 /etc/nova/nova.conf 默认 \
flat_interface INTERFACE_NAME # openstack
config--集 /etc/nova/nova.conf 默认 \
public_interface INTERFACE_NAME
```

2．启动的服务，并将它们配置为在系统引导时启动：

```
# 服务 openstack-新星-网络启动
# 服务 openstack 新星元数据 api 的开始
# chkconfig openstack 新星网络上
# chkconfig openstack-新星-元数据-api 上
```

创建初始的网络

在启动之前您的第一个实例，您必须创建该实例将连接到的必要的虚拟网络基础设施。此网络通常提供*从*实例的互联网访问。您可以启用互联网访问*到*的各个实例使用一个*浮动 IP 地址*和合适*的安全组*的规则。Admin 租客拥有这一网络，因为它为多个租户提供外部网络访问。

此网络共享相同*的子网*与物理网络连接到外部*接口*上计算节点相关联。您应该指定独家片的这个子网，防止外部网络上的其他设备的干扰。

请注意

控制器节点上执行这些命令。

若要创建网络

1. 源 admin 租客凭据：

```
$ 源管理-openrc.sh
```

2. 创建网络：

NETWORK_CIDR 替换与物理网络相关联的子网。

```
$ 新星网络创建演示-网 — — 桥梁 br100 — — 多主机 T \
— — 固定范围 v4 NETWORK_CIDR
```

例如，使用 203.0.113.24 到 203.0.113.32 的 IP 地址范围的 203.0.113.0/24 独家切片：

```
$ 新星网络创建演示-网 — — 桥梁 br100 — — 多主机 T \
— — 固定范围 v4 203.0.113.24/29
```

请注意

该命令提供无输出。

3.验证网络的创建：

```
$ 新星网-列表
+--------------------------------------+----------+-------------
-----+
|ID |标签 |CIDR |
+--------------------------------------+----------+-------------
-----+
|84b34a65-a762-44d6-8b5e-3b461a53f513 |演示网 |203.0.113.24/29 |
+--------------------------------------+----------+-------------
-----+
```

接下来的步骤

你 OpenStack 环境现在包括启动一个基本的实例所需的核心组件。你可以启动一个实例或将更多的服务添加到您的环境，在下面的章节.

8。添加仪表板

表的内容

OpenStack 仪表，也被称为地平线，是一个 Web 界面，使云管理员和用户能够管理各种 OpenStack 资源和服务。

仪表板使基于 web 的交互与 OpenStack 计算云控制器通过 OpenStack Api。

这些指令显示配置与 Apache web 服务器部署示例。

在你后安装和配置仪表板，你可以完成以下任务：

- 自定义您的仪表板。请参阅自定义仪表板 *OpenStack 云管理员指南》*中一节。
- 设置会话存储的仪表板。请参阅有一段叫"设置会话存储的仪表板"[71].

系统要求

你在安装 OpenStack 仪表板之前，你必须满足以下系统要求：

- OpenStack 计算安装。启用标识服务的用户和项目管理。

  请注意的身份服务和计算终结点的 Url。

- 标识服务用户使用 sudo 的权限。因为 Apache 不从根用户服务的内容，用户必须运行仪表板作为标识服务用户使用 sudo 特权。

- Python 2.6 或 2.7。Python 版本必须支持 Django。Python 版本应该运行在任何系统上，包括 Mac OS X.安装必备组件可能会与不同的平台。

然后，安装并配置仪表板上一个节点，它可以与身份服务联系。

这样，他们可以通过在他们的本地计算机上的 web 浏览器访问仪表板，为用户提供以下信息：

- 公共 IP 地址，他们可以访问仪表板
- 用户名称和密码，他们可以访问仪表板

您的 web 浏览器，那你的用户，必须支持 HTML5 和吃饼干和 JavaScript 启用。

请注意

若要使用仪表板的 VNC 客户端，浏览器必须支持 HTML5 画布和 HTML5 Websocket。

> 有关支持 noVNC 的浏览器的详细信息，请参阅 https://github.com/kanaka/noVNC/blob/master/README.md 和 https://github.com/kanaka/ noVNC/wiki/浏览器支持，分别。

安装仪表板

你可以在安装和配置了仪表板之前，满足的要求被称为"系统要求"部分 [69].

请注意

当您安装只有对象存储和身份的服务，即使您安装仪表板时，它拉不上来的项目，无法使用。

有关如何部署该仪表板的详细信息，请参阅中的部署主题开发人员文档.

1. 可以联系身份服务为根的节点上安装的仪表板：

```
# yum 安装 memcached python memcached mod_wsgi openstack 仪表板
```

2. 修改缓存 ['默认'] ['位置'] 中 /etc/openstackdashboard/local_settings 来匹配那些在 /etc/sysconfig/memcached 中设置的值。

打开 /etc/openstack-dashboard/local_settings，查找这一行：

```
缓存 = {
默认： {
'后端'： 'django.core.cache.backends.memcached.MemcachedCache'，
'位置'： '127.0.0.1:11211'
}
}
```

备注

- 的地址和端口必须匹配的设置在 /etc/sysconfig/memcached。

  如果您更改 memcached 的设置，您必须重新启动 Apache web 服务器以使更改生效。

- 可以为会话存储使用 memcached 选项之外的选择。设置会话后端通过 *SESSION_ENGINE* 选项。

- 若要更改时区，使用仪表板或编辑 /etc/ openstack-仪表板/local_settings 文件。

  更改下面的参数： TIME_ZONE ="UTC"

3. 更新在 local_settings.py ALLOWED_HOSTS，包括你想要访问从仪表板的地址。

   编辑 /etc/openstack-dashboard/local_settings：

```
ALLOWED_HOSTS = [localhost，我桌面]
```

4. 本指南假定您正在运行的仪表板上的控制器节点。你很容易可以在单独的服务器上，运行仪表板通过更改相应的设置在 local_settings.py.

   编辑 /etc/openstack-dashboard/local_settings 和 OPENSTACK_HOST 改为你的身份服务的主机名：

```
OPENSTACK_HOST ="控制器"
```

5. 确保系统的 SELinux 策略被配置为允许到 HTTP 服务器的网络连接。

```
# 关于 setsebool-P httpd_can_network_connect
```

6. 启动 Apache web 服务器和 memcached：

```
# 服务 httpd 启动
# memcached 服务启动
# chkconfig httpd 上
# chkconfig memcached 上
```

7. 您现在可以访问的仪表板，在 http://controller/dashboard.

与任何你创建的 OpenStack 标识服务的用户的凭据登录。

设置会话存储的仪表板

仪表板使用 Django 的会话框架来处理用户会话数据。然而，您可以使用任何可用的会话的后端。在您的 local_settings 文件中自定义的会话后端通过 SESSION_ENGINE 设置 （在 Fedora/RHEL/CentOS 上：/etc/openstack-dashboard/local_settings，在 Debian 和 Ubuntu 上的： /etc/ openstack-仪表板 /local_settings.py 和 openSUSE：/srv/www/openstack-dashboard/openstack_dashboard /local/local_settings.py）。

以下各节描述每个选项的利弊，因为它涉及到部署的仪表板。

本地内存缓存

本地内存存储是回来，最快和最简单的会话结束设置，因为它没有任何的外部依赖项。它具有以下显著缺点：

- 跨进程或工人没有共享的存储。

- 在进程终止后没有持久性。

本地内存后端作为默认为启用地平线仅仅因为它没有任何依赖项。不建议为生产使用，或甚至为严重的事态发展

键-值存储

您可以使用应用程序例如 Memcached 或穿红衣的外部高速缓存。这些应用程序提供持久性和共享的存储，是有用的小规模部署和/或发展。

Memcached

Memcached 是一个高性能、 分布式内存对象缓存系统提供的任意数据的小块内存中键-值存储。

要求：

• Memcached 服务运行并且可以访问。

• 安装的 Python 模块 python memcached。

穿红衣

穿红衣的是一个开放源码的 BSD 许可，先进的键 / 值存储。它通常被称为一个数据结构服务器。

要求：

• 穿红衣的服务运行并且可以访问。

• Python 模块穿红衣和 django-穿红衣的安装.

初始化和配置数据库

备份数据库会话是可扩展，宿存，和可接受高并发性和高可用性。

但是，数据库支持的会话是一个较慢的会话存储，在沉重的使用情况下的高开销。正确配置您的数据库部署也可以是一项重大任务，是远远超出了本文档的范围。

1. 启动 mysql 命令行客户端：

```
$ mysql-u 根-p
```

2. 输入 MySQL root 用户的密码提示时。

3. 要配置的 MySQL 数据库，创建破折号数据库：

```
mysql >创建数据库破折号 ；
```

4. 创建一个 MySQL 的用户的新创建的破折号数据库的数据库的完全控制。*DASH_DBPASS* 替换为新用户的密码：

```
mysql >授予所有权限上 dash.* 到 'dash'@'%' 确定的
'DASH_DBPASS';mysql >授予所有权限上 dash.* 到
'dash'@'localhost' 确定的 'DASH_DBPASS';
```

5. Enter 在 mysql 时退出 > 提示退出 MySQL。

6. 在 local_settings 文件中 （在 Fedora/RHEL/CentOS 上：/etc/openstackdashboard/local_settings，在 Ubuntu/Debian 上的：/etc/openstackdashboard/local_settings.py 和 openSUSE：/srv/www/openstackdashboard/openstack_dashboard/local/local_settings.py），更改这些选项：

```
SESSION_ENGINE = 'django.core.cache.backends.db.DatabaseCache'
数据库 = {
默认： {
# 数据库配置在这里 '引擎'：
'django.db.backends.mysql',
NAME：短跑
'用户'： '冲'，
'密码'： 'DASH_DBPASS',
主机： localhost，默认字符集 '： 'utf8'
    }
}
```

7. 配置后 local_settings 如图所示，您可以运行 **manage.py syncdb** 命令来填充这个新创建的数据库。

```
$ /usr/share/openstack-dashboard/manage.py syncdb
```

注意在 openSUSE 路径是 /srv/ www/openstack-dashboard/ manage.py。

结果，返回下面的输出：

```
安装自定义 SQL...
正在安装索引...
DEBUG:django.db.backends:(0.008) 创建索引 'django_session_c25c2c28' 上
'django_session' ('expire_date') ； ; args=()
装置没有发现。
```

8. 对 Ubuntu： 如果你想要避免的警告，当你重新启动 apache2 时，一个黑洞目录在目录下创建仪表板，，如下所示：

```
# mkdir-p /var/lib/dash/.blackhole
```

9. 重新启动 Apache 来接起来的默认站点和符号链接设置：

在 Ubuntu： 上

```
# /etc/init.d/apache2 重新启动
```

在 Fedora/RHEL/CentOS：

```
# 服务重新启动 httpd

# 服务 apache2 重启
```

在 openSUSE：

```
# 立即重新启动 apache2.service
```

10. 在 Ubuntu 上，重启 nova api 服务，以确保 API 的服务器可以连接到的仪表板不会出现错误：

```
# nova api 重新启动服务
```

缓存的数据库

为了减轻数据库查询的性能问题，您可以使用 Django **cached_db** 会话后端，利用您的数据库和缓存执行写通过缓存的基础设施和高效的检索。

如前文所述，通过配置您的数据库和缓存，启用此混合设置。然后，设置以下值：

```
SESSION_ENGINE ="django.contrib.sessions.backends.cached_db"
```

饼干

如果您使用 Django 1.4 或更高版本， **signed_cookies** 后端避免了服务器的负载和扩展的问题。

这个后端将会话数据存储在 cookie 中，这存储用户的浏览器。背部的最终用途加密的签名技术，以确保会话数据不是在运输过程中篡改。这不是加密 ； 相同会话数据是由攻击者仍然可读。

这台发动机的优点是它不需要附加依赖项或基础设施的开销，和它无限期地，只要正在进入一个正常的 cookie 存储的适合的会话数据量的尺度。

最大的坏处是它将会话数据放置到存储在用户的机器上，并把它输送过铁丝网。它还限制可以存储的会话数据的数量。

请参阅 Django 基于 cookie 的会话文档。

接下来的步骤

你 OpenStack 环境现在包括仪表板。你可以启动实例或将更多的服务添加到您的环境，在下面的章节。

9。添加数据块存储服务

表的内容

OpenStack 块存储服务工作，通过一系列的命名煤渣-守护进程的互动 *，坚持不懈地驻留在主机机器或机器上。从单个节点或跨多个节点，您可以运行的二进制文件。你也可以在其他 OpenStack 服务相同的节点上运行它们。以下各节介绍块存储服务组件和概念，并向您展示如何配置和安装的数据块存储服务。

数据块存储

数据块存储服务使您能够管理卷、 卷快照、 卷类型。它包括以下组件：

- 煤渣 api： 接受 API 请求并将它们路由到煤渣卷行动。
- 煤渣卷： 响应请求读取和写入数据块存储数据库来维护状态，与其他的进程 （就像煤渣-调度程序） 通过消息队列和直接在数据块存储提供的硬件或软件进行交互。它可以与存储提供商通过驱动程序体系结构的各种交互。
- 煤渣调度器守护程序： 像 nova 调度器，挑选最优块存储提供程序节点在其上创建卷。
- 消息队列： 路线之间的数据块存储服务流程的信息。

数据块存储服务与交互与计算提供卷为实例。

配置数据块存储服务控制器

请注意

这种情况下的*控制器节点*上配置 OpenStack 块存储服务和假定第二个节点提供存储通过 cindervolume 服务。

> 有关如何配置第二个节点，请参阅有一段叫"配置一个数据块存储服务节点"[78].

您可以配置 OpenStack 使用各种存储系统。此示例使用 LVM。

1. 安装适当的程序包的块存储服务：

```
# yum 安装 openstack 煤渣
```

2. 配置的块存储，使用您的数据库。

运行以下命令来设置连接选项，在 [数据库] 部分中，它是在 /etc/cinder/cinder.conf 文件中，将 *CINDER_DBPASS* 替换的块存储数据库，您将在稍后的步骤中创建的密码：

```
# openstack config--集 /etc/cinder/cinder.conf \
数据库连接 mysql://cinder:CINDER_DBPASS@控制器/cinder
```

3. 使用您设置以 root 身份创建一个煤渣数据库登录的密码：

```
# mysql-u 根-p mysql >创建数据库
渣 ；
mysql >授予所有权限上 cinder.* 到 'cinder'@'localhost' \ 确定由
'CINDER_DBPASS';
mysql >授予所有权限上 cinder.* 到 'cinder'@'%' \
由标识 'CINDER_DBPASS';
```

4. 创建数据库表中的数据块存储服务：

