Red Hat OpenStack Platform 13

安全强化指南

最佳实践、合规性和安全强化

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 安全强化指南

最佳实践、合规性和安全强化

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南提供有关强化 Red Hat OpenStack Platform 环境安全性的良好实践建议和概念性信息。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

使用直接文档反馈(DDF)功能

使用 **添加反馈** DDF 功能，用于特定句子、段落或代码块上的直接注释。

1. 以 *Multi-page HTML* 格式查看文档。
2. 请确定您看到文档右上角的 **反馈** 按钮。
3. 用鼠标指针高亮显示您想评论的文本部分。
4. 点 **添加反馈**。
5. 在**添加反馈**项中输入您的意见。
6. 可选：添加您的电子邮件地址，以便文档团队可以联系您以讨论您的问题。
7. 点 **Submit**。

第 1 章 安全简介

使用 Red Hat Openstack Platform (RHOSP)提供的工具，在规划中以及操作方面优先考虑安全性，以满足用户对隐私及其数据安全性的期望。未能实施安全标准会导致停机或数据泄露。您的用例可能会受到需要传递审计和合规性流程的法律约束。备注

按照本指南中的说明强化环境的安全性。但是，这些建议不能保证安全性或合规。您必须根据您的环境的唯一要求来评估安全性。

有关强化 Ceph 的详情，请参考数据安全性和强化指南。

1.1. RED HAT OPENSTACK PLATFORM SECURTIY

默认情况下，Red Hat OpenStack Platform (RHOSP) director 使用以下工具和访问控制来创建 overcloud，以提高安全性：

SElinux

SELinux 通过提供需要每个进程对每个操作有显式权限的访问控制，为 RHOSP 提供安全增强。

Podman

Podman 作为容器工具是 RHOSP 的安全选项，因为它不使用需要具有 root 访问权限的进程的客户端/服务器模型。

系统访问限制

您只能使用 director 在 overcloud 部署期间为 heat-admin 创建的 SSH 密钥或您在 overcloud 上创建的 SSH 密钥登录到 overcloud 节点。您不能使用带有密码的 SSH 登录 overcloud 节点，或使用 root 登录 overcloud 节点。

您可以根据机构的需要和信任级别使用以下额外安全功能配置 director：

- 公共 TLS 和 TLS-everywhere
- 硬件安全模块与 OpenStack Key Manager (barbican)集成
- 签名的镜像和加密卷
- 使用工作流执行对密码和 fernet 密钥进行轮转

1.2. 安全区

安全区是共享常见安全问题的通用资源、应用程序、网络和服务器。安全区应共享相同的身份验证和授权要求和用户。

您可能需要重新定义您自己的安全区，以根据云的唯一架构、环境中可接受的信任级别以及标准化要求来更精细。区域及其信任要求可能会因云实例是公共、私有还是混合而有所不同。

部署安全强化型 OpenStack 云（从至少到最受信任）所需的最小区如下：

- **公共区** ：公共区托管面向公共的 API，以及浮动 IP 和 SNAT 等 neutron 外部网络，用于实例的外部连接。此区域是云基础架构的不受信任的区域。它指的是通过公共或您区域以外的网络或 Red Hat OpenStack Platform 部署外部的网络进行云访问。

遍历此区域的任何具有保密性或完整性要求的数据都应使用编译控制进行保护。

- **guest zone** ：guest zone 托管项目网络，无论是 VXLAN 还是 GENEVE。公共云和私有云供应商对实例使用没有严格控制，或者允许不受限制的互联网访问实例。

如果您是私有云提供商，如果实施控制以模拟实例和相关项目，则可将其视为内部和可信网络，以包括底层硬件和物理网络。

- **存储访问区域** ：存储访问区域用于存储管理、监控和集群，以及存储流量本身。

在此网络中传输的数据需要高级别的完整性和机密性。也可能具有强大的可用性要求。

除复制要求外，请勿使此网络可从云外部访问，但存储设备组件以外的存储设备从安全角度来说是敏感的。

- **管理区域** ：管理区域包括 undercloud、主机操作系统、服务器硬件、物理网络和 Red Hat OpenStack Platform director control plane。

管理网络用于系统管理、监控或备份。不要托管此区域上的 OpenStack API 或控制接口。

- admin **zone** ：admin zone 主机用于系统管理、监控或备份的流量。但是，是没有托管 OpenStack API 或控制接口的地方。

1.3. 连接安全区

必须仔细配置跨多个安全区（具有不同信任级别或身份验证要求）的组件。这些连接通常是网络架构的弱点，并且应始终配置为满足所连接任何区域的最高信任级别。在很多情况下，安全控制连接的区应该是主要问题，因为攻击的可能性较高。区域满足点显示额外的攻击服务，并为攻击者将其攻击迁移到部署更敏感的部分的机会。

在某些情况下，OpenStack 操作器可能希望考虑确保集成点比它所在的任何区域更高的标准。根据上面的 API 端点示例，如果这些区域没有完全隔离，则可能会对公共 API 端点为目标。

OpenStack 的设计使得安全区分离比较困难。由于核心服务通常至少跨越两个区域，因此在将安全控制应用到它们时，必须考虑特殊考虑。

1.4. 威胁分类、执行者和攻击向量

大多数类型的云部署、公共、私有或混合都暴露于某种形式的攻击。本节对攻击者进行分类，并总结了每个安全区中潜在的攻击类型。

1.4.1. 威胁者

威胁者是一个抽象的、用于指代您可能需要防御的一类行为的方式。能够使用更加强大的安全控制，这是成功攻击缓解和防止所需的安全控制。安全性是根据要求进行的平衡便利、防御和成本方面的问题。在某些情况下，无法保护云部署免受此处描述的所有威胁执行者。在部署 OpenStack 云时，您必须决定部署和使用的平衡位置。

作为风险评估的一部分，您还必须考虑您存储和任何可访问资源的数据类型，因为这将影响某些参与者。但是，即使您的数据对威胁者没有吸引力，它们也可能很吸引于您的计算资源，例如，参与 botnet 或运行未授权的加密者减法。

- Nation-State Actors - 这是最强大的行为。Nation-state actors 可以使用巨大的资源来进行攻击。他们拥有超越其他任何参与者的功能。在没有严格控制（包括人工和技术）的情况下，很能防御此类的攻击。

- 严重的机构组织 - 这个类代表具有强大能力和经济的攻击者组。他们能够为攻击方法的开发和研究提供大量资金。近年来，一些兴起的组织（例如 Russian Business Network，它是一个大型网络犯罪组织），已证明网络攻击如何成为一种商品。工业间谍通常属于这类严重犯罪组织。

- 高能力组 - 这是指通常不是商业资助的"Hacktivist"类型组织，但会对服务提供商和云操作员造成严重威胁。
- 仅有动机行为的个人 - 这些攻击者包括许多人，如恶意员工、受损的客户或小型工业激发。
- 脚本 Kiddies - 这些攻击者不针对特定的组织，而是运行自动化漏洞扫描和利用漏洞。它们通常看似微不足道，但可能会对一个机构构成声誉风险。

以下实践可帮助缓解上述发现的一些风险：

- 安全更新 - 您必须考虑底层物理基础架构的端到端安全，包括网络、存储和服务器硬件。这些系统需要自己的安全强化实践。对于 Red Hat OpenStack Platform 部署，您应该有一个计划定期测试和部署安全更新。
- 访问管理 - 当授予系统对个人的访问权限时，您应该应用 *最小特权的原则*，并且仅向他们授予实际需要的粒度系统特权。您可以使用 AAA （访问、授权和核算）实践帮助强制执行此策略。这种方法还可以帮助缓解系统管理员中恶意执行者和排字错误的风险。
- 管理内部 - 您可以通过应用谨慎分配基于角色的访问控制（最低访问权限）、在内部接口上使用加密以及使用身份验证/授权安全（如集中式身份管理）来缓解恶意人员的威胁。您还可以考虑额外的非技术选项，例如将职责分离和不定期的作业角色轮转。

1.4.2. 出站攻击和建议风险

对于云部署的潜在外向滥用，应仔细考虑因素。云部署往往会具有大量可用资源；通过黑客或有权权（如恶意员工）在云内建立了点的攻击者，可以将这些资源用于恶意目的。具有计算服务的云为理想的 DDoS 和 brute 强制引擎。由于公共云用户非常无法可靠，因此这个问题对于公共云而言尤其重要，而且可以快速启动大量可处理实例以进行出站攻击。防止方法包括出口安全组、流量检查、入侵检测系统、客户教育和认知以及欺诈和滥用缓解策略。对于可通过或访问公共网络（如互联网）访问的部署，最好检测流程和基础架构，并解决出站 abuse 的问题。

1.5. 支持软件

支配整个红帽解决方案堆栈是安全的软件供应链。红帽安全战略的基石是，这种战略性重要的做法和程序集合的目标是提供具有安全内置前期和长期支持的解决方案。红帽采取的具体步骤包括：

- 保持上游关系和社区参与，以帮助从一开始专注于安全性。
- 根据安全和性能跟踪记录选择和配置软件包。
- 从相关源代码构建二进制文件（而不是只接受上游构建）。
- 应用一系列检查和质量保证工具，以防止大量潜在安全问题和回归问题。
- 数字签名所有已发布的软件包，并通过经过加密验证的分发频道进行发布。
- 提供单一统一的补丁和更新分发机制。基于 OpenStack 的 Red Hat Enterprise Linux 和 KVM 组件也经过通用标准认证。这涉及执行物理站点访问的第三方审核员，以及咨询员工有关遵守良好做法，例如，关于供应链或开发。

此外，红帽还维护了一个专门的安全团队，针对我们的产品分析威胁和漏洞，并通过客户门户网站提供相关建议和更新。这个团队决定哪些问题很重要，而不是与大多数理论问题相关的问题。红帽产品安全团队在维护了专业技术方面，并对与订阅产品关联的上游社区做出了大量贡献。红帽安全公告 （红帽安全公告）的一个主要部分提供了影响红帽解决方案的安全缺陷通知 - 通常会在漏洞首次发布之日提供相关的补丁程序。

第 2 章 DOCUMENTATION

文档应提供 OpenStack 环境的一般描述，并涵盖所有使用的系统（如生产、开发或测试）。记录系统组件、网络、服务和软件通常会提供全面覆盖和考虑安全问题、攻击向量和可能的安全区桥接点的视角。系统清单可能需要捕获临时资源，如虚拟机或虚拟磁盘卷，否则在传统 IT 环境中是持久资源。

2.1. 系统角色和类型

记录环境中使用的角色。通常组成 OpenStack 安装的两种定义型节点类型是：

- **基础架构节点** - 它们运行与云相关的服务，如 OpenStack API 提供程序（如 neutron）、消息排队服务、存储管理、监控、联网和其他服务，以支持云的操作和配置。
- **计算、存储或其他资源节点** - 为云上运行的实例提供计算和存储容量。

2.2. 硬件清单

没有编写文档严格的合规要求的云可能会受益于配置管理数据库(CMDB)。CMDB 通常用于硬件资产跟踪和整个生命周期管理。通过使用 CMDB，组织可以快速识别云基础架构硬件，如计算节点、存储节点或网络设备。CMDB 可以协助识别网络中存在的资产，这些资产可能因为维护不足、不足保护或被取代和忘记而存在漏洞。如果底层硬件支持必要的自动发现功能，OpenStack 调配系统可以提供一些基本的 CMDB 功能。

2.3. 软件清单

与硬件一样，应该记录 OpenStack 部署中的所有软件组件。示例包括：

- 系统数据库，如 MySQL 或 mongoDB
- OpenStack 软件组件，如 Identity 或 Compute
- 在评估库、应用程序或软件中的漏洞影响时，支持组件（如负载均衡器、反向代理、DNS 或 DHCP 服务）是至关重要的、软件组件的权威列表。

第 3 章 加密和密钥管理

设备间通信是严重的安全问题。通过网络进行安全通信方法变得越来越重要，如 Heartbleed 等重要漏洞或更高级的攻击，如 BEAST 和 CRIME。但是，加密只是更大的安全策略的一个部分。端点的威胁意味着攻击者不再需要破坏所使用的加密，但可以在系统处理时查看和操作消息。

本章将回顾有关配置传输层安全(TLS)来保护内部和外部资源的功能，并将调用应有特定关注的特定系统类别。

OpenStack 组件通过各种协议相互通信，并且通信可能涉及敏感数据或机密数据。攻击者可以尝试窃听频道，以便访问敏感信息。因此，所有组件都必须使用安全通信协议相互通信。

3.1. TLS 和 SSL 简介

有些情况下，需要确保 OpenStack 部署中网络流量的机密性或完整性。您通常会使用加密措施（如 TLS 协议）进行配置。在典型的部署中，通过公共网络传输的所有流量都应安全强化，但安全的良好做法预期内部流量也必须得到保护。不足以保护安全区分离。如果攻击者获得对 hypervisor 或主机资源的访问权限，破坏 API 端点或任何其他服务，则他们不能轻松地注入或捕获消息、命令或其他影响云的管理功能。

您应该安全强化所有带有 TLS 的区域，包括管理区服务和服务内通信。TLS 提供了相应的机制，可确保用户与 OpenStack 服务通信以及 OpenStack 服务本身之间的用户身份验证、非回复、机密性和完整性。

由于安全套接字层(SSL)协议中公布的漏洞，请考虑在首选使用 TLS 1.2 或更高版本，并且所有情况下都禁用了 SSL，除非您需要与过时的浏览器或库兼容。

3.2. 公钥基础架构

公钥基础架构(PKI)是一种框架，用于提供加密算法、密码模式和协议来保护数据和身份验证。它由一组系统和流程组成，以确保可以加密流量，同时验证各方的身份。此处描述的 PKI 配置集是 PKIX 工作组开发的互联网工程任务组(IETF)公共密钥基础架构(PKIX)配置文件。PKI 的核心组件是：

- 数字证书 - 签名的公钥证书是具有可验证实体数据的数据结构，其公钥以及一些其他属性。这些证书由证书颁发机构(CA)发布。由于证书由信任的 CA 签名，经过验证后，与实体关联的公钥保证与上述实体关联。用于定义这些证书的最常见标准是 X.509 标准。RFC5280 中详细介绍当前标准的 X.509 v3，并由 RFC6818 更新。CA 签发证书作为证明在线实体身份的机制。CA 通过从证书创建消息摘要并使用其私钥加密摘要来数字签名证书。

- 实体 - 证书主题的用户、进程或系统。端实体将其证书请求发送到注册授权中心(RA)进行批准。若获得批准，MRA 将请求转发到认证认证机构(CA)。认证认证机构会验证请求以及信息是否正确，是否生成证书并签名。然后，这个签名证书会发送到证书存储库。

- 依赖方 - 接收数字签名证书的端点，该证书可以被验证，并引用证书中列出的公钥。依赖方应位于验证链的位置，确保它没有出现在 CRL 中，还必须能够验证证书上的到期日期。

- 证书颁发机构(CA)- CA 是可信实体，由最终方和依赖证书认证政策、管理处理和证书颁发方使用方。

- 注册授权中心(RA)- CA 委派特定管理功能的可选系统，其中包括在由 CA 签发证书前对最终用户进行身份验证的功能。

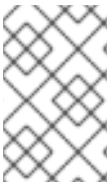
- 证书撤销列表(CRL)- 证书撤销列表(CRL)是已撤销的证书序列号列表。在 PKI 模型中，呈现这些证书的最终实体不应被信任。因为几个原因（如密钥威胁、CA 威胁）会发生吊销。

- CRL 签发者 - 一个可选系统，CA 委派证书撤销列表的发布。

- 证书存储库 - 最终用户实体证书和证书撤销列表存储并查询的位置 - 有时被称为证书捆绑包。

3.3. 认证授权

许多组织拥有拥有自己的认证认证机构(CA)、证书策略和管理，它们用于为内部 OpenStack 用户或服务发布证书。公共安全区是面向互联网的组织还需要被广泛认可的公共 CA 签名的证书。对于管理网络的加密通信，建议不使用公共 CA。相反，建议大多数部署都部署自己的内部 CA。

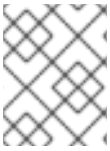


注意

有效的 TLS 依赖于提供 DNS 中域或子域的部署，该部署可以由通配符使用，或者由公共或内部 CA 使用的特定证书问题。为确保 TLS 证书可以有效地验证，对平台服务的访问需要通过这些 DNS 记录。

建议 OpenStack 云架构考虑将独立的 PKI 部署用于内部系统和面向客户的服务。这允许云部署器维护对 PKI 基础架构的控制，并使内部系统请求、签名和部署证书变得更加容易。高级配置可能会将独立的 PKI 部署用于不同的安全区。这使得 OpenStack 操作器能够维护环境的加密分离，确保签发的证书不能被另一个环境识别。

用于在面向互联网的云端点（或客户不应安装标准操作系统提供证书捆绑包外）上支持 TLS 的证书应使用安装在操作系统证书捆绑包中的证书颁发机构进行置备。



注意

创建和签署证书存在管理、策略和技术挑战。除了这里推荐的指导外，云架构师或操作员可能还希望寻求行业领导和供应商的建议。

3.4. 使用 DIRECTOR 配置加密

默认情况下，overcloud 使用未加密的端点作为其服务。这意味着 overcloud 配置需要额外的环境文件，才能为其公共 API 端点启用 SSL/TLS。*高级 Overcloud 自定义指南* 描述了如何配置 SSL/TLS 证书，并将其作为 overcloud 创建过程的一部分包括 ：https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html/advanced_overcloud_customization/sect-enabling_ssltls_on_the_overcloud

3.5. TLS 库

OpenStack 生态系统中的某些组件、服务和应用程序可以配置为使用 TLS 库。OpenStack 中的 TLS 和 HTTP 服务通常使用 OpenSSL 实施，后者具有一个经过 FIPS 140-2 验证的模块。但是，请考虑每个应用程序或服务仍可能会在使用 OpenSSL 库时引入弱点。

3.6. 弃用 TLS 1.0

重要

需要 FedRAMP-authorized 系统才能离开 TLS 1.0。推荐的级别是 1.2，只有需要广泛的兼容性时才可以接受 1.1。如需更多信息，请参阅 https://www.fedramp.gov/assets/resources/documents/CSP_TLS_Requirements.pdf。

对于 Red Hat OpenStack Platform 13 部署，HAProxy 不接受 TLS 1.0 连接，该连接处理 TLS 启用 API 的 TLS 连接。这通过 **no-tlsv10** 选项实现。对于启用了 **InternalTLS** 的 HA 部署，控制器平面上的跨节点流量也会加密。这包括 RabbitMQ、MariaDB 和 Redis 等等。MariaDB 和 Redis 已被弃用 TLS1.0，对

RabbitMQ 的相同弃用应该被上游向后移植。

3.6.1. 检查是否使用了 TLS 1.0

您可以使用 **cipherscan** 来决定部署是否提供 TLS 1.0。Cipherscan 可以从 https://github.com/mozilla/cipherscan 克隆。这个示例输出显示从 **horizon** 收到的结果：

注意

从非生产环境系统运行 **cipherscan**，因为它可能会在首次运行时安装额外的依赖项。

```
$ ./cipherscan https://openstack.lab.local
...............................
Target: openstack.lab.local:443

prio  ciphersuite                protocols  pfs                curves
1     ECDHE-RSA-AES128-GCM-SHA256  TLSv1.2    ECDH,P-256,256bits  prime256v1
2     ECDHE-RSA-AES256-GCM-SHA384  TLSv1.2    ECDH,P-256,256bits  prime256v1
3     DHE-RSA-AES128-GCM-SHA256    TLSv1.2    DH,1024bits         None
4     DHE-RSA-AES256-GCM-SHA384    TLSv1.2    DH,1024bits         None
5     ECDHE-RSA-AES128-SHA256      TLSv1.2    ECDH,P-256,256bits  prime256v1
6     ECDHE-RSA-AES256-SHA384      TLSv1.2    ECDH,P-256,256bits  prime256v1
7     ECDHE-RSA-AES128-SHA         TLSv1.2    ECDH,P-256,256bits  prime256v1
8     ECDHE-RSA-AES256-SHA         TLSv1.2    ECDH,P-256,256bits  prime256v1
9     DHE-RSA-AES128-SHA256        TLSv1.2    DH,1024bits         None
10    DHE-RSA-AES128-SHA           TLSv1.2    DH,1024bits         None
11    DHE-RSA-AES256-SHA256        TLSv1.2    DH,1024bits         None
12    DHE-RSA-AES256-SHA           TLSv1.2    DH,1024bits         None
13    ECDHE-RSA-DES-CBC3-SHA       TLSv1.2    ECDH,P-256,256bits  prime256v1
14    EDH-RSA-DES-CBC3-SHA         TLSv1.2    DH,1024bits         None
15    AES128-GCM-SHA256            TLSv1.2    None                None
16    AES256-GCM-SHA384            TLSv1.2    None                None
17    AES128-SHA256                TLSv1.2    None                None
18    AES256-SHA256                TLSv1.2    None                None
19    AES128-SHA                   TLSv1.2    None                None
20    AES256-SHA                   TLSv1.2    None                None
21    DES-CBC3-SHA                 TLSv1.2    None                None

Certificate: trusted, 2048 bits, sha256WithRSAEncryption signature
TLS ticket lifetime hint: None
NPN protocols: None
OCSP stapling: not supported
Cipher ordering: server
Curves ordering: server - fallback: no
Server supports secure renegotiation
Server supported compression methods: NONE
TLS Tolerance: yes

Intolerance to:
 SSL 3.254          : absent
 TLS 1.0            : PRESENT
 TLS 1.1            : PRESENT
 TLS 1.2            : absent
 TLS 1.3            : absent
 TLS 1.4            : absent
```

在扫描服务器时，Cipherscan 公告对特定 TLS 版本的支持，这是它将协商的最高 TLS 版本。如果目标服务器正确跟随 TLS 协议，它将响应相互支持的最高版本，这可能低于最初公告的 Cipherscan。如果服务器继续使用该特定版本与客户端建立连接，则不能接受到该协议版本。如果没有建立连接（通过指定版本或任何较低版本），那么该版本的协议将被视为存在。例如：

```
Intolerance to:
 SSL 3.254          : absent
 TLS 1.0            : PRESENT
 TLS 1.1            : PRESENT
 TLS 1.2            : absent
 TLS 1.3            : absent
 TLS 1.4            : absent
```

在这个输出中，**TLS 1.0** 和 **TLS 1.1** 的容错性报告为 **PRESENT**，这意味着无法建立连接，并且 Cipherscan 无法连接，同时公告这些 TLS 版本的支持。因此，最好在扫描的服务器上不启用这些协议（以及任何较低）版本。

3.7. 加密算法、密码模式和协议

您应该只使用 TLS 1.2。其他版本（如 TLS 1.0 和 1.1）容易受到多个攻击的影响，被许多政府机构和管制的行业禁止。在您的环境中应禁用 TLS 1.0。TLS 1.1 可能用于广泛的客户端兼容性，但在启用此协议时谨慎操作。只有在存在强制兼容性要求时才启用 TLS 版本 1.1，如果您了解相关的风险，才启用 TLS 版本 1.1。由于存在多个公共漏洞，不得使用所有版本的 SSL （前身为 TLS）。

当您使用 TLS 1.2 并控制客户端和服务器时，密码套件应限制为 **ECDHE-ECDSA-AES256-GCM-SHA384**。如果不同时控制端点，且正在使用 TLS 1.1 或 1.2，则更多常规 **HIGH:!aNULL:!eNULL:!DES:!3DES:!SSLv3:!TLSv1:!CAMELLIA** 是合理的密码选择。

注意

本指南不作为加密参考，不说明您应该在 OpenStack 服务中启用或禁用的特定算法或密码模式。

3.8. TLS PROXIES 和 HTTP 服务

OpenStack 端点是向公共网络上的最终用户和管理网络上的其他 OpenStack 服务提供 API 的 HTTP 服务。目前，您可以使用 TLS 加密外部请求。要在 Red Hat OpenStack Platform 中配置，您可以在 HAproxy 之后部署 API 服务，该服务能够建立和终止 TLS 会话。

如果软件终止无法提供性能不足，硬件加速器可能会成为替代选项。这种方法需要在平台上进行额外的配置，而并非所有硬件负载均衡器都可能与 Red Hat OpenStack Platform 兼容。务必要注意任何所选 TLS 代理处理的请求大小。

3.9. PERFECT FORWARD SECRECY

为完美转发保密配置 TLS 服务器需要仔细规划密钥大小、会话 ID 和会话票据。此外，对于多服务器部署，共享状态也是重要的考虑因素。实际部署可能会考虑启用此功能来提高性能。这可以以安全加固的方式完成，但需要特别考虑关键管理。此类配置超出了本指南的范围。

3.10. 使用 BARBICAN 管理 SECRET

OpenStack Key Manager (barbican)是 Red Hat OpenStack Platform 的机密管理器。您可以使用 barbican API 和命令行来集中管理 OpenStack 服务使用的证书、密钥和密码。barbican 目前支持以下用例：

- **对称加密密钥** - 用于块存储(cinder)卷加密、临时磁盘加密和 Object Storage (swift)对象加密。
- **非对称密钥和证书** - glance 镜像签名和验证，Octavia TLS 负载均衡。

在本发行版本中，barbican 提供与 cinder、swift、Octavia 和 Compute (nova)组件的集成。例如，您可以为以下用例使用 barbican：

- **支持加密的卷** - 您可以使用 barbican 来管理您的 Cinder 加密密钥。此配置使用 LUKS 对连接到您的实例的磁盘进行加密，包括引导磁盘。密钥管理方面由用户执行。
- **Glance Image Signing** - 您可以配置镜像服务(glance)，以验证上传的镜像没有被篡改。镜像首先使用存储在 barbican 中的密钥签名，镜像会在每次使用前进行验证。

