Red Hat OpenStack Services on OpenShift 18.0

执行安全操作

在 OpenShift 环境中的 Red Hat OpenStack Services 中操作安全服务

Last Updated: 2024-10-08

Red Hat OpenStack Services on OpenShift 18.0 执行安全操作

在 OpenShift 环境中的 Red Hat OpenStack Services 中操作安全服务

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

在 OpenShift 环境中的 Red Hat OpenStack Services 中备份和恢复安全信息、轮转密码和操作安全服务。

目录

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

在 JIRA 中提供文档反馈

使用 Create Issue 表单在 OpenShift (RHOSO)或更早版本的 Red Hat OpenStack Platform (RHOSP)上提供有关 Red Hat OpenStack Services 文档的反馈。当您为 RHOSO 或 RHOSP 文档创建问题时，这个问题将在 RHOSO Jira 项目中记录，您可以在其中跟踪您的反馈的进度。

要完成 Create Issue 表单，请确保您已登录到 JIRA。如果您没有红帽 JIRA 帐户，您可以在 https://issues.redhat.com 创建一个帐户。

1. 点击以下链接打开 **Create Issue** 页面： Create Issue
2. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节号以及问题的详细描述。不要修改表单中的任何其他字段。
3. 点 **Create**。

第 1 章 在 RHOSO 中为数据库帐户轮转用户名或密码

您可以通过编辑 **openstackcontrolplanes** 自定义资源来轮转 OpenShift (RHOSO)上 Red Hat OpenStack Services 中用于数据库帐户的用户名和密码。

1.1. 为数据库帐户生成一个新的用户名和密码

您可以通过编辑包含数据库帐户的模板的 YAML 文件，为数据库帐户生成新用户名和密码。数据库帐户与 MariaDBAccount YAML 资源对应。MariaDBAccount YAML 是指包含密码的实际数据库用户名和 Secret。MariaDBAccount 的名称以及 MariaDBAccount 指代的用户名不同。MariaDBAccount 是一个资源标识符，它使用短划线来隔离每个单词，MariaDB 用户名使用下划线来隔离每个单词。

当您更改 yaml 文件中的 databaseAccount: 参数的值时，这将创建一个新的 MariaDBAccount 资源，该资源具有随机生成的用户名和 secret，其中包含密码。此资源会创建一个 pod，它在目标数据库中创建实际 MariaDB 数据库用户名。

重要

为了避免中断的长时间运行的迁移，请使用维护窗口。例如，块存储(Cinder)卷迁移或计算服务(Nova)主机迁移。

流程

1. 要检索包含数据库详细信息的 YAML 文件的名称，请使用 **oc get openstackcontrolplanes** 命令。
2. 要访问 YAML 文件，请使用 **oc edit openstackcontrolplanes <YAML_File_Name>** 命令。
   - 将 <YAML_File_Name> 替换为在第 1 步中获取的 YAML 文件的名称。
3. 在 YAML 文件中，找到您要为其生成新用户名和密码的数据库帐户。
4. 在 **databaseAccount:** 标签旁边，更改数据库帐户的名称。自动生成新数据库帐户和密码，服务会自动切换到新的数据库帐户。

注意

您必须使用短划线分隔 MariaDBAccount 名称中的每个单词。

5. 可选： 要删除旧的数据库帐户，请使用 **oc delete mariadbaccounts <mariadbaccount_name>** 命令。MariaDBAccount 会一直保持，直到资源被更新为使用新的 MariaDBAccount。
   - 将 <mariadbaccount_name> 替换为您要删除的数据库帐户的名称。

第 2 章 在 OPENSHIFT 上的 RED HAT OPENSTACK SERVICES 中，基于角色的安全访问控制

在 OpenShift (RHOSO) 18.0 上对 Red Hat OpenStack Services (keystone)进行身份验证的所有用户都使用安全基于角色的访问控制(SRBAC)。SRBAC 是更新的基于角色的访问控制式身份验证模式，在所有 RHOSO 部署中都存在。现在，所有策略都使用这个模式。

2.1. RED HAT OPENSTACK SERVICES ON OPENSHIFT 中的 SRBAC

Keystone 使用用户角色（属于范围特定的角色）实施安全角色。您可以为项目或系统范围的用户或组分配角色。

2.1.1. 角色

角色定义用户可以执行的操作。项目和系统范围内提供了三个 OpenStack 角色。

admin

在任何范围内授予 admin 角色包括资源和 API 的所有创建、读取、更新或删除操作。

成员

member 角色可以创建、读取、更新和删除由其所属的范围拥有的资源。

读取器

reader 角色用于只读操作，无论它应用到的范围是什么。此角色可以查看应用资源的范围。

2.1.2. 影响范围

范围是执行操作的上下文。所有 OpenStack 资源都归特定项目所有，因此通过将角色分配给项目范围来授予这些资源的访问权限。

2.1.3. Personas

项目管理员

此用户角色包括跨项目资源创建、读取、更新和删除操作，包括添加和删除用户和其他项目。**授予用户 admin，在任何范围内授予用户对所有可用范围内所有 API 的完整 API 访问权限。**