```
# 苏-s /bin/sh-c"煤渣-db 同步管理"煤渣
```

5. 创建一个煤渣用户。

数据块存储服务使用此用户标识服务进行身份验证。

使用服务租客和给用户管理员角色：

```
$ 重点用户创建 – – 名称 = 渣 – – 通过 =CINDER_PASS -
电子邮件 =cinder@example.com $重点用户-角色添加 – –
用户 = 渣 – – 租客 = 服务 – – 角色 = 管理员
```

6．编辑的 /etc/cinder/cinder.conf 配置文件：

```
# openstack config--集 /etc/cinder/cinder.conf 默
认 \ auth_strategy 重点# openstack config--集的
/etc/cinder/cinder.conf keystone_authtoken \
auth_uri http://控制器： 5000 # openstack config--
集的 /etc/cinder/cinder.conf keystone_authtoken \
auth_host 控制器# openstack config--集的
/etc/cinder/cinder.conf keystone_authtoken \
auth_protocol http # openstack config--设置的
/etc/cinder/cinder.conf keystone_authtoken \
auth_port 35357 # openstack config--设置的
/etc/cinder/cinder.conf keystone_authtoken \
admin_user 煤渣# openstack config--集的
/etc/cinder/cinder.conf keystone_authtoken \
admin_tenant_name 服务# openstack config--集的
/etc/cinder/cinder.conf keystone_authtoken \
admin_密码 CINDER_PASS
```

7．配置数据块存储使用 Qpid 消息代理：

```
# openstack config--集 /etc/cinder/cinder.conf \
默认 rpc_backend cinder.openstack.common.rpc.impl_qpid
# openstack config--集 /etc/cinder/cinder.conf \
默认 qpid_hostname 控制器
```

8．数据块存储服务注册与标识服务，以便其他 OpenStack 服务可以找到它：

```
$ 重点服务创建 — — 名称 = 渣 — — 类型 = 卷 — — 描述 =
"OpenStack 块存储"
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk ' / 卷 / {打印 $2}'）。
— — publicurl = http: / /控制器： 8776 /v1/%\ (tenant_id\) s。
— — internalurl = http: / /控制器： 8776 /v1/%\ (tenant_id\) s。
— — adminurl = http: / /控制器： 8776 /v1/%\ (tenant_id\) s
```

9．注册一个服务和终结点的数据块存储服务 API 版本 2：

```
$ 重点服务创建 — — 名称 = cinderv2 — — 类型 = volumev2 — — 描述 =
"OpenStack 块存储 v2"
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk ' / volumev2 / {打印 $2}'）。
— — publicurl = http: / /控制器： 8776 /v2/%\ (tenant_id\) s。
```

```
— — internalurl = http: / /控制器: 8776 /v2/%\ (tenant_id\) s。
— — adminurl = http: / /控制器: 8776 /v2/%\ (tenant_id\) s
```

10. 启动和数据块存储将服务配置为在系统引导时启动：

```
# 服务 openstack-煤渣-api 的开始
# 服务 openstack-煤渣-调度程序的开始
# chkconfig openstack 煤渣 api 上
# chkconfig openstack 煤渣-调度器上
```

配置一个数据块存储服务节点

你的控制器节点上配置的服务后，配置第二个系统是一个数据块存储服务节点。此节点包含服务的卷的磁盘。

您可以配置 OpenStack 使用各种存储系统。此示例使用 LVM。

1. 使用中的说明第 2 章，"基本环境配置"[6]对系统进行配置。请注意以下几点差异从控制器节点的安装说明：

   - 将主机名设置为 block1，并使用 10.0.0.41 作为管理网络接口上的 IP 地址。确保每个系统上主机文件中列出的 IP 地址和主机名为控制器节点和数据块存储服务节点。
   - 按照中的说明被称为"网络时间协议 （NTP）"[17] 条为从控制器节点同步。

2. 创建 LVM 的物理和逻辑卷。本指南假定用于此目的的第二个磁盘 /dev/ 康体：

```
# pvcreate/dev/康体发展局
# vgcreate 煤渣卷/dev/康体发展局
```

3. 将筛选器条目添加到设备部分中要保持 LVM 从扫描设备虚拟机所使用的 /etc/lvm/lvm.conf 文件：

```
设备 {......
筛选器 = ["a/sda1 /"、"康体发展局 /", "r /.*
/"]......
}
```

请注意

数据块存储主机上，您必须为 LVM 中添加所需的物理卷。运行 **pvdisplay** 命令，得到一个列表或所需的卷。

过滤数组中的每个项目的开始要么接受，或 r 为拒绝。需要在数据块存储主机的物理卷有以开头的名称。该数组必须以结束"r /.* /"拒绝未列出的任何设备。

在此示例中，/dev/sda1 是节点的操作系统卷驻留，而 /dev/sdb 是保留为 cindervolumes 的卷的卷。

4. 操作系统配置完后，安装适当的程序包的块存储服务：

```
# yum 安装 openstack 煤渣 scsi-目标-utils
```

5. 从控制器中，复制的 /etc/cinder/cinder.conf 配置文件或执行以下步骤来设置重点凭据：

```
# openstack config--集 /etc/cinder/cinder.conf 默认 \ auth_strategy 重点# openstack config--集的 /etc/cinder/cinder.conf keystone_authtoken \
auth_uri http://控制器： 5000 # openstack config--集的 /etc/cinder/cinder.conf keystone_authtoken \
auth_host 控制器# openstack config--集的 /etc/cinder/cinder.conf keystone_authtoken \
auth_protocol http # openstack config--设置的 /etc/cinder/cinder.conf keystone_authtoken \
auth_port 35357 # openstack config--设置的 /etc/cinder/cinder.conf keystone_authtoken \
admin_user 煤渣# openstack config--集的 /etc/cinder/cinder.conf keystone_authtoken \
admin_tenant_name 服务# openstack config--集的 /etc/cinder/cinder.conf keystone_authtoken \
admin_密码 CINDER_PASS
```

6. 配置数据块存储使用 Qpid 消息代理：

```
# openstack config--集 /etc/cinder/cinder.conf \
默认 rpc_backend cinder.openstack.common.rpc.impl_qpid
# openstack config--集 /etc/cinder/cinder.conf \
默认 qpid_hostname 控制器
```

7. 配置数据块存储使用你的 MySQL 数据库。编辑 /etc/煤渣 /cinder.conf 文件，并将以下项添加到 [数据库] 部分。*CINDER_DBPASS* 替换所选的数据块存储数据库的密码：

```
# openstack config--集 /etc/cinder/cinder.conf \
数据库连接 mysql://cinder:CINDER_DBPASS@控制器 /cinder
```

8. 配置数据块存储使用图像服务。数据块存储需要访问的图像来创建可引导卷。编辑 /etc/cinder/cinder.conf 文件，并更新 [默认] 节中的 glance_host 选项：

```
# openstack config--集 /etc/cinder/cinder.conf \
默认 glance_host 控制器
```

9. 配置 iSCSI 目标服务以发现数据块存储卷。如果它已不存在的 /etc/tgt/targets.conf 文件的开头添加以下行：

```
包括 /etc/煤渣/卷 / *
```

10. 启动和数据块存储将服务配置为在系统引导时启动：

```
# 服务 openstack 煤渣卷启动
# 服务 tgtd 开始
# chkconfig openstack 煤渣卷上
# chkconfig tgtd 上
```

验证数据块存储的安装

若要验证数据块存储是安装和配置正确，请创建新的卷。

有关如何管理卷的详细信息，请参阅 *OpenStack 用户指南*.

1. 源文件演示-openrc.sh：

```
$ 源演示-openrc.sh
```

2. 使用**煤渣创建**命令来创建新的卷：

```
$ 煤渣创建 – – 显示名称 myVolume 1
+---------------------+--------------------------------------+
|属性 |值 |
+---------------------+--------------------------------------+
|附件 |[] |
|availability_zone |新星 |
|可启动 |虚假 |
|created_at |2014-04-17T10:28:19.615050 |
|display_description |没有人 |
|display_name |myVolume |
|加密 |虚假 |
|id |5e691b7b-12e3-40b6-b714-7f17550db5d1 |
```

```
|元数据 |{} |
|大小 |1 |
|snapshot_id |没有人 |
|source_volid |没有人 |
|状态 |创建 |
|volume_type |没有人 |
+--------------------+--------------------------------------+
```

3.确定该卷已正确地创建用**煤渣列表**的命令：

```
$ 煤渣列表
--------------------------------------+-----------+-------------
-+------
+-------------+----------+-------------+
|ID |状态 |显示名称 |大小 |
卷类型 |可启动 |附加到
|+--------------------------------------+-----------+-----------
---+------
+-------------+----------+-------------+
|5e691b7b-12e3-40b6-b714-7f17550db5d1 |可用 |myVolume |1 |
没有人 |false | |
+--------------------------------------+-----------+------------
--+------
+-------------+----------+-------------+
```

如果没有可用的状态值，卷创建失败。检查/var/日志/煤渣/目录的日志文件在控制器和体积的节点，以获得有关失败的信息。

接下来的步骤

你 OpenStack 环境现在包括数据块存储。你可以启动实例或将更多的服务添加到您的环境，在下面的章节。

10。添加对象存储

表的内容

在一起以提供对象存储和检索通过 REST API OpenStack 对象存储的服务工作。对于此示例体系结构中，您必须已经安装了标识服务，也被称为基石。

对象存储服务

对象存储服务是一个高可扩展性和持久的多租户对象存储系统对于大量的非结构化数据在通过 rest 风格的 HTTP API 成本低。

它包括以下组件：

- 代理服务器 （swift-代理-服务器）。接受对象存储 API 和原始的 HTTP 请求上，传文件，修改元数据，创建的容器。它也是文件或容器

的列表，使 web 浏览器。为了提高性能，代理服务器可以使用可选的缓存，通常使用 memcache 部署。

- 帐户服务器 （swift-帐户-服务器）。管理定义与对象存储服务的帐户。
- 容器 （swift-容器-服务器） 的服务器。管理映射的容器或文件夹内的对象存储服务.
- 对象的服务器 （swift-对象-服务器）。管理实际的对象，如文件的存储节点上.
- 的定期进程数。对大型数据存储区执行日常任务。复制服务确保一致性和可用性通过群集。其他定期进程包括审计、 更新和收割者。
- 可配置 WSGI 中间件处理身份验证。通常的身份服务。

对象存储系统要求

硬件： OpenStack 对象存储设计在商品化的硬件上运行。

请注意

当您安装只有对象存储和身份的服务时，不能使用仪表板，除非你也安装计算和图像服务。

表 10.1。建议的硬件

服务器	推荐的硬件	备注
对象存储对象服务器	处理器： 双四核 内存： 8 或 12 GB RAM 磁盘空间： 为每 GB 的成本优化 网络： 一个 1 GB 网络接口卡 (NIC)	磁盘空间量取决于多少你可以有效地融入机架。您想要优化这些最好每 GB 的成本的同时仍获得行业标准失败率。在 rackspace 公司，我们的存储服务器当前正在运行相当通用 4U 服务器与 24 2T SATA 驱动器和 8 核的处理能力。RAID 存储驱动器上的不是需要，不建议使用。斯威夫特的磁盘的使用模式是最坏的情况可能为 RAID，和性能下降非常快使用 RAID 5 或 6。 作为一个例子，rackspace 公司运行云文件存储服务器 24 2T SATA 驱动器和 8 核的处理能力。大多数服务支持一个工人或并发值，在设置。这允许有效地利用核心的可用的服务。
对象存储容器/帐户服务器	处理器： 双四核 内存： 8 或 12 GB RAM 网络： 一个 1 GB	由于跟踪与 SQLite 数据库优化的 IOPS。

	网络接口卡 (NIC)	
对象存储代理服务器	处理器： 双四核 网络： 一个 1 GB 网络接口卡 (NIC)	更高的网络吞吐量提供更好的性能，支持很多的 API 要求。 优化您的代理服务器为获得最佳的 CPU 性能。代理服务器服务是更多的 CPU 和网络 I/O 密集型。如果您正在使用 10 GB 网络连接到代理服务器，或将终止在代理服务器的 SSL 通信，需要更大的 CPU 功率的。

操作系统：OpenStack 对象存储当前运行在 Ubuntu，RHEL，CentOS，Fedora，openSUSE 或 SLES。

网络： 内部建议 1 Gbps 或 10 Gbps。对于 OpenStack 对象存储，外部网络应将外面的世界连接到代理服务器，和存储网络要被孤立的专用网络或多个专用网络上。

数据库： 为 OpenStack 对象存储，SQLite 数据库是 OpenStack 对象存储容器和帐户管理过程的一部分。

权限： 您可以安装 OpenStack 对象存储作为根或 sudo 权限的用户如果您配置 sudoers 文件以启用的所有权限。

计划对象存储网络

为节约网络资源和确保网络管理员了解网络的需求和提供访问 Api 和存储网络的必要的公共 IP 地址，这一节提供的建议和所需的最小大小。建议至少 1000 Mbps 的吞吐量。

本指南介绍了以下网络：

- 一个强制性的公共网络。连接到代理服务器。
- 强制性存储网络。不可以从外部访问群集。所有节点都连接到此网络。
- 一种可选的复制网络。不可以从外部访问群集。致力于在存储节点之间的复制流量。必须配置在环。

此图显示公共网络、 存储网络和可选的复制网络的基本体系结构。

默认情况下，所有 OpenStack 对象存储服务，以及在存储节点上，rsync 守护进程配置为在他们的 STORAGE_LOCAL_NET IP 地址上侦听。

如果您配置一个复制网络环中，帐户、 容器和对象的服务器侦听的 STORAGE_LOCAL_NET 和 STORAGE_REPLICATION_NET 的 IP 地址。Rsync 守护进程只能 STORAGE_REPLICATION_NET IP 地址上侦听。

公共网络 （公开可路由 IP 范围）

提供公共 IP 访问到云基础设施内的 API 端点。

最小尺寸： 每个代理服务器的 IP 地址。

存储网络 （RFC1918 IP 范围内，不公开可路由）

管理对象存储基础结构中的所有服务器间通信。

最小尺寸： 为每个存储节点和代理服务器的一个 IP 地址。

推荐大小： 如上所述，与空间扩张到最大您的群集大小。例如，255 或 CIDR 24。

复制网络 （RFC1918 IP 范围内，不公开可路由）

管理对象存储基础结构中的存储服务器之间复制相关通信。

推荐大小： 至于 STORAGE_LOCAL_NET。

对象存储安装体系结构的例子

- 节点： 运行一个或多个 OpenStack 对象存储服务主机机器。
- 代理节点： 运行代理服务.
- 存储节点： 运行的帐户、 容器和对象服务。包含的 SQLite 数据库。
- 环： 一套 OpenStack 对象存储数据到物理设备之间的映射。
- 副本： 对象的一个副本。默认情况下，三份副本维护群集中。
- 区： 群集，独立故障特性相关的逻辑上单独一节.
- 地区 （可选）： 群集，代表如城市或国家的不同物理位置的逻辑上单独一节。类似于区但代表物理位置的群集，而不是逻辑段部分。

为了提高可靠性和性能，您可以添加额外的代理服务器。

本文档描述了作为一个单独的分区，在圆环的每个存储节点。在最低限度，推荐五个区。区域是一组是从其他节点 （单独的服务器、 网

络、 电源、 甚至地理） 尽可能像孤立的节点。这枚戒指可以保证每个副本存储在一个单独的分区中。此关系图中显示为最小安装一个可能的配置：

安装对象存储

虽然您可以为开发或测试的目的，一台服务器上安装 OpenStack 对象存储，多服务器安装允许你想要在一个生产分布式对象存储系统的高可用性和冗余。

要为发展目的从源代码执行单节点的安装，请使用 Swift 都在同一个指令 （Ubuntu） 或 DevStack （多个发行版）。请参阅 http:// swift.openstack.org/development_saio.html 手动的说明或 http://devstack.org 为所有在一包括身份验证标识服务 （重点） v2.0 的 API。

警告

在本指南中，我们建议安装和配置标识服务，因此，它实现了身份 API v2.0。实施访问控制列表 (Acl)，所以您必须使用 2.0 版的 API 来避免在不同的领域，将会使两个用户可以访问相同的对象中有相同的用户名称时，是没有意识到的域的对象存储服务。

在您开始之前

如果你在一个新的服务器上安装，有操作系统安装媒体可用的副本。

这些步骤假定您已经设置了您的操作系统中所示的软件包的存储库 OpenStack 包.

本文档演示如何通过使用以下类型的节点安装群集：

- 一个代理节点运行斯威夫特-代理-服务器进程。代理服务器的代理请求到相应的存储节点。
- 运行控制存储的帐户数据库、 集装箱数据库，以及实际的斯威夫特-帐户-服务器、斯威夫特-容器-服务器和 swiftobject 服务器进程的五个存储节点存储对象。

请注意

最初，可以用更少的存储节点，但是最小的五个建议对生产群集。

一般的安装步骤

1. 创建 swift 用户对象存储服务可以使用身份服务进行身份验证。选择一个密码并指定 swift 用户电子邮件地址。使用服务租客，并给用户管理员角色：