如需更多信息，请参阅 Barbican 指南： https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/manage_secrets_with_openstack_key_manager/

第 4 章 身份和访问管理

Identity 服务(keystone)为云用户提供身份验证和授权，它是安全考虑的一个重要组件。

身份服务可以直接提供最终用户身份验证，或者可以配置为使用外部身份验证方法来符合组织的安全策略和要求。

4.1. 身份验证

身份验证是任何现实世界 OpenStack 部署不可或缺的一部分。请仔细考虑该系统设计的这一方面。本主题的完整处理超出了本指南的范围，但以下部分将介绍一些关键主题。

在大多数情况下，身份验证是确认身份的过程，即用户实际声明的身份。熟悉的示例是在登录系统时提供用户名和密码。

OpenStack Identity service (keystone)支持多种身份验证方法，包括用户名和密码、LDAP 和其他外部身份验证方法。身份验证成功后，身份服务为用户提供了用于后续服务请求的身份验证令牌。

传输层安全性(TLS)使用 X.509 证书在服务和个人之间提供身份验证。虽然 TLS 的默认模式仅是服务器端的身份验证，但您应该考虑使用证书进行客户端身份验证，因为它在美国政府标准中强制使用。

4.2. 无效的登录尝试

Identity Service (keystone)可以在重复失败的登录尝试后限制对帐户的访问。重复登录尝试的模式通常是对暴力攻击的指示。这种类型的攻击在公共云部署中更为普遍。您还可以使用在配置次数失败的登录尝试后阻止帐户的外部身份验证系统缓解这个问题。然后，帐户只能通过进一步的管理干预解锁。

检测技术也可用于缓解损坏。检测涉及频繁检查访问控制日志，以识别未经授权的访问帐户尝试。可能的补救包括查看用户密码的强度，或通过防火墙规则阻止攻击的网络源。您可以在 keystone 服务器中添加限制连接数量的防火墙规则；这有助于降低攻击的有效性。

此外，检查帐户活动对于不常见的登录时间和可疑操作（如禁用帐户）非常有用。

4.3. 授权

身份服务支持组和角色的概念。用户在组拥有角色列表时从属于组。OpenStack 服务引用尝试访问该服务的用户的角色。OpenStack 策略强制器中间件考虑与每个资源关联的策略规则，然后考虑用户的 group/roles 和关联，以确定是否允许访问所请求的资源。

4.4. 建立表单访问控制策略

在配置角色、组和用户之前，您应该记录您所需的 OpenStack 安装访问控制策略。该策略应与组织的任何法规或法律要求保持一致。将来的对访问控制配置的修改应当与正式策略一致。该策略应包括用于创建、删除、禁用和启用帐户以及为帐户分配特权的条件和流程。定期检查策略，并确保配置符合批准的策略。

4.5. SERVICE AUTHORIZATION

云管理员必须定义具有每个服务 admin 角色的用户。此服务帐户提供相应的服务来验证用户。

计算和对象存储服务可以配置为使用身份服务来存储身份验证信息。身份服务支持 TLS 的客户端身份验证，可能启用。TLS 客户端身份验证除了提供用户标识的可靠性外，还提供额外的身份验证因素。当用户名和密码被泄露时，它会降低未授权访问的风险。但是，对于每次部署都可能无法正常工作的用户，还有

额外的管理开销和成本来为用户发布证书。

云管理员应保护敏感配置文件不受未授权的修改。这可以使用强制访问控制框架（如 SELinux）进行配置，包括 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 文件和 X.509 证书。

使用 TLS 的客户端身份验证需要向服务发布证书。这些证书可以由外部或内部证书颁发机构进行签名。OpenStack 服务默认检查对可信 CA 的证书签名请求的有效性，如果签名无效或者 CA 不可信，连接将失败。云部署器可能会使用自签名证书；在这种情况下，必须禁用有效检查，或者证书应标记为可信。要禁用自签名证书验证，请在 **/etc/nova/api.paste.ini** 文件的 **[filter:authtoken]** 部分中设置 **insecure=False**。此设置还禁用其他组件的证书。请注意，对于容器化服务，确切的文件路径可能有所不同。

4.6. 令牌

用户通过身份验证后，会生成一个令牌，以便授权和访问 OpenStack 环境。令牌可以具有变量生命周期，但到期的默认值是一小时。推荐的到期值应设置为较低的值，允许有足够的时间让内部服务完成任务。如果令牌在任务完成前过期，云可能会变得无响应或停止提供服务。计算服务使用期间的加快时间示例将是将磁盘镜像传输到 hypervisor 以便进行本地缓存所需的时间。

令牌通常在身份服务响应的更大上下文结构中传递。这些响应也提供各种 OpenStack 服务目录。每一服务均列出其名称、内部访问端点、管理和公共访问。身份服务支持令牌撤销。此清单作为 API 撤销令牌，列出已撤销的令牌和单独的 OpenStack 服务，以便缓存令牌来查询已撤销的令牌，并将它们从缓存中移除，并将相同的功能附加到已缓存的已撤销令牌列表中。Fernet 令牌是 Red Hat OpenStack Platform 13 中唯一支持的令牌。

4.6.1. Fernet 令牌

Fernet 令牌现在是默认的令牌提供程序。Fernet 是明确设计为在 API 令牌中使用的安全消息传递格式。Fernet 令牌是非持久性（无需持久存于数据库）、轻量级（在 180 到 240 字节）并且减少了运行云所需的操作开销。身份验证和授权元数据捆绑到消息打包的载荷中，然后作为 Fernet 令牌进行加密并签名（在 180 到 240 字节）。

与 UUID、PKI 和 PKIZ 令牌不同，Fernet 令牌不需要持久性。keystone 令牌数据库不再受到身份验证的副作用。在使用 Fernet 令牌时，不再需要从令牌数据库修剪已过期的令牌。由于 Fernet 令牌是非持久性的，因此不必复制它们。只要每个 keystone 节点共享相同的存储库，就可以在节点间立即创建和验证 Fernet 令牌。

与 PKI 和 PKIZ 令牌相比，Fernet 令牌的大小较小；通常保持在 250 字节限值下。对于 PKI 和 PKIZ 令牌，较大的服务目录将导致令牌长度较长。Fernet 令牌没有此模式，因为加密载荷的内容保持为最小值。

有关 Fernet 令牌和轮转 Fernet 密钥的详情，请参考 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/deploy_fernet_on_the_overcloud/。

4.7. KEYSTONE 域

Keystone 域是高级别的安全界限，逻辑上对项目、用户和组进行分组。因此，它们可用于集中管理所有基于 keystone 的身份组件。随着帐户域的引入，服务器、存储和其他资源现在可以逻辑地分组到多个项目（以前称为租户），后者本身可以分组到一个类似于 master 帐户的容器中。此外，可以在帐户域中管理多个用户，并分配对每个项目不同的角色。

Identity V3 API 支持多个域。不同域的用户可能会在不同的身份验证后端中表示。它们甚至可能具有不同的属性，它们必须映射到一组角色和特权，这些属性在策略定义中使用，以访问各种服务资源。

如果规则只能指定对 admin 用户和属于项目的用户的访问权限，则映射可能很简单。在其他情况下，云管理员可能需要批准每个项目的映射例程。

特定于域的身份验证驱动程序允许使用域特定的配置文件为多个域配置 Identity 服务。启用驱动程序并设置域特定配置文件位置，发生在 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 文件的 **[identity]** 部分中。例如：

```
[identity]
domain_specific_drivers_enabled = True
domain_config_dir = /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/
```

任何没有域特定配置文件的域将使用主 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 文件中的选项。

4.8. 使用 LDAP 进行身份验证

您可以使用外部身份提供程序(IdP)向 OpenStack 服务供应商(SP)进行身份验证。在本例中，Service Providers 是 OpenStack 云提供的服务。

对于身份验证和授权服务，OpenStack 身份模型将外部身份验证数据库视为单独的 keystone 域。每个外部身份验证机制都与 keystone 域关联，支持多个共存域。您可以使用角色为外部域中的用户授予对云中资源的访问权限；这种方法也适用于跨域多项目部署。这种方法还对每个组件策略也有影响，因为并非所有 OpenStack 角色都可以在策略中映射到外部验证的用户。例如，如果外部身份验证数据库中的用户需要管理访问权限，则通常还需要额外的配置，并且需要与 **admin** 域中的 **admin** 用户类似。

外部身份验证提供了一种方式，可以使用现有凭证来访问使用一组凭证在多个授权云中提供的多个端点（如服务器、卷和数据库）的云资源，而无需多次置备额外的身份或登录。凭据由用户的身份提供程序维护。

身份服务可以将用户凭据存储在 SQL 数据库中，也可以使用与 LDAP 兼容的目录服务器。身份数据库可能独立于其他 OpenStack 服务使用的数据库，以减少存储的凭据的风险。

当您使用用户名和密码进行身份验证时，Identity 不会强制对密码强度、过期或身份验证尝试失败的策略。希望强制实施更强大的密码策略的组织应考虑使用身份扩展或外部身份验证服务。

LDAP 简化了身份验证集成到机构的现有目录服务和用户帐户管理流程中的集成。OpenStack 中的身份验证和授权策略可能会被委派给其他服务。典型的用例是寻求部署私有云的组织，并且已经在 LDAP 系统中拥有员工和数据库用户。使用这个作为身份验证授权，对 Identity 服务的请求会被委派给 LDAP 系统，然后根据其策略授权或拒绝。身份验证成功后，Identity 服务会生成用于访问授权服务的令牌。

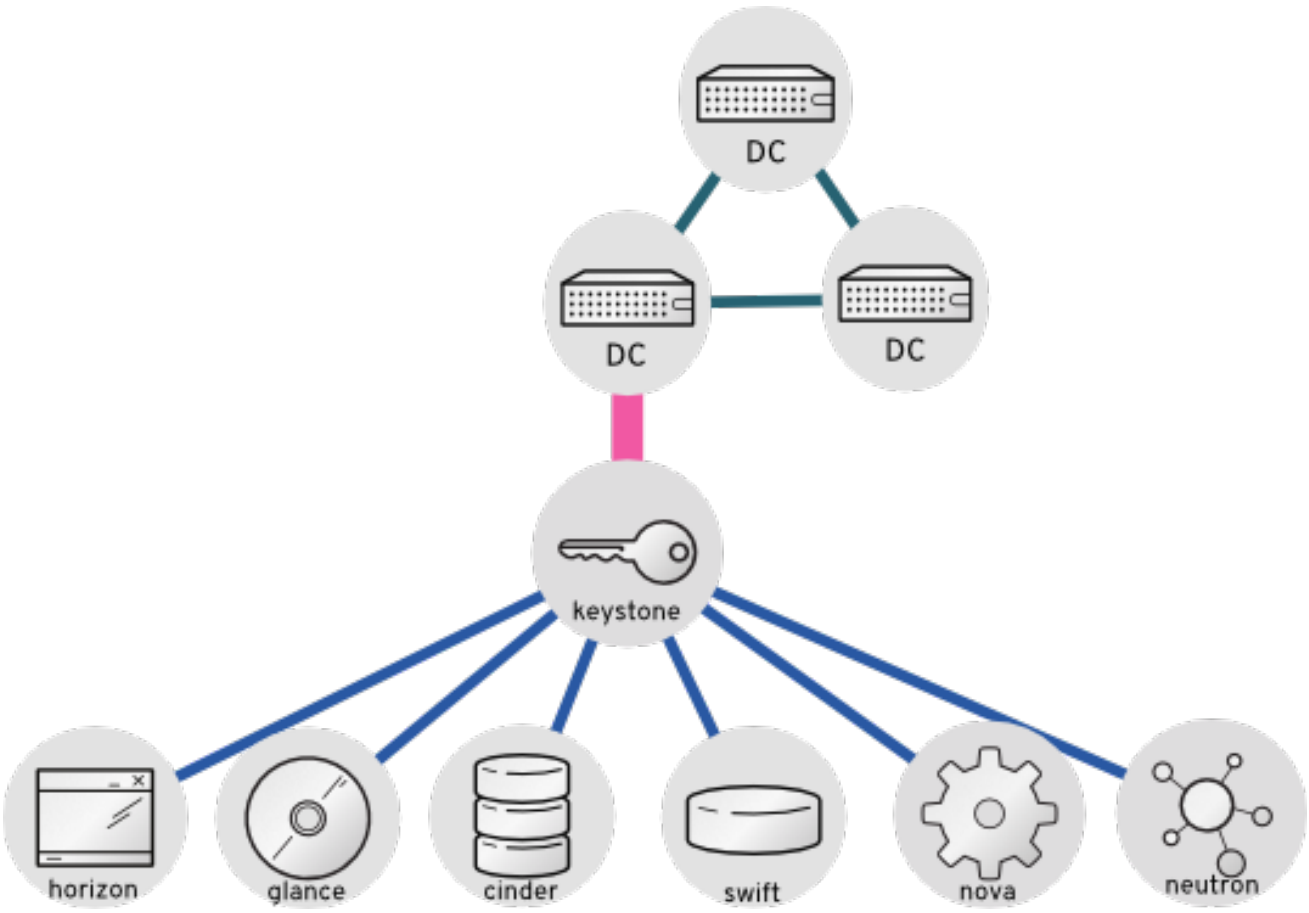
请注意，如果 LDAP 系统为用户定义了属性，如 admin、finance 和 HR 等，则这些属性必须映射到身份内的角色和组中，以供各种 OpenStack 服务使用。**/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** 文件将 LDAP 属性映射到 Identity 属性。

4.9. 与基于 LDAP 的服务集成

身份服务(keystone)可以验证存储在基于 LDAP 的服务中的用户帐户，如 Microsoft Active Directory Domain Services (AD DS)和 Red Hat Identity Management (IdM)。在这种情况下，keystone 对 LDAP 用户数据库身份验证具有只读访问权限，并保持对分配给经过身份验证的用户的 authZ 权限的管理。authZ 功能（权限、角色、项目）仍然由 keystone 执行，其中使用 keystone 管理工具将权限和角色分配给 LDAP 帐户。

4.9.1. LDAP 集成如何工作

在下图中，keystone 使用加密的 LDAPS 连接连接到 Active Directory 域控制器。当用户登录到 horizon 时，keystone 会收到提供的用户凭证，并将其传递给 authZ 的 Active Directory。



有关将 OpenStack 与 AD DS 和 IdM 集成的详情，请参考集成指南 ：
https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/integrate_with_identity_service/

第 5 章 策略（POLICY）

每个 OpenStack 服务都包含由访问策略管理的资源。例如，资源可能包含以下功能：

- 创建和启动实例的权限
- 将卷附加到实例的功能

作为 Red Hat OpenStack Platform (RHOSP)管理员，您可能需要创建自定义策略来引入具有不同级别访问权限的新角色，或更改现有角色的默认行为。您还需要在更改后验证 API 访问策略，并在策略无法正常工作时对其进行调试。验证和调试生产部署之外的策略，其中语法错误可能会导致停机，并且应用错误的授权可能会对安全性或可用性造成负面影响。

5.1. 检查现有策略

通常，服务的策略文件存在于 **/etc/$service** 目录中。例如，Compute (nova)的 **policy.json** 文件的完整路径为 **/etc/nova/policy.json**。

有两个重要的架构更改会影响到您如何找到现有策略：

- Red Hat OpenStack Platform 现已容器化。
  - 如果您从服务容器内查看策略文件（如果存在）位于传统路径中：
    /etc/$service/policy.json
  - 如果您从服务容器外查看策略文件（如果存在）位于以下路径中：
    /var/lib/config-data/puppet-generated/$service/etc/$service/policy.json
- 每个服务都有代码中提供的默认策略，且文件仅在您手动创建的文件或使用 **oslopolicy** 工具生成时可用。要生成策略文件，请使用容器中的 **oslopolicy-policy-generator**，如下例所示：

```
docker exec -it keystone oslopolicy-policy-generator --namespace keystone
```

默认情况下，生成的策略通过 osly.policy CLI 工具推送到 stdout。

5.2. 了解服务策略

服务策略文件语句是别名定义或规则。别名定义位于文件的顶部。以下列表包含计算(nova)生成的 **policy.json** 文件中别名定义的说明：

- "context_is_admin": "role:admin"
  当 **rule:context_is_admin** 在目标后出现时，策略会检查用户是否在允许该操作前使用管理上下文操作。
- "admin_or_owner": "is_admin:True or project_id:%(project_id)s"
  当 **admin_or_owner** 出现在目标后，策略会检查用户是 admin，或者其项目 ID 与目标对象的拥有项目 ID 匹配，然后再允许该操作。
- "admin_api": "is_admin:True
  当 **admin_api** 在目标后出现时，策略会检查用户是否是 admin，然后再允许该操作。

5.3. 策略语法

policy.json 文件支持某些操作器，以便您可以控制这些设置的目标范围。例如，以下 keystone 设置包含只有 admin 用户可以创建用户的规则：

```
"identity:create_user": "rule:admin_required"
```

: 字符左侧的部分描述了特权，右侧的 部分则定义哪些人可以使用特权。您还可以使用 Operator 来进一步控制范围：

- **!** - 没有任何用户（包括 admin）都可执行此操作。
- **@** 和 **""** - 任何用户都可以执行此操作。
- **not**, **and**, **or** - 可以使用标准的操作符。

例如，以下设置意味着没有用户有权创建新用户：

```
"identity:create_user": "!"
```

5.4. 使用策略文件进行访问控制

若要覆盖默认规则，请编辑相应 OpenStack 服务的 **policy.json** 文件。例如，Compute 服务在 nova 目录中有一个 **policy.json**，当您从容器内部查看时，这是容器化服务的正确位置。

注意

- 您必须在暂存环境中对策略文件进行彻底测试，然后才能在生产环境中实施它们。
- 您必须检查对访问控制策略的任何更改是否不会意外地降低任何资源的安全性。另外，对 **policy.json** 文件的任何更改都会立即有效，不需要重启服务。

5.5. 示例：创建高级用户角色

要自定义 keystone 角色的权限，请更新服务的 **policy.json** 文件。这意味着您可以更精细地定义分配给一类用户的权限。这个示例使用以下权限 *为您的部署* 创建一个高级用户角色：

- 启动实例。
- 停止一个实例。
- 管理连接到实例的卷。

此角色的目的是为某些用户授予其他权限，而无需授予 **admin** 访问权限。要使用这些权限，您必须为自定义角色授予以下权限：

- 启动一个实例：" **os_compute_api:servers:start": "role:PowerUsers"**
- 停止实例：" **os_compute_api:servers:stop": "role:PowerUsers"**
- 将实例配置为使用特定卷：" **os_compute_api:servers:create:attach_volume": "role:PowerUsers"**
- 列出附加到实例的卷：" **os_compute_api:os-volumes-attachments:index": "role:PowerUsers"**
- Attach a volume: **"os_compute_api:os-volumes-attachments:create": "role:PowerUsers"**

- 查看附加卷的详情："**os_compute_api:os-volumes-attachments:show": "role:PowerUsers"**
- 更改附加到实例的卷："**os_compute_api:os-volumes-attachments:update": "role:PowerUsers"**
- 删除附加到实例的卷："**os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"**

注意

当您修改 **policy.json** 文件时，您将覆盖默认策略。因此，**PowerUsers** 的成员是唯一可以执行这些操作的用户。要允许 **管理员用户** 保留这些权限，您可以为 *admin_or_power_user 创建规则*。您还可以使用一些基本条件逻辑来定义 **role:PowerUsers 或 role:Admin**。

1. 要确保在命令行会话中使用 keystone v3 API，请提供定义 v3 端点和设置的 rc 文件：

```
OS_AUTH_URL=http://controller-hostname.lab.local:5000/v3
OS_USERNAME=username
OS_PASSWORD=password
OS_USER_DOMAIN_NAME=Default
OS_PROJECT_DOMAIN_NAME=Default
OS_PROJECT_NAME=project-name
OS_IDENTITY_API_VERSION=3
```

2. 创建自定义 keystone 角色：

```
$ openstack role create PowerUsers
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 7061a395af43455e9057ab631ad49449 |
| name      | PowerUsers                       |
+-----------+----------------------------------+
```

3. 在角色中添加现有用户，并将角色分配给项目：

```
$ openstack role add --project [PROJECT_NAME] --user [USER_ID] [PowerUsers-ROLE_ID]
```

注意

角色分配仅与一个项目配对。这意味着，当您为用户分配角色时，您也同时定义 target 项目。如果您希望用户收到相同的角色，但对于不同的项目，您必须单独为他们分配该角色，但以不同的项目为目标。

4. 查看默认 nova 策略设置：

```
$ oslopolicy-policy-generator --namespace nova
```

5. 通过将以下条目添加到 **/var/lib/config-data/puppet-generated/nova/etc/nova/policy.json** 来为新的 **PowerUsers** 角色创建自定义权限：

注意

在部署前测试您的策略更改，以验证它们是否按预期工作。

```
{
"os_compute_api:servers:start": "role:PowerUsers",
"os_compute_api:servers:stop": "role:PowerUsers",
"os_compute_api:servers:create:attach_volume": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:index": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:create": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:show": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:update": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"
}
```

保存此文件并重新启动 nova 容器时，您将实施这些更改。添加到 **PowerUsers** keystone 角色的用户会收到这些权限。

5.6. 示例：根据属性限制访问

您可以创建策略，以根据发出该 API 调用的用户属性限制对 API 调用的访问。例如，以下默认规则指出，如果从管理上下文运行，或者令牌的用户 ID 与与目标关联的用户 ID 匹配时，允许删除密钥对。

```
"os_compute_api:os-keypairs:delete": "rule:admin_api or user_id:%(user_id)s"
```

注意：* 新实施的功能无法保证在每个服务中使用每个发行版本。因此，使用目标服务现有策略的约定编写规则非常重要。有关查看这些策略的详情，请参阅检查现有策略。* 所有策略都应该在非生产环境中为部署它们的每个版本进行严格测试，因为策略无法保证跨版本兼容。

根据上例，您可以制作 API 规则，以根据他们自己是否拥有资源来扩展或限制用户访问。另外，属性也可以与其他限制结合使用，以形成规则，如下例所示：

```
"admin_or_owner": "is_admin:True or project_id:%(project_id)s"
```

考虑以上示例，您可以创建一个仅限于管理员和用户的唯一规则，然后使用该规则来进一步限制操作：

```
"admin_or_user": "is_admin:True or user_id:%(user_id)s"
"os_compute_api:os-instance-actions": "rule:admin_or_user"
```

有关可用的 **policy.json** 语法选项的更多信息，请参阅策略语法。

5.7. 审计用户和角色

您可以使用 Red Hat OpenStack Platform 中的工具构建每个用户和相关权限的角色分配报告。

1. 运行 **openstack role list** 命令，查看环境中当前的角色：

```
openstack role list -c Name -f value

swiftoperator
ResellerAdmin
```

```
admin
_member_
heat_stack_user
```

2. 运行 **openstack role assignment list** 命令，以列出作为特定角色成员的所有用户。例如，要查看具有 admin 角色的所有用户，请运行以下命令：

```
$ openstack role assignment list --names --role admin
+-------+-----------------------------------+-------+-----------------+------------+--------+-----------+
| Role  | User                              | Group | Project         | Domain     | System | Inherited |
+-------+-----------------------------------+-------+-----------------+------------+--------+-----------+
| admin | heat-cfn@Default                  |       | service@Default |            |        | False     |
| admin | placement@Default                 |       | service@Default |            |        | False     |
| admin | neutron@Default                   |       | service@Default |            |        | False     |
| admin | zaqar@Default                     |       | service@Default |            |        | False     |
| admin | swift@Default                     |       | service@Default |            |        | False     |
| admin | admin@Default                     |       | admin@Default   |            |        | False     |
| admin | zaqar-websocket@Default           |       | service@Default |            |        | False     |
| admin | heat@Default                      |       | service@Default |            |        | False     |
| admin | ironic-inspector@Default          |       | service@Default |            |        | False     |
| admin | nova@Default                      |       | service@Default |            |        | False     |
| admin | ironic@Default                    |       | service@Default |            |        | False     |
| admin | glance@Default                    |       | service@Default |            |        | False     |
| admin | mistral@Default                   |       | service@Default |            |        | False     |
| admin | heat_stack_domain_admin@heat_stack |       |                 | heat_stack |        | False
|
| admin | admin@Default                     |       |                 |            | all    | False     |
+-------+-----------------------------------+-------+-----------------+------------+--------+-----------+
```

注意

您可以使用 **-f {csv,json,table,value,yaml}** 参数导出这些结果。

5.8. 审计 API 访问

您可以审核给定角色可以访问的 API 调用。为每个角色重复此过程将生成各个角色的可访问 API 的综合报告。对于您需要的步骤：

- 作为目标角色中的用户提供的身份验证文件。
- JSON 格式的访问令牌。
- 您希望审计的每个服务的 API 策略文件。

流程

1. 首先，在所需角色中提供用户的身份验证文件。
2. 捕获生成的 Keystone 令牌并将其保存到文件中。您可以通过运行任何 openstack-cli 命令并使用 --debug 选项进行此操作，该选项将提供的令牌输出到 stdout。您可以复制此令牌并将其保存到访问文件中。使用以下命令作为单一步骤进行此操作：

```
openstack token issue --debug 2>&1 | egrep ^'{\"token\":' > access.file.json
```

3. 创建策略文件。这可以在托管感兴趣的容器化服务的 overcloud 节点上完成。以下示例为 cinder 服务创建一个策略文件：

```
ssh heat-admin@CONTROLLER-1 sudo docker exec cinder_api \
oslopolicy-policy-generator \
--config-file /etc/cinder/cinder.conf \
--namespace cinder > cinderpolicy.json
```