项目成员

项目成员用户角色适用于被授予权限的用户，使其消耗项目范围内的资源。此用户角色可以在分配到的项目中创建、列出、更新和删除资源。此用户角色表示赋予项目读取器的所有权限。

项目读取器

项目读取器用于授予查看项目中非敏感资源的权限的用户。在项目中，将 reader 角色分配给需要检查或查看资源或审核员的最终用户，他们只需要查看单个项目中的特定于项目的资源，供审计的 Project-reader 用户角色不会解决所有审计用例。

系统管理员

系统管理员通常是可影响部署行为的云管理员。**授予用户 admin，在任何范围内授予用户对所有可用范围内所有 API 的完整 API 访问权限。**

系统成员和系统读取器

系统成员和系统读取器具有相同的授权，可以查看 keystone 中的所有资源。如果审计不需要访问敏感信息，**系统读取** 者对审核员很有用。

注意

基于 **域范围的** 其他用户角色不可用。

2.2. 在 OPENSHIFT 上的 RED HAT OPENSTACK SERVICES 中分配角色

使用 OpenShift 上的 Red Hat OpenStack Services (RHOSO)的 SRBAC，您可以在项目或系统范围内将用户分配给 admin、member 或 reader 的角色。

流程

- 使用类似如下的语法，为用户提供一个具有项目范围的角色：

  示例命令

  ```
  $ openstack role add --user user1 --user-domain Default --project demo --project-domain Default <role>
  ```

- 使用类似如下的语法为用户提供一个系统范围的角色：

  示例命令

  ```
  $ openstack role add --user user1 --user-domain Default --system all  <role>
  ```

  将 <role> 替换为 reader、member 或 admin。

第 3 章 IDENTITY SERVICE (KEYSTONE)简介.

作为云管理员，您可以管理项目、用户和角色。

Red Hat OpenStack Services on OpenShift (RHOSO)数据平面中的项目是组织单元，其中包含一组资源。您可以将 OpenStack 用户分配给这些项目中的角色。角色定义这些用户能够对给定项目中的资源执行的操作。可以在多个项目中为用户分配角色。

每个 RHOSO 数据平面部署必须至少包含一个用户分配给项目中的角色。作为云管理员，用户可以在 OpenStack 中执行以下操作：

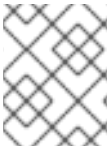
- 添加、更新和删除项目与用户。
- 将用户分配给一个或多个角色，并更改或删除这些分配。
- 管理相互独立的项目和用户。

您还可以使用 Identity 服务(keystone)配置用户身份验证，以控制对服务和端点的访问。

3.1. 使用 CLOUDS.YAML 进行身份验证

先决条件

- 管理员已为您创建一个项目，并为您提供了一个 **clouds.yaml** 文件来访问云。
- 已安装 **python-openstackclient** 软件包。



注意

要在云上执行 **openstack** 客户端命令，您必须指定 **clouds.yaml** 文件中详述的云名称。您可以通过两种方式之一指定云名称：

在每个命令 **中使用--os-cloud** 选项：

```
$ openstack flavor list --os-cloud <cloud_name>
```

如果您访问多个云，则使用此选项。

在 **bashrc** 文件中为云名称创建一个环境变量：

```
`export OS_CLOUD=<cloud_name>`
```

第 4 章 管理用户

作为 OpenStack 管理员，您可以在仪表板中添加、修改和删除用户。用户可以是一个或多个项目的成员。您可以独立于彼此管理项目和用户。

4.1. 使用仪表板创建用户

您可以为用户分配主项目和角色。使用 OpenStack Dashboard (horizon)创建的用户默认为 Identity 服务用户。您可以通过配置 Identity 服务中包含的 LDAP 供应商来集成 Active Directory 用户。

流程

1. 以 admin 用户身份登录控制面板。
2. 选择 **Identity > Users**。
3. 点 **Create User**.
4. 输入用户的用户名、电子邮件和初始密码。
5. 从 **Primary Project** 列表中选择一个项目。
6. 从 **Role** 列表中选择用户的角色。默认角色是 **member**。
7. 点 **Create User**.