```
$ 重点用户创建 – – 名称 = swift – – 通过 =SWIFT_PASS \
– – 电子邮件 =swift@example.com
$ 重点用户-角色添加 – – 用户 = swift – – 租客 = 服务 – – 角色 = 管理员
```

2. 对象存储服务创建一个服务条目：

```
$ 重点服务创建 – – 名称 = swift – – 类型 = 对象存储。
– – 描述 ="OpenStack 对象存储"
+-------------+----------------------------------+
|属性 |值 |
+-------------+----------------------------------+
|描述 |OpenStack 对象存储 |
|id |eede9296683e4b5ebfa13f5166375ef6 |
|名称 |斯威夫特 |
|类型 |对象存储 |
+-------------+----------------------------------+
```

请注意

服务 ID 随机生成的是不同于此处所示。

3.指定 API 端点为对象存储服务通过使用返回的服务 id。当你指定终结点时，您为公共 API、 内部 API 和 admin API 提供的 Url。在本指南中，使用控制器主机名称：

```
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk ' 对象存储 / {打印
$2}') \
— — publicurl ='http: / /控制器: 8080 /v1/AUTH_%(tenant_id) s' \
— — internalurl ='http: / /控制器: 8080 /v1/AUTH_%(tenant_id) s' \
— — adminurl = http: / /控制器: 8080
+-------------+----------------------------------------------
--+
|属性 |值 |
+-------------+----------------------------------------------
--+
|adminurl |http://controller:8080 / |
|id |9e3ce428f82b40d38922f242c095982e |
|internalurl |http://controller:8080/v1/AUTH_ （tenant_id） %s |
|publicurl |http://controller:8080/v1/AUTH_ （tenant_id） %s |
|区域 |regionOne |
|service_id |eede9296683e4b5ebfa13f5166375ef6 |
+-------------+----------------------------------------------
--+
```

4．在所有节点上创建的配置目录：

```
# mkdir-p/等/斯威夫特
```

5．在所有节点上创建 /etc/swift/swift.conf：

```
[斯威夫特-哈希] # 随机的唯一字符串，可以永远
不会改变 （不要失去）
swift_hash_path_prefix =
xrfuniounenqjnw
swift_hash_path_suffix =
fLIbertYgibbitZ
```

请注意

/Etc/swift/swift.conf 中的前缀和后缀的值应设置为一些随机的文本字符串，以一种盐时用作哈希来确定在圆环中的映射。此文件必须在群集中的每个节点上相同 ！

下一步，设置你的存储节点和节点代理。此示例使用常见的身份验证件标识服务。

安装和配置存储节点

请注意

对象存储在任何支持扩展属性 （XATTRS） 的文件系统上工作。XFS 显示最佳的总体性能的迅速用例经过大量的测试和基准测试在 rackspace 公司。它也是唯一已被彻底测试的文件系统。请参阅 *OpenStack 配置参考*的补充建议。

1. 安装存储节点软件包：

```
# yum 安装 openstack-斯威夫特-帐户 openstack-斯威夫特-容器 \
openstack-斯威夫特-对象 xfsprogs xinetd
```

2. 对于每个节点的设备上，您想要使用的存储，设置 XFS 卷 /dev/sdb 作为一个例子）。使用每个驱动器的单个分区。例如，在一个具有 12 个磁盘服务器您可以使用一个或两个磁盘的操作系统，它不应该在这一步涉及到。其他 10 或 11 的磁盘应该带有单个分区，然后格式化在 XFS 分区。

```
# fdisk/dev/康体发展局
# mkfs.xfs /dev/sdb1 #回声"/ dev/sdb1 /srv/node/sdb1 xfs noatime,
nodiratime, nobarrier, logbufs =
8 0 0">> fstab/等 /
# mkdir-p /srv/node/sdb1
# 装载 /srv/节点/sdb1
# chown-R 斯威夫特： 斯威夫特/srv/节点
```

3. 创建 /etc/rsyncd.conf：

```
uid = 迅速
gid = 迅速
的日志文件
=
/var/log/
rsyncd.lo
```

```
g pid 文件
=
/var/run/
rsyncd.pi
d 地址 =
STORAGE_L
OCAL_NET_
IP
[帐户] 最大连接 = 2 路径 = /srv/node/
只读 = 假锁文件 =
/var/lock/account.lock

[集装箱] 最大连接 = 2 路径 = /srv/node/
只读 = 假锁文件 =
/var/lock/container.lock

[对象] 最大连接 = 2 路径 = /srv/node/
只读 = 假锁文件 =
/var/lock/object.lock
```

4. （可选） 如果您想要将 rsync 和复制到复制网络流量分开设置而不是 STORAGE_LOCAL_NET_IP STORAGE_REPLICATION_NET_IP：

```
地址 = STORAGE_REPLICATION_NET_IP
```

5. 编辑 /etc/xinetd.d/rsync 中的以下行：

```
禁用 = 否
```

6. 启动 xinetd 服务：

```
# 服务 xinetd 启动
```

请注意

Rsync 服务不要求身份验证，所以在一个本地的私人网络上运行它。

7.创建迅速侦察缓存目录并设置它的权限：

```
# mkdir-p /var/swift/recon
# chown-R 斯威夫特： 斯威夫特 /var/swift/recon
```

安装和配置代理服务器节点

代理服务器需要的每个请求和查找帐户、容器或对象的位置并正确地将请求路由。代理服务器还可以处理 API 发出的请求。可以通过将它配置在 /etc/swift/proxy-server.conf 文件中启用帐户管理。

请注意

对象存储过程运行下一个单独的用户和组、 设置的配置选项，和称为 swift： 斯威夫特。默认的用户是迅速的。

1. 安装 swift 代理服务：

```
# yum 安装 openstack-斯威夫特-代理 memcached python swiftclient pythonkeystone 身份验证令牌
```

2. 修改 memcached 在一个当地的非公共网络上的默认接口上侦听。

   编辑 /etc/sysconfig/memcached 文件：

```
选项 ="-l PROXY_LOCAL_NET_IP"
```

3. 启动 memcached 服务并将其配置为在系统引导时启动：

```
# memcached 服务启动
# chkconfig memcached 上
```

4. 编辑 /etc/swift/proxy-server.conf：

```
[默认] bind_port =
8080 用户 = 斯威夫特

[管道： 主] 管道 = 运行状况检查缓存 authtoken keystoneauth 代理服务器

[app:proxy-服务器] 使用 = 蛋：斯威
夫特 #proxy
allow_account_management = true
account_autocreate = true

[筛选器： keystoneauth] 使用 = 蛋： 斯威夫特
#keystoneauth operator_roles = 成员、 管理员、
swiftoperator

[筛选器： authtoken] paste.filter_factory =
keystoneclient.middleware.auth_token:filter_factory

# 延迟执行授权决定需要支持匿名引用令牌少 # 用法 （'.： *')。
```

```
delay_auth_decision = true

# auth_ * 设置是指重点服务器 auth_protocol = http
auth_host =控制器 auth_port = 35357

# 服务租客和 swift 用户名和密码创建在梯形 admin_tenant_name = 服务 admin_user =
swift admin_password = SWIFT_PASS

[过滤器： 缓存] 使用 = 蛋：
斯威夫特 #memcache

[筛选器： catch_errors] 使用 =
蛋： swift #catch_errors

[筛选器： 运行状况检查] 使用 =
蛋： 斯威夫特 #healthcheck
```

请注意

如果你运行多个 memcache 服务器，把多个之所以列表放在 /etc/swift/proxy-server.conf 文件的 [过滤器：缓存] 部分：

```
10.1.2.3:11211,10.1.2.4:11211
```

只有代理服务器使用 memcache。

5. 创建帐户、 容器和对象的戒指。生成器命令创建一个生成器文件与几个参数。该参数与 18 的值表示 2 ^18 th，分区大小调整为的值。"隔断电源"将该值设置的基础的你期待你整个的戒指要使用的存储总量。值 3 表示的每个对象的副本的数量正在要限制不止一次移动一个分区的小时数的最后一个值。

```
# cd/等/斯威夫特
# 斯威夫特-环-生成器 account.builder 创建 18 3 1
# 斯威夫特-环-生成器 container.builder 创建 18 3 1
# 斯威夫特-环-生成器 object.builder 创建 18 3 1
```

6. 为每个节点上的每个存储设备将条目添加到每个环：

```
# 斯威夫特-环-生成器 account.builder 添加 z 区
-STORAGE_LOCAL_NET_IP： 6002
[RSTORAGE_REPLICATION_NET_IP： 6005] /设备 100 #斯威夫特-
环-生成器 container.builder 添加 z 区
-STORAGE_LOCAL_NET_IP_1： 6001
[RSTORAGE_REPLICATION_NET_IP： 6004] /设备 100 #斯威夫特-
```

```
环-生成器 object.builder 添加 z 区-STORAGE_LOCAL_NET_IP_1:
6000 [RSTORAGE_REPLICATION_NET_IP: 6003] /设备 100
```

请注意

如果你不想使用专用的网络进行复制，则必须省略可选的 *STORAGE_REPLICATION_NET_IP* 参数。

例如，如果一个存储节点上 IP 10.0.0.1 第 1 区有一个分区，该存储节点具有从复制网络地址 10.0.1.1。此分区的挂载点是 /srv/node/sdb1，和 /etc/rsyncd.conf 中的路径 /srv/节点/，该设备将是 sdb1，命令如下：

```
# 斯威夫特-环-生成器 account.builder 添加
z1-10.0.0.1:6002R10.0.1.1:6005/ sdb1 100 #斯威夫特-环-
生成器 container.builder 添加
z1-10.0.0.1:6001R10.0.1.1:6004/ sdb1 100 #斯威夫特-环-
生成器 object.builder 添加
z1-10.0.0.1:6000R10.0.1.1:6003/ sdb1 100
```

请注意

如果你假设在一个节点，每个区域的五个区，从 1 开始区。对于每个额外的节点，递增 1 区。

7. 验证每个环的环内容：

```
# 斯威夫特-环-生成器 account.builder
# 斯威夫特-环-生成器 container.builder
# 斯威夫特-环-生成器 object.builder
```

8. 平衡指环王：

```
# 斯威夫特-环-生成器 account.builder 再平衡
# 斯威夫特-环-生成器 container.builder 再平衡
# 斯威夫特-环-生成器 object.builder 再平衡
```

请注意

再平衡环，可以花些时间。

9. Account.ring.gz,container.ring.gz 和 object.ring.gz 文件复制到每个代理服务器和存储节点中 swift。

10. 确保 swift 用户拥有的所有配置文件：

```
# chown-R 斯威夫特： 斯威夫特 swift
```

11. 启动的代理服务并将其配置为在系统引导时启动：

```
# 服务 openstack-斯威夫特-代理启动
# chkconfig openstack-斯威夫特-代理服务器上
```

在存储节点上启动服务

现在，环文件在每个存储节点上，您可以启动的服务。在每个存储节点上，运行以下命令：

```
# 在服务 \ openstack-斯威夫特-对象 openstack-斯威夫特-对象-复制器 openstack-swiftobject-updater openstack-斯威夫特-对象-审计师 \ openstack-斯威夫特-容器 openstack-斯威夫特-容器-复制器 openstackswift-容器-updater openstack-斯威夫特-容器-审计师 \ openstack-斯威夫特-帐户 openstack-斯威夫特-帐户-复制 openstack-swiftaccount-收割者 openstack-斯威夫特-帐户-核数师 ；做 \ 服务 $service 开始 ；chkconfig $service 上 ；做
```

请注意

要立即开始迅速的所有服务，请运行命令：