4. 使用这些文件，您现在可以审核有关访问 cinder API 的角色：

```
oslopolicy-checker --policy cinderpolicy.json --access access.file.json
```

第 6 章 强化基础架构和虚拟化

您的操作程序应该有一个计划了解新的漏洞和安全更新。硬件和软件供应商通常宣布存在漏洞，并提供解决这些问题的解决方法和补丁。

红帽产品安全团队维护站点以帮助您了解安全更新：

- https://www.redhat.com/mailman/listinfo/rhsa-announce
- https://www.redhat.com/wapps/ugc/protected/notif.html
- https://access.redhat.com/security/security-updates/#/?q=openstack&p=1&sort=portal_publication_date%20desc&rows=10&portal_advisory_type=Security

注意

除了跟踪更新外，请确保您的进程和部署还可以容纳常规安全更新的安装。另外，使用以下指南：

- 在设计流程时包括实例安全更新。
- 重新引导计算和管理节点以获取内核更新。
- 定期更新托管的镜像服务(glance)镜像，以确保新创建的实例有最新的更新。

6.1. 虚拟机监控程序

当您评估虚拟机监控程序平台时，请考虑管理程序要在其上运行的硬件的可支持性。此外，还要考虑硬件中提供的其他功能，以及您选择作为 OpenStack 部署一部分的虚拟机监控程序支持这些功能。为此，管理程序各自具有自己的硬件兼容性列表(HCLs)。当选择兼容硬件时，提前知道基于硬件的虚拟化技术在安全性方面非常重要。

6.1.1. 管理程序与裸机

务必要识别 Linux 容器或裸机系统与使用 KVM 等虚拟机监控程序之间的差别。具体来说，此安全指南的重点主要是基于管理程序和虚拟化平台。但是，如果您的实施需要使用裸机或容器化环境，您必须注意部署该环境的具体区别。

对于裸机，请确保在重新置备和停用之前节点已被正确清理数据。另外，在重新使用节点前，您必须提供保证硬件没有被篡改或被破坏。如需更多信息，请参阅 https://docs.openstack.org/ironic/queens/admin/cleaning.html

6.1.2. hypervisor 内存优化

某些虚拟机监控程序使用内存优化技术，可过量使用虚拟机的内存。这是一个实用的功能，允许您部署非常密度的计算集群。这种技术的一种方法是通过去除重复数据或内存页面共享：当两个虚拟机在内存中有相同的数据时，有好处是将它们引用同一内存。通常，这通过 Copy-On-Write (COW)机制执行，如内核相同的页面合并(KSM)。这些机制容易受到攻击：

- 内存去除重复数据系统会受到 side-channel 攻击的影响。在学术研究中，攻击者可以通过分析攻击者虚拟机上的内存访问时间来识别在邻居虚拟机上运行的软件包和版本，以及软件下载和其他敏感信息。因此，一个虚拟机可以推断出另一个状态的问题，这可能不适用于不是所有项目信任的多项目环境，或者共享同一级别的信任

- 更重要的是，对 KSM 进行了 多行类型攻击，以保证跨虚拟机修改可执行内存。这意味着，恶意实例可以获取对同一计算主机上其他实例的代码执行访问权限。

如果部署器需要强大的项目分离（与公共云和一些私有云一样），则部署器应禁用 KSM：

- 要禁用 KSM，请参阅 取消激活 KSM。

6.2. PCI PASSTHROUGH

PCI 透传允许实例直接访问节点上的某一硬件。例如，这可用于允许实例访问视频卡或 GPU，为高性能计算提供计算的统一设备架构(CUDA)。这个功能涉及两种类型的安全风险：直接内存访问和硬件延迟。

直接内存访问(DMA)是允许某些硬件设备访问主机计算机中任意物理内存地址的功能。视频卡通常具有此功能。但是，不应为实例指定任意物理内存访问，因为它可以完全查看同一节点上运行的主机系统和其他实例。硬件供应商使用输入/输出内存管理单元(IOMMU)来管理 DMA 访问。您应该确认管理程序已配置为使用此硬件功能。

当实例对固件或某个设备的一部分进行恶意修改时，会发生硬件检测。由于此设备由其他实例或主机操作系统使用，因此恶意代码可能会分散到这些系统。最终结果是，一个实例可以在其安全区外运行代码。这是一个重大的破坏，因为难以重置物理硬件的状态，并可能导致其他暴露，如访问管理网络。

由于与 PCI 透传关联的风险和复杂性，应默认禁用它。如果为特定需要启用，则需要适当的进程，以帮助确保在重复使用前清理硬件。

6.3. SELINUX

强制访问控制通过限制 QEMU 进程的权限限制为仅需要什么内容来限制尝试攻击的影响。在 Red Hat OpenStack Platform 上，SELinux 配置为以单独的安全上下文运行每个 QEMU 进程。SELinux 策略已预先配置为 Red Hat OpenStack Platform 服务。

OpenStack 的 SELinux 策略旨在帮助保护虚拟机监控程序主机和虚拟机免受两个主要威胁向量：

- 管理程序威胁 - 虚拟机内运行的受损的应用程序会攻击管理程序以访问底层资源。例如，当虚拟机能够访问虚拟机监控程序操作系统、物理设备或其他应用程序时。这种威胁向量代表对虚拟机监控程序的威胁，对物理硬件的威胁以及公开其他虚拟机和网络段的风险。

- 虚拟机（多项目）威胁 - 虚拟机内运行的受损的应用程序会攻击虚拟机监控程序访问或控制另一台虚拟机及其资源。这是虚拟化独有的威胁向量，它代表了大量风险，因为单个应用程序中的漏洞可能会被破坏虚拟机文件镜像。这个虚拟网络攻击是一个主要关注，因为保护实际网络的管理技术不会直接应用到虚拟环境。每个基于 KVM 的虚拟机都是由 SELinux 标记的进程，能够有效地围绕每个虚拟机建立安全边界。此安全边界由 Linux 内核监控和强制实施，限制虚拟机对其边界之外的资源的访问权限，如主机数据文件或其他虚拟机。

无论虚拟机中运行的客户机操作系统是什么，都提供了红帽的基于 SELinux 的隔离。可以使用 Linux 或 Windows 虚拟机。

6.3.1. labels 和 Categories

基于 KVM 的虚拟机实例使用自己的 SELinux 数据类型进行标记，称为 **svirt_image_t**。内核级别保护可防止未经授权的系统进程（如恶意软件）操作磁盘上的虚拟机镜像文件。当虚拟机关闭时，镜像存储为 **svirt_image_t**，如下所示：

```
system_u:object_r:svirt_image_t:SystemLow image1
system_u:object_r:svirt_image_t:SystemLow image2
system_u:object_r:svirt_image_t:SystemLow image3
```

```
system_u:object_r:svirt_image_t:SystemLow image4
```

svirt_image_t 标签 (svirt_image_t 标签)会唯一标识磁盘上的镜像文件，允许 SELinux 策略限制访问。当基于 KVM 的计算镜像时，SELinux 会将随机数字标识符附加到镜像。SELinux 能够为每个虚拟机监控程序节点分配最多 524,288 虚拟机，但大多数 OpenStack 部署不太可能遇到这个限制。本例显示了 SELinux 类别标识符：

```
system_u:object_r:svirt_image_t:s0:c87,c520 image1
system_u:object_r:svirt_image_t:s0:419,c172 image2
```

6.3.2. SELinux 用户和角色

SELinux 管理用户角色。它们可以通过 **-Z** 标志查看，也可以使用 **semanage** 命令查看。在 hypervisor 上，只有管理员应该可以访问系统，并且应对管理用户和系统上的任何其他用户都有适当的上下文。

6.4. 容器化服务

有些服务（如 nova、glance 和 keystone）现在在容器内运行。这种方法有助于通过更轻松地将更新应用到服务来提高安全性。在其自己的容器中运行每个服务还提高了在同一裸机上共存的服务之间的隔离。如果任何一个服务受到攻击的影响，这非常有用，这可以防止对相邻服务进行轻松访问。

注意

挂载到容器的主机机器上的任何路径都可以用作在容器和主机间传输数据的挂载点（如果它们被配置为 ro/rw）。

如果您要更新任何配置文件，则考虑某些管理实践，假定容器化服务是临时的：

- 不要更新您可能在物理节点的主机操作系统中找到的任何配置文件，例如 **/etc/cinder/cinder.conf**。这是因为容器化服务不引用此文件。
- 不要更新在容器中运行的配置文件。这是因为在重启容器后任何更改都会丢失。

相反，如果您需要为容器化服务添加任何更改，则需要更新用于看到容器的配置文件。这些文件在初始部署期间由 puppet 生成，并包含对云运行很重要的敏感数据，应相应地处理。它们存储在 **/var/lib/config-data/puppet-generated/** 中。例如：

- keystone: **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf**
- cinder: **/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf**
- nova: **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf**

容器重启后将应用对这些文件所做的任何更改。

6.5. 强化计算部署

任何 OpenStack 部署的主要安全问题之一是围绕敏感文件的安全性和控制，如 **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** 文件。此配置文件包含许多敏感选项，包括配置详情和服务密码。所有这些敏感文件都应赋予严格的文件级别权限，并通过文件完整性监控(FIM)工具（如 AIDE）进行监控。这些工具将采用已知良好状态的目标文件的哈希值，然后定期使用文件的哈希并将其与已知良好的哈希进行比较。如果发现警报已意外修改，则可以创建警报。

可以通过移动到文件中包含的目录并运行 **ls -lh** 命令来检查文件的权限。这将显示有权访问该文件的权限、所有者和组，以及其他信息，如上次修改文件以及创建该文件的时间。

/var/lib/nova 目录包含有关给定 Compute 节点上的实例的信息。此目录应被视为敏感，并严格强制执行文件权限。此外，它应定期备份，因为它包含与该主机关联的实例的信息和元数据。

如果您的部署不需要全虚拟机备份，请考虑排除 **/var/lib/nova/instances** 目录，因为它与该节点上运行的每个实例的组合空间相同。如果您的部署需要完整的虚拟机备份，则需要确保成功备份该目录。

注意

存储在存储子系统（如 Ceph）中的数据也应考虑敏感，因为如果网络或逻辑访问允许，可以从存储子系统检索完整的虚拟机镜像。

6.6. 固件更新

物理服务器使用复杂的固件来启用和操作服务器硬件，以及轻型管理卡，它们可能有自己的安全漏洞，从而可能允许系统访问和中断。为了解决这个问题，硬件供应商会发布固件更新，这些更新与操作系统更新分开安装。您需要一个可操作的安全进程，在常规时间表上检索、测试并实现这些更新，请注意，固件更新通常需要重新启动物理主机才能生效。

6.7. 块存储

OpenStack Block Storage (cinder)是一种向自助服务提供软件（服务和库）的服务，用于管理永久的块级存储设备。这会创建按需访问块存储资源，以用于 Compute (nova)实例。这通过将块存储池虚拟化为各种后端存储设备（可以是软件实施或传统硬件存储产品）来创建软件定义型存储。这的主要功能是管理块设备的创建、附加和分离。消费者不需要了解后端存储设备的类型或其所在的位置。

计算实例使用行业标准存储协议（如 iSCSI、ATA over Ethernet 或 Fibre-Channel）存储和检索块存储。这些资源可使用 OpenStack 原生标准 HTTP RESTful API 管理和配置。

6.7.1. 卷 Wiping

擦除块存储设备的方法有多种。传统方法是将 **lvm_type** 设置为 thin，然后使用 **volume_clear** 参数。或者，如果卷加密功能我们已描述，如果删除了卷加密密钥，则不需要卷加密功能。

注意

在以前的版本中，**lvm_type=default** 用于表示擦除。虽然此方法 **仍可以正常工作，但不建议** 设置 secure delete。

volume_clear 参数可以接受 **zero** 或 **shred** 作为参数。**零** 可将单个从零写到设备。**shred** 操作将写入预定的位模式的三个通过。

6.7.2. 将配置文件的用户/组所有权设置为 root/cinder

配置文件包含组件可以平稳运行所需的关键参数和信息。如果非特权用户被有意或意外地修改或删除任何 par ater 或文件本身，则会导致严重可用性问题导致拒绝服务到其他最终用户。因此，此类关键配置文件的用户所有权必须设置为 root，并且组所有权必须设置为 **cinder**。

主机上不存在 **cinder** 组。使用 *ls -l* 将显示组所有者的 GID。

```
sudo ls -l /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
```

```
-rw-r-----. 1 root 42407 188012 Dec 11 09:34 /var/lib/config-data/puppet-
generated/cinder/etc/cinder/cinder.conf

sudo stat -L -c "%U %G" /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
root UNKNOWN
```

使用以下命令，检查 **cinder.conf** 配置文件的用户和组所有权是否已分别设置为 root 和 cinder：

```
sudo docker exec -it cinder_api stat -L -c "%U %G" /etc/cinder/cinder.conf
root cinder
```

6.7.3. 为配置文件设置严格的权限

检查 **cinder.conf** 配置文件的权限是否已设置为 **640** 或 stricter。

```
$ stat -L -c "%a" /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
```

6.7.4. 使用 keystone 进行身份验证

在 **/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf** 中，检查 **[DEFAULT]** 部分下的 **auth_strategy** 的值是否已设置为 **keystone**，而不是 **noauth**。

6.7.5. 为身份验证启用 TLS

在 **/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf** 中，检查 **[keystone_authtoken]** 部分下的 **auth_uri** 的值设置为一个以 **https:// ' 开头的身份 API 端点，以及 [keystone_authtoken] 下参数 'insecure 的值** 是否设为 **False**。

6.7.6. 确保块存储使用 TLS 与 Compute 通信

在 **cinder.conf** 中，检查 **[DEFAULT]** 部分下的 **glance_api_servers** 的值是否已设置为以 **https://** 开头，并且参数 **glance_api_insecure** 的值设置为 **False**。

6.7.7. 设置请求的正文的最大大小

如果没有定义每个请求的最大正文大小，攻击者可以制作一个大容量的任意 OSAPI 请求，从而导致服务崩溃，最终导致服务攻击。分配 t he maximum 值可确保任何恶意的超值被禁止确保服务持续可用。

查看 **cinder.conf** 中的 **[oslo_middleware]** 部分下的 **max_request_body_size** 是否已设置为 **114688**。

6.7.8. 启用卷加密

未加密的卷数据使卷托管平台对于攻击者而言尤为高值目标，因为它允许攻击者为许多不同的虚拟机读取数据。此外，物理存储介质库来自其他计算机被盗、重新挂载和访问。加密卷数据和卷备份可帮助缓解这些风险，并为卷托管平台提供防御性。块存储(c inder)可以在写入磁盘前对卷数据进行加密，因此请考虑启用卷加密，并使用 Barbican 进行私钥存储。

6.8. 网络

OpenStack Networking 服务(neutron)使最终用户或项目能够定义和使用网络资源。OpenStack 网络提供了一个面向项目的 API，用于定义云中实例的网络连接和 IP 寻址，除了编排网络配置之外。通过过渡到以 API 为中心的网络服务，云架构师和管理员应考虑确保物理和虚拟网络基础架构和服务的良好实践。

OpenStack 网络采用插件架构设计，通过开源社区或第三方服务提供 API 的可扩展性。当您评估您的架构设计要求时，务必要确定 OpenStack 网络核心服务中提供的功能、第三方产品提供的任何其他服务，以及在物理基础架构中实施哪些补充服务。

本节概述了实施 OpenStack 网络时应考虑的流程和良好做法。

6.8.1. 网络架构

OpenStack 网络是一种独立服务，可在多个节点上部署多个进程。这些流程相互交互和其他 OpenStack 服务。OpenStack 网络服务的主要流程是 neutron-server，它是公开 OpenStack 网络 API 的 Python 守护进程，并将项目请求传递给一组插件以进行额外的处理。

OpenStack 网络组件有：

- **Neutron 服务器(neutron-server 和 neutron-*-plugin)** - neutron-server 服务在 Controller 节点上运行，为网络 API 及其扩展（或插件）提供服务。它还强制执行每个端口的网络模型和 IP 地址。neutron-server 需要直接访问持久数据库。代理可以通过 neutron-server 间接访问数据库，它们使用 AMQP （高级消息队列协议）进行通信。

- **Neutron 数据库** - 数据库是 neutron 信息的集中式来源，API 记录数据库中的所有事务。这允许多个 Neutron 服务器共享相同的数据库集群，保持它们同步，并允许持久性网络配置拓扑。

- **插件代理(neutron-*-agent)** - 在每个计算节点和网络节点（与 L3 和 DHCP 一起）上运行，以管理本地虚拟交换机(vswitch)配置。启用的插件决定了哪些代理被启用。这些服务需要消息队列访问，并且根据所使用的插件来访问外部网络控制器或 SDN 实施。些插件，如 OpenDaylight (ODL)和 Open Virtual Network (OVN)，不需要计算节点上任何 python 代理，只需要启用的 Neutron 插件来进行集成。

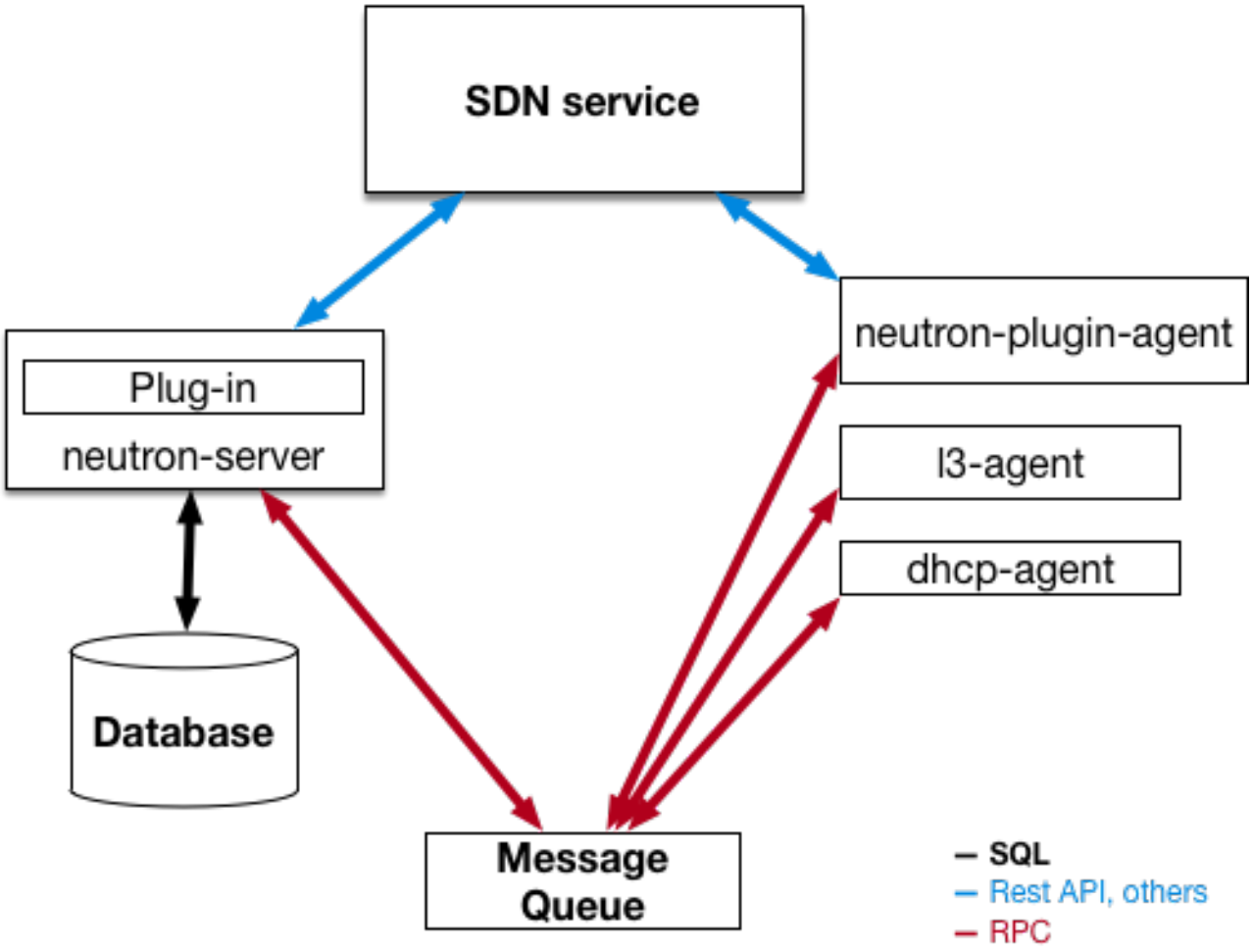
- **DHCP 代理(neutron-dhcp-agent)** - 为项目网络提供 DHCP 服务。这个代理在所有插件中都是一样的，负责维护 DHCP 配置。neutron-dhcp-agent 需要消息队列访问。可选基于插件。

- **元数据代理(neutron-metadata-agent,neutron-ns-metadata-proxy)** - 提供用于应用实例操作系统配置和用户提供的初始脚本('userdata')的元数据服务。该实施要求在 L3 或 DHCP 代理命名空间中运行 **neutron-ns-metadata-proxy** 来截获 cloud-init 发送的元数据 API 请求，以便代理到元数据代理。

- **L3 代理(neutron-l3-agent)** - 为项目网络上的虚拟机的外部网络访问提供 L3/NAT 转发。需要消息队列访问。可选基于插件。

- **网络提供程序服务(SDN server/services)** - 为项目网络提供额外的网络服务。这些 SDN 服务可以通过 REST API 等通信频道与 neutron-server、neutron-plugin 和 plugin-agents 交互。

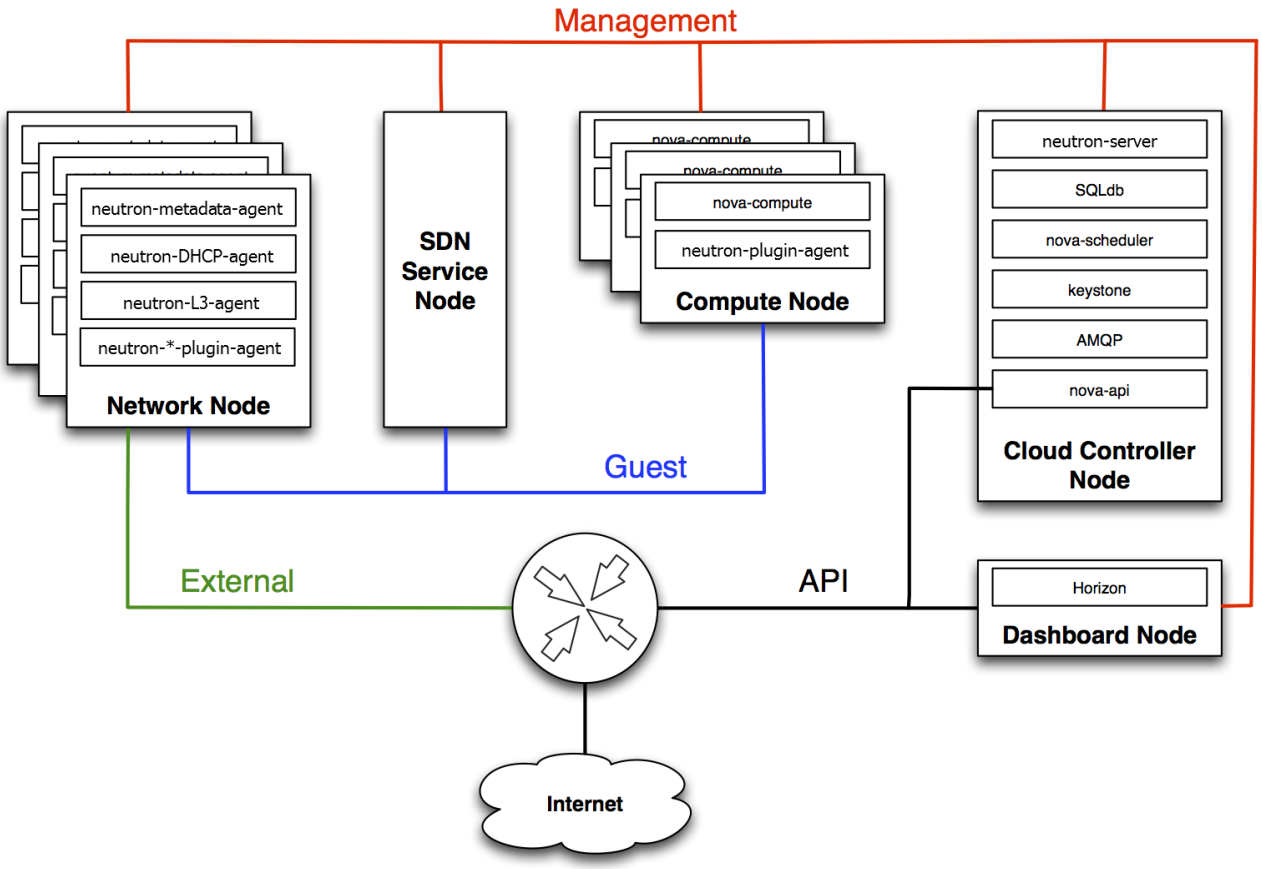
下图显示了 OpenStack 网络组件的架构和网络流图：



请注意，当使用分布式虚拟路由(DVR)和第 3 层高可用性(L3HA)时，此方法会有很大变化。这些模式改变了 neutron 的安全环境，因为 L3HA 在路由器之间实现 VRRP。部署需要正确调整大小和强化，以帮助缓解对路由器的 DoS 攻击，以及路由器之间的本地网络流量必须被视为敏感，以帮助解决 VRRP 欺骗的问题。DVR 将网络组件（如路由）移到 Compute 节点，同时仍需要网络节点。因此，计算节点需要访问和从公共网络访问，从而增加其暴露，并要求客户需要额外的安全考虑，因为它们需要确保防火墙规则和安全模型支持此方法。