```
# 斯威夫特 init 所有开始
```

要知道更多关于 swift init 命令，运行：

```
$ 男人斯威夫特 init
```

验证安装成功

你可以从代理服务器或对标识服务具有访问权限的任何服务器运行这些命令。

1. 请确保您的凭据 admin openrc.sh 文件中正确设置和源：

```
$ 源管理-openrc.sh
```

2. 运行下面的 **swift** 命令：

```
$ swift stat
帐户： AUTH_11b9758b7049476d9b48f7a91ea11493
容器： 0
```

```
对象： 0
字节数： 0
内容类型： 文本 ; charset = utf-8
X-时间戳：1381434243.83760
X-反式-Id：txdcdd594565214fb4a2d33 0052570383
X-放-时间戳：1381434243.83760
```

3. 运行以下 **swift** 的命令来上传文件到一个容器。如果需要本地创建的文件 test.txt 和 test2.txt 的测试文件。

```
$ 快速上传 myfiles test.txt
$ 快速上传 myfiles test2.txt
```

4. 运行以下的 **swift** 命令，从 myfiles 容器下载所有文件：

```
$ 斯威夫特下载 myfiles test2.txt [标题 0.267s、总 0.267s、0.000s MB/s]
test.txt [标题 0.271s、 总 0.271s、0.000s MB/s]
```

添加另一个代理服务器

提供额外的可靠性和您的群集的带宽，您可以添加代理服务器。您可以设置一个额外的代理节点同样的方式，您设置了代理服务器的第一个节点，但与额外的配置步骤。

你有两个以上代理服务器之后，您必须加载平衡它们 ；（什么客户端用来连接到您的存储） 您存储终结点也会更改。你可以选择从不同的负载平衡策略。例如，您可以使用 DNS 轮循或软件或硬件负载平衡器 （就像磅） 前面的两个代理。你可以然后你存储 URL 指向负载平衡器，配置初始代理节点和完成这些步骤以添加代理服务器。

1. 更新的 memcache 服务器列表中的 /etc/swift/proxy-server.conf 文件添加的代理服务器。如果你运行多个 memcache 服务器，使用这种模式为每个代理服务器的配置文件中的多个之所以列表：

```
10.1.2.3:11211,10.1.2.4:11211
```

```
[过
滤器：
缓存]
使用
= 蛋：
```

```
斯威
大特
#me
mca
che
mem
cac
he_
ser
ver
s =
PRO
XY_
LOC
AL_
NET
_IP:
112
11
```

2. 复制到所有节点，包括新的代理节点环信息。同时，还要确保环信息获取到的所有存储节点。

3. 你同步所有节点后，请确保该管理员有钥匙在 swift 和环文件的所有权是正确的.

接下来的步骤

你 OpenStack 环境现在包括对象存储。你可以启动一个实例或将更多的服务添加到您的环境，在下面的章节.

11。添加业务流程服务

表的内容

使用业务流程模块来创建云计算资源使用一种叫做热的模板语言。集成的项目名称是热。

业务流程服务概述

业务流程服务提供基于模板的业务流程描述云应用程序中通过运行 OpenStack API 调用来生成运行的云计算应用程序。该软件将 OpenStack 其他核心组件集成到一个文件模板系统。模板使您可以创建大多数 OpenStack 资源类型如实例，浮动 IPs、 卷、 安全组、 用户和等等。此外，还提供一些更先进的功能，如实例的高可用性，实例自动缩放）和嵌套堆栈。通过提供与其他 OpenStack 核心项目非常紧密集成，所有 OpenStack 核心项目可以都得到一个较大的用户基础。

该服务使部署整合与业务流程服务直接或通过自定义的插件。

业务流程服务包括以下组件：

- 热命令行客户端。CLI，与热 api 来运行 AWS CloudFormation Api 进行通信。最终开发人员也可以直接使用业务流程 REST API。
- 热 api 组件。提供了通过 RPC 将他们发送到热引擎处理 API 请求 OpenStack 本机 REST API。
- 热-api-cfn 组件。提供 AWS 查询 API，与 AWS CloudFormation 兼容，通过 RPC 将他们发送到热引擎处理 API 请求。
- 热引擎。编排模板的发射和回 API 使用者提供事件。

安装业务流程服务

1. 控制器节点上安装业务流程模块：

```
# yum 安装 openstack-热-api openstack-热-引擎 \ openstack-热-api-cfn
```

2. 在配置文件中，指定数据库的业务流程服务存储的数据的位置。这些例子与热用户的控制器节点上使用一个 MySQL 数据库。*HEAT_DBPASS* 替换数据库用户的密码：

```
# openstack config--集 /etc/heat/heat.conf \
数据库连接 mysql://heat:HEAT_DBPASS@控制器烘
```

3. 使用你以前设置以 root 身份登录并创建热的数据库用户的密码：

```
$ mysql-u 根-p mysql >创建数据
库热 ；
mysql >授予所有权限上 heat.* 到 'heat'@'localhost' \ 确定由 'HEAT_DBPASS';
mysql >授予所有权限上 heat.* 到 'heat'@'%' \
由标识 'HEAT_DBPASS';
```

4．创建的热服务表：

```
# 苏-s /bin/sh-c"热-管理 db_sync"热
```

请注意

忽略 DeprecationWarning 错误。

5．创建热用户的业务流程服务可以使用身份服务进行身份验证。使用服务租客，并给用户管理员角色：

```
$ 重点用户创建 — — 名称 = 热 — — 通过 =HEAT_PASS \
— — 电子邮件 =heat@example.com
$ 重点用户-角色添加 — — 用户 = 热--租客 = 服务 — — 角色 = 管理员
```

6．编辑 /etc/heat/heat.conf 文件，以更改 [keystone_authtoken] 和 [ec2authtoken] 节将凭据添加到业务流程服务：

```
[keystone_authtoken] auth_host =控制器 auth_port = 35357
auth_protocol = http auth_uri = http://控制器: 5000 /v 2.0
admin_tenant_name = 服务 admin_user = 热 admin_password = HEAT_PASS

[] ec2authtoken
auth_uri = http://控制器: 5000 /v 2.0
```

7．与标识服务注册的热和 CloudFormation Api，以便其他 OpenStack 服务可以找到这些 Api。注册服务和指定终结点：

```
$ 重点服务创建 — — 名称 = 热 — — 类型 = 业务流程。
— — 描述 ="业务流程"
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk ' / 业务流程 / {打印
$2}'） \
— — publicurl = http: / /控制器: 8004 /v1/%\ (tenant_id\) s。
— — internalurl = http: / /控制器: 8004 /v1/%\ (tenant_id\) s。
— — adminurl = http: / /控制器: 8004 /v1/%\ (tenant_id\) s
$ 重点服务创建 — — 名称 = 热 cfn — — 类型 = cloudformation。
```

```
— — 描述 ="业务流程 CloudFormation"
$ 重点创建终结点。
— — 服务 id = $（重点服务列表 | awk ' / cloudformation / {打印
$2}'） \
— — publicurl = http: / /控制器： 8000 /v1 \
— — internalurl = http: / /控制器： 8000 /v1 \
— — adminurl = http: / /控制器： 8000 /v 1
```

8. 创建的 heat_stack_user 角色。

此角色用于创建业务流程模块的用户的默认角色。

运行以下命令来创建 heat_stack_user 角色：

```
$ 重点角色创建 — — 名称 heat_stack_user
```

9. 配置的元数据和 waitcondition 的服务器 Url。

运行下面的命令来修改 /etc/热/heat.conf 文件的 [默认] 部分：

```
# openstack config--集 /etc/heat/heat.conf \
默认 heat_metadata_server_url http://10.0.0.11： 8000
# openstack config--集 /etc/heat/heat.conf \ 默认
heat_waitcondition_server_url http://10.0.0.11： 8000 /v1/
waitcondition
```

请注意

该示例使用控制器 （10.0.0.11） 的 IP 地址而不是控制器主机名称由于我们的示例体系结构不包括 DNS 设置。请确保实例可以解析控制器主机名称，如果您选择的 Url 中使用它。

10. 启动热 api、热-api-cfn 和热发动机服务并将它们配置为在系统引导时启动：

```
# 服务 openstack-热-api 的开始
# 服务 openstack-热-api-cfn 开始
# 服务 openstack 热机启动
# chkconfig openstack 热 api 上
# chkconfig openstack-热-api-cfn 上
# chkconfig openstack 热引擎上
```

验证业务流程服务的安装

要验证的业务流程服务安装并正确配置，请确保您的凭据都正确设置了演示 openrc.sh 文件中。源文件，如下：

```
$ 源演示-openrc.sh
```

业务流程模块使用模板来描述堆栈。要了解有关的模板语言，请参阅模板指南热开发人员文档中.

使用以下内容的测试 stack.yml 文件中创建一个测试模板：

```
$ NET_ID = $(新星网列表 | awk ' / 演示网 / {打印 $2}')
$ 热堆栈创建-f 测试-stack.yml。
-P"ImageID = cirros-0.3.2-x86_64 ; 用户名 = $NET_ID"testStack
+--------------------------------------+------------+--------------------
+----------------------+ |id |stack_name |stack_status |creation_time
|+--------------------------------------+------------+------------------
--
+----------------------+
|477d96b4-d547-4069-938d-32ee990834af |testStack |CREATE_IN_PROGRESS |
2014-04-06T15:11:01Z |
+--------------------------------------+------------+--------------------
+----------------------+
```

验证堆栈与**热堆栈列表**命令已成功创建：

```
$ 热堆栈列表
+--------------------------------------+------------+-----------------
+----------------------+ |id |stack_name |stack_status |creation_time
|+--------------------------------------+------------+---------------
--
+----------------------+
|477d96b4-d547-4069-938d-32ee990834af |testStack |CREATE_COMPLETE |
2014-04-06T15:11:01Z |
+--------------------------------------+------------+-----------------
+----------------------+
```

接下来的步骤

你 OpenStack 环境现在包括业务流程。你可以启动一个实例或将更多的服务添加到您的环境，在下面的章节.

12。添加遥测模块

表的内容

遥测提供监测和计量 OpenStack 云计算框架。它也被称为是测云项目。

遥测

遥测模块：

- 高效地收集有关 CPU 和网络成本的计量数据。
- 收集的数据，通过监测从服务发送的通知或通过轮询的基础设施。
- 配置收集的数据，满足各种操作的类型。访问和插入通过 REST API 的计量数据。
- 扩展框架，以收集自定义使用数据的附加插件插件
- 生产签署计量的邮件不会被否定。

该系统由下列基本组件组成：

- 计算代理 （测云-代理-计算）。运行在每个计算节点和

资源利用率统计的民意调查。在未来，可能有其他类型的代理，但现在我们将专注于创建计算代理。

- 中央代理 （测云-代理-中部）。在中央管理服务器轮询资源利用率统计资源不依赖于实例或计算节点上运行。

- 一位收藏家 （测云 — — 收集器）。在要监视消息队列 （通知和计量数据来自代理） 的一个或多个中央管理服务器上运行。通知邮件的处理和变成了计量的消息和送回消息总线中使用相应的主题。遥测消息写入到数据存储而无需修改。

- 报警通告程序 （测云-报警-[通告程序]）。在允许设置警报基于阈值评价为样本的集合的一个或多个中央管理服务器上运行。
- 一种数据存储区。数据库能够处理 （从一个或多个收集器实例） 并行写入和读取 （从 API 服务器上）。

- API 服务器 （测云 api）。运行在一个或多个中央管理服务器可从数据存储区提供对数据的访问上。

这些服务通过使用标准 OpenStack 消息传递总线进行通信。只收集器和 API 服务器可以访问的数据存储区。

安装遥测模块

遥测提供 API 服务，提供了一个收集器和一系列的不同的代理。您可以在计算节点的节点上安装这些代理之前，必须使用此过程在控制器节点上安装核心组件。

1. 控制器节点上安装遥测服务：

```
# yum 安装 openstack-测云-api openstack-测云-收藏家 \
openstack-测云-通知 openstack-测云-中央 openstack-测
云-报警 \ python ceilometerclient
```

2. 遥测服务使用一个数据库来存储信息。在配置文件中指定数据库的位置。示例使用 MongoDB 数据库上的控制器节点：

```
# yum 安装 mongodb 服务器 mongodb
```

请注意

默认情况下 MongoDB 配置为： 在创建几个 1 GB 文件 lib/mongodb/杂志/目录来支持数据库的日记功能。

如果你需要尽量减少分配支持数据库的日记功能，然后将 smallfiles 配置键设置为 true，在 /etc/mongodb.conf 配置文件中的空间。此配置可以减少每个日志文件的大小为 512 MB。

Smallfiles 配置密钥的详细信息请参阅 MongoDB 文档在 http://docs.mongodb.org/manual/ 参考/配置-选项 / #smallfiles .

详细的步骤完全禁用数据库的日记功能的说明请参阅 http://docs.mongodb.org/manual/tutorial/manage-journaling/ .

3. 配置 MongoDB，使它在控制器管理 IP 地址上侦听。编辑 /etc/mongodb.conf 文件，并修改 bind_ip 键：

```
bind_ip = 10.0.0.11
```

4. 启动 MongoDB 服务器，并将其配置为在系统引导时启动：

```
# 服务的 mongod 开始
# chkconfig mongod 上
```

5. 创建的数据库和一个测云数据库用户：

```
# 蒙哥 —— 主机控制器--eval ' db
=
db.getSiblingDB("ceilometer
") ；db.addUser ({用户："测云"、
pwd："CEILOMETER_DBPASS"，角
色： ["读写"，"dbAdmin"]})'
```

6. 遥测将服务配置为使用数据库：

```
# openstack config--集
/etc/ceilometer/ceilometer.conf \ 数据库连接
mongodb： / /
测云： @控制器 CEILOMETER_DBPASS：27017
/ceilometer
```

7. 你必须定义一个密钥用来作为遥测服务节点之间共享的秘密。使用 **openssl** 来生成一个随机的令牌并将其存储在配置文件中：

```
# CEILOMETER_TOKEN = $（openssl 兰德-十六进制 10）
# 回声 $CEILOMETER_TOKEN # openstack config--设置
```

```
/etc/ceilometer/ceilometer.conf 出版商 metering_secret
$CEILOMETER_TOKEN
```

8. 配置 Qpid 访问：

```
# openstack config--集 /etc/ceilometer/ceilometer.conf \
默认 rpc_backend ceilometer.openstack.common.rpc.impl_qpid
```

9. 创建测云用户遥测服务所使用的身份服务进行身份验证。使用服务租客，并给用户管理员角色：