6.8.2. 物理服务器上的 Neutron 服务放置

本节介绍包含控制器节点、网络节点和一组用于运行实例的计算节点的标准架构。要为物理服务器建立网络连接，典型的 neutron 部署具有四个不同的物理数据中心网络：



- **管理网络** - 用于 OpenStack 组件之间的内部通信。此网络上的 IP 地址应该只在数据中心内访问，并被视为 Management Security 区域。默认情况下，Management network 角色由 *Internal API* 网络执行。

- **客户机网络** - 用于云部署内的虚拟机数据通信.此网络的 IP 寻址要求取决于所使用的 OpenStack 网络插件，以及项目提出的虚拟网络的网络配置选择。此网络被视为 Guest Security 区域。

- **外部网络** - 在某些部署场景中为虚拟机提供互联网访问。此网络上的 IP 地址应该可以被 Internet 上的任何人访问。此网络被视为位于公共安全区中。此网络由 neutron *External* 网络提供。这些 neutron VLAN 托管在外部网桥上。它们不是由 Red Hat OpenStack Platform director 创建，而是在部署后由 neutron 创建。

- **公共 API 网络** - 向项目公开所有 OpenStack API，包括 OpenStack 网络 API。此网络上的 IP 地址应该可以被 Internet 上的任何人访问。这可能与外部网络相同，因为可以为外部网络创建一个使用 IP 分配范围小于 IP 块中完整范围的 IP 地址范围的子网。此网络被视为位于公共安全区中。

建议您将此流量划分为单独的区。如需更多信息，请参阅下一章节。

6.8.3. 安全区

建议您使用安全区的概念来保持关键系统相互独立。在实际意义上，这意味着使用 VLAN 和防火墙规则隔离网络流量。该短语以细致的细节完成，结果应该是只有需要连接到 neutron 的服务才能这样做。

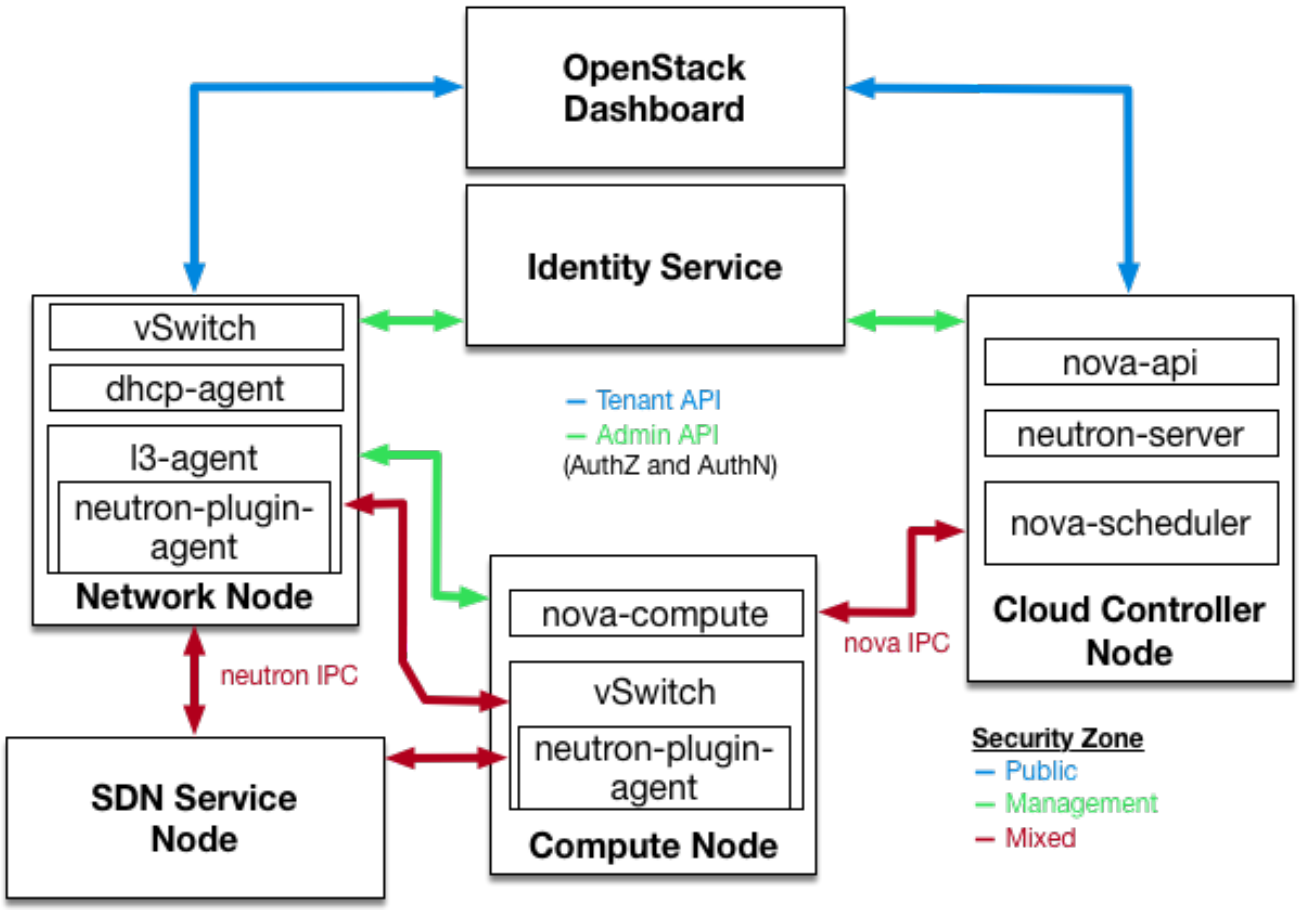
在下图中，您可以看到已创建区来分离某些组件：

- 仪表板：可以访问公共网络和管理网络。

- Keystone：可以访问管理网络。

- Compute 节点：可以访问管理网络和计算实例。

- 网络节点： 可根据所使用的 neutron-plugin 访问管理网络、计算实例和可能公共网络。
- SDN 服务节点：管理服务、计算实例，以及可能根据使用和配置的公共产品。



.

6.8.4. 网络服务

在设计您的 OpenStack 网络基础架构的初始架构阶段，务必要确保提供适当的专业知识，以帮助设计第一台物理网络基础架构，以确定正确的安全控制和审计机制。

OpenStack 网络添加了一个虚拟化网络服务层，让项目能够构建自己的虚拟网络。目前，这些虚拟化服务不像其传统网络的对应部分一样成熟。在采用这些虚拟化服务之前，请考虑这些虚拟化服务的当前状态，因为它指明了您在虚拟化和传统网络边界上实施的控制。

6.8.5. 使用 VLAN 和隧道进行 L2 隔离

OpenStack 网络可以使用两种不同的机制来针对各个项目/网络组合使用两种不同的机制：VLAN (IEEE 802.1Q 标记)或使用 VXLAN 或 GRE 封装的 L2 隧道进行。OpenStack 部署的范围和规模决定了您应该用于流量隔离或隔离的方法。

6.8.6. vlans

VLAN 在一个特定物理网络中以数据包的形式实现，它包括 IEEE 802.1Q 头，其中带有一个特定的 VLAN ID (VID) 字段值。共享同一物理 netw ork 的 VLAN 网络在网络的 L2 层相互隔离，它们甚至可以有重叠的 IP 地址空间。支持 VLAN 网络的每个不同物理网络都被视为单独的 VLAN trun k，不同的空间为 VID 值。有效的 VID 值为 1 到 4094。

VLAN 配置复杂性取决于您的 OpenStack 设计要求。要允许 OpenStack 网络更有效地使用 VLAN，您必须分配 VLAN 范围（每个项目一个），并将每个 Compute 节点物理交换机端口切换到 VLAN 中继端口。

6.8.7. L2 隧道

网络隧道将各个项目/网络组合与唯一的"隧道"组合封装，用于识别属于该组合的网络流量。项目的 L2 网络连接独立于物理本地或底层网络设计。通过在 IP 数据包内封装流量，该流量可以跨第 3 层边界，无需预先配置的 VLAN 和 VLAN 中继。隧道为网络流量添加了一个模糊的层，减少了个别项目流量的可见性会给监控点造成一个监控点。

OpenStack 网络目前支持 GRE 和 VXLAN 封装。提供 L2 隔离的选择取决于部署中将创建的项目网络的范围和大小。

6.8.8. 访问控制列表

计算(Compute)支持使用 OpenStack 网络服务进行项目网络流量访问控制。通过安全组，管理员可以和项目指定流量类型，以及允许通过虚拟接口端口的方向(ingress/egress)。安全组规则是有状态 L2-L4 流量过滤器。

6.8.9. L3 路由和 NAT

OpenStack 网络路由器可以连接多个 L2 网络，还可提供网关，将一个或多个专用 L2 网络连接到共享外部网络，如用于访问互联网的公共网络。

L3 路由器在将路由器链接到外部网络的网关端口上提供基本的网络地址转换(SNAT 和 DNAT)功能。默认情况下，此路由器 SNAT （源 NAT）所有出口流量，并支持浮动 IP，它会创建一个静态一对一的双向映射，从外部网络上的公共 IP 到附加到路由器的其他子网的私有 IP。浮动 IP （通过 DNAT）为实例提供外部入站连接，可以从一个实例移到另一个实例。

考虑使用每个项目 L3 路由和浮动 IP 来更精细的项目实例连接。应将特殊考虑给连接到公共网络或使用浮动 IP 的实例。建议仔细考虑安全组，以仅过滤对需要外部公开的服务的访问。

6.8.10. 服务质量(QoS)

默认情况下，服务质量(QoS)策略和规则由云管理员管理，这会导致项目无法创建特定的 QoS 规则，或将预期策略附加到端口。在某些用例中，如某些电信应用程序，管理员可能会信任项目，从而使他们能够创建和附加自己的策略到端口。这可以通过修改 **policy.json** 文件来实现。

从 Red Hat OpenStack Platform 12，neutron 支持入口和出口流量的带宽限制 QoS 规则。这个 QoS 规则名为 **QosBandwidthLimitRule**，它接受每秒 kilobits 测量的两个非负整数：

- **max-kbps** ：带宽
- **max-burst-kbps**: burst 缓冲

在 neutron Open vSwitch、Linux 网桥和 SR-IOV 驱动程序中实施了 **QoSBandwidthLimitRule**。但是，对于 SR-IOV 驱动程序，不使用 **max-burst-kbps** 值，如果设置，则会被忽略。

对于所有离开一个虚拟机的网络流量，QoS 规则 **QosDscpMarkingRule** 在服务的类型头（IPv4 (RFC 2474)）或网络类头（IPv6）中设置 DSCP（Differentiated Service Code Point ）值，相关的规则会在其中应用。这是一个 6 位标头，有 21 个有效值，表示数据包的丢弃优先级，因为它跨网络应满足拥塞。防火墙也可以使用它来匹配其访问控制列表的有效或无效流量。

6.8.11. 负载平衡

OpenStack 负载均衡服务(octavia)为 Red Hat OpenStack Platform director 安装提供负载平衡即服务(NFV)实施。为了实现负载平衡，octavia 支持启用多个供应商驱动程序。参考供应商驱动程序(Amphora provider driver)是一个开源、可扩展和高度可用的负载供应商。它通过管理一组虚拟机（统称为 amphorae-），实现负载平衡服务的交付。

有关负载均衡服务的更多信息，请参阅使用 Octavia 进行负载平衡即服务 指南。

6.8.12. 强化网络服务

本节讨论 OpenStack 网络配置良好实践，因为它们适用于您的 OpenStack 部署中的项目网络安全性。

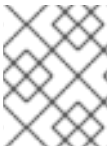
6.8.13. 限制 API 服务器的绑定地址：neutron-server

要限制 OpenStack Networking API 服务将网络套接字绑定到传入客户端连接的接口或 IP 地址，请在 **/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** 文件中指定 **bind_host** 和 **bind_por t** ：

```
# Address to bind the API server
bind_host = IP ADDRESS OF SERVER

# Port the bind the API server to
bind_port = 9696
```

6.8.14. 限制 OpenStack 网络服务的 DB 和 RPC 通信

OpenStack 网络服务的不同组件使用消息传递队列或数据库连接与 OpenStack 网络中其他组件通信。



注意

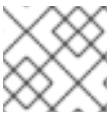
建议您按照 第 15.3 节 “队列身份验证和访问控制” 中针对需要 RPC 通信的所有组件进行操作。

6.8.15. 项目网络服务工作流

OpenStack 网络为用户提供网络资源的自助服务配置。云架构师和操作员在让用户能够创建、更新和销毁可用网络资源方面评估其设计用例非常重要。

6.8.16. 网络资源策略引擎

OpenStack 网络内的策略引擎及其配置文件(**policy.json**)提供了在项目网络方法和对象上向用户提供精细的授权的方法。OpenStack N etworking 策略定义会影响网络可用性、网络安全和整个 OpenStack 安全性。云架构师和操作员应仔细评估他们对用户和项目访问权限管理网络资源的政策。



注意

检查默认网络资源策略非常重要，因为可以修改此策略以满足您的安全状态。

如果您的 OpenStack 部署提供了多个外部访问点到不同的安全区，您务必要限制将多个 vNIC 附加到多个外部访问 po ints的项，这可能会导致这些安全区桥接，并可能导致无法预见的安全隐患。您可以使用计算提供的主机聚合功能，或将第 1 个项目实例拆分为具有不同虚拟网络配置的多个项目中，以帮助缓解这一风险。如需有关主机聚合的更多信息，请参阅 link:https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html/configuring_the_compute_service_for_instance_creation/creating-and-managing-host-aggregates[Creating and managing host aggregates]。

6.8.17. 安全组

安全组是安全组规则的集合。通过安全组及其规则，管理员可以和项目指定流量类型和方向 (ingress/egress)，允许通过虚拟接口端口。在 OpenStack 网络中创建虚拟接口端口时，它将与安全组关联。可以在默认安全组中添加规则，以便根据每个部署更改行为。

在使用 Compute API 修改安全组时，更新的安全组将应用到实例上的所有虚拟接口端口。这是因为计算安全组 API 是基于实例的，而不是基于端口的，如 neutron 中找到。

6.8.18. 配额

配额提供限制可用于项目的网络资源数量的功能。您可以为所有项目强制执行默认配额。要查看配额选项，请参阅 **/var/lib/config-data/puppet-generated /neutron/etc/neutron/neutron.conf**。

OpenStack 网络还通过配额扩展 API 支持按项目配额限制。要启用每个项目配额，您必须在 **/var/lib/config-data/puppet-generated/neutron/e tc/neutron/neutron.conf** 中设置 **quota_driver** 选项。例如：

```
quota_driver = neutron.db.quota_db.DbQuotaDriver
```

6.8.19. 缓解 ARP 欺骗

OpenStack 网络具有内置功能，可帮助缓解对实例的 ARP 欺骗。除非为生成的风险赋予了谨慎考虑，否则不应禁用此设置。

6.8.20. 将配置文件的用户/组所有权设置为 root/neutron

配置文件包含组件可以平稳运行所需的关键参数和信息。如果非特权用户意外或意外修改或删除任何半虚拟化计量或文件本身，则会导致严重可用性问题导致向其他最终用户拒绝服务。因此，此类关键配置文件的用户所有权必须设置为 root，并且组所有权必须设置为 neutron。

主机上不存在 **neutron** 组。使用 *ls -l* 将显示组所有者的 GID。

```
sudo ls -l /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf
-rw-r-----. 1 root 42435 41748 Dec 11 09:34 /var/lib/config-data/puppet-
generated/neutron/etc/neutron/neutron.conf

sudo stat -L -c "%U %G" /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf
root UNKNOWN
```

确保 **neuton.conf** 配置文件的用户和组所有权分别设置为 **root** 和 **neutron** ：

```
sudo docker exec -it neutron_api stat -L -c "%U %G" /etc/neutron/neutron.conf
root neutron
```

6.8.21. 为配置文件设置严格权限

检查 **neutron.conf** 配置文件的权限是否已设置为 **640** 或 stricter：

```
$ stat -L -c "%a" /var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf
```

6.8.22. 使用 Keystone 进行身份验证

在 **/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** 中，检查 **[DEFAULT]** 部分下的 **auth_strategy** 的值是否被设置为 **keystone**，而不是 **noauth** 2 或 **noauth2**。

6.8.23. 使用安全协议进行身份验证

在 **/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** 中检查 **[keystone_authtoken]** 部分下的 **auth_uri** 的值是否设置为以 'https 开头的身份 API 端点：

6.8.24. 在 Neutron API 服务器上启用 TLS

在 **/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** 中，确保 **[DEFAULT]** 部分下的参数 **use_ssl** 设置为 **True**。

第 7 章 网络时间协议

您需要确保 Red Hat OpenStack Platform 集群中的系统在系统间具有准确且一致的时间戳。

7.1. 为什么持续时间很重要

在整个机构中，对操作和安全需求都很重要：

识别安全事件

持续计时可以帮助您关联受影响系统上的事件的时间戳，以便您可以了解事件序列。

身份验证和安全系统

安全系统可能会对时间偏移敏感，例如：

- 基于 kerberos 的身份验证系统可能会拒绝验证受时钟偏移影响的客户端。
- 传输层安全性(TLS)证书取决于有效时间源。如果客户端和服务器系统时间超过 *Valid From* 日期范围，则服务器 TLS 连接的客户端会失败。

Red Hat OpenStack Platform 服务

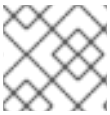
某些核心 OpenStack 服务特别依赖于准确计时，包括高可用性(HA)和 Ceph。

7.2. NTP 设计

网络时间协议(NTP)以分级设计进行组织。每个层称为 stratum。在层次结构的顶部是 stratum 0 设备，如原子时钟。在 NTP 层次结构中，Straum 0 设备为公开可用的级别 1 和 stratum 2 NTP 时间服务器提供参考。

不要将数据中心客户端直接连接到公开可用的 NTP stratum 1 或 2 服务器。直接连接的数量会对公共 NTP 资源造成不必要的压力。相反，在您的数据中心中分配一个专用时间服务器，并将客户端连接到该专用服务器。

将实例配置为从专用时间服务器（而不是它们所在的主机）接收时间。



注意

在 Red Hat OpenStack Platform 环境中运行的服务容器仍从它们所在的主机接收时间。

第 8 章 使用 DIRECTOR 配置安全强化

使用 Red Hat OpenStack Platform director 在部署过程中应用安全强化值。OpenStack 配置文件由 director 管理，如果直接编辑，则可覆盖。

在运行 **openstack overcloud deploy** 时，请记住，除了您要进行的任何更改外，您始终还需要包括部署 overcloud 所需的全部环境文件。

8.1. 使用 SSH 横幅文本

您可以设置标语，向通过 SSH 连接的所有用户显示控制台消息。您可以使用环境文件中的以下参数向 **/etc/issue** 添加横幅文本。考虑自定义此示例文本以满足您的要求。

```
resource_registry:
  OS::TripleO::Services::Sshd: ../puppet/services/sshd.yaml

parameter_defaults:
  BannerText: |
   ******************************************************************
   * This system is for the use of authorized users only. Usage of  *
   * this system may be monitored and recorded by system personnel. *
   * Anyone using this system expressly consents to such monitoring *
   * and is advised that if such monitoring reveals possible        *
   * evidence of criminal activity, system personnel may provide    *
   * the evidence from such monitoring to law enforcement officials.*
   ******************************************************************
```

要将这个更改应用到您的部署，请将设置保存为名为 **ssh_banner.yaml** 的文件，然后将它传递给 **overcloud 部署命令**，如下所示：& **lt;full environment** > 表示您仍然必须包含所有原始部署参数。例如：

```
    openstack overcloud deploy --templates \
      -e <full environment> -e  ssh_banner.yaml
```

8.2. 系统事件的审核

维护所有审计事件的记录可帮助您建立系统基准、执行故障排除或分析导致特定结果的事件序列。审计系统可以记录许多类型的事件，如更改系统时间、强制/诊断访问控制的更改，以及创建/删除用户或组。

可以使用环境文件创建规则，然后由 director 注入到 **/etc/audit/audit.rules** 中。例如：

```
    resource_registry:
      OS::TripleO::Services::Auditd: /usr/share/openstack-tripleo-heat-
templates/puppet/services/auditd.yaml
    parameter_defaults:
      AuditdRules:
        'Record Events that Modify User/Group Information':
          content: '-w /etc/group -p wa -k audit_rules_usergroup_modification'
          order  : 1
        'Collects System Administrator Actions':
          content: '-w /etc/sudoers -p wa -k actions'
          order  : 2
```

```
    'Record Events that Modify the Systems Mandatory Access Controls':
      content: '-w /etc/selinux/ -p wa -k MAC-policy'
      order  : 3
```

8.3. 管理防火墙规则

防火墙规则在部署过程中自动应用到 overcloud 节点上，并且仅用于公开 OpenStack 正常工作所需的端口。您可以根据需要指定额外的防火墙规则。例如，要为 Zabbix 监控系统添加规则：

```
parameter_defaults:
  ControllerExtraConfig:
    tripleo::firewall::firewall_rules:
      '301 allow zabbix':
        dport: 10050
        proto: tcp
        source: 10.0.0.8
        action: accept
```

您还可以添加限制访问的规则。规则定义中使用的数字将决定规则的优先级。例如，RabbitMQ 的规则号默认为 **109**。如果要保留它，请将其切换到使用较低值：

```
parameter_defaults:
  ControllerExtraConfig:
    tripleo::firewall::firewall_rules:
      '098 allow rabbit from internalapi network':
        dport: [4369,5672,25672]
        proto: tcp
        source: 10.0.0.0/24
        action: accept
      '099 drop other rabbit access':
        dport: [4369,5672,25672]
        proto: tcp
        action: drop
```

在本例中，**098** 和 **099** 是任意选择的数字，其小于 RabbitMQ 的规则号 **109**。要确定规则的数量，您可以检查适当节点上的 iptables 规则；对于 RabbitMQ，您将检查控制器：

```
iptables-save
[...]
-A INPUT -p tcp -m multiport --dports 4369,5672,25672 -m comment --comment "109 rabbitmq" -m
state --state NEW -j ACCEPT
```

或者，您还可以从 puppet 定义中提取端口要求。例如，RabbitMQ 的规则存储在 **puppet/services/rabbitmq.yaml** 中：

```
tripleo.rabbitmq.firewall_rules:
  '109 rabbitmq':
    dport:
      - 4369
      - 5672
      - 25672
```

可以为规则设置以下参数：

- **端口** ：与规则关联的端口。被 **puppetlabs-firewall** 弃用.
- **D** port ：与规则关联的目标端口。
- **S** PORT ：与规则关联的源端口。
- **Proto** ：与规则关联的协议。默认为 *tcp*
- **操作** ：与规则关联的操作策略。默认为 *accept*
- **跳到** ：要跳到的链。
- **State** ：与规则关联的状态数组。默认为 [*NEW*]
- **Source** ：与规则关联的源 IP 地址。
- **iniface** ：与规则关联的网络接口。
- **链** ：与规则关联的链。默认为 *INPUT*
- **destination** ：与规则关联的目标 cidr。
- **Extras** ： Puppet **labs-firewall** 模块支持的任何其他参数的哈希。

8.4. 使用 AIDE 入侵检测

AIDE （高级入侵检测环境）是一个文件和目录完整性检查程序。它用于检测未授权文件篡改或更改的事件。例如，如果更改了系统密码文件，AIDE 可以提醒您。

AIDE 的工作原理是分析系统文件，然后编译文件哈希的完整性数据库。然后，数据库充当一个比较点，以验证文件和目录的完整性并检测更改。

director 包括 AIDE 服务，允许您向 AIDE 配置中添加条目，然后供 AIDE 服务用来创建完整性数据库。例如：

```
resource_registry:
  OS::TripleO::Services::Aide: ../puppet/services/aide.yaml

parameter_defaults:
  AideRules:
    'TripleORules':
      content: 'TripleORules = p+sha256'
      order: 1
    'etc':
      content: '/etc/ TripleORules'
      order: 2
    'boot':
      content: '/boot/ TripleORules'
      order: 3
    'sbin':
      content: '/sbin/ TripleORules'
      order: 4
    'var':
      content: '/var/ TripleORules'
      order: 5
    'not var/log':
```

```
    content: '!/var/log.*'
    order: 6
  'not var/spool':
    content: '!/var/spool.*'
    order: 7
  'not nova instances':
    content: '!/var/lib/nova/instances.*'
    order: 8
```

注意

上面的示例没有主动维护或基准测试，因此您应该选择符合您的要求的 AIDE 值。

1. 声明了一个名为 **TripleORules** 的别名，以避免每次都重复相同的属性。
2. 别名接收 **p+sha256** 属性。在 AIDE 术语中，这会被解释为：监视所有文件权限 **p**，带有完整性 checksum **sha256**。

有关 AIDE 配置文件可用属性的完整列表，请参见 的 AIDE MAN 页面（https://aide.github.io/）。

完成以下内容以将更改应用到您的部署：

1. 将设置保存为 **/home/stack/templates/** 目录中的名为 **aide.yaml** 的文件。
2. 您必须确保 **OS::TripleO::Services::Aide** 参数的值已从相对路径改为绝对路径：

```
  OS::TripleO::Services::Aide: /usr/share/openstack-tripleo-heat-
templates/puppet/services/aide.yaml
```

3. 在 **openstack overcloud deploy** 命令中包含 **/home/stack/templates/aide.yaml** 环境文件，以及特定于您环境的所有其他必要 heat 模板和环境文件：

```
openstack overcloud deploy --templates
...
-e /home/stack/templates/aide.yaml
```