```
$ 重点用户创建 —— 名称 = 测云 —— 通过 =CEILOMETER_PASS -
电子邮件 =ceilometer@example.com $重点用户-角色添加 — —
用户 = 测云 — — 租客 = 服务 — — 角色 = 管理员
```

10. 遥测将服务配置为通过身份验证的身份服务。

    在 /etc/测云/ceilometer.conf 文件中的 auth_strategy 值设置为重点：

```
# openstack config--集 /etc/ceilometer/ceilometer.conf \
默认 auth_strategy 基石
```

11. 将凭据添加到遥测服务的配置文件：

```
# openstack config--集
/etc/ceilometer/ceilometer.conf \
keystone_authtoken auth_host 控制器#
openstack config--集
/etc/ceilometer/ceilometer.conf \
keystone_authtoken admin_user 测云#
openstack config--集
/etc/ceilometer/ceilometer.conf \
keystone_authtoken admin_tenant_name 服务
# openstack config--设置
/etc/ceilometer/ceilometer.conf \
keystone_authtoken auth_protocol http #
openstack config--设置
/etc/ceilometer/ceilometer.conf \
keystone_authtoken auth_uri http://控制器:
5000 # openstack config--设置
/etc/ceilometer/ceilometer.conf \
keystone_authtoken admin_password
CEILOMETER_PASS # openstack config--设置
/etc/ceilometer/ceilometer.conf \
service_credentials os_auth_url http://控
```

```
制器： 5000 /v 2.0# openstack config--集
/etc/ceilometer/ceilometer.conf \
service_credentials os_username 测云
# openstack config--集
/etc/ceilometer/ceilometer.conf \
service_credentials os_tenant_name 服
务# openstack config--集
/etc/ceilometer/ceilometer.conf \
service_credentials os_password
CEILOMETER_PASS
```

12. 寄存器遥测服务与标识服务以便其他 OpenStack 服务可以找到它。使用**重点**命令来注册该服务并指定的终结点：

```
$ 重点服务创建 – – 名称 = 测云 – – 类型 = 计量。
– – 描述 ="遥测"
$ 重点创建终结点。
– – 服务 id = $（重点服务列表 | awk '/ 计量 / {打印 $2}'）。
– – publicurl = http: / /控制器： 8777 \
– – internalurl = http: / /控制器： 8777 \
– – adminurl = http: / /控制器： 8777
```

13. 开始 openstack-测云-api，openstack-测云-中央，openstack-测云-集热器，和服务并将它们配置为在系统引导时启动：

```
# 服务 openstack-测云-api 的开始
# 服务 openstack-测云-通知开始
# 服务 openstack-测云-中央开始
# 服务 openstack-测云 – – 收集器启动
# 服务 openstack-测云-报警-计算器开始
# 服务 openstack 测云报警通知程序的开始
# chkconfig openstack 测云 api 上
# chkconfig openstack 测云通知上
# chkconfig openstack 测云中央上
# chkconfig openstack 测云器上
# chkconfig openstack-测云-报警-计算器上
# chkconfig openstack-测云-报警-通告程序上
```

安装的计算代理的遥测

遥测提供 API 服务，提供了一个收集器和一系列的不同的代理。此过程详细介绍如何安装在计算节点运行的代理。

1. 在计算节点上安装遥测服务：

```
# yum 安装 openstack 测云计算 python-ceilometerclient
pythonpecan
```

2. 在 /etc/nova/nova.conf 文件中设置以下选项：

```
# openstack config--集
/etc/nova/nova.conf 默认 \
instance_usage_audit True #
openstack config--集
/etc/nova/nova.conf 默认 \
instance_usage_audit_period 小时#
openstack config--默认设置
/etc/nova/nova.conf \
notify_on_state_change
vm_and_task_state
```

请注意

Notification_driver 选项是多值的选项，哪个 openstack 配置不能正确设置。请参阅有一段叫"OpenStack 软件包"[19].

编辑 /etc/nova/nova.conf 文件，并将以下行添加到 [默认] 部分：

```
[默认]...notification_driver =
nova.openstack.common.notifier.rpc_notifier
notification_driver =
ceilometer.compute.nova_notifier
```

3. 重新启动计算服务：

```
# 服务 openstack-新星-计算重新启动
```

4. 你必须设置你以前定义的机密密钥。遥测服务节点共享此密钥为共享的秘密：

```
# openstack config--集的 /etc/ceilometer/ceilometer.conf 出版商 \
metering_secret CEILOMETER_TOKEN
```

5. 配置 QPid 访问：

```
# openstack config--集的 /etc/ceilometer/ceilometer.conf
默认 qpid_hostname 控制器
```

6．添加身份服务凭据：

```
# openstack config--集
/etc/ceilometer/ceilometer.conf \
keystone_authtoken auth_host 控制器#
openstack config--集
/etc/ceilometer/ceilometer.conf \
keystone_authtoken admin_user 测云#
openstack config--设置
/etc/ceilometer/ceilometer.conf \
keystone_authtoken admin_tenant_name 服务
# openstack config--设置
/etc/ceilometer/ceilometer.conf \
keystone_authtoken auth_protocol http #
openstack config--设置
/etc/ceilometer/ceilometer.conf \
keystone_authtoken admin_password
CEILOMETER_PASS # openstack config--设置
/etc/ceilometer/ceilometer.conf \
service_credentials os_username 测云#
openstack config--集
/etc/ceilometer/ceilometer.conf \
service_credentials os_tenant_name 服务#
openstack config--设置/等/测云
/ceilometer.conf \ service_credentials
os_password CEILOMETER_PASS # openstack
config--集
/etc/ceilometer/ceilometer.conf \
service_credentials os_auth_url http://控
制器： 5000 /v 2.0
```

7．启动服务并将其配置为在系统引导时启动：

```
# openstack 测云计算服务的开始
# chkconfig openstack 测云计算上
```

配置为遥测图像服务

1．若要检索图像样本，您必须配置图像服务总线向其发送通知

运行以下命令：

```
# openstack-config--设置 /etc/glance/glance-api.conf 默认
notification_driver 消息传递
# openstack config--集的 /etc/glance/glance-api.conf 默认
rpc_backend qpid
```

2．重新启动图像服务，与他们新的设置：

```
# 服务 openstack-眼-api 重新启动
# 服务 openstack-眼-注册表重新启动
```

添加遥测数据块存储服务代理

1．要检索量的样品，你必须配置块存储服务来发送通知到公车

控制器和卷节点上运行以下命令：

```
# openstack config--设置 /etc/cinder/cinder.conf 默认
control_exchange 煤渣# openstack config--设置
/etc/cinder/cinder.conf 默认 notification_driver
cinder.openstack.common.notifier.rpc_notifier
```

2．重新启动数据块存储服务，与他们的新设置。

控制器在节点上：

```
# 服务 openstack-煤渣-api 重新启动
# 服务 openstack-煤渣-调度程序重新启动
```

卷在节点上：

```
# 服务 openstack 煤渣卷重新启动
```

对象存储为配置服务遥测

1．来检索对象存储统计信息，遥测服务需要访问的对象存储与 ResellerAdmin 的作用。为 os_tenant_name 房客这个角色给你 os_username 用户：

```
$ 重点角色创建 – – 名称 = ResellerAdmin
+----------+----------------------------------+
|属性 |值 |
+----------+----------------------------------+
|id |462fa46c13fd4798a95a3bfbe27b5e54 |
```

```
|名称 |ResellerAdmin |
+----------+----------------------------------+

$ 重点用户-角色添加 – – 租客服务--用户测云。
– – 角色 462 fa46c13fd4798a95a3bfbe27b5e 54
```

2. 您还必须添加对象存储处理传入和传出流量遥测中间件的。将这些行添加到 /etc/swift/proxy-server.conf 文件：

```
[筛选器：测云]使用 = 蛋：测云
#swift
```

3. 该同一文件的管道参数添加测云：

```
[管道：主]管道 = 运行状况检查缓存 authtoken keystoneauth 测云代理服务器
```

4. 重新启动其新的设置与服务：

```
# 服务 openstack-斯威夫特-代理服务器重新启动
```

验证遥测的安装

若要测试的遥测安装，下载图像，从图像服务，和使用**测云**命令来显示使用情况统计信息。

1.使用测云仪列表命令来测试对遥测的访问：

```
$ 测云仪-列表

+------------+-------+-------+----------------------------------
----
+---------+----------------------------------+
|名称 |类型 |单位 |资源 ID |用户
ID |项目 ID |
+------------+-------+-------+----------------------------------
----
+---------+----------------------------------+
|图像 |衡量 |图像 |acafc7c0-40aa-4026-9673-b879898e1fc2 |没有一个
```

```
|efa984b0a914450e9a47788ad330699d |
|image.size |衡量 |B |acafc7c0-40aa-4026-9673-b879898e1fc2 |没有一
个
|efa984b0a914450e9a47788ad330699d |
+------------+-------+-------+-------------------------------
----
+---------+----------------------------------+
```

2．下载图像，从图像服务：

```
$ 看一眼图像下载"cirros-0.3.2-x86_64"> cirros.img
```

3．调用测云仪 `list` 命令的再一次验证下载已被检测到并通过遥测存储：

```
$ 测云仪-列表

+----------------+-------+-------+-----------------------------
--------
+---------+----------------------------------+
|名称 |类型 |单位 |资源 ID |
用户 ID |项目 ID
|+----------------+-------+-------+-----------------------------
---------
+---------+----------------------------------+
|图像 |衡量 |图像 |acafc7c0-40aa-4026-9673-b879898e1fc2 |
没有人 |efa984b0a914450e9a47788ad330699d | |image.download |三角洲
|B |acafc7c0-40aa-4026-9673-b879898e1fc2 |
没有人 |efa984b0a914450e9a47788ad330699d |
|image.serve |三角洲 |B |acafc7c0-40aa-4026-9673-b879898e1fc2 |
没有人 |efa984b0a914450e9a47788ad330699d |
|image.size |衡量 |B |acafc7c0-40aa-4026-9673-b879898e1fc2 |
没有人 |efa984b0a914450e9a47788ad330699d |
+----------------+-------+-------+-----------------------------
--------
+---------+----------------------------------+
```

4.你现在可以为各种仪表使用情况统计信息：

```
$ 测云统计-m image.download-p
60+--------+--------------------+--------------------+-------
+------------+------------+-----------+------------+----------
```

```
+----------------------------+----------------------------+
|时期 |期间开始 |期间结束 |计数 |民
|最大值 |总和 |Avg |持续时间 |持续时间开始
|持续时间结束 |
+--------+--------------------+--------------------+-------
+------------+------------+------------+------------+----------
+----------------------------+----------------------------+
|60 |2013-11-18T18:08:50 |2013-11-18T18:09:50 |1 |13167616.0
|13167616.0 |13167616.0 |13167616.0 |0.0 |2013-11-18T18:09:05。
334000 |2013-11-18T18:09:05.334000 |
+--------+--------------------+--------------------+-------
+------------+------------+------------+------------+----------
+----------------------------+----------------------------+
```

接下来的步骤

你 OpenStack 环境现在包括遥测。你可以启动实例或将更多的服务添加到您的环境在前面的章节。

13。添加数据库服务

表的内容

使用*数据库模块*创建云数据库资源。集成的项目名称是*宝库*.

警告

这一章是进展中的工作。它可能包含不正确的信息，并会经常更新。

数据库服务概述

数据库服务提供可扩展和可靠的云资源调配功能，这两个关系和非关系数据库引擎。用户可以快速、 轻松地利用数据库功能没有处理复杂的管理任务的负担。云用户和数据库管理员可以调配和管理多个数据库实例，根据需要。

数据库服务提供的高性能水平，资源隔离，并自动执行复杂的管理任务，如部署、 配置、 修补、 备份、 恢复和监测。

过程流示例。 这里是一个高级过程流示例使用数据库服务：

1. 管理员设置基础设施：

    a. OpenStack 管理员安装数据库服务。

    b. 她创建一个图像，对于每种类型的数据库管理员想要有 （一个 MySQL，另一个用于 MongoDB，等等）。

    c. OpenStack 管理员更新数据存储区使用新的图像，使用**宝库-管理**命令。

2. 最终用户使用的数据库服务：

    a. 现在的基本基础设施的设置，最终用户可以创建一个宝库实例（数据库） 每当用户想使用**宝库创建**命令。

b. 最终用户通过使用**收藏列表**命令来获取实例的 ID，然后使用**宝库显示*实例 Id***命令来获取的 IP 地址获取的宝库实例的 IP 地址。

c. 最终用户现在可以访问使用典型的数据库访问命令的宝库实例。MySQL 的例子：

```
$ mysql-u 对象-pmypass-h trove_ip_address mydb
```

组件： 数据库的服务包括以下组件：

- python troveclient 命令行客户端。与 troveapi 组件进行通信的 CLI。
- 宝库 api 组件。提供支持 JSON 来调配和管理宝库实例 OpenStack 本机 RESTful API。
- 宝库指挥服务。在主机上运行和接收来自想要更新信息的主机上的客人实例消息。
- 宝库-任务管理器服务。仪器的支持资源调配的情况下，管理的生命周期的情况下，并在实例上执行操作的复杂系统流动。
- 宝库 guestagent 服务。在客人实例中运行。管理并执行对数据库本身操作。

安装数据库服务

此过程将在控制器节点上安装数据库模块。

先决条件。 本章假定你已经有了工作 OpenStack 环境与在至少安装了以下组件： 计算、 图像服务身份。

在控制器上安装数据库模块：

1. 安装所需的包：

```
# yum 安装 openstack 宝库 python troveclient
```

2. 准备 OpenStack：

a. 管理员 openrc.sh 源文件。

```
$ 源 ~/admin-openrc.sh
```

b. 创建一个计算使用身份服务进行身份验证的宝库用户。使用服务租客和给用户管理员角色：

```
$ 重点用户创建 – – 名称 = 宝库 – – 通过 =TROVE_PASS \
– – 电子邮件 =trove@example.com
$ 重点用户-角色添加 – – 用户 = 宝库 – – 租客 = 服务 – – 角色 = 管理员
```

3．编辑下面的配置文件，采取以下操作为每个文件：

- trove.conf
- 宝库 taskmanager.conf
- 宝库 conductor.conf

a．编辑每个文件的 [默认] 节和为 OpenStack 服务 Url，日志记录和邮件的配置和 SQL 连接设置适当的值：