8.4.1. 使用复杂 AIDE 规则

可以使用前面描述的格式创建复杂规则。例如：

```
MyAlias = p+i+n+u+g+s+b+m+c+sha512
```

以上将转换为以下指令：监控权限、索引节点、链接数、用户、组、大小、块数、mtime、ctime，使用 sha256 进行校验和生成。

请注意，别名应始终具有 **1** 的顺序位置，这意味着它位于 AIDE 规则的顶部，并递归地应用到以下所有值。

如下为要监控的目录。请注意，可以使用正则表达式。例如，我们为 **var** 目录设置了监控，但使用了 not（**!**）语句（**'!/var/log.*'** 和 **'!/var/spool.*'**）进行了覆盖。

8.4.2. 其他 AIDE 值

以下 AIDE 值也可用：

AideConfPath ： aide 配置文件的完整路径，默认为 **/etc/aide.conf**。如果不要求更改文件位置，建议与默认路径保持一致。

AideDBPath ：AIDE 完整性数据库的完整路径。这个值可以被配置，以便操作员声明自己的完整路径，因为 AIDE 数据库文件可能会将节点存储在只读文件挂载中。

AideDBTempPath ：AIDE 完整性临时数据库的完整路径。当 AIDE 初始化新数据库时，会创建此临时文件。

AideHour ：此值是将 hour 属性设置为 AIDE cron 配置的一部分。

AideMinute ：此值是将 minute 属性设置为 AIDE cron 配置的一部分。

AideCronUser ：此值是将 linux 用户设置为 AIDE cron 配置的一部分。

AideEmail ：此值设置每次执行 cron 运行时接收 AIDE 的电子邮件地址。

AideMuaPath ：此值设置邮件用户代理的路径，用于将 AIDE 报告发送到 **AideEmail** 中设置的电子邮件地址。

8.4.3. AIDE 的 Cron 配置

AIDE director 服务允许您配置 cron 作业。默认情况下，它将向 **/var/log/audit/**; 发送通知；如果要使用电子邮件警报，则启用 **AideEmail** 参数，以将警报发送到配置的电子邮件地址。请注意，依赖于电子邮件中的关键警报容易受到系统中断和意外消息过滤的影响。

8.4.4. 考虑系统升级的影响

当执行升级时，AIDE 服务将自动重新生成一个新的完整性数据库，以确保正确重新计算所有升级的文件来拥有更新的校验和。

如果 **openstack overcloud deploy** 被作为后续运行调用到初始部署，并且更改了 AIDE 配置规则，则 director AIDE 服务将重建数据库，以确保新的配置属性封装在完整性数据库中。

8.5. 回顾 SECURETTY

securetty 允许您禁用任何控制台设备(tty)的 root 访问权限。此行为由 **/etc/securetty** 文件中的条目管理。例如：

```
resource_registry:
  OS::TripleO::Services::Securetty: ../puppet/services/securetty.yaml

parameter_defaults:
  TtyValues:
    - console
    - tty1
    - tty2
    - tty3
    - tty4
    - tty5
    - tty6
```

8.6. IDENTITY SERVICE 的 CADF 审计

全面的审计流程可帮助您检查 OpenStack 部署的持续安全性。这对 keystone 尤为重要，因为其在安全模型中的角色。

Red Hat OpenStack Platform 采用 Cloud Auditing Data Federation (CADF)作为审计事件的数据格式，keystone 服务为 Identity 和 Token 操作生成 CADF 事件。您可以使用 **KeystoneNotificationFormat** 为 keystone 启用 CADF 审计：

```
parameter_defaults:
  KeystoneNotificationFormat: cadf
```

8.7. 检查 LOGIN.DEFS 值

要强制对新系统用户（非keystone）的密码要求，director 可以按照以下示例参数将条目添加到 **/etc/login.defs** 中：

```
resource_registry:
  OS::TripleO::Services::LoginDefs: ../puppet/services/login-defs.yaml

parameter_defaults:
  PasswordMaxDays: 60
  PasswordMinDays: 1
  PasswordMinLen: 5
  PasswordWarnAge: 7
  FailDelay: 4
```

第 9 章 强化仪表板服务

本章介绍了使用 OpenStack Dashboard (horizon)的 Red Hat OpenStack Platform 部署的安全强化注意事项。

控制面板为用户提供一个自助服务门户，用于调配自己的资源（管理员设置的限制）。例如，您可以使用控制面板定义实例类别、上传虚拟机(VM)镜像、管理虚拟网络、创建安全组、启动实例，以及通过管理控制台远程访问实例。

如果仪表板具有与 OpenStack API 相同的敏感度，则应该可以对待控制面板，同时具有对云资源和配置的访问权限。

9.1. 规划仪表板部署

本节论述了在部署 Dashboard (horizon)服务前需要考虑的安全问题。

9.1.1. 选择域名

建议您将仪表板部署到二级域中，如 **https://example.com**，而不是在共享子域（任何级别上）上部署仪表板，如 **https://openstack.example.org** 或 **https://horizon.openstack.example.org**。它还建议您避免将仪表板部署到裸机内部域，如 **https://horizon/**。这些建议基于浏览器 **same-origin-policy** 的限制。

这种方法可帮助您将 Cookie 和安全令牌与其他域隔离，因为您可能无法对内容进行完全控制。当部署到子域上时，仪表板的安全性等同于在同一第二级别域中部署的最低安全应用程序。

您可以通过避免由 Cookie 支持的会话存储和配置 HTTP Strict Transport Security (HSTS) （本指南中所述），来进一步缓解这一风险。

9.1.2. 配置 ALLOWED_HOSTS

Web 服务可能会受到与虚拟 HTTP 主机标头关联的威胁的影响。为了帮助缓解这一点，请考虑将 **ALLOWED_HOSTS** 设置配置为使用由 OpenStack 控制面板提供的 FQDN。

配置后，如果传入 HTTP 请求的 **Host:** 标头中的值与此列表中的任何值不匹配，则将引发错误，并且请求者将无法继续。

Horizon 基于 python Django Web 框架构建，这需要设置 **ALLOWED_HOSTS** 以防止潜在的 **HTTP Host:** 标头的恶意操作。将此值设置为 FQDN，仪表板应该可从其中访问。对于 director，此设置由 **HorizonAllowedHosts** 管理。

9.2. 了解常见的 WEB 服务器漏洞

本章论述了如何帮助缓解一些常见 Web 服务器漏洞。

9.2.1. 跨站点脚本(XSS)

OpenStack 控制面板可以自定义，并允许大多数字段中设置整个 Unicode 字符；这种可扩展性可能会引入跨站点脚本(XSS)漏洞。Horizon 包括可帮助开发人员避免意外创建 XSS 漏洞的工具，但只有在开发人员正确使用这些漏洞时才能工作。建议您审核任何自定义仪表板，并特别注意以下功能：

- **mark_safe** 功能。
- **is_safe** - 与自定义模板标签一起使用时。

- **safe** 模板标签。
- 任何自动转义都关闭，任何可能会评估错误转义的数据的 JavaScript。

9.2.2. 跨站点请求 Forgery (CSRF)

OpenStack 仪表板旨在禁止开发人员通过自定义仪表板引入跨站点脚本漏洞，因为可能会引入威胁。应当对使用多个 JavaScript 实例的仪表板进行审核，如不当使用 **@csrf_exempt** decorator。任何未遵循这些推荐的安全设置的仪表板，都应在 CORS (Cross Origin 资源共享)限制之前仔细评估。

您应该将 Web 服务器配置为为每个响应发送限制的 CORS 标头，仅允许仪表板域和协议。例如：**Access-Control-Allow-Origin: https://example.com/**。您不应该允许通配符源。

9.2.3. 跨报脚本(XFS)

disallow_iframe_embed 设置不允许 Dashboard 在 iframe 中嵌入。旧浏览器仍容易受到 Cross-Frame Scripting (XFS)漏洞的影响，因此此选项可以为不需要的部署添加额外的安全强化。

您可以使用以下参数允许 iframe 嵌入：

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disallow_iframe_embed: false
```

9.2.4. 为 Dashboard 流量使用 HTTPS 加密

建议您使用 HTTPS 加密仪表板流量。您可以将证书配置为使用可识别的证书(CA)的有效可信证书来实现此目的。只有在所有用户浏览器中预安装信任 root 时，私有机构签发的证书才适当。

配置 HTTP 请求到仪表板域，以重定向到完全限定的 HTTPS URL。

有关基于 director 的部署，请参阅 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html/advanced_overcloud_customization/sect-enabling_ssltls_on_the_overcloud。

9.2.5. HTTP 严格传输安全性(HSTS)

HTTP 严格传输安全性(HSTS)可防止浏览器在最初建立安全连接后进行后续的不安全连接。如果您已在公共或不受信任的区域中部署了 HTTP 服务，HSTS 尤为重要。

对于基于 director 的部署，默认启用此设置：

```
enable_secure_proxy_ssl_header: true
```

9.3. 缓存仪表板内容

9.3.1. 前端缓存

不建议在控制面板中使用前端缓存工具，因为它呈现了直接从 OpenStack API 请求生成的动态内容。因此，**varnish** 等前端缓存层可能会阻止显示正确的内容。控制面板使用 Django，它直接提供从 Web 服务提供的静态媒体，并且已经从 Web 主机缓存中受益。

9.3.2. 会话后端

对于基于 director 的部署，horizon 的默认会话后端是 **django.contrib.sessions.backends.cache**，它与 memcached 的组合。由于性能原因，这种方法首选使用 local-memory 缓存，对于高可用性和负载平衡的安装而言是更安全的，而且能够将缓存与多个服务器共享，同时仍将其视为单一缓存。

您可以在 director 的 **horizon.yaml** 文件中查看这些设置：

```
horizon::cache_backend: django.core.cache.backends.memcached.MemcachedCache
horizon::django_session_engine: 'django.contrib.sessions.backends.cache'
```

9.4. 检查 SECRET 密钥

仪表板依赖于一些安全功能的共享 **SECRET_KEY** 设置。secret 密钥应该是随机生成的字符串，至少 64 个字符，它必须在所有活跃的仪表板实例间共享。对这个密钥的威胁可能会允许远程攻击者执行任意代码。轮转此密钥会导致现有用户会话和缓存无效。不要将此密钥提交到公共存储库。

对于 director 部署，此设置作为 **HorizonSecret** 值进行管理。

9.5. 配置会话 COOKIE

控制面板会话 Cookie 可以通过浏览器技术（如 JavaScript）进行交互。对于随处 TLS 的 director 部署，您可以使用 **HorizonSecureCookies** 设置强化此行为。

注意

切勿配置 CSRF 或会话 Cookie，以使用带前点的通配符域。

9.6. 静态介质

仪表板的静态介质应部署到仪表板域的子域并由 Web 服务器提供。还可以接受使用外部内容交付网络 (CDN)。此子域不应设置 Cookie 或提供用户提供的内容。介质还应通过 HTTPS 提供。

仪表板的默认配置使用 **django_compressor** 在提供之前压缩和减去 CSS 和 JavaScript 内容。这个过程应该在部署仪表板前静态完成，而不是使用默认的 in-request 动态压缩，并与部署的代码或 CDN 服务器一起复制生成的文件。应在非生产构建环境中进行压缩。如果这不实际，请考虑完全禁用资源压缩。不应在生产环境中安装在线压缩依赖项（无 Node.js）。

9.7. 验证密码复杂性

OpenStack Dashboard (horizon)可以使用密码验证检查来强制实施密码复杂性。

您可以指定用于密码验证的正则表达式，并为失败的测试显示帮助文本。以下示例要求用户创建长度为 8 到 18 个字符之间的密码：

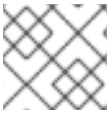
```
parameter_defaults:
  HorizonPasswordValidator: '^.{8,18}$'
  HorizonPasswordValidatorHelp: 'Password must be between 8 and 18 characters.'
```

要将此更改应用到您的部署，请将设置保存为名为 **horizon_password.yaml** 的文件，然后将它传递给 **overcloud deploy** 命令，如下所示：& **lt;full environment** > 表示您仍然必须包含所有原始部署参数。例如：

```
openstack overcloud deploy --templates \
  -e <full environment> -e  horizon_password.yaml
```

9.8. 强制管理员密码检查

以下设置默认设置为 **True**，但可以根据需要使用环境文件来禁用它。



注意

只有在完全理解潜在的安全影响后，这些设置才应设置为 **False**。

Dashboard 的 **local_settings.py** 文件中的 **ENFORCE_PASSWORD_CHECK** 设置在 *Change Password* 表单上显示一个 *Admin Password* 字段，这有助于验证管理员是否启动密码更改。

您可以使用环境文件禁用 **ENFORCE_PASSWORD_CHECK** ：

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::enforce_password_check: false
```

9.9. DISALLOW IFRAME EMBEDDING

注意

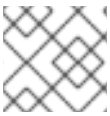
只有在完全理解潜在的安全影响后，这些设置才应设置为 **False**。

DISALLOW_IFRAME_EMBED 设置不允许在 iframe 中嵌入仪表板。旧浏览器仍容易受到 Cross-Frame Scripting (XFS)漏洞的影响，因此此选项可以为不需要的部署添加额外的安全强化。默认设置为 **True**，但可以根据需要使用环境文件来禁用它。例如，您可以使用以下参数允许 iframe 嵌入：

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disallow_iframe_embed: false
```

9.10. 禁用密码显示

以下设置默认设置为 **True**，但可以根据需要使用环境文件来禁用它。



注意

只有在完全理解潜在的安全影响后，这些设置才应设置为 **False**。

密码显示按钮可在仪表板上查看他们要输入的密码。可以使用 **DISABLE_PASSWORD_REVEAL** 参数切换这个选项：

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disable_password_reveal: false
```

9.11. 显示仪表板的登录横幅

许多管制行业，如 HIPAA、PCI-DSS 和美国。政府，要求您显示用户登录横幅。控制面板服务在容器内运行，因此您需要自定义 Dashboard 容器镜像以应用横幅。有关自定义 Dashboard 容器的更多信息，请参阅 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/introduction_to_the_openstack_dashboard/index#dashboard-customization。

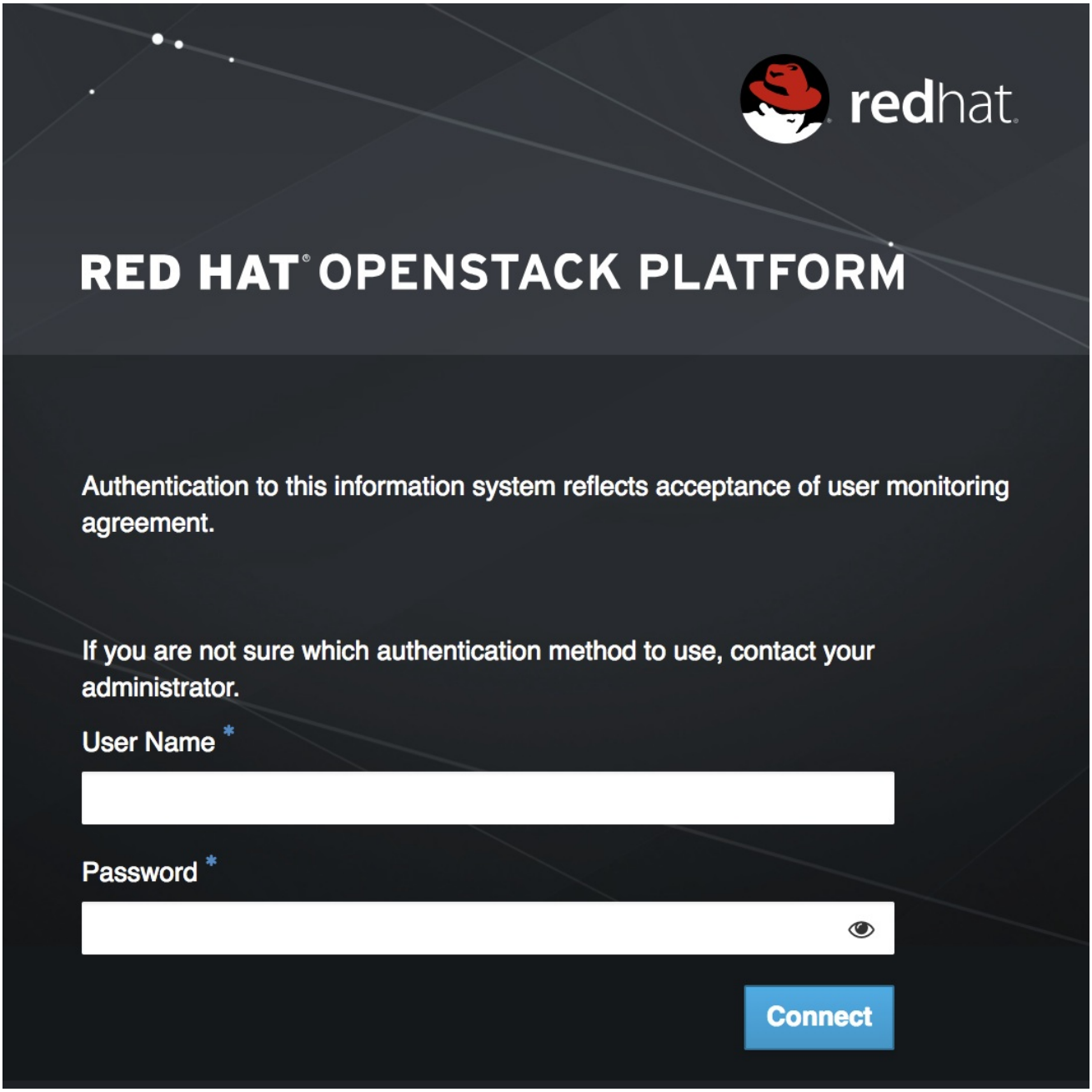
9.11.1. 自定义主题

在自定义 Dashboard 容器中，您可以通过手动编辑 **/usr/share/openstack-dashboard/openstack_dashboard/themes/rcue/templates/auth/login.html** 文件来创建一个登录横幅：

1. 在 **{% include 'auth/_login.html' %}** 部分前立即输入所需的 logon banner。请注意，允许 HTML 标签。例如：

```
<snip>
<div class="container">
  <div class="row-fluid">
    <div class="span12">
      <div id="brand">
        <img src="../../static/themes/rcue/images/RHOSP-Login-Logo.svg">
      </div><!--/#brand-->
    </div><!--/.span*-->

    <!-- Start of Logon Banner -->
    <p>Authentication to this information system reflects acceptance of user monitoring agreement.</p>
    <!-- End of Logon Banner -->

    {% include 'auth/_login.html' %}
  </div><!--/.row-fluid→
</div><!--/.container-->

{% block js %}
  {% include "horizon/_scripts.html" %}
{% endblock %}

  </body>
</html>
```

更新的仪表板类似如下：



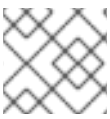
9.12. 限制文件上传的大小

您可以选择配置仪表板来限制文件上传的大小；此设置可能需要各种安全强化策略。

LimitRequestBody - 这个值（以字节为单位）限制您可以使用仪表板（如镜像和其他大型文件）传输的文件的最大大小。

重要

红帽尚未正式测试此设置。建议您在将其部署到生产环境中之前，彻底测试此设置的影响。



注意

如果值太小，文件上传将失败。

例如，此设置将每个文件上传限制为最大 10 GB (**10737418240**)。您需要调整这个值以适合您的部署。

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf/httpd.conf**

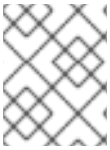
```
<Directory />
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/10-horizon_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/15-horizon_ssl_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```



注意

这些配置文件由 Puppet 管理，因此每当您运行 **openstack overcloud deploy** 过程时，任何未管理的更改都会被覆盖。

9.13. 调试仪表板服务

建议在生产环境中将 **DEBUG** 设置保留为 **False**。如果设置为 **True**，则 Django 可能会输出堆栈追踪到包含敏感 Web 服务器状态信息的浏览器用户。此外，**DEBUG** 模式具有禁用 **ALLOWED_HOSTS** 的影响，这可能是不必要的结果，具体取决于您的要求。

第 10 章 强化共享文件系统服务(MANILA)

共享文件系统服务(manila)提供了一组服务，用于在多项目云环境中管理共享文件系统。它类似于 OpenStack 通过块存储服务(cinder)项目提供基于块的存储管理的方式。使用 manila，您可以创建一个共享文件系统并管理其属性，如可见性、可访问性和使用配额。

有关 manila 的更多信息，请参阅存储指南： https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/storage_guide/

10.1. MANILA 的安全注意事项

Manila 通过 keystone 注册，允许您使用 **manila 端点** 命令找到 API。例如：

```
$ manila endpoints
+-------------+-----------------------------------------+
| manila      | Value                                   |
+-------------+-----------------------------------------+
| adminURL    | http://172.18.198.55:8786/v1/20787a7b...|
| region      | RegionOne                               |
| publicURL   | http://172.18.198.55:8786/v1/20787a7b...|
| internalURL | http://172.18.198.55:8786/v1/20787a7b...|
| id          | 82cc5535aa444632b64585f138cb9b61        |
+-------------+-----------------------------------------+

+-------------+-----------------------------------------+
| manilav2    | Value                                   |
+-------------+-----------------------------------------+
| adminURL    | http://172.18.198.55:8786/v2/20787a7b...|
| region      | RegionOne                               |
| publicURL   | http://172.18.198.55:8786/v2/20787a7b...|
| internalURL | http://172.18.198.55:8786/v2/20787a7b...|
| id          | 2e8591bfcac4405fa7e5dc3fd61a2b85        |
+-------------+-----------------------------------------+
```

默认情况下，manila API 服务仅侦听端口 **8786，其支持** IPv4 和 IPv6。

Manila 使用多个配置文件；这些文件存储在 **/var/lib/config-data/puppet-generated/manila/** 中：