```
[默认] log_dir = /var/log/trove
trove_auth_url = http://控制器: 5000 /v
2.0 nova_compute_url = http://控制器: 8774
/v 2 cinder_url = http://控制器: 8776 /v
1 swift_url = http://控制器: 8080/v1/AUTH_
sql_connection =
mysql://trove:TROVE_DBPASS @控制器/trove
notifier_queue_hostname =控制器
```

b．设置这些配置项来配置要使用的 Qpid 消息代理的数据库模块：

```
# openstack config--集 /etc/trove/trove-api.conf \
默认 rpc_backend qpid
# openstack config--集 /etc/trove/trove-taskmaster.conf \
默认 rpc_backend qpid
# openstack config--集 /etc/trove/trove-conductor.conf \
默认 rpc_backend qpid # openstack config--集 /etc/trove/trove-api.conf
默认 \ qpid_hostname 控制器# openstack config--集
/etc/trove/trove-taskmaster.conf 默认 \ qpid_hostname 控制器#
openstack config--默认设置 /etc/trove/trove-conductor.conf \
qpid_hostname 控制器
```

4．编辑 api paste.ini 文件的 [筛选器: authtoken] 部分，使它与匹配的清单如下所示：

```
[筛选器:
authtoken]
auth_host
=控制器
```

```
auth_port
= 35357
auth_prot
ocol =
http
admin_use
r = 宝库
admin_pas
sword =
ADMIN_PAS
S
admin_tok
en =
ADMIN_TOK
EN
admin_ten
ant_name =
服务
signing_d
ir =
/var/cach
e/trove
```

5. 编辑 trove.conf 文件，所以它包括适当默认数据存储和网络标签的正则表达式的值，如下所示：

```
[默认] default_datastore = mysql......#
Config 选项为显示的 IP 地址，nova 发放
add_addresses = True network_label_regex =
^ NETWORK_LABEL$......
```

6. 编辑的宝库 taskmanager.conf 文件，所以它包含如下所示连接到 OpenStack 计算服务所需的适当的服务凭据：

```
[默认]......
# 交谈的新星通过 novaclient 配置选项。
# 这些选项是用于在您的重点配置管理员用户。# 它代理的收到来自用户将发送到通过此
admin 用户凭据设置，# 基本上像客户端通过代理令牌新星的令牌。
nova_proxy_admin_user = admin nova_proxy_admin_pass = ADMIN_PASS
nova_proxy_admin_tenant_name = 服务......
```

7. 准备的宝库 admin 数据库：

```
$ mysql-u 根-p mysql >创建数据库宝库 ; mysql >授予所有权限
在 trove.* 到 trove@'localhost' 确定的 'TROVE_DBPASS';
```

```
mysql >授予所有权限在 trove.* 到 trove@'%' 确定的
'TROVE_DBPASS';
```

8. 准备数据库服务：

a. 初始化数据库：

```
# 苏-s /bin/sh-c"宝库-管理 db_sync"宝库
```

b. 创建一个数据存储。您需要创建一个单独的数据存储为每种类型的数据库，您想要使用，例如，MySQL，MongoDB，卡桑德拉。此示例显示如何创建一个数据存储为一个 MySQL 数据库：

```
# 苏-s /bin/sh-c"宝库-管理 datastore_update mysql '"宝库
```

9. 创建一个宝库图像。

创建您想要使用，例如，MySQL，MongoDB，卡桑德拉的数据库的类型的图像。

这幅图像必须有宝库客人安装代理，并且它必须配置为连接到您的 OpenStack 环境的 troveguestagent.conf 文件。要正确配置的宝库 guestagent.conf 文件，在您使用来构建您的映像客人实例上执行以下步骤：

- 将以下行添加到宝库-guestagent.conf：

```
rpc_backend = qpid
qpid_host =控制器
nova_proxy_admin_user =
admin
nova_proxy_admin_pass =
ADMIN_PASS
nova_proxy_admin_tenant_
name = 服务
trove_auth_url = http://
控制器: 35357/v2.0
```

10. 更新数据存储区，使用新的图像，使用**宝库-管理**命令。

此示例显示了如何创建一个 MySQL 5.5 数据存储：

```
# 宝库-管理--config-file=/etc/trove/trove.conf
datastore_version_update。
mysql mysql 5.5 mysql glance_image_ID mysql-服务器-5.5 1
```

11. 你必须注册数据库模块与标识服务，以便其他 OpenStack 服务可以找到它。注册服务和指定的终结点：

```
$ 重点服务创建 – – 名称 = 宝库 – – 类型 = 数据库。
– – 描述 ="OpenStack 数据库服务"
$ 重点创建终结点。
– – 服务 id = $（重点服务列表 | awk ' / 宝库 / {打印 $2}'）。
– – publicurl = http: / /控制器: 8779 /v1.0/%\ (tenant_id\) s。
– – internalurl = http: / /控制器: 8779 /v1.0/%\ (tenant_id\) s。
– – adminurl = http: / /控制器: 8779 /v1.0/%\ (tenant_id\) s
```

12. 启动数据库服务，并将它们配置为在系统引导时启动：

```
# 服务 openstack-宝库-api 的开始
# 服务 openstack-宝库-任务管理器启动
# 服务 openstack-宝库-导体的开始
# chkconfig openstack 宝库 api 上
# chkconfig openstack-宝库-任务管理器上
# chkconfig openstack 宝库导体上
```

验证数据库服务的安装

要验证的数据库服务安装并正确配置，请尝试执行宝库的命令：

1. 演示 openrc.sh 源文件。

```
$ 源 ~/demo-openrc.sh
```

2. 检索宝库实例列表：

```
$ 宝库列表
```

您应该看到类似于这样的输出：

```
+----+------+-----------+-------------------+--------+----------
-+-------+
|id |名称 |数据存储 |datastore_version |状态 |flavor_id |大小 |
+----+------+-----------+-------------------+--------+----------
-+-------+
+----+------+-----------+-------------------+--------+----------
-+-------+
```

3. 假设您已创建的数据库，你想要的并更新了数据存储区使用的图像类型的图像，您现在可以创建一个宝库实例 （数据库）。要执行此操作，请使用宝库**创建**命令。

此示例演示了您如何创建 MySQL 5.5 的数据库：

```
$ 宝库创建名称 2 – – 大小 = 2 – – 数据库 =数据库名称 \
– – 用户用户：密码– – datastore_version mysql 5.5。
– – 数据存储 mysql
```

14。启动实例

表的内容

实例是一个 OpenStack 规定在计算节点的虚拟机。本指南说明如何启动一个最小的实例，使用 *CirrOS* 图像添加到您的环境中 "配置图像服务"[35] 章 5 章。在这些步骤中，您在您的控制器节点或任何与适当的 OpenStack 客户端库的系统上使用命令行界面 (CLI)。若要使用仪表板，请参阅 *OpenStack 用户指南*。

启动实例使用 OpenStack 联网 （中子）或遗产网络 (novanetwork)。有关详细信息，请参阅 *OpenStack 用户指南*。

请注意

这些步骤引用在前面的章节中创建的示例组件。您必须调整某些值，例如 IP 地址以匹配您的环境。

启动实例带 OpenStack 网络（子）

要生成一个密钥对

大多数云图像支持*公共密钥身份验证*，而不是传统的用户名 / 密码身份验证。在启动之前一个实例，你必须生成使用 **ssh 凯基**公/私密钥对并将公钥添加到您 OpenStack 环境。

1．源演示租客凭据：

```
$ 源演示-openrc.sh
```

2．生成一个密钥对：

```
$ ssh 凯基
```

3．将公共密钥添加到您的 OpenStack 环境：

```
$ nova 密钥对 – – 添加 – – 酒吧关键 ~/.ssh/id_rsa.pub 试用密钥
```

请注意

该命令提供无输出。

4.验证公钥的加法：

```
$ nova 密钥对列表
+----------+-------------------------------------------------+
|名称 |指纹 |
+----------+-------------------------------------------------+
|试用密钥 |6 c: 74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

要启动一个实例

要启动一个实例，您必须至少指定风味、 图像名称、 网络、 安全组、 键和实例名称。

1.一种风味指定虚拟资源的分配配置文件，包括处理器、 内存和存储。

列表中可用的口味：

```
$ nova 的味道-列表
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
|ID |名称 |Memory_MB |磁盘 |短暂 |交换 |Vcpu 可 |
RXTX_Factor |Is_Public |
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
|1 |m1.tiny |512 |1 |0 | |1 |1.0
|True |
```

```
|2 |m1.small |2048 |20 |0 | |1 |1.0
|True |
|3 |m1.medium |4096 |40 |0 | |2 |1.0
|True |
|4 |m1.large |8192 |80 |0 | |4 |1.0
|True |
|5 |m1.xlarge |16384 |160 |0 | |8 |1.0
|True |
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
```

您的第一个实例使用的 `m1.tiny` 风味。

请注意

你还可以引用香精的 `id`。

2.请列出可用的图像：

```
$ nova 图像列表
+--------------------------------------+---------------------+--
------
+---------+
|ID |名称 |状态 |
服务器 |
+--------------------------------------+---------------------+--
------
+---------+
|acafc7c0-40aa-4026-9673-b879898e1fc2 |cirros-0.3.2-x86_64 |活动 |
     |
+--------------------------------------+---------------------+--
------
+---------+
```

您的第一个实例使用的 `cirros-0.3.2-x86_64` 图像。

3. 列出可用的网络：

```
$ 中子网列表
+--------------------------------------+----------
+------------------------------------------------------+
|id |名称 |子网
```

```
                                    |
+--------------------------------------+----------
+-----------------------------------------------------+
|3c612b5a-d1db-498a-babb-a4c50e344cb1 |演示网
|20bcd3fd-5785-41feac42-55ff884e3180 192.168.1.0/24 |
|9bce64a3-a963-4c05-bfcd-161f708042d1 |ext 网
|b54a8d85-b434-4e85a8aa-74873841a90d 203.0.113.0/24
|+--------------------------------------+----------
+-----------------------------------------------------+
```

您的第一个实例使用演示网租客网络。然而，你必须引用此网络使用 ID 而不名称。

4. 列出可用的安全组：

```
$ nova secgroup-列表
+--------------------------------------+---------+-------------+
|Id |名称 |描述 |
+--------------------------------------+---------+-------------+
|ad8d4ea5-3cad-4f7d-b164-ada67ec59473 |默认 |默认 |
+--------------------------------------+---------+-------------+
```

您的第一个实例使用默认安全组。默认情况下，该安全组实现一个防火墙，阻止对实例的远程访问。如果你想要允许远程访问到您的实例，启动它，然后配置远程访问.

5. 启动该实例：

替换的演示网租客网络 ID 为 *DEMO_NET_ID* 。

```
$ nova 引导 — — 风味 m1.tiny — — 图像 cirros-0.3.2-x86_64 — — nic 的用
户名 =DEMO_NET_ID \ — — 安全组默认 — — 关键名称演示关键演示实例 1
+--------------------------------------
+------------------------------------------------------------+
|属性 |价值
                            |
+--------------------------------------
+------------------------------------------------------------+
|OS-DCF:diskConfig |手动
| |OS-EXT-AZ:availability_zone |新星
| |OS-EXT-STS:power_state |0
                            |
```

```
|OS-EXT-STS:task_state |调度
                            |
|OS-EXT-STS:vm_state |建筑
                            |
|OS-SRV-USG:launched_at |-
                            |
|OS-SRV-USG:terminated_at |-
                            |
|accessIPv4 |
                            |
|accessIPv6 |
                            |
|adminPass |vFW7Bp8PQGNo
                            |
|config_drive |
                            |
|创建 |2014-04-09T19:24:27Z
                            |
|风味 |m1.tiny (1)
                            |
|主机 Id |
                            |
|id |
05682b91-81a1-464c-8f40-8b3da7ee92c5 |
|图像 |cirros-0.3.2-x86_64
(acafc7c0-40aa-4026-9673-b879898e1fc2) |
|key_name |试用密钥
                            |
|元数据 |{}
                            |
|名称 |演示实例 1
                            |
|os-扩展-卷: volumes_attached |[]
                            |
|进展 |0
                            |
|security_groups |默认
```

```
                              |
|状态 |生成
                              |
|tenant_id |7cf50047f8df4824bc76c2fdf66d11ec
                              |
|更新 |2014-04-09T19:24:27Z
                              |
|user_id |0e47686e72114d7182f7569d70c519c9
                              |
+--------------------------------------
+-------------------------------------------------------------+
```

6.请检查您的实例的状态：

```
$ nova 列表
+--------------------------------------+----------------+--------
+------------+-------------+-------------------------+ |ID |名称 |状态 |
任务
状态 |电源状态 |网络 |
+--------------------------------------+----------------+--------
+------------+-------------+-------------------------+
|05682b91-81a1-464c-8f40-8b3da7ee92c5 |演示实例 1 |活动 |-
|运行 |演示网 = 192.168.1.3 |
+--------------------------------------+----------------+--------
+------------+-------------+-------------------------+
```

状态更改从生成到主动当您的实例完成后生成过程。

若要访问您的实例使用一个虚拟控制台

- 获得您的实例的*虚拟网络计算 (VNC)*会议 URL，并从 web 浏览器访问它：

```
$ nova get-vnc-控制台演示实例 1 novnc
+-------
+-----------------------------------------------------------------
---------------------
+
|类型 |Url
                    |
+-------
```

```
+--------------------------------------------------------------------
--------------------
+ |novnc |http://controller:6080/vnc_auto.html? 令牌 =
2f6dd985-f906-4bfcb566-e87ce656375b |+-------
+--------------------------------------------------------------------
--------------------
+
```

请注意

如果您的 web 浏览器运行的主机上，不能解决的*控制器*主机名，您可以替换*控制器*管理接口的 IP 地址在您的控制器节点。

CirrOS 图像包括传统的用户名/密码身份验证，并提供这些凭据登录提示符。后登录到 CirrOS，我们建议您验证使用 **ping** 的网络连接.

验证演示网房客网关：

```
$ ping-c 4 192.168.1.1PING 192.168.1.1 （192.168.1.1）
56(84) 字节的数据。
64 字节从 192.168.1.1： icmp_req = 1 ttl = 64 时间 = 0.357 ms
64 字节从 192.168.1.1: icmp_req = 2 ttl = 64 时间 = 0.473 ms
64 字节从 192.168.1.1: icmp_req = 3 ttl = 64 时间 = 0.504 ms
64 字节从 192.168.1.1: icmp_req = 4 ttl = 64 时间 = 0.470 ms

---为 192.168.1.1 ping 统计--4 数据包传输，4 收到，0%数据包丢失，时间
2998ms rtt min/平均/最大/mdev = 0.357/0.451/0.504/0.055 女士
```

验证 ext 净外部网络：

```
$ ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
64 字节从 174.143.194.225： icmp_req = 1 ttl = 53 时间 = 17.4 ms
64 字节从 174.143.194.225： icmp_req = 2 ttl = 53 时间 = 17.5 ms
64 字节从 174.143.194.225： icmp_req = 3 ttl = 53 时间 = 17.7 ms
64 字节从 174.143.194.225： icmp_req = 4 ttl = 53 时间 = 17.5 ms

— — openstack.org 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间
3003ms rtt min/平均/最大/mdev = 17.431/17.575/17.734/0.143 女士
```

远程访问您的实例

1. 将规则添加到默认安全组：

a. 允许 *ICMP* (ping)：

```
$ nova secgroup 添加规则默认值为-1-1 icmp 0.0.0.0/0
+-------------+-----------+---------+-----------+-----------
---+
|IP 协议 |从端口 |到端口 |IP 范围 |源组 |
+-------------+-----------+---------+-----------+-----------
---+
|icmp |-1 |-1 |0.0.0.0/0 | |
+-------------+-----------+---------+-----------+-----------
---+
```

b. 允许安全外壳 (SSH) 访问：

```
$ nova secgroup 添加规则默认 tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+-----------
---+
|IP 协议 |从端口 |到端口 |IP 范围 |源组 |
+-------------+-----------+---------+-----------+-----------
---+
|tcp |22 |22 |0.0.0.0/0 | |
+-------------+-----------+---------+-----------+-----------
---+
```

2. 在 ext 净外部网络上创建一个*浮动 IP 地址*：

```
$ 中子 floatingip-创建 ext 网创建新的
floatingip:
+---------------------+----------------------------------------+
|字段 |值 |
+---------------------+----------------------------------------+
|fixed_ip_address | |
|floating_ip_address |203.0.113.102 |
|floating_network_id |9bce64a3-a963-4c05-bfcd-161f708042d1 |
|id |05e36754-e7f3-46bb-9eaa-3521623b3722 |
|port_id | |
|router_id | |
|状态 |下来 |
|tenant_id |7cf50047f8df4824bc76c2fdf66d11ec |
```

```
+--------------------+--------------------------------------+
```

3．与您的实例相关联的浮动 IP 地址：

```
$ 浮动 ip 副教授演示-实例 1 新星 203.0.113.102
```

请注意

该命令提供无输出。

4.检查您的浮动 IP 地址的状态：

```
$ nova 列表
+--------------------------------------+----------------+-------
-
+------------+-------------+-----------------------------------
-----+
|ID |名称 |状态 |任务
状态 |电源状态 |网络 |
+--------------------------------------+----------------+-------
-
+------------+-------------+-----------------------------------
-----+
|05682b91-81a1-464c-8f40-8b3da7ee92c5 |演示实例 1 |活动 |-
|运行 |演示网 = 192.168.1.3，203.0.113.102 |
+--------------------------------------+----------------+-------
-
+------------+-------------+-----------------------------------
-----+
```

5．验证外部网络上使用 **ping** 从控制器节点或任何主机的网络连接：

```
$ ping-c 4 203.0.113.102PING 203.0.113.102 (203.0.113.112)
56(84) 字节的数据。
64 字节从 203.0.113.102： icmp_req = 1 ttl = 63 时间 = 3.18 ms
64 字节从 203.0.113.102： icmp_req = 2 ttl = 63 时间 = 0.981 ms
64 字节从 203.0.113.102： icmp_req = 3 ttl = 63 时间 = 1.06 ms
64 字节从 203.0.113.102： icmp_req = 4 ttl = 63 时间 = 0.929 ms

---203.0.113.102 ping--4 传输的信息包，4 收到的统计，0%数据包丢失，rtt
时间 3002ms 分钟，avg，马克斯，.= 0.929/1.539/3.183/0.951 女士
```

6．访问您使用 SSH 从控制器节点或外部网络上的任何主机的实例：

```
$ ssh cirros@203.0.113.102 不能建立主机"203.0.113.102 （203.0.113.102）"的
```

```
真实性。RSA 密钥指纹是 ed: 05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9。
你确定你想要继续连接 （是/否）？是的警告： 永久性地将 '203.0.113.102' （RSA） 添加到已知主机的列表。
$
```

请注意

如果您的主机不包含在前面的步骤中创建公钥/私钥对，SSH 会提示与 cirros 用户关联的默认密码。

如果您的实例不会启动或似乎工作作为你期望，请参阅 *OpenStack 操作指南*为更多的信息或使用一个许多其他选项寻求援助。我们想要你的工作环境 ！

启动实例与旧版网络（新星网络）

要生成一个密钥对

大多数云图像支持*公共密钥身份验证*，而不是传统的用户名 / 密码身份验证。在启动之前一个实例，你必须生成使用 **ssh 凯基**公/私密钥对并将公钥添加到您 OpenStack 环境。

1. 源演示租客凭据：

```
$ 源演示-openrc.sh
```

2. 生成一个密钥对：

```
$ ssh 凯基
```

3. 将公共密钥添加到您的 OpenStack 环境：

```
$ nova 密钥对 — — 添加 — — 酒吧关键 ~/.ssh/id_rsa.pub 试用密钥
```

请注意

该命令提供无输出。

4.验证公钥的加法：

```
$ nova 密钥对列表
+----------+-------------------------------------------------+
|名称 |指纹 |
+----------+-------------------------------------------------+
```

```
|试用密钥 |6 c: 74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

要启动一个实例

要启动一个实例，您必须至少指定风味、 图像名称、 网络、 安全组、 键和实例名称。

1.一种风味指定虚拟资源的分配配置文件，包括处理器、 内存和存储。

列表中可用的口味：

```
$ nova 的味道-列表
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
|ID |名称 |Memory_MB |磁盘 |短暂 |交换 |Vcpu 可 |
RXTX_Factor |Is_Public |
+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
|1 |m1.tiny |512 |1 |0 | |1 |1.0
|True |
|2 |m1.small |2048 |20 |0 | |1 |1.0
|True |
|3 |m1.medium |4096 |40 |0 | |2 |1.0
|True |
|4 |m1.large |8192 |80 |0 | |4 |1.0
|True |
|5 |m1.xlarge |16384 |160 |0 | |8 |1.0
|True
|+----+-----------+-----------+------+-----------+------+-------
+-------------+-----------+
```

您的第一个实例使用的 `m1.tiny` 风味。

请注意

你还可以引用香精的 `id`。

2.请列出可用的图像：

```
$ nova 图像列表
```

```
+--------------------------------------+--------------------+--
------
+--------+
|ID |名称 |状态 |
服务器
|+--------------------------------------+--------------------+-
-------
+--------+
|acafc7c0-40aa-4026-9673-b879898e1fc2 |cirros-0.3.2-x86_64 |活动 |
     |
+--------------------------------------+--------------------+--
------
+--------+
```

您的第一个实例使用的 `cirros-0.3.2-x86_64` 图像。

3.请列出可用的网络：

请注意

你必须源 `admin` 租客凭据的这一步，然后源演示租客凭据的其余步骤。

```
$ 源管理-openrc.sh
```

```
$ 新星网-列表
+--------------------------------------+----------+-------------
-----+
|ID |标签 |CIDR |
+--------------------------------------+----------+-------------
-----+
|7f849be3-4494-495a-95a1-0f99ccb884c4 |演示网 |203.0.113.24/29 |
+--------------------------------------+----------+-------------
-----+
```

您的第一个实例使用演示网租客网络。然而，你必须引用此网络使用 `ID` 而不名称。

4．列出可用的安全组：

```
$ nova secgroup-列表
+--------------------------------------+---------+-------------+
```

```
|Id |名称 |描述 |
+--------------------------------------+---------+-------------+
|ad8d4ea5-3cad-4f7d-b164-ada67ec59473 |默认 |默认 |
+--------------------------------------+---------+-------------+
```

您的第一个实例使用默认安全组。默认情况下，该安全组实现一个防火墙，阻止对实例的远程访问。如果你想要允许远程访问到您的实例，启动它，然后配置远程访问.

5. 启动该实例：

替换的演示网租客网络 ID 为 *DEMO_NET_ID* 。

```
$ nova 引导 — — 风味 m1.tiny — — 图像 cirros-0.3.2-x86_64 — — nic 的用
户名 =DEMO_NET_ID \ — — 安全组默认 — — 关键名称演示关键演示实例 1
+--------------------------------------
+------------------------------------------------------------+
|属性 |价值
                            |
+--------------------------------------
+------------------------------------------------------------+
|OS-DCF:diskConfig |手动
                            |
|OS-EXT-AZ:availability_zone |新星
                            |
|OS-EXT-STS:power_state |0
                            |
|OS-EXT-STS:task_state |调度
                            |
|OS-EXT-STS:vm_state |建筑
                            |
|OS-SRV-USG:launched_at |-
                            |
|OS-SRV-USG:terminated_at |-
                            |
|accessIPv4 |
                            |
|accessIPv6 |
                            |
```

```
|adminPass |ThZqrg7ach78
                            |
|config_drive |
                            |
|创建 |2014-04-10T00:09:16Z
                            |
|风味 |m1.tiny (1)
                            |
|主机 Id |
| |id |45ea195cc469-43eb-83db-1a663bbad2fc | |图像
|cirros-0.3.2-x86_64
(acafc7c0-40aa-4026-9673-b879898e1fc2) |
|key_name |试用密钥
                            |
|元数据 |{}
                            |
|名称 |演示实例 1
                            |
|os-扩展-卷： volumes_attached |[]
                            |
|进展 |0
                            |
|security_groups |默认
                            |
|状态 |生成
                            |
|tenant_id |93849608fe3d462ca9fa0e5dbfd4d040
| |更新 |2014-04-10T00:09:16Z
| |user_id |8397567baf4746cca7a1e608677c3b23
                            |
+--------------------------------------
+------------------------------------------------------------+
```

6．检查您的实例的状态：

```
$ nova 列表
+--------------------------------------+----------------+-------
-
+------------+-------------+-----------------------+
```

```
|ID |名称 |状态 |任务
状态 |电源状态 |网络 |
+--------------------------------------+----------------+-------
-
+------------+-------------+-----------------------+
|45ea195c-c469-43eb-83db-1a663bbad2fc |演示实例 1 |活动 |-
|运行 |演示网 = 203.0.113.26 |
+--------------------------------------+----------------+-------
-
+------------+-------------+-----------------------+
```

状态更改从生成到主动当您的实例完成后生成过程。

若要访问您的实例使用一个虚拟控制台

- 获得您的实例的*虚拟网络计算 (VNC)*会议 URL，并从 web 浏览器访问它：

```
$ nova get-vnc-控制台演示实例 1 novnc
+-------
+---------------------------------------------------------------------
----------------------
+
|类型 |Url
                    |
+-------
+---------------------------------------------------------------------
----------------------
+ |novnc |http://controller:6080/vnc_auto.html? 令牌 =
2f6dd985-f906-4bfcb566-e87ce656375b |+-------
+---------------------------------------------------------------------
----------------------
+
```

请注意

如果您的 web 浏览器运行的主机上，不能解决的*控制器*主机名，您可以替换*控制器*管理接口的 IP 地址在您的控制器节点。

CirrOS 图像包括传统的用户名/密码身份验证，并提供这些凭据登录提示符。后登录到 CirrOS，我们建议您验证使用 **ping** 的网络连接.

验证演示网网络：

```
$ ping-c 4 openstack.orgPING openstack.org (174.143.194.225)
56(84) 字节的数据。
64 字节从 174.143.194.225： icmp_req = 1 ttl = 53 时间 = 17.4 ms
64 字节从 174.143.194.225： icmp_req = 2 ttl = 53 时间 = 17.5 ms
64 字节从 174.143.194.225： icmp_req = 3 ttl = 53 时间 = 17.7 ms 64 字节
从 174.143.194.225： icmp_req = 4 ttl = 53 时间 = 17.5 ms

— — openstack.org 坪统计--4 传输的信息包，4 收到，0%数据包丢失，时间
3003ms rtt min/平均/最大/mdev = 17.431/17.575/17.734/0.143 女士
```

远程访问您的实例

1．将规则添加到默认安全组：

a．允许 *ICMP* (ping)：

```
$ nova secgroup 添加规则默认值为-1-1 icmp
0.0.0.0/0+-------------+-----------+---------+-----------+--
------------+
|IP 协议 |从端口 |到端口 |IP 范围 |源组 |
+-------------+-----------+---------+-----------+-----------
---+
|icmp |-1 |-1 |0.0.0.0/0 | |
+-------------+-----------+---------+-----------+-----------
---+
```

b．允许安全外壳 (SSH) 访问：

```
$ nova secgroup 添加规则默认 tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+-----------
---+
|IP 协议 |从端口 |到端口 |IP 范围 |源组 |
+-------------+-----------+---------+-----------+-----------
---+
|tcp |22 |22 |0.0.0.0/0 | |
+-------------+-----------+---------+-----------+-----------
---+
```

2．验证外部网络上使用 **ping** 从控制器节点或任何主机的网络连接：

```
$ ping-c 4 203.0.113.26PING 203.0.113.26 (203.0.113.26)
56(84) 字节的数据。
```

```
64 字节从 203.0.113.26： icmp_req = 1 ttl = 63 时间 = 3.18 ms
64 字节从 203.0.113.26： icmp_req = 2 ttl = 63 时间 = 0.981 ms
64 字节从 203.0.113.26： icmp_req = 3 ttl = 63 时间 = 1.06 ms
64 字节从 203.0.113.26： icmp_req = 4 ttl = 63 时间 = 0.929 ms

---203.0.113.26 ping--4 传输的信息包，4 收到的统计，0%数据包丢失，rtt
时间 3002ms 分钟，avg，马克斯，.= 0.929/1.539/3.183/0.951 女士
```

3．访问您使用 SSH 从控制器节点或外部网络上的任何主机的实例：