```
api-paste.ini
manila.conf
policy.json
rootwrap.conf
rootwrap.d

./rootwrap.d:
share.filters
```

建议您将 manila 配置为以非 root 服务帐户运行，并更改文件权限，以便只有系统管理员可以修改它们。Manila 期望只有管理员能够写入配置文件，服务只能通过 **manila** 组中的组成员身份来读取它们。其他用户不能读取这些文件，因为它们包含服务帐户密码。

注意

只有 root 用户应该可以写入 **rootwrap.conf** 中的 **manila-rootwrap** 配置，**manila-rootwrap** 命令会过滤 **rootwrap.d/share.filters** 中的共享节点。

10.2. MANILA 的网络和安全模型

manila 中的共享驱动程序是一个 Python 类，可以为后端设置来管理共享操作，其中一些特定于供应商。后端是 **manila-share** 服务的实例。Manila 具有许多不同的存储系统的共享驱动程序，支持商业供应商和开源解决方案。每个共享驱动都支持一个或多个后端模式：**共享服务器**，**无共享服务器**。管理员使用 **driver_handles_share_servers** 在 **manila.conf** 中指定它来选择模式。

共享服务器是导出共享文件系统的逻辑网络附加存储(NAS)服务器。现在，后端存储系统非常复杂，可以隔离不同 OpenStack 项目之间的数据路径和网络路径。

在属于创建项目用户的隔离网络上创建由 manila 共享驱动程序置备的共享服务器。**共享** 服务器模式可以配置扁平网络或网段网络，具体取决于网络提供程序。

对于不同的模式，可以使用不同的驱动程序来使用相同的硬件。根据所选模式，您可能需要通过配置文件提供额外的配置详情。

10.3. 共享后端模式

每个共享驱动程序支持至少一个可用驱动程序模式：

- 共享服务器 - **driver_handles_share_servers = True** - 共享驱动程序创建共享服务器并管理共享服务器生命周期。
- 无共享服务器 - **driver_handles_share_servers = False** - 管理员（而不是共享驱动程序）使用网络接口管理裸机存储，而不依赖于存在共享服务器。

无共享服务器模式 - 在这个模式中，驱动程序将不会设置共享服务器，因此不需要设置任何新的网络接口。假设由驱动程序管理的存储控制器具有需要的所有网络接口。驱动程序直接创建共享，而无需之前创建共享服务器。要使用在这个模式下运行的驱动程序创建共享，manila 不需要用户创建任何私有共享网络。

注意

在 **无共享服务器模式** 中，manila 将假定导出任何共享的网络接口可被所有项目访问。

在没有共享服务器 模式中，共享驱动程序无法处理共享服务器生命周期。管理员应该处理可能需要提供项目隔离所需的存储、网络和其他主机侧配置。在此模式中，管理员可以将存储设置为导出共享的主机。OpenStack 云中的所有项目共享一个通用的网络管道。缺少隔离可能会影响安全性和服务质量。当使用不处理共享服务器的共享驱动程序时，云用户无法确保不可信用户通过树形访问其共享，可以遍历其文件系统的顶级目录。在公共云中，所有网络带宽都由一个客户端使用，因此管理员应该小心谨慎。网络平衡可以由任何方法实现，而不一定要借助 OpenStack 工具。

共享服务器模式 - 在此模式中，驱动程序可以创建共享服务器并将其插入到现有的 OpenStack 网络。Manila 确定是否需要新的共享服务器，并提供共享驱动程序所需的所有网络信息来创建必要的共享服务器。

在处理共享服务器的驱动程序模式中创建共享时，用户必须提供一个共享网络，以便他们希望导出其共享。Manila 使用此网络为此网络上的共享服务器创建网络端口。

用户可以在共享服务器中配置 **安全服务，且没有 共享服务器** 后端模式。但是，**如果没有共享服务器** 后端模式，管理员必须在主机上手动设置所需的身份验证服务。在 **共享服务器** 模式中，manila 可以配置用户在其生成的共享服务器上识别的安全服务。

10.4. MANILA 的网络要求

Manila 可以与不同的网络类型集成： **flat**、**GRE**、**VLAN**、**VXLAN**。

注意

Manila 仅将网络信息存储在数据库中，以及由网络提供程序提供的实际网络。Manila 支持使用 OpenStack Networking 服务(neutron)和"standalone"预配置网络。

在 **共享服务器** 后端模式中，共享驱动程序为每个共享网络创建和管理共享服务器。这个模式可以分为两个变体：

- **共享服务器** 后端模式的扁平网络
- **共享服务器** 后端模式中的分段网络

用户可以使用 OpenStack Networking (neutron)服务中的网络和子网来创建共享网络。如果管理员决定使用 **StandAloneNetworkPlugin**，则用户不需要提供任何网络信息，因为管理员在配置文件中预配置此设置。

注意

由某些共享驱动程序生成的共享服务器是使用计算服务创建的 Compute 服务器。其中一些驱动程序不支持网络插件。

创建共享网络后，manila 检索由网络提供程序确定的网络信息：网络类型、分段标识符（如果网络使用分段）和 CIDR 表示法中分配网络的 IP 块。

用户可以创建指定安全要求的安全服务，如 AD 或 LDAP 域或 Kerberos 域。Manila 假定创建共享服务器的子网可以访问在安全服务中引用的任何主机，这限制了可以使用此模式的情况数量。

注意

一些共享驱动程序可能不支持所有类型的分段，请参阅您使用的驱动程序规格。

10.5. 使用 MANILA 的安全服务

Manila 可以通过与网络身份验证协议集成来限制对文件共享的访问。每个项目都可以有自己的身份验证域，域独立于云的 Keystone 身份验证域。此项目域可用于向 OpenStack 云中运行的应用提供授权(AuthZ)服务，包括 manila。可用的身份验证协议包括 LDAP、Kerberos 和 Microsoft Active Directory 身份验证服务。

10.5.1. 安全服务简介

在创建了共享并获取其导出位置后，用户没有挂载它的权限，并使用文件操作。用户需要明确授予新共享的访问权限。

客户端身份验证和授权(authN/authZ)可与安全服务一同执行。如果共享驱动程序和后端支持，manila 可以使用 LDAP、Kerberos 或 Microsoft Active 目录。

注意

在某些情况下，需要明确指定一个安全服务，例如 NetApp、EMC 和 Windows 驱动程序需要使用 CIFS 协议创建共享。

10.5.2. 安全服务管理

安全服务 是一个 manila 实体，它抽象了一组选项，它们为特定共享文件系统协议（如 Active Directory 域或 Kerberos 域）定义安全区。安全服务包含 manila 创建加入给定域的服务器所需的所有信息。

通过使用 API，用户可以创建、更新、查看和删除安全服务。安全服务在以下假设中设计：

- 项目提供安全服务的详情。
- 管理员关心安全服务：它们配置此类安全服务的服务器端。
- 在 manila API 中，**security_service** 与 **share_networks** 关联。
- 共享驱动程序使用安全服务中的数据来配置新创建的共享服务器。

在创建安全服务时，您可以选择以下身份验证服务之一：

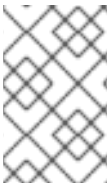
- **LDAP** - 轻量级目录访问协议。用于通过 IP 网络访问和维护分布式目录信息服务的应用程序协议。
- **Kerberos** - 在票据的基础上工作的网络身份验证协议，允许节点通过非安全网络进行通信，以安全的方式证明其身份。
- **Active Directory** - Microsoft 为 Windows 域网络开发的目录服务。使用 LDAP、Microsoft 的 Kerberos 版本和 DNS。

Manila 允许您使用以下选项配置安全服务：

- 在项目网络中使用的 DNS IP 地址。
- 安全服务的 IP 地址或主机名。
- 安全服务的域。
- 项目使用的用户或组名称。
- 如果您指定了用户名，则用户的密码。

现有的安全服务实体可以与共享网络实体关联，该实体向 manila 告知一组共享的安全性和网络配置。您还可以看到指定共享网络的所有安全服务列表，并将它们与共享网络关联。

管理员和用户作为共享所有者可以通过 IP 地址、用户、组或 TLS 证书创建通过身份验证的访问规则来管理对共享的访问。身份验证方法取决于您配置和使用的共享驱动程序和安全服务。然后，您可以将后端配置为使用特定的身份验证服务，该服务可在没有 manila 和 keystone 的情况下操作客户端。



注意

不同共享驱动程序支持不同的身份验证服务。有关支持不同驱动程序的功能的详情，请参考 https://docs.openstack.org/manila/latest/admin/share_back_ends_feature_support_mappi

通过驱动程序支持特定身份验证服务并不意味着可以使用任何共享文件系统协议进行配置。支持的共享文件系统协议包括 NFS、CEPHFS、CIFS、GlusterFS 和 HDFS。有关特定驱动程序及其安全服务配置的详情，请查看驱动程序厂商的文档。

有些驱动支持安全服务，而其他驱动则不支持上述任何安全服务。例如，带有 NFS 或 CIFS 共享文件系统协议的通用驱动程序只支持通过 IP 地址进行身份验证的方法。

注意

在大多数情况下，支持 CIFS 共享文件系统协议的驱动程序可以配置为使用 Active Directory 并通过用户身份验证管理访问权限。

- 支持 GlusterFS 协议的驱动程序可使用 TLS 证书进行身份验证。
- 使用 IP 地址支持 NFS 协议身份验证的驱动程序是唯一支持的选项。
- 由于 HDFS 共享文件系统协议使用 NFS 访问，因此也可以将其配置为使用 IP 地址进行身份验证。

生产 manila 部署的建议配置是使用 CIFS 共享协议创建共享，并将其添加到 Microsoft Active Directory 目录服务中。使用这个配置，您将获得集成 Kerberos 和 LDAP 方法的集中式数据库和服务。

10.6. 共享访问控制

用户可以指定哪些特定客户端有权访问其创建的共享。由于 Keystone 服务，由各个用户创建的共享仅对同一项目中本身和其他用户可见。Manila 允许用户创建"公开"可见的共享。这些共享在属于其他 OpenStack 项目的用户的仪表板中可见，如果所有者授予访问权限，它们甚至可以挂载这些共享（如果它们可从网络访问）。

在创建共享时，使用 key **--public** 为您的其他项目共享公共，以便在共享列表中看到它，并查看其详细信息。

根据 **policy.json** 文件，管理员和共享所有者的用户可以通过创建访问规则来管理对共享的访问。使用 **manila access-allow**、**manila access-deny** 和 **manila access-list** 命令，您可以相应地授予、拒绝和列出对指定共享的访问权限。

注意

Manila 不提供存储系统的端到端管理。您仍然需要单独保护后端系统不受未授权访问的影响。因此，如果某人破坏后端存储设备，则 manila API 提供的保护仍然可以被循环处理，从而获得带外的访问。

当只创建共享时，没有与其关联的默认访问规则，并挂载它的权限。这可以在挂载配置中使用导出协议中看到。例如，存储上存在一个 NFS 命令 **exportfs** 或 **/etc/exports** 文件，该文件控制每个远程共享并定义可以访问它的主机。如果 nobody 可以挂载共享，则为空。对于远程 CIFS 服务器，有一个 **net conf list** 命令可以显示配置。**hosts deny** 参数应由共享驱动程序设置为 **0.0.0.0/0**，这意味着拒绝任何主机挂载共享。

使用 manila，您可以通过指定这些支持的共享访问级别之一来允许或拒绝对共享的访问：

- **rw** - 读取和写入(RW)访问。这是默认值。
- **ro**- 只读(RO)访问。

注意

当管理员为某些某些编辑器或贡献者提供读写(RW)访问权限时，RO 访问级别在公共共享方面很有用，并为其余用户(viewer)提供只读(RO)访问权限。

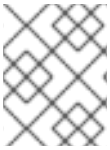
您还必须指定以下支持的身份验证方法之一：

- **ip** - 使用 IP 地址验证实例。可以通过以 CIDR 表示法以正确的 IPv4 或 IPv6 地址为客户端提供 IP 访问。
- **认证** - 使用 TLS 证书验证实例。将 TLS 身份指定为 **IDENTKEY**。有效值是证书通用名称(CN)中长到 64 个字符的字符串。
- **user** - 由指定用户或组名称进行授权。有效值是一个字母数字字符串，可以包含一些特殊字符，长度为 4 到 32 个字符。

注意

支持的身份验证方法取决于您使用的共享驱动程序、安全服务和共享文件系统协议。支持的共享文件系统协议有 MapRFS、CEPHFS、NFS、CIFS、GlusterFS 和 HDFS。支持的安全服务包括 LDAP、Kerberos 协议或 Microsoft Active Directory 服务。

要验证是否为共享正确配置了访问规则(ACL)，您可以列出其权限。



注意

为共享选择安全服务时，您需要考虑共享驱动程序是否可以使用可用的身份验证方法创建访问规则。支持的安全服务包括 LDAP、Kerberos 和 Microsoft Active Directory。

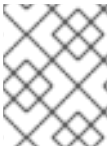
10.7. 共享类型访问控制

共享类型是一个由管理员定义的*服务类型（type of service）*，它包括一个项目可见的描述信息，以及一个项目不可见的、称为*额外规格（extra specifications）*的键值对列表。**manila-scheduler** 使用额外的规格来做出调度决策，驱动程序控制共享创建。

管理员可以创建和删除共享类型，也可以管理额外规格在 manila 中赋予它们的含义。项目可以列出共享类型，并可使用它们来创建新共享。共享类型可以被创建为 **public** 和 **private**。这是定义其他项目是否可以在共享类型列表中看到的共享类型的可见性级别，并使用它来创建新共享。

默认情况下，共享类型创建为公共。在创建共享类型时，使用 **--is_public** 参数设置为 **False** 来使您的共享类型私有设为私有，这将阻止其他项目在共享类型列表中看到它，并使用它创建新的共享。另一方面，**公共** 共享类型可供云中的每个项目使用。

Manila 允许管理员为项目授予或拒绝对 **私有** 共享类型的访问权限。您还可以获取有关指定私有共享类型访问权限的信息。



注意

由于它们的额外规格有助于在用户创建共享之前过滤或选择后端，因此您可以使用对特定后端选择的客户端限制客户端。

例如，**admin** 项目中的管理员用户可以创建名为 **my_type** 的专用共享类型，并在列表中看到它。在下面的控制台示例中，会省略了登录和注销，并且提供了环境变量来显示当前登录的用户。

■

```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ manila type-list --all
+----+--------+-----------+-----------+--------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs           | optional_extra_specs |
+----+--------+-----------+-----------+--------------------------------+----------------------+
| 4..| my_type| private   | -         | driver_handles_share_servers:False| snapshot_support:True |
| 5..| default| public    | YES       | driver_handles_share_servers:True | snapshot_support:True |
+----+--------+-----------+-----------+--------------------------------+----------------------+
```

demo 项目中的 **demo** 用户可以列出类型，但名为 **my_type** 的私有共享类型对他不可见。

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
+----+--------+-----------+-----------+--------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs           | optional_extra_specs |
+----+--------+-----------+-----------+--------------------------------+----------------------+
| 5..| default| public    | YES       | driver_handles_share_servers:True| snapshot_support:True|
+----+--------+-----------+-----------+--------------------------------+----------------------+
```

管理员可以为 demo 项目授予专用共享类型的访问权限，其项目 ID 等于 **df29a37db5ae48d19b349fe947fada46** ：

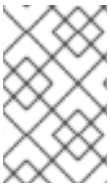
```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ openstack project list
+----------------------------------+--------------------+
| ID                               | Name               |
+----------------------------------+--------------------+
| ...                              | ...                |
| df29a37db5ae48d19b349fe947fada46 | demo               |
+----------------------------------+--------------------+
$ manila type-access-add my_type df29a37db5ae48d19b349fe947fada46
```

因此，**demo** 项目中的用户可以查看私有共享类型，并在创建共享时使用它：

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
+----+--------+-----------+-----------+--------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs           | optional_extra_specs |
+----+--------+-----------+-----------+--------------------------------+----------------------+
```

```
| 4..| my_type| private   | -          | driver_handles_share_servers:False| snapshot_support:True |
| 5..| default| public    | YES        | driver_handles_share_servers:True | snapshot_support:True |
+----+--------+-----------+------------+-----------------------------------+-----------------------+
```

要拒绝对指定项目的访问，请使用 **manila type-access-remove <share_type> <project_id>**。

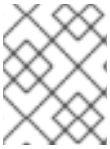


注意

在演示共享类型的目的中，请考虑有两个后端：LVM 作为公共存储和 Ceph 作为私有存储的情况。在这种情况下，您可以使用 **用户/组 身份**验证方法授予对特定项目的访问权限，并控制访问权限。

10.8. 策略（POLICY）

共享文件系统服务 API 使用基于角色的访问控制策略进行设计。这些策略决定了哪些用户能够以某种方式访问某些 API，并在服务的 **policy.json** 文件中定义。



注意

配置文件 **policy.json** 可以在任何位置放置。默认预期预期 **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 的路径。

每当向 manila 发出 API 调用时，策略引擎使用适当的策略定义来确定是否可以接受调用。策略规则决定在哪些情况下允许 API 调用。当规则是空字符串：**""** 时，**/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 文件具有始终允许的操作的规则："" ；基于用户角色或规则的规则。以下是 manila 的 **policy.json** 文件的片段。预计在 OpenStack 版本之间可能会有变化。

```
{
    "context_is_admin": "role:admin",
    "admin_or_owner": "is_admin:True or project_id:%(project_id)s",
    "default": "rule:admin_or_owner",
    "share_extension:quotas:show": "",
    "share_extension:quotas:update": "rule:admin_api",
    "share_extension:quotas:delete": "rule:admin_api",
    "share_extension:quota_classes": "",
}
```

用户必须分配给您在策略中引用的组和角色。使用用户管理命令时，服务会自动完成。

注意

对 **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** 的任何更改都会立即生效，这允许在 manila 运行时实施新的策略。手动修改策略可能会带来意外的副作用。Manila 不提供默认策略文件；所有默认策略都在代码库内。您可以通过执行：**oslopolicy-sample-generator --config-file=var/lib/config-data/puppet-generated/manila/etc/manila/manila-policy-generator.conf**从 manila 代码生成默认策略

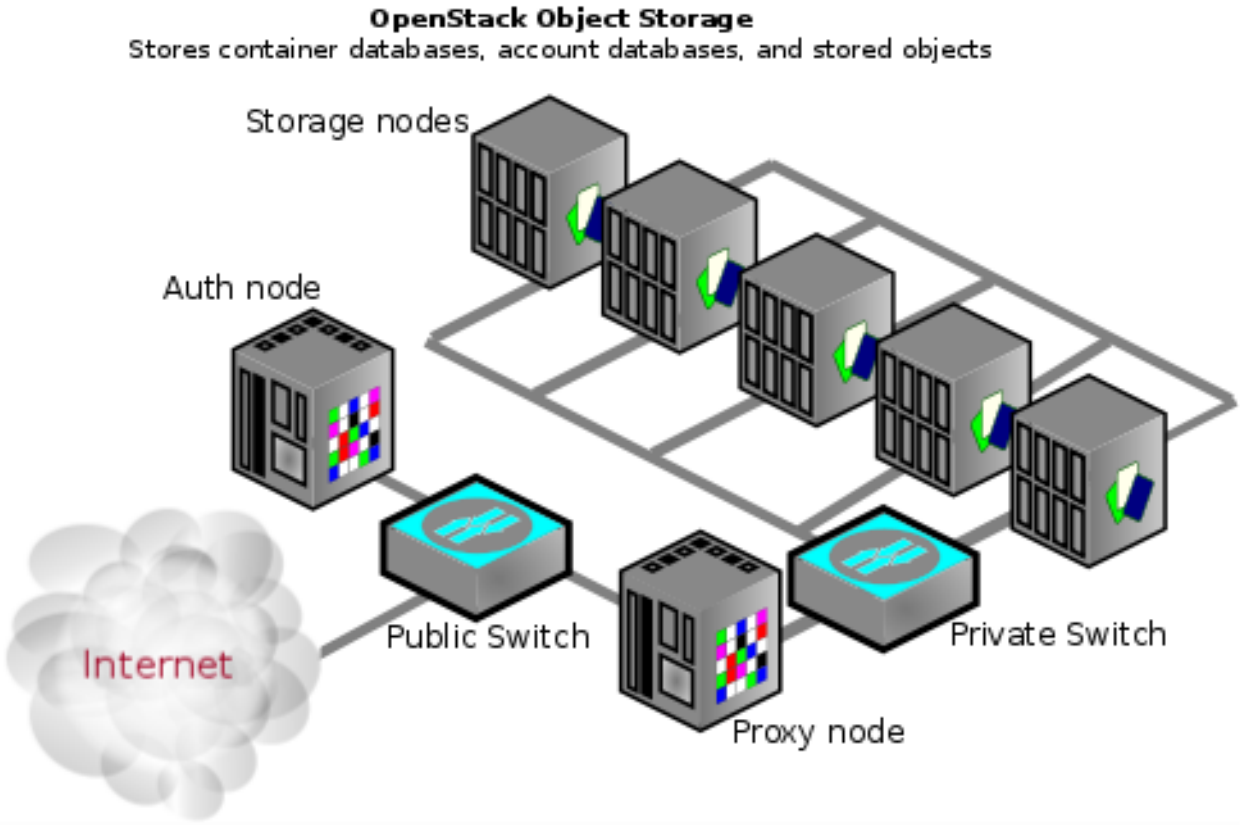
第 11 章 对象存储

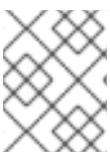
Object Storage (swift)服务通过 HTTP 存储和检索数据。对象（数据的blob）存储在组织层次结构中，它可以配置为提供匿名只读访问、ACL 定义访问甚至临时访问。Swift 支持多种通过中间件实施的基于令牌的身份验证机制。

应用程序使用行业标准 HTTP RESTful API 在对象存储中存储和检索数据。后端 swift 组件遵循相同的 RESTful 模型，但一些 API （如管理持久性）会保持对集群的私有状态。

swift 的组件属于以下主要组：

- 代理服务
- 身份验证服务
- 存储服务
  - 帐户服务
  - 容器服务
  - 对象服务





注意

对象存储安装不必面向互联网，也不必通过公共交换机作为组织内部网络基础架构的一部分，作为私有云。

11.1. 网络安全性

对 swift 的安全强化开始保护网络组件。如需更多信息，请参阅网络章节。

为了高可用性，使用 rsync 协议在存储服务节点间复制数据。此外，在客户端端点和云环境之间转发数据时，代理服务与存储服务通信。

注意

Swift 不使用加密，也不会与节点间通信进行身份验证。这是因为 swift 出于性能的原因使用原生 rsync 协议，且不会对 rsync 通信使用 SSH。这是您在架构图中看到私有交换机或专用网络([V]LAN)的原因。此数据区域还应与其他 OpenStack 数据网络分开。

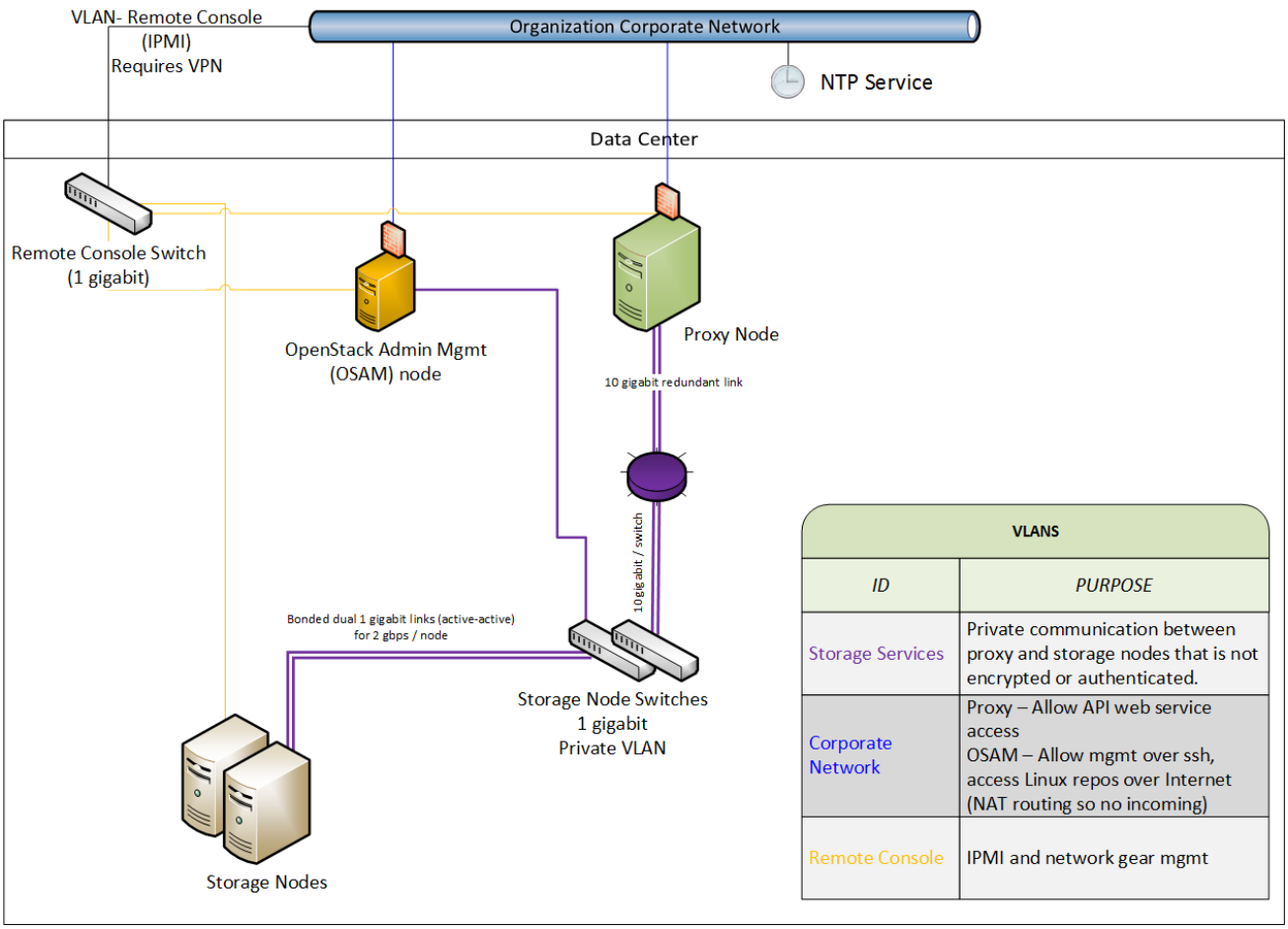
注意

在数据区中为您的存储节点使用私有(V) LAN 网络段。

这要求代理节点具有两个接口（物理或虚拟）：

- 一个接口，作为消费者访问的公共接口。
- 另一个是私有接口，可访问存储节点。

下图演示了一个可能的网络架构，它使用具有管理节点(OSAM)的对象存储网络架构：



11.2. 常规服务安全性

11.2.1. 以非 root 用户身份运行服务

建议将 swift 配置为在非 root (**UID 0**)服务帐户下运行。一个建议是，director 部署的用户名 **swift** 并带有一个主组 **swift**。对象存储服务包括 **proxy-server**、**container-server**、**account-server**。

11.2.2. 文件权限

/var/lib/config-data/puppet-generated/swift/etc/swift/ 目录包含有关环拓扑和环境配置的信息。建议以下权限：