```
$ ssh cirros@203.0.113.26 不能建立主机"203.0.113.26 （203.0.113.26）"的真实
性。RSA 密钥指纹是 ed： 05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9。
你确定你想要继续连接 （是/否）？是的警告： 永久性地将 '203.0.113.26' （RSA） 添
加到已知主机的列表。
$
```

请注意

如果您的主机不包含在前面的步骤中创建公钥/私钥对，SSH 会提示与 cirros 用户关联的默认密码。

如果您的实例不会启动或似乎工作作为你期望，请参阅 *OpenStack 操作指南*为更多的信息或使用一个许多其他选项寻求援助。我们想要你的工作环境 ！

附录 A.保留用户 Id

在 OpenStack,某些用户 Id 是保留的用于运行特定 OpenStack 服务和自己的特定 OpenStack 文件。这些用户是建立包分配。下表概述了。

表 A.1。保留的用户 Id

名称	描述	ID
测云	OpenStack 测云守护程序	166
煤渣	OpenStack 煤渣守护程序	165
一眼	OpenStack 眼守护程序	161
热	OpenStack 热守护程序	187
基石	OpenStack 重点守护程序	163
中子	OpenStack 中子守护程序	164

新星	OpenStack 新星守护程序	162
斯威夫特	OpenStack 斯威夫特守护程序	160
宝库	OpenStack 宝藏守护程序	未知的 FIXME

每个用户属于一个用户组与用户的名称相同。

附录 B. 社区支持

表的内容

以下资源都是可用来帮助您运行和使用 OpenStack。OpenStack 社区不断提高，并将添加到 OpenStack 的主要特征，但如果你有任何问题，不要犹豫，问。使用以下资源以获得 OpenStack 支持及故障排除您的安装。

文档

可用 OpenStack 文档，请参见 docs.openstack.org .

若要在文档上提供的反馈，加入并使用 < openstack-docs@lists.openstack.org > 邮件列表 OpenStack 文档邮寄列表或报告一个 bug .

下列书籍解释如何安装 OpenStack 云及其相关联的组件：

- *Debian 7.0 安装指南*
- *OpenSUSE 和 SUSE Linux 企业服务器安装指南*
- *红帽企业 Linux，CentOS 和 Fedora 安装指南*
- *Ubuntu 12.04/14.04 （LTS） 安装指南*

下列书籍解释如何配置和运行 OpenStack 云：

- *云管理员指南*
- *配置参考*
- *操作指南*
- *高可用性指南*
- *安全指南*
- *虚拟机映像指南*

下列书籍解释如何使用 OpenStack 仪表板和命令行客户端：

- *API 快速开始*
- *最终用户指南*
- *Admin 用户指南*
- *命令行界面参考*

下面的文档提供借鉴和指导信息 OpenStack Api：

- OpenStack API 的完整参考 (HTML)
- API 的完整参考 (PDF)
- *OpenStack 块存储服务 API v2 参考*
- *OpenStack 计算 API v2 和扩展参考*
- *OpenStack 身份服务 API v2.0 参考*
- *OpenStack 图像服务 API v2 参考*
- *OpenStack 网络 API v2.0 参考*
- *OpenStack 对象存储 API v1 参考*

*培训指南*提供云行政和管理的软件培训。

ask.openstack.org

在设置或测试 OpenStack，你可能会有关于如何完成特定任务的问题，或在一项功能是在哪里不能正常工作的情况。使用 ask.openstack.org 网站问问题并获得答案。当您访问 http://ask.openstack.org 网站时，扫描最近问的问题，看看是否已回答

你的问题。如果没有，问一个新问题。一定要在标题中给出清晰、 简明的概括和描述中尽可能提供尽可能详细的信息。粘贴在你的命令输出或堆栈跟踪、 屏幕快照的链接和可能有用的任何其他信息。

OpenStack 邮件列表

一个伟大的方式来得到的答案和见解是发布您的问题或疑难场景到 OpenStack 邮件列表中。你可以从学习和帮助其他人可能会有类似的问题。要订阅或查看存档，请转到 http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack。你可能感兴趣的具体项目或发展，你可以找到在 wiki 的其他邮寄列表中。所有邮件列表的描述是可在 http://wiki.openstack.org/MailingLists .

OpenStack wiki

OpenStack wiki 包含范围广泛的主题，但某些信息可能很难找到或几页深。幸运的是，维基搜索功能使您能够通过标题或内容进行搜索。如果你搜索的特定信息，如有关联网或新星，你可以找到大量的相关材料。更多是正在添加所有的时间，所以一定要经常回来检查。你可以在任何 OpenStack wiki 页面的右上角找到搜索框。

Launchpad Bug 地区

OpenStack 社区值您的集和测试工作，希望您的反馈意见。要登录一个 bug，你必须注册在 https://launchpad.net/+login 的启动帐户。你可以查看现有的 bug 和报告 bug 据点 Bug 区域中。使用搜索功能来确定是否该 bug 已报道或已定。如果它仍然看起来像你的错误是未报告，填写一个 bug 报告。

一些小贴士：

- 给出清晰、 简明的概括。
- 在描述中尽可能的提供尽可能详细。粘贴在你的命令输出或堆栈跟踪、 屏幕快照的链接和可能有用的任何其他信息。
- 一定要包括软件和包的版本，您使用的尤其是如果您使用一个开发分支，如，"朱诺释放"vs git commit bc79c3ecc55929bac585d04a03475b72e06a3208.
- 任何特定于部署的信息是有帮助的例如是否使用 Ubuntu 14.04 或正在执行多节点的安装。

以下几个据点的 Bug 方面是可用的：

- 的 bug： OpenStack 块存储 （煤渣）
- 的 bug： OpenStack 计算 (nova)
- 的 bug： OpenStack 仪表板 （地平线）
- 的 bug： OpenStack 身份 （重点）
- 的 bug： OpenStack 图像服务 （眼）
- 的 bug： OpenStack 联网 （中子）
- 的 bug： OpenStack 对象存储 (swift)
- 的 bug： 裸露的金属 （讽刺）
- 的 bug： 数据处理服务 (撒哈拉)
- 错误： 数据库服务 （宝库）
- 的 bug： 业务流程 (热)
- 的 bug： 遥测 （测云）
- 的 bug： 队列服务 （马可尼）
- 的 bug： OpenStack API 文档 （api.openstack.org）
- 的 bug： OpenStack 文档 (docs.openstack.org)

OpenStack IRC 频道

OpenStack 社区生活在 #openstack IRC 频道上的 Freenode 网络。你可以挂出来，问的问题，或得到即时的反馈为紧迫和紧迫的问题。要安装一个 IRC 客户端或使用基于浏览器的客户端，请转到 http://webchat.freenode.net/。您还可以使用 Mac OS X http://colloquy.info/） 的国际讨论会，mIRC （Windows， http:// www.mirc.com/） 或 XChat （Linux）。当你在 IRC 频道，想要分享代码或命令的输出时，普遍接受的方法是使用粘贴 Bin。OpenStack 项目有一个在 http://paste.openstack.org。只是粘贴你较多的文字或日志在 web 窗体中，你得到一个 URL，你可以粘贴到该频道。的

OpenStack IRC 频道是 #openstack 上封锁。你可以在 https://wiki.openstack.org/wiki/IRC 中找到所有 OpenStack IRC 频道的列表.

文档反馈

若要在文档上提供的反馈，加入并使用 < openstack-docs@lists.openstack.org > 邮件列表 OpenStack 文档邮寄列表或报告一个 bug .

OpenStack 分发程序包

以下的 Linux 发行版提供 OpenStack 社区支持的软件包：

- **Debian:** http://wiki.debian.org/OpenStack
- **CentOS、软呢帽和红帽企业 Linux:** http://openstack.redhat.com/
- **openSUSE 和 SUSE Linux 企业服务器:** http://en.opensuse.org/Portal:OpenStack
- **Ubuntu:** https://wiki.ubuntu.com/ServerTeam/CloudArchive

术语表

API

应用程序编程接口。

API 端点

守护程序、 工作人员或客户端与通信访问 API 的服务。API 端点可以提供任意数量的服务，如身份验证、 销售数据、 性能指标、 计算 VM 命令、人口普查数据，等等。

数据块存储

OpenStack 核心项目，它使您能够管理卷、 卷快照、 卷类型。数据块存储的项目名称是煤渣。

CirrOS

设计上，如 OpenStack 乌云密布的测试图像用作最小 Linux 发行版。

云控制器节点

一个节点，运行网络、 卷、 API、 调度器和图像服务。每个服务可能分成单独的节点可伸缩性或可用性。

计算

OpenStack 核心项目，提供计算服务。计算服务的项目名称是新星。

计算节点

运行 nova 计算守护进程管理提供范围广泛的服务，例如 web 应用程序的虚拟机实例的节点和分析。

控制器节点

云控制器节点的的替代词。

数据库服务

为这两个关系和非关系数据库引擎提供可扩展和可靠的云数据库作为-一-服务功能的综合的项目。数据库服务的项目名称是宝库。

DOC7

动态主机配置协议。配置设备在连接到网络，以便他们可以在该网络上通信通过使用互联网协议 (IP) 的网络协议。议定书 》 是在哪里 DHCP 客户端请求的配置数据，如客户端-服务器模型、 IP 地址、 默认路由和一个或多个 DNS 服务器地址从 DHCP 服务器中实现的。

DHCP 代理

OpenStack 网络代理，提供了虚拟网络的 DHCP 服务。

终结点

请参阅 API 端点。

外部网络

一个网络段，通常使用例如互联网访问。

防火墙

用来限制主机和/或实施在计算使用 iptables、arptables、ip6tables、和土壤中的节点之间的通信。

平面网络

网络控制器提供了虚拟的网络，使计算服务器进行交互和公共网络。所有机器都必须都有一个公用和专用网络接口。一个扁平的网络是专用网络接口，它由与平经理 flat_interface 选项控制。

浮动 IP 地址

一个 IP 地址，以便实例具有相同的公共 IP 地址每个启动的时间，一个项目可以将 VM 相关联。你创建的浮动 IP 地址池，并将它们分配给实例启动时保持一致维护 DNS 分配的 IP 地址。

网关

IP 地址，通常被分配到路由器上，不同的网络之间传递的网络流量。

泛型接收卸载 (GRO)

某些网络接口驱动程序的功能，将很多小收到的数据包组合成一个大的数据包才运送到内核 IP 堆栈。

虚拟机管理程序

仲裁并控制对实际的底层硬件的虚拟机访问软件。

IaaS

基础设施作为 —— 即服务。IaaS 是资源调配模型在其中一个组织外包数据中心的存储、 硬件、 服务器和网络组件的物理组件。一个服务提供商拥有所有的设备，并负责房屋、 操作和维护它。客户端通常支付在每次使用的基础上。IaaS 是提供云计算服务模式。

ICMP

因特网控制消息协议，由控制消息的网络设备使用。例如， **ping** 使用 ICMP 来测试连接性。

标识服务

OpenStack 核心项目，提供的用户映射到他们可以访问的 OpenStack 服务中央目录。它也注册为 OpenStack 服务终结点。它充当一个共同的身份验证系统。标识服务的项目名称是重点。

图像服务

OpenStack 核心项目，提供了为磁盘和服务器映像的发现、 登记、 和交付服务。图像服务的项目名称是一眼。

实例隧道网络

一个网络段使用例如计算节点和网络节点之间的交通隧道。

接口

提供连接到另一个设备或介质的物理或虚拟设备。

基于内核的虚拟机 （KVM）

OpenStack 支持虚拟机管理程序。

图层-3 (L3) 代理

3 层 （路由） 为提供服务的虚拟网络的 OpenStack 网络代理。

负载平衡器

负载平衡器是属于云帐户的逻辑设备。它用来分发多个后端系统或基于作为其配置的一部分定义的标准的服务之间的工作负载。

逻辑卷管理器 (LVM)

提供了一种分配空间在大容量存储设备上，比传统的分区方案更灵活的方法。

管理网络

用于管理，不可以访问公共互联网网络段。

消息代理

软件包用于提供 AMQP 消息内计算的能力。默认的包是 RabbitMQ。

多主机

遗产 (nova) 网络的高可用性模式。每个计算节点处理 NAT 和 DHCP，并充当网关的所有的 Vm 上它。一个计算节点上的联网故障并不影响在其他计算节点上的 Vm。

网络地址转换 (NAT)

修改 IP 地址信息，同时在运输的过程。计算和网络支持。

网络时间协议 (NTP)

保持一个时钟的节点或主机正确通过一个可信的、准确的时间源与通信的一种方法。

网络

向 OpenStack 计算提供网络连通性的抽象层的核心 OpenStack 项目。联网的项目名称是中子。

对象存储

OpenStack 核心项目，提供最终一致和冗余的存储和检索的固定数字内容。OpenStack 对象存储的项目名称是迅速的。

OpenStack

OpenStack 是一个云操作系统，控制大的计算、 存储和整个数据中心，通过一个仪表板，同时赋予其用户提供通过 web 界面资源让管理员控制所有托管网络资源池。OpenStack 是领有根据 Apache 许可证 2.0 开放源码项目。

业务流程

一个综合的项目，协调多个云计算应用程序为 OpenStack。业务流程的项目名称是热。

插件

提供实际的执行，对于网络 Api，还是对于计算的 Api，根据上下文的软件组件。

混杂模式

原因要将它接收的所有通信都传递到主机，而不是通过仅向它的帧的网络接口。

公共密钥身份验证

使用键，而不是密码的身份验证方法。

Rest 风格

一种 web 服务 API，它使用的休息，或具象状态传输。休息是超媒体系统的体系结构，用于万维网的风格。

作用

假定用户，使他们能够执行特定的一组操作的人格。一个角色包括一组权利和特权。承担这一角色的用户将继承这些权利和特权。

路由器

不同网络间传输网络通信的物理或虚拟网络设备。

安全组

一套的过滤规则应用于计算实例的网络流量。

服务目录

标识服务目录的替代词。

子网

逻辑细分的 IP 网络。

遥测

一个综合的项目，提供了计量和测量设施的 OpenStack。遥测的项目名称是测云。

租客

一组用户，用来隔离对计算资源的访问。一个项目替代的名词。

宝库

OpenStack 项目，它提供对应用程序的数据库服务。

用户

在标识服务，每个用户与一个或多个租户，关联和在计算中，可以与角色、 项目或两者相关联。

虚拟机 (VM)

在虚拟机监控程序上运行的操作系统实例。多个虚拟机可以在同一时间运行在同一台物理主机上。

虚拟网络

虚拟化的网络功能，如交换、 路由、 负载平衡、 和安全使用的虚拟机和叠加组合在物理网络基础设施上的通用术语。

虚拟网络计算 (VNC)

开放源码的 GUI 和 CLI 工具用于 Vm.支持远程控制台访问的计算。

虚拟专用网络 (VPN)

由计算 cloudpipes，专门用于在每个项目的基础上创建 Vpn 的实例的形式提供。

Microsoft Translator

原文

You have to choose a password while configuring the service and later remember to use the same password when accessing it.