```
chown -R root:swift /var/lib/config-data/puppet-generated/swift/etc/swift/*
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type f -exec chmod 640 {} \;
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type d -exec chmod 750 {} \;
```

此限制仅允许 root 修改配置文件，同时仍允许服务读取这些文件，因为 **swift** 组中的成员资格。

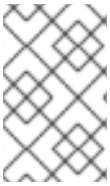
11.3. 保护存储服务

以下是各种存储服务的默认侦听端口：

- 帐户服务 - **TCP/6002**
- 容器服务 - **TCP/6001**
- 对象服务 - **TCP/6000**
- rsync - **TCP/873**

注意

如果使用 ssync 而不是 rsync，则使用对象服务端口来维护其持久性。



注意

在存储节点不会进行身份验证。如果您能够连接到其中一个端口中的存储节点，您可以在不进行身份验证的情况下访问或修改数据。为了帮助缓解此问题，您应该遵循之前使用私有存储网络的建议。

11.3.1. Object Storage 帐户术语

swift 帐户不是用户帐户或凭据。存在以下区别：

- Swift 帐户 - 容器的集合（而不是用户帐户或身份验证）。您使用的验证系统将确定哪些用户与帐户相关联以及如何访问该帐户。
- Swift 容器 - 对象的集合。容器的元数据可用于 ACL。ACL 的使用取决于所使用的身份验证系统。
- Swift 对象 - 实际数据对象。对象级别的 ACL 也可用于元数据，并且取决于所使用的身份验证系统。

在每个级别，您具有控制用户访问的 ACL；根据所使用的身份验证系统进行解释 ACL。最常见的身份验证提供程序类型是 Identity Service (keystone)；自定义身份验证提供程序也可用。

11.4. 保护代理服务

代理节点应至少有两个接口（物理或虚拟）：一个公共和一个私有接口。您可以使用防火墙或服务绑定来帮助保护公共接口。面向公众的服务是一种 HTTP Web 服务器，用于处理端点客户端请求、验证它们并执行适当的操作。私有接口不需要任何侦听的服务，而是用来建立与私有存储网络上的存储节点的传出连

接。

11.4.1. HTTP 侦听端口

director 将 Web 服务配置为在非 root 用户（没有 UID 0）用户下运行。使用大于 **1024** 的端口号，避免以 root 用户身份运行任何部分 web 容器。通常，使用 HTTP REST API （和执行自动身份验证）的客户端会检索从身份验证响应所需的完整 REST API URL。OpenStack REST API 允许客户端向一个 URL 进行身份验证，然后重定向为为实际服务使用完全不同的 URL。例如：客户端可以对 **https://identity.cloud.example.org:55443/v1/auth** 进行身份验证，并使用其身份验证密钥和存储 URL （代理节点或负载均衡器的 URL）获得响应。**https://swift.cloud.example.org:44443/v1/AUTH_8980**

11.4.2. 负载均衡器

如果无法使用 Apache 的选项不可行，或者您想要卸载 TLS 工作的性能，则可能会使用专用的网络设备负载均衡器。这是使用多个代理节点时提供冗余和负载平衡的常见方法。

如果您选择卸载 TLS，请确保负载均衡器和代理节点之间的网络链接位于私有(V) LAN 段中，以便网络中的其他节点（可能被破坏）不能有线tap (sniff)未加密的流量。如果发生此类漏洞，攻击者就可以访问端点客户端或云管理员凭证，并访问云数据。

您使用的身份验证服务将决定如何在对端点客户端的响应中配置不同的 URL，允许它们使用负载均衡器而不是单独的代理节点。

11.5. 对象存储身份验证

Object Storage (swift)使用 WSGI 模型来为中间件功能提供，不仅提供通用可扩展性，也用于对端点客户端进行身份验证。身份验证提供程序定义存在哪些角色和用户类型。有些使用传统的用户名和密码凭证，另一些可能会利用 API 密钥令牌甚至客户端 x.509 证书。自定义供应商可以使用自定义中间件进行集成。

对象存储默认附带两个身份验证中间件模块，其中任一模块可用作开发自定义身份验证中间件的示例代码。

11.5.1. Keystone

Keystone 是 OpenStack 中常用的身份提供程序。它还可用于在对象存储中进行身份验证。

11.6. 加密 AT-REST SWIFT 对象

Swift 可以与 Barbican 集成，以透明加密和解密存储(at-rest)对象。at-rest 加密与传输中的加密不同，引用在磁盘上存储的同时加密的对象。

Swift 以透明方式执行这些加密任务，对象在上传到 swift 时自动加密，然后在提供给用户时自动解密。此加密和解密使用同样的(symmetric)密钥进行，该密钥存储在 Barbican 中。

如需更多信息，请参阅 Barbican 集成指南： https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/manage_secrets_with_openstack_key_manager/

11.7. 其他项目

在每个节点上的 **/var/lib/config-data/puppet-generated/swift/etc/swift/swift.conf** 中，有一个 **swift_hash_path_prefix** 设置和 **swift_hash_path_suffix** 设置。提供了它们，以减少所存储对象的哈希冲突几率，一个用户会覆盖另一个用户的数据。

这个值应该最初使用加密安全随机数字生成器设置，并在所有节点中保持一致。确保它具有正确的 ACL，并且您有备份副本以避免数据丢失。

第 12 章 监控和日志记录

日志管理是监控 OpenStack 部署安全状态的重要组件。除了组成您的 OpenStack 部署的组件活动外，日志还为您提供与管理员、项目和实例的 BAU 操作相关的见解。

日志不仅对主动的安全性和持续合规活动至关重要，而且它们也是调查和事件响应的宝贵信息来源。例如，分析 keystone 访问日志可能会提醒您登录失败、其频率、原始 IP 以及事件是否仅限于选择帐户，以及其他信息。

director 包括使用 AIDE 的入侵检测功能，以及 keystone 的 CADF 审计。如需更多信息，请参阅 director 强化章节。

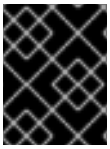
12.1. 强化监控基础架构

集中式日志记录系统是入侵者的高值目标，因为成功漏洞可能会允许他们通过记录的事件清除或篡改。建议您使用以下命令强化监控平台。此外，请考虑定期备份这些系统，并在出现中断或 DoS 时进行故障转移规划。

12.2. 要监控的事件示例

事件监控是一种更主动的方法来保护环境，提供实时检测和响应。存在多个工具，有助于监控。对于 OpenStack 部署，您需要监控硬件、OpenStack 服务和云资源的使用情况。

本节描述了您可能需要了解的一些示例事件。



重要

此列表并不完整。您需要考虑可能应用到特定网络的额外用例，并且可能会认为异常行为。

- 检测没有日志生成事件是高值的事件。这种差距可能会指示服务失败，甚至是临时关闭日志记录或修改日志级别以隐藏其跟踪的入侵者。
- 未调度的应用程序事件（如启动或停止事件）可能会产生安全隐患。
- OpenStack 节点上的操作系统事件，如用户登录或重启。它们可以为系统的正确使用和不当使用提供有价值的洞察力。
- 网络桥接关闭。由于服务中断的风险，这是一个可操作的事件。
- Compute 节点上的 iptables 刷新事件，并导致实例无法访问。

为了减少用户、项目或域删除用户、项目或域删除中的安全风险，讨论了系统中的通知，并让 OpenStack 组件根据终止实例、断开 CPU 和存储资源等情况来响应这些事件。

安全监控控制，如入侵检测软件、防病毒软件和清除软件检测和删除工具可生成日志，显示攻击或入侵发生的时间和方式。这些工具可以在 OpenStack 节点上部署时提供一层保护。项目用户可能还想在其实例上运行此类工具。

第 13 章 项目的数据隐私

OpenStack 旨在支持具有不同数据要求的项目之间的多租户。云操作员需要考虑其适用的数据隐私问题和法规。本章解决了数据所驻留的方面，并对 OpenStack 部署进行处理。

13.1. 数据隐私问题

本节介绍 OpenStack 部署中数据隐私的一些问题。

13.1.1. 数据驻留

过去数年来，数据隐私和隔离一直被认为是云采用的主要障碍。对于拥有云数据以及云操作员是否最终信任此数据的关注，过去是否有严重的问题。

某些 OpenStack 服务有权访问属于项目的数据和元数据，或引用项目信息。例如，存储在 OpenStack 云中的项目数据可能包括以下项目：

- 对象存储对象.
- 计算实例临时文件系统存储.
- 计算实例内存.
- 块存储卷数据。
- 用于计算访问的公钥。
- 镜像服务中的虚拟机镜像。
- 实例快照.
- 传递给 Compute configuration-drive 扩展的数据。

由 OpenStack 云存储的元数据包括以下项目（此列表不是正确的）：

- 组织名称.
- 用户的"名称"。
- 正在运行的实例、存储桶、对象、卷和其他与配额相关的项目的数量或大小。
- 运行实例或存储数据的小时数。
- 用户的 IP 地址。
- 用于计算镜像绑定的内部生成的私钥。

13.1.2. 数据分布

良好做法建议，Operator 必须清理云系统介质（数字和非数字），然后再分发机构控制前，或者在发布重复使用前前清理云系统介质。隔离方法应该根据信息的特定安全域和敏感度实施适当的强度和完整性。

注意

NIST Special Publication 800-53 Revision 4 在这个主题上使用一个特定的视图：

> The sanitization process removes information from the media such that the information cannot be retrieved or reconstructed. Sanitization techniques, including clearing, purging, cryptographic erase, and destruction, prevent the disclosure of information to unauthorized individuals when such media is reused or released for disposal.

根据 NIST 推荐的安全控制，在开发常规数据分布和清理指南时，云操作员应考虑以下问题：

- 跟踪、文档和验证媒体清理和处理操作。
- 测试清理设备和程序以验证性能。
- 在将这些设备连接到云基础架构之前，可以清理可移植的可移动存储设备。
- 销毁无法清理的云系统介质。

因此，OpenStack 部署需要解决以下做法（其他位置）：

- 保护数据纠删代码
- 实例内存清理
- 块存储卷数据
- 计算实例临时存储
- 裸机服务器清理

13.1.3. 没有安全擦除的数据

在 OpenStack 中，可能会删除一些数据，但无法象上述 NIST 标准上下文中那样安全地清除。这通常适用于存储在数据库中的最大或全部定义元数据和信息。这可能修复数据库和/或系统配置，以自动审查和定期释放空间。

13.1.4. 实例内存清理

特定于各种虚拟机监控程序是实例内存的处理。此行为在计算中定义，但它通常是在实例创建时（或创建实例）时清理内存的最佳 hypervisor。

13.1.5. 加密 Cinder 卷数据

强烈建议使用 OpenStack 卷加密功能。这在卷加密下的 Data Encryption 部分中讨论。当使用这个功能时，通过安全地删除加密密钥来完成数据破坏性。最终用户可以在创建卷时选择此功能，但请注意，管理员必须首先执行一次性设置卷加密功能。

如果没有使用 OpenStack 卷加密功能，其他方法通常更难以启用。如果使用后端插件，则可能会独立执行加密或非标准覆盖解决方案。OpenStack 块存储的插件将以各种方式存储数据。许多插件特定于供应商或技术，而其他插件是围绕文件系统（如 LVM 或 ZFS）的更多 DIY 解决方案。在插件、供应商和文件系统之间，安全销毁数据的方法会有所不同。

有些后端（如 ZFS）将支持写时复制以防止数据泄露。在这些情况下，从未写入的块读取始终为零。其他后端（如 LVM）可能不会原生支持此功能，因此 cinder 插件在将它们传递给用户之前负责覆盖之前写入的块。务必要检查您选择的卷后端提供哪些保证，并查看未提供这些保证可能可用的补救。

13.1.6. 镜像服务延迟删除功能

镜像服务具有延迟删除功能，该功能将在定义的时间段内擦除镜像。如果此行为是一个安全问题，请考虑禁用此功能；您可以通过编辑 **glance-api.conf** 文件并将 **delayed_delete** 选项设置为 **False** 来执行此操作。

13.1.7. 计算软删除功能

Compute 具有 soft-delete 功能，它允许在定义的时间段内删除处于软删除状态的实例。此时间段内可以恢复实例。要禁用 soft-delete 功能，请编辑 **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** 文件，并将 **reclaim_instance_interval** 选项留空。

13.1.8. Compute 实例的临时存储

请注意，OpenStack Ephemeral disk encryption 功能提供了一种改进临时存储隐私和隔离的方法，在活跃使用期间以及数据被销毁时。与加密块存储一样，只需删除加密密钥以有效地销毁数据。

在临时存储创建和销毁过程中提供数据隐私的替代措施将依赖于所选的虚拟机监控程序和计算插件。

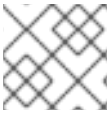
计算的 libvirt 驱动程序可能直接在文件系统上或 LVM 中维护临时存储。文件系统存储通常不会在删除数据时覆盖数据，尽管没有向用户置备脏扩展。

当使用基于块的 LVM 支持的临时存储时，计算软件安全清除块以防止信息泄漏。在以前的版本中，有与不正确的临时块存储设备相关的安全漏洞信息。

文件系统存储是一种比 LVM 作为脏扩展的临时块存储设备更安全的解决方案，无法向用户置备。但是，务必要注意用户数据不会被销毁，因此建议加密后备文件系统。

13.2. 裸机配置的安全强化

对于裸机置备基础架构，您应该考虑安全强化基板管理控制器(BMC)，特别是 IPMI。例如，您可以在 provisioning 网络中隔离这些系统，配置非默认和强大的密码，以及禁用不需要的管理功能。如需更多信息，请参阅厂商强化这些组件的指导。



注意

如果可能，请考虑评估基于 Redfish 的 BMC 而不是旧的 BMC。

13.2.1. 硬件识别

在部署服务器时，可能没有可靠的方法将其与攻击者的服务器区分开来。这个功能可能依赖于硬件/BMC 进行某种程度，但通常它似乎没有内置在服务器中识别的方法。

13.3. 数据加密

存在 选项，用于加密项目数据（无论其存储在磁盘上或通过网络传输），如下面描述的 OpenStack 卷加密功能。在将自己的数据发送到其供应商之前，这比一般建议用户对自己的数据进行加密。

代表项目加密数据的重要性主要与攻击者可以访问项目数据的供应商承担的风险相关。政府可能提出要求，以及每个策略的要求，在私人合同中，甚至针对公共云提供商的私人合同而遵守相关的要求。在选择项目加密策略前，请考虑获取风险评估和法律建议。

每个实例或每个对象加密更首选，按项目降序、每个项目、每主机和云聚合进行降序排列。本建议与实施的复杂性和难度相反。目前，在某些项目中，很难或不可能实施加密，如每个项目也是如此。实施者应该优先考虑加密项目数据。

通常，数据加密与可靠地销毁项目和每个实例数据（只要丢弃密钥）紧密相关。请注意，在这样做时，以可靠和安全的方式销毁这些密钥非常重要。

存在为用户加密数据的机会：

- Object Storage 对象
- 网络数据

13.3.1. 卷加密

OpenStack 中的卷加密功能支持每个项目的隐私。支持以下功能：

- 创建和使用加密卷类型，通过仪表板或命令行界面启动
- 启用加密并选择参数，如加密算法和密钥大小
- iSCSI 数据包中包含的卷数据是加密的
- 如果原始卷已加密，支持加密备份
- 显示卷加密状态的仪表板。包括表示卷已加密，并包含加密参数，如算法和密钥大小
- 使用密钥管理服务接口

13.3.2. 临时磁盘加密

临时磁盘加密功能可以解决数据隐私。临时磁盘是虚拟主机操作系统使用的临时工作空间。如果没有加密，可以在这个磁盘上访问敏感用户信息，并在卸载磁盘后保留 vestigial 信息。支持以下临时磁盘加密功能：

13.3.2.1. 创建和使用加密的 LVM 临时磁盘

注意

Compute 服务目前只支持 LVM 格式的加密临时磁盘。

计算配置文件(**/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf**)在 **ephemeral_storage_encryption** 部分中具有以下默认参数：

- **cipher = aes-xts-plain64** - 此字段设置用于加密临时存储的密码和模式。NIST 建议为磁盘存储特别推荐的 AES-XTS，使用 XTS 加密模式用于 AES 加密的名称。可用的密码取决于内核支持。在命令行中，键入 **cryptsetup benchmark** 以确定可用选项（以及查看基准测试结果）或转至 /proc/crypto。
- **enabled = false** - 要使用临时磁盘加密，请设置 option: **enabled = true**
- **key_size = 512** - 请注意，来自后端密钥管理器的密钥大小限制可能需要使用 **key_size = 256**，这仅提供 AES 密钥大小为 128 位。XTS 还需要自己的"tweak 密钥"，除了加密密钥 AES 之外。这通常表示为单个大密钥。在这种情况下，使用 512 位设置，256 位将由 AES 和 256 位 256 位用于 XTS。（请参阅 NIST）

13.3.2.2. 与密钥管理服务交互

密钥管理服务将通过基于每个项目提供临时磁盘加密密钥来支持数据隔离。后端密钥存储支持临时磁盘加密以增强安全性。使用密钥管理服务时，如果不再需要临时磁盘，只需删除密钥即可覆盖临时磁盘存储区域。

13.3.3. Object Storage 对象

Object Storage (swift)支持存储节点上静态的对象数据的可选加密。对象数据的加密旨在降低在未经授权方获得对磁盘的物理访问权限时，用户数据正在读取的风险。

静态数据的加密是由中间件实现的，该加密可能包括在代理服务器 WSGI 管道中。这个功能是 swift 集群的内部，而不是通过 API 公开。客户端不知道数据由此功能在内部向 swift 服务加密；内部加密数据不应通过 swift API 返回到客户端。

以下数据在 swift 中被加密：

- 对象内容，如对象 **PUT** 请求正文的内容。
- 具有非零内容的对象的实体标签(**ETag**)。
- 所有自定义用户对象元数据值。例如，使用带有 **X-Object-Meta-** 前缀的标头的 **PUT** 或 **POST** 请求发送元数据。

以上列表中不包含的任何数据或元数据都不会加密，包括：

- 帐户、容器和对象名称
- 帐户和容器自定义用户元数据值
- 所有自定义用户元数据名称
- 对象内容类型值
- 对象大小
- 系统元数据

13.3.4. 块存储性能和后端

启用操作系统时，可以使用 Intel 和 AMD 处理器中当前可用的硬件加速功能来增强 OpenStack 卷加密性能。OpenStack 卷加密功能和 OpenStack Ephemeral Disk Encryption 功能都使用 **dm-crypt** 保护卷数据。**DM-crypt** 是 Linux 内核版本 2.6 及更高版本中的透明磁盘加密功能。在使用硬件加速时，两个加密功能的性能影响会最小化。

13.3.5. 网络数据

Compute 节点的项目数据可以通过 IPsec 或其他隧道加密。这种做法在 OpenStack 中不是常见或标准，而是可用于动机和感兴趣的实施人员的选项。同样，加密的数据在网络上传输时仍然加密。

13.4. 密钥管理

为了解决项目数据隐私经常提及的问题，OpenStack 社区中会非常关注，使数据加密更为普遍。最终用户在将其数据保存到云之前比较简单，而这是项目对象（如媒体文件、数据库存档等）的可行路径。在某些情况下，客户端加密用于加密由需要客户端交互的虚拟化技术（如显示密钥）保存的数据，以解密数据以供以后使用。

barbican 可以帮助项目更无缝地加密数据，并使其可以访问，而无需给用户带来密钥管理。作为 OpenStack 的一部分提供加密和密钥管理服务，可简化数据恢复的采用，并有助于解决客户对隐私或滥用数据的问题。

卷加密和临时磁盘加密功能依赖于密钥管理服务（如 barbican）来创建和安全强化密钥的存储。

第 14 章 管理实例安全

在虚拟化环境中运行实例的一个好处是安全控制的新机会，这些控制在部署到裸机时通常不可用。某些技术可以应用于虚拟化堆栈，从而提高 OpenStack 部署的信息保证。具有强大安全要求的 Operator 可能需要考虑部署这些技术，但并不适用于所有情况。在某些情况下，因为存在明确的业务需求，可能会排除在云中使用技术。同样，一些技术会检查实例数据，如运行状态，这可能对系统的用户不必要。

本章介绍了这些技术以及可用于帮助提高实例或底层节点的安全性的情况。也会突出显示可能的隐私问题，其中包括数据直通、内省或熵源。

14.1. 为实例提供熵

本章使用术语 *熵* 来指代实例可用的随机数据的质量和来源。加密技术通常依赖于随机性，这要求从高质量的熵池中提取。实例通常难以获得足够的熵来支持这些操作；这称为熵。此条件可以在实例中作为看似不相关的操作而定。例如，引导时间较慢的原因可能是等待 SSH 密钥生成的实例。此条件还可能会使用户从实例内部使用较差质量熵源的风险，从而使应用程序在云中运行的总体安全。

需要的是，您可以通过为实例提供高质量熵来源来帮助解决这些问题。这可以通过在云中有足够的硬件随机数生成器(HRNG)来支持实例。在这种情况下，有足够的内容特定于域。对于日常操作，现代 HRNG 可能会生成足够的熵来支持 50-100 计算节点。高带宽 HRNG，如 Intel Ivy Bridge 和较新的处理器可用的 RdR 和指令可能会处理更多节点。对于给定的云，架构师需要了解应用要求，以确保有足够的熵。

Virtio RNG 是一个随机数字生成器，默认使用 **/dev/random** 作为熵的来源。它还可以配置为使用硬件 RNG 或熵收集守护进程(EGD)等工具来提供一种通过部署公平地分发熵的方法。您可以使用 **hw_rng** 元数据属性在实例创建时启用 Virtio RNG。

14.2. 将实例调度到节点

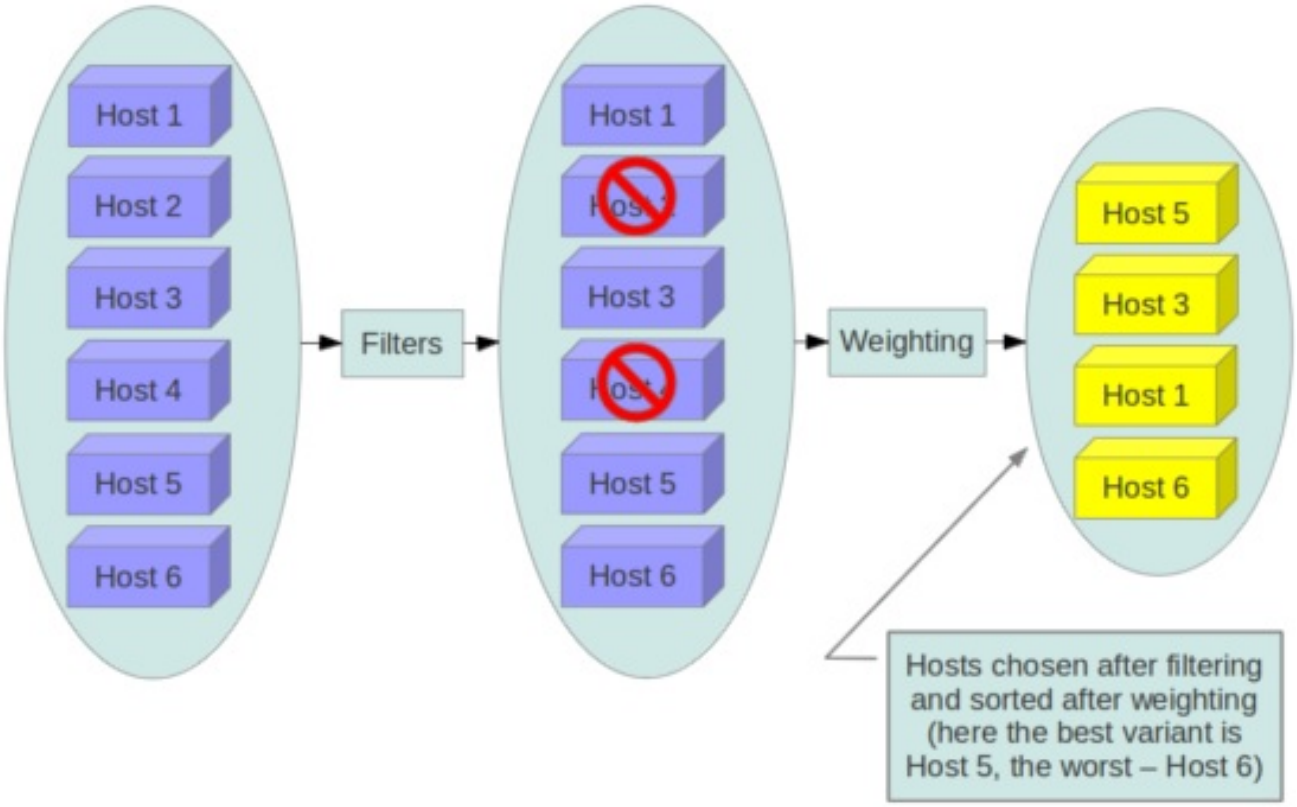
在创建实例之前，必须选择镜像实例化的主机。此选择由 **nova-scheduler** 执行，它决定了如何分配计算和卷请求。

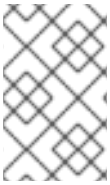
FilterScheduler 是计算的默认调度程序，但存在其他调度程序。这个功能与 *filter hints* 协同工作，用于确定应当启动实例的位置。通过这种主机选择流程，管理员可以满足许多不同的安全性和合规性要求。如果数据隔离是主要关注，您可以选择让项目实例尽可能驻留在同一主机上。相反，出于可用性或容错原因，您可以尝试使实例驻留在许多不同的主机上。

过滤器调度程序位于以下主要类别下：

- *基于资源的过滤器* - 根据虚拟机监控程序主机集的系统资源使用情况确定实例的放置，并可以在空闲或已用的属性上触发，如 RAM、IO 或 CPU 使用率。
- *基于镜像的过滤器* - 根据所使用的镜像元数据创建实例，如使用的虚拟机或镜像类型的操作系统。
- *基于环境的过滤器* - 根据外部详细信息（如特定 IP 范围、跨可用区）或与另一个实例相同的主机上确定实例的放置。
- *自定义条件* - 根据用户或管理员提供标准（如信任或元数据解析）创建实例创建。

可以同时应用多个过滤器。例如，**ServerGroupAffinity** 过滤器检查是否在特定主机的成员上创建实例，并且 **ServerGroupAntiAffinity** 过滤器检查是否没有在另一组主机上创建相同的实例。请注意，这两个过滤器通常同时启用，并且永远不会相互冲突，因为它们每次检查给定属性的值时都不能为 true。





注意

DiskFilter 过滤器能够过量处理磁盘空间。虽然通常没有问题，但这可能会对精简置备的存储设备有问题。此过滤器应当与应用经过良好测试的配额一起使用。此功能已被弃用，不应在 Red Hat OpenStack Platform 12 后使用。

重要

考虑禁用解析用户提供的对象的过滤器，或者可以操作（如元数据）。

14.3. 使用可信镜像

在云环境中，用户处理预安装的镜像或他们上传的映像。在这两种情况下，用户都应该能够确保它们使用的镜像没有被篡改。能够验证镜像是安全性的基础。需要信任链，从镜像源到使用它的目的地。这可以通过对从可信源获取的镜像签名，并在使用前验证签名来实现。下面将讨论获取和创建验证的镜像的各种方法，然后介绍镜像签名验证功能。

14.3.1. 创建镜像

OpenStack 文档提供如何创建和将镜像上传到镜像服务的指导。另外，假设您有一个安装和强化客户机操作系统的过程。以下项目将提供有关如何将镜像传输到 OpenStack 的其他指导。获取镜像有多种选项。每种方法均有特定的步骤，可以帮助验证镜像的来源。

- 选项 1：包含来自可信源的引导介质.例如，您可以从官方的红帽源下载镜像，然后执行额外的 checksum 验证。

- 选项 2：使用 OpenStack 虚拟机镜像指南。在这种情况下，您希望遵循您的组织操作系统强化指南。
- 选项 3 ：使用自动化镜像构建器。以下示例使用 Oz 镜像构建器。OpenStack 社区最近创建了名为 **disk-image-builder** 的较新工具，它尚未遭遇安全评估。

在本例中，**RHEL 6 CCE-26976-1** 帮助在 Oz 中实施 NIST 800-53 第 AC-19 (d)。

```
<template>
<name>centos64</name>
<os>
  <name>RHEL-6</name>
  <version>4</version>
  <arch>x86_64</arch>
  <install type='iso'>
  <iso>http://trusted_local_iso_mirror/isos/x86_64/RHEL-6.4-x86_64-bin-DVD1.iso</iso>
  </install>
  <rootpw>CHANGE THIS TO YOUR ROOT PASSWORD</rootpw>
</os>
<description>RHEL 6.4 x86_64</description>
<repositories>
  <repository name='epel-6'>
  <url>http://download.fedoraproject.org/pub/epel/6/$basearch</url>
  <signed>no</signed>
  </repository>
</repositories>
<packages>
  <package name='epel-release'/>
  <package name='cloud-utils'/>
  <package name='cloud-init'/>
</packages>
<commands>
  <command name='update'>
  yum update
  yum clean all
  sed -i '/^HWADDR/d' /etc/sysconfig/network-scripts/ifcfg-eth0
  echo -n > /etc/udev/rules.d/70-persistent-net.rules
  echo -n > /lib/udev/rules.d/75-persistent-net-generator.rules
  chkconfig --level 0123456 autofs off
  service autofs stop
  </command>
</commands>
</template>
```

考虑避免手动镜像构建过程，因为它很复杂且容易出错。此外，使用 Oz 等自动化系统进行镜像构建，或用于启动后镜像强化的配置管理实用程序（如 Chef 或 Puppet），使您能够生成一致的镜像，以及跟踪基础镜像合规性到其相应的强化指南。

如果订阅公共云服务，您应该与云供应商进行检查，以了解用于生成其默认镜像的流程概述。如果供应商允许您上传自己的镜像，则需要确保在使用它来创建实例之前验证您的镜像没有被修改。要做到这一点，请参考以下 _ 验证镜像签名_ 一节，或者以下段落（如果无法使用签名）。

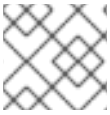
镜像服务(glance)用于将镜像上传到节点上的 Compute 服务。此传输应该通过 TLS 进一步强化。镜像位于节点上后，会使用基本校验和进行检查，然后根据要启动的实例大小扩展其磁盘。如果稍后一次启动同一镜像，在此节点上使用相同的实例大小，它将从同一扩展的镜像启动。由于在启动前，这个扩展的镜像

不会被重新验证，因此存在风险。除非在生成的镜像中手动检查文件，否则用户不会了解这个用户。要帮助缓解这个问题，请参阅有关验证镜像签名的主题部分。

14.3.2. 验证镜像签名

与镜像签名相关的某些功能现在包括在 OpenStack 中。从 Red Hat OpenStack Platform 13 开始，镜像服务可以验证这些签名的镜像，并提供完整的信任链，计算服务可以选择在镜像引导前执行镜像签名验证。在镜像引导前成功签名验证可确保签名的镜像没有改变。启用此功能后，可以检测到对镜像的未授权修改（例如，将镜像修改为包含恶意软件或 rootkits）。

您可以通过在 **/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** 文件中将 **verify_glance_signatures** 标志设置为 **True** 来启用实例签名验证。启用后，计算服务会在从 glance 检索时自动验证签名的实例。如果这个验证失败，引导过程不会启动。



注意

启用这个功能后，没有签名（未签名镜像）的镜像也会失败验证，引导过程不会启动。

14.4. 迁移实例

OpenStack 和底层虚拟化层为 OpenStack 节点之间的镜像实时迁移提供，允许您无缝地执行 Compute 节点的滚动升级，而无需实例停机时间。但是，实时迁移也存在重大风险。要了解涉及的风险，以下是实时迁移过程中执行的高级步骤：

1. 在目标主机上启动实例
2. 转让内存
3. 停止客户机和同步磁盘
4. 传输状态
5. 启动客户机

注意

某些操作（如冷迁移、调整大小和她）都会导致在网络上将实例的数据传送到其他服务。

14.4.1. 实时迁移风险

在实时迁移过程的不同阶段，实例的运行时和磁盘的内容通过纯文本通过网络传输。因此，在使用实时迁移时需要解决多个风险。以下非排除列表详细介绍了其中一些风险：

- 拒绝服务(DoS)：如果迁移过程中出现某种失败，则实例可能会丢失。
- 数据泄露：必须安全地处理内存或磁盘传输。
- 数据操作：如果内存或磁盘传输没有被安全处理，则攻击者可以在迁移过程中操作用户数据。
- 代码注入：如果内存或磁盘传输没有被安全处理，则攻击者可以在迁移过程中操作可执行文件（在磁盘上或内存中）。

14.4.2. 实时迁移缓解方案

有多种方法可帮助缓解与实时迁移相关的一些风险。以下部分描述了它们：

14.4.2.1. 禁用实时迁移

目前，在 OpenStack 中默认启用实时迁移。默认情况下，实时迁移是仅限管理员的任务，因此用户无法启动此操作，只有管理员（可能可以被信任）。实时迁移可以通过将以下行添加到 nova **policy.json** 文件中来禁用：

```
"compute_extension:admin_actions:migrate": "!",
"compute_extension:admin_actions:migrateLive": "!",
```

或者，当阻止 TCP 端口 **49152** 到 **49261** 时，实时迁移可能会失败，或者确保 nova 用户在计算主机之间没有免密码 SSH 访问。

请注意，实时迁移的 SSH 配置被明显锁定：创建了一个新用户(nova_migration)，SSH 密钥仅限于该用户，且仅在白名单网络上使用。然后，打包程序脚本会限制可运行的命令（例如，在 libvirt 套接字上的 netcat）。

14.4.2.2. 迁移网络

实时迁移流量以纯文本形式传输正在运行的实例的磁盘和内存内容，并且当前默认托管在 Internal API 网络中。

14.4.2.3. 加密实时迁移

如果有足够的要求（如升级）来保持启用实时迁移，则 libvirtd 可以为实时迁移提供加密的隧道。但是，此功能不会在 OpenStack Dashboard 或 nova-client 命令中公开，只能通过 libvirtd 手动配置进行访问。然后，实时迁移过程会更改为以下高级别步骤：

1. 实例数据从 hypervisor 复制到 libvirtd。
2. 在源和目标主机上的 libvirtd 进程间创建了一个加密的隧道。
3. 目标 libvirtd 主机将实例复制到底层虚拟机监控程序。

注意

对于 Red Hat OpenStack Platform 13，建议使用隧道迁移，这在将 Ceph 用作后端时默认启用。如需更多信息，请参阅 https://docs.openstack.org/nova/queens/configuration/config.html#libvirt.live_migration_t

14.5. 监控、警报和报告

实例是能够在主机之间复制的服务器镜像。因此，最好在物理和虚拟主机之间应用日志。应该记录操作系统和应用程序事件，包括访问主机和数据的事件、用户添加和删除、特权更改等等。考虑将结果导出到收集日志事件的日志聚合器，与它们相关联进行分析，并存储它们以对其进行参考或进一步操作。一个常见的工具是 ELK 堆栈或 Elasticsearch、Logtash 和 Kibana。

注意

这些日志应定期检查，甚至在由网络操作中心(NOC)执行的实时视图中进行监控。

您需要进一步确定哪些事件将触发随后发送到操作响应器的警报。

如需更多信息，请参阅 监控工具配置指南

14.5.1. 更新和补丁

管理程序运行独立的虚拟机。此管理程序可以在操作系统中运行，也可直接在硬件（称为裸机）上运行。hypervisor 的更新不会传播到虚拟机。例如，如果部署正在使用 KVM，并且设置了 CentOS 虚拟机，则对 KVM 的更新将不会更新 CentOS 虚拟机上运行的任何内容。

考虑将虚拟机清除所有权分配给所有者，然后负责虚拟机的强化、部署和持续功能。您还应该有一个计划定期部署更新，同时首先在一个类似生产环境的环境中进行测试。

14.5.2. 防火墙和实例配置集

最常见的操作系统包括基于主机的防火墙，以实现额外的安全层。虽然实例应该尽可能运行一些应用程序（到单一用途实例的点，如果可能），但实例上运行的所有应用程序都应进行性能分析，以确定应用需要访问哪些系统资源，运行它所需的最低特权级别，以及预期网络流量将进入并来自于虚拟机。该预期流量应添加到基于主机的防火墙中，如允许的流量（或列入白名单），以及任何必要的日志记录和管理通信（如 SSH 或 RDP）。防火墙配置中应明确拒绝所有其他流量。

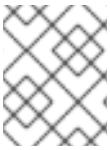
在 Linux 实例中，上面的应用程序配置文件可与 **audit2allow** 等工具一起使用，以构建 SELinux 策略来进一步保护大多数 Linux 发行版上的敏感系统信息。SELinux 结合使用用户、策略和安全上下文来划分应用运行所需的资源，并将其从不需要的其他系统资源进行分段。

注意

Red Hat OpenStack Platform 默认启用了 SELinux，策略为 OpenStack 服务自定义。考虑根据需要定期检查这些策略。

14.5.2.1. 安全组

OpenStack 为主机和网络提供安全组，为给定项目中的实例添加防御性。它们类似于基于主机的防火墙，因为它们根据端口、协议和地址允许或拒绝传入流量。但是，安全组规则仅应用于传入流量，而基于主机的防火墙规则则可应用到传入和传出流量。主机和网络安全组规则也可以发生，以冲突并拒绝合法流量。考虑检查是否正确为正在使用的网络配置了安全组。如需了解更多详细信息，请参阅本指南中的安全组。



注意

除非特别需要禁用它们，否则您应该启用安全组和端口的安全性。要基于防御性方法进行构建，建议您对实例应用粒度规则。

14.5.3. 访问实例控制台

默认情况下，实例的控制台可以通过虚拟控制台远程访问。这对于故障排除非常有用。Red Hat OpenStack Platform 使用 VNC 进行远程控制台访问。

- 考虑使用防火墙规则锁定 VNC 端口。默认情况下，**nova_vnc_proxy** 使用 **6080** 和 **13080**。
- 确认 VNC 流量由 TLS 加密。对于基于 director 的部署，以 **UseTLSTransportForVnc** 开始。

14.5.4. 证书注入

如果需要 SSH 到实例，您可以将 Compute 配置为在创建时自动将所需的 SSH 密钥注入到实例中。

如需更多信息，请参阅 https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/instances_and_images_guide/#section-create-images

第 15 章 消息排队

消息队列服务有助于 OpenStack 中的进程间通信。这可以通过这些消息队列服务后端完成：

- RabbitMQ - Red Hat OpenStack Platform 默认使用 RabbitMQ。
- qpid

RabbitMQ 和 Qpid 都是高级消息队列协议(AMQP)框架，为对等通话提供消息队列。队列实施通常部署为集中式或分散的队列服务器池。

消息队列有效促进了跨 OpenStack 部署的命令和控制功能。允许访问队列后，不会执行进一步的授权检查。可以通过队列访问的服务会验证实际消息有效负载中的上下文和令牌。但是，您必须注意令牌的过期日期，因为令牌可能会重新显示，并可授权基础架构中的其他服务。

OpenStack 不支持消息级的信任，如消息签名。因此，您必须保护消息传输本身并进行身份验证。对于高可用性(HA)配置，您必须执行队列到队列身份验证和加密。

15.1. 消息传递安全性

本节讨论 OpenStack 中使用的三个最常见消息队列解决方案的安全强化方法：RabbitMQ 和 Qpid。

15.2. 消息传递传输安全性

基于 AMQP 的解决方案(Qpid 和 RabbitMQ)支持使用 TLS 的传输级别安全性。

考虑为您的消息队列启用传输级别加密。将 TLS 用于消息传递客户端连接可保护通信不受篡改，并将通信转换至消息传递服务器。下文包含有关如何为两个流行消息传递服务器配置 TLS 的下方： Qpid 和 RabbitMQ。在配置消息传递服务器用来验证客户端连接的可信证书颁发机构(CA)捆绑包时，建议只限于用于您的节点的 CA，最好是内部管理的 CA。可信 CA 的捆绑包将决定哪些客户端证书将被授权，并传递设置 TLS 连接的 client-server 验证步骤。

注意

安装证书和密钥文件时，请确保对文件权限进行了限制，例如使用 **chmod 0600**，并且所有权仅限于消息传递服务器上的消息传递服务器守护进程用户，以防止消息传递服务器上的其他进程和用户进行未经授权的访问。

15.2.1. RabbitMQ 服务器 SSL 配置

以下行应添加到系统范围的 RabbitMQ 配置文件中，通常为 **/etc/rabbitmq/rabbitmq.config** ：

```
[
  {rabbit, [
    {tcp_listeners, [] },
    {ssl_listeners, [{"<IP address or hostname of management network interface>", 5671}] },
    {ssl_options, [{cacertfile,"/etc/ssl/cacert.pem"},
                   {certfile,"/etc/ssl/rabbit-server-cert.pem"},
                   {keyfile,"/etc/ssl/rabbit-server-key.pem"},
                   {verify,verify_peer},
                   {fail_if_no_peer_cert,true}]}
  ]}
].
```

注意

tcp_listeners 选项被设置为 **[]**，以防止其侦听非 SSL 端口。**ssl_listeners** 选项应限制为仅侦听服务的管理网络。

15.3. 队列身份验证和访问控制

RabbitMQ 和 Qpid 提供身份验证和访问控制机制，用于控制对队列的访问。

简单身份验证和安全层(SASL)是 Internet 协议中身份验证和数据安全性的框架。除简单的用户名和密码外，RabbitMQ 和 Qpid 均提供 SASL 和其他可插入身份验证机制，从而提高安全性。虽然 RabbitMQ 支持 SASL，但 OpenStack 中的支持目前不允许请求特定的 SASL 身份验证机制。OpenStack 中的 RabbitMQ 支持通过未加密的连接或用户名和密码加 X.509 客户端证书进行用户名和密码身份验证，以建立安全的 TLS 连接。

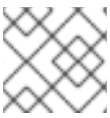
考虑在所有 OpenStack 服务节点上为与消息传递队列的客户端连接配置 X.509 客户端证书，并在可能的情况下（目前只是 Qpid）通过 X.509 客户端证书进行身份验证。在使用用户名和密码时，应为每个服务和节点创建帐户，以便精细审核对队列的访问。

在部署前，请考虑排队服务器使用的 TLS 库。Qpid 使用 Mozilla 的 NSS 库，而 RabbitMQ 使用 Erlang 的 TLS 模块（使用 OpenSSL）。

15.3.1. RabbitMQ 的 OpenStack 服务配置

本节介绍 OpenStack 服务的典型 RabbitMQ 配置：

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_kombu
rabbit_use_ssl = True
rabbit_host = RABBIT_HOST
rabbit_port = 5671
rabbit_user = compute01
rabbit_password = RABBIT_PASS
kombu_ssl_keyfile = /etc/ssl/node-key.pem
kombu_ssl_certfile = /etc/ssl/node-cert.pem
kombu_ssl_ca_certs = /etc/ssl/cacert.pem
```



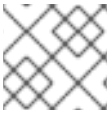
注意

将 **RABBIT_PASS** 替换为合适的密码。

15.3.2. Qpid 的 OpenStack 服务配置

本节介绍 OpenStack 服务的典型 Qpid 配置：

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_qpid
qpid_protocol = ssl
qpid_hostname = <IP or hostname of management network interface of messaging server>
qpid_port = 5671
qpid_username = compute01
qpid_password = QPID_PASS
```



注意

将 **QPID_PASS** 替换为合适的密码。

另外，如果将 SASL 与 Qpid 搭配使用，则通过添加来指定 SASL 机制：

```
qpid_sasl_mechanisms = <space separated list of SASL mechanisms to use for auth>
```

15.4. 消息队列进程隔离和策略

每个项目提供许多服务来发送和接收消息。每个发送消息的二进制文件都应该使用来自队列的消息（如果仅回复）。

消息队列服务进程应相互隔离，以及计算机上的其他进程。

15.4.1. 命名空间

Neutron 和 Open vSwitch (OVS)网络在命名空间中运行，但某些 OVS 主机服务没有，包括 vswitchd、libvirtd 和 QEMU。在运行容器化 control plane 时，所有 control plane 服务都默认在网络命名空间中运行。强烈建议在计算虚拟机监控程序上运行的所有服务（因为它们运行 AMQP 服务）都强烈建议使用网络命名空间。这种方法有助于防止在实例和管理网络之间桥接网络流量。

第 16 章 强化 API 端点

与 OpenStack 云互动的过程从查询 API 端点开始。虽然公共和私有端点有不同的挑战，但这些都是高价值资产，在被破坏时可能会带来巨大的风险。

本章推荐对面向公共和私有的 API 端点的安全性增强。

16.1. API 端点配置建议

本节论述了强化 API 端点的建议。

16.1.1. 内部 API 通信

OpenStack 同时提供面向公共的、内部管理员和私有 API 端点。默认情况下，OpenStack 组件使用公开定义的端点。建议将这些组件配置为在正确的安全域中使用 API 端点。内部管理端点允许对 keystone 进一步提升访问，因此可能需要进一步隔离它。

服务根据 OpenStack 服务目录选择对应的 API 端点。这些服务可能没有遵循列出的公共或内部 API 端点值。这会导致内部管理流量路由到外部 API 端点。

16.1.2. 在 Identity 服务目录中配置内部 URL

Identity 服务目录应该了解您的内部 URL。虽然默认不使用此功能，但可以通过 配置来提供。另外，当此行为成为默认值后，它应该与预期更改兼容。

考虑将配置的端点与网络级别隔离，只要它们具有不同的访问级别。Admin 端点可供云管理员访问，因为它提供了对内部或公共端点不支持的 keystone 操作的升级访问。内部端点设计为使用云内部（例如，OpenStack 服务），通常无法在部署网络外访问。公共端点应启用 TLS，并且部署外的唯一 API 端点供云用户操作。

director 自动注册端点的内部 URL。如需更多信息，请参阅 https://github.com/openstack/tripleo-heat-templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map

16.1.3. 为内部 URL 配置应用程序

您可以强制某些服务使用特定的 API 端点。因此，建议任何联系另一个服务的 API 的 OpenStack 服务都必须明确配置，才能访问正确的内部 API 端点。

每个项目可能出现定义目标 API 端点不一致的方法。OpenStack 的未来版本希望通过使用 Identity 服务目录一致地解析这些不一致的情况。

16.1.4. 粘贴和中间件

OpenStack 中的大多数 API 端点和其他 HTTP 服务都使用 Python Paste Deploy 库。从安全角度来说，这个库支持通过应用程序的配置操作请求过滤器管道。这个链中的每个元素都被称为中间件。在管道中更改过滤顺序或添加额外的中间件可能会产生无法预计的安全影响。

通常，实施者添加中间件来扩展 OpenStack 的基本功能。考虑谨慎考虑通过向 HTTP 请求管道添加非标准软件组件所引入的潜在风险。

16.1.5. API 端点进程隔离和策略

您应该隔离 API 端点进程，特别是驻留在公共安全域中的进程应该尽可能被隔离。在部署允许的情况下，API 端点应部署到单独的主机上，以增加隔离。

16.1.6. 命名空间

Linux 使用命名空间将进程分配到独立的域中。本指南的其他部分更详细地涵盖了系统划分。

16.1.7. 网络策略

API 端点通常跨越多个安全区，因此您必须特别注意 API 进程分离。例如，在网络设计级别上，您可以考虑只限制对指定系统的访问。如需更多信息，请参阅安全区的指导。

仔细建模时，您可以使用网络 ACL 和 IDS 技术来强制实施网络服务之间的明确点对点通信。作为关键的跨域服务，这种类型的显式实施也适用于 OpenStack 的消息队列服务。

要强制执行策略，您可以配置服务、基于主机的防火墙（如 iptables）、本地策略(SELinux)和可选的全局网络策略。

16.1.8. 强制访问控制

您应该将 API 端点进程相互隔离，以及机器上的其他进程。这些进程的配置应受到 Discretionary Access Controls (DAC)和强制访问控制(MAC)的限制。这些增强的访问控制的目标是帮助 API 端点安全破坏。

16.1.9. API 端点速率限制

速率限制是一种控制基于网络的应用程序接收的事件频率的方法。如果没有可靠的速率限制，则可能会导致应用程序容易受到各种拒绝服务攻击的影响。对于 API，这尤其适用，其性质设计为接受类似请求类型和操作的高频率。

建议所有端点（特别是公共）都提供额外的保护层，例如使用物理网络设计、速率限制代理或 Web 应用程序防火墙。

在配置并实施任何速率限制功能时，Operator 会仔细规划并考虑其 OpenStack 云中用户和服务的独立性能需求。

请注意，对于 Red Hat OpenStack Platform 部署，所有服务都放在负载均衡代理后面。

第 17 章 实施联邦

警告

红帽目前不支持联合。这个功能只应用于测试，不应部署在生产环境中。

17.1. 使用红帽单点登录与 IDM 联合

您可以使用 Red Hat Single Sign-On (RH-SSO)来联合您的 IdM 用户进行 OpenStack 身份验证(authN)。通过联邦，您的 IdM 用户无需向任何 OpenStack 服务公开凭据，即可登录 OpenStack 控制面板。相反，当 Dashboard 需要用户的凭证时，它会将用户转发到 Red Hat Single Sign-On (RH-SSO)，并允许他们在其中输入其 IdM 凭证。因此，RH-SSO 会返回到用户成功通过身份验证的 Dashboard，然后允许用户访问项目。

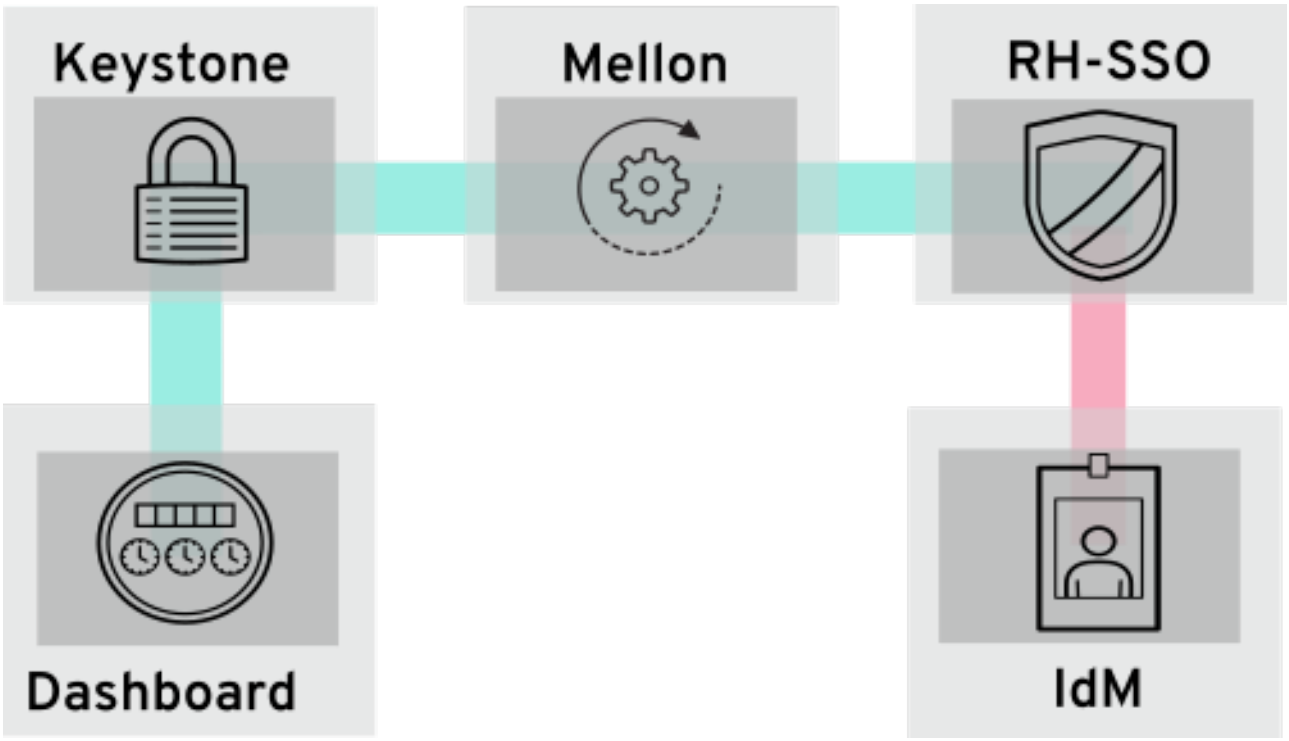
17.2. 联邦工作流

本节描述了 keystone、RH-SSO 和 IdM 如何相互交互。OpenStack 中的联邦使用身份提供程序和服务提供商的概念：

身份提供程序 (IdP)- 存储用户帐户的服务。在这种情况下，IdM 中保存的用户帐户使用 RH-SSO 向 Keystone 提供。

Service Provider (SP)- 需要从 IdP 中的用户进行身份验证的服务。在本例中，keystone 是授予 Dashboard 对 IdM 用户访问权限的服务提供商。

在下图中，keystone (SP)与 RH-SSO (IdP)通信，后者提供必要的 SAML2 WebSSO。RH-SSO 也可以作为其他 IdP 的通用适配器。在此配置中，您可以把 keystone 指向 RH-SSO，RH-SSO 将把请求转发到其支持的身份提供程序（称为身份验证模块），目前包括 IdM 和 Active Directory。这可以通过使用 Service Provider (SP)和身份提供程序(IdP)交换元数据来完成，然后为每个系统管理员做出信任决定。结果是 IdP 可以放心地做出断言，而 SP 然后可以接收这些断言。



如需更多信息，请参阅联邦指南： https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/13/html-single/federate_with_identity_service/