4.2. 使用仪表板编辑用户

您可以更新用户详情，包括主项目。

流程

1. 以 admin 用户身份登录控制面板。
2. 选择 **Identity > Users**。
3. 在 **Actions** 列中，单击 **Edit**。
4. 在 **Update User** 窗口中，您可以更新 **User Name**, **Email**, and **Primary Project**。
5. 单击 **Update User**。

4.3. 使用仪表板启用或禁用用户

您可以使用仪表板禁用用户。此操作不可逆，与删除用户不同。

限制：

- 您不能一次禁用或启用多个用户。
- 您不能将用户的主项目设置为 active。

结果是您已禁用的用户无法：

- 登录控制面板。

- 获取 OpenStack 服务的访问权限。
- 在控制面板中执行任何 user-project 操作。

流程

1. 在控制面板中以 admin 用户身份，选择 **Identity > Users**。
2. 在 **Actions** 列中，单击箭头，然后选择 **Enable User** 或 **Disable User**。在 **Enabled** 列中，值然后更新为 **True** 或 **False**。

4.4. 使用仪表板删除用户

您必须是具有管理角色的用户，才能删除其他用户。此操作无法撤销。

流程

1. 在控制面板中以 admin 用户身份，选择 **Identity > Users**。
2. 选择您要删除的用户。
3. 单击 **Delete Users**。此时会显示 **Confirm Delete Users** 窗口。
4. 单击 **Delete Users** 以确认操作。

第 5 章 管理角色

Red Hat OpenStack Services on OpenShift (RHOSO)使用安全基于角色的访问控制(SRBAC)来管理对其资源的访问。角色定义用户可以执行的操作。默认情况下，有三个预定义的角色：

- 查看特定项目的权限的 reader 角色。
- 在指定项目中创建和管理资源的成员角色。
- 一个管理角色，使非管理员用户能够管理环境。

您还可以创建特定于环境的自定义角色。

5.1. 了解 OPENSTACK ADMIN 角色

当您为用户分配 OpenStack 项目中的 **admin** 角色时，此用户具有查看、更改、创建和删除任何项目的任何资源的权限。此用户可以创建可在项目间访问的共享资源，如公开可用的 glance 镜像或提供商网络。此外，具有 **admin** 角色的用户可以创建或删除用户，并管理角色。

为用户授予 **admin** 角色的项目是执行 **openstack** 命令的默认项目。例如，如果一个 **admin** 用户在一个名为 **development** 的项目中运行以下命令，将会在名为 **development** 项目中创建一个名为 **internal-network** 的网络。

```
openstack network create internal-network
```

admin 用户可以通过使用-- **project** 参数在任意项目中创建 **internal-network** ：

```
openstack network create internal-network --project testing
```

5.2. 使用 CLI 查看角色

作为管理员，您可以查看现有角色的详情。

流程

1. 列出可用的预定义角色：

```
$ openstack role list
```

1. 查看指定角色的详情：

```
$ openstack role show admin
```

注意

要获取有关与每个角色关联的权限的详细信息，您必须审核其对每个 API 调用的访问权限。

5.3. 在 OPENSHIFT 上的 RED HAT OPENSTACK SERVICES 中分配角色

使用 OpenShift 上的 Red Hat OpenStack Services (RHOSO)的 SRBAC，您可以在项目或系统范围内将用户分配给 admin、member 或 reader 的角色。

流程

- 使用类似如下的语法，为用户提供一个具有项目范围的角色：

  示例命令

  ```
  $ openstack role add --user user1 --user-domain Default --project demo --project-domain Default <role>
  ```

- 使用类似如下的语法为用户提供一个系统范围的角色：

  示例命令

  ```
  $ openstack role add --user user1 --user-domain Default --system all  <role>
  ```

  将 <role> 替换为 reader、member 或 admin。

第 6 章 管理组

您可以使用 Identity Service (keystone)组为多个用户帐户分配一致的权限。

6.1. 使用仪表板配置组

您可以使用控制面板管理 keystone 组成员资格。但是，您必须使用命令行为组分配角色权限。

6.1.1. 创建组

1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Groups**。
3. 单击 **+Create Group**。
4. 输入组的名称和描述。
5. 单击 **Create Group**。

6.1.2. 管理组成员身份

您可以使用控制面板管理 keystone 组成员资格。

1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Groups**。
3. 单击您要编辑的组的 **管理成员**。
4. 使用 **Add users** 向组中添加用户。如果要删除用户，选中其复选框并点 **Remove users**。

第 7 章 配额管理

作为云管理员，您可以为项目设置和管理配额。每个项目都被分配了资源，项目用户被授予使用这些资源的访问权限。这可让多个项目使用单个云，而不会相互干扰权限和资源。创建新项目时，会预先配置一组资源配额。配额包括可分配给项目的 VCPU 数、实例、RAM 和浮动 IP 数。您可以使用控制面板为新的和现有项目设置或修改计算和块存储配额。如需更多信息，请参阅管理项目。

7.1. 查看项目的计算配额

运行以下命令，以列出当前设置的配额值。

流程

```
$ openstack quota show [PROJECT-ID]
```

Example

```
 openstack quota show f0ba064c24ca4176ac55a45635ca561f
+----------------------+-------+
| Resource             | Limit |
+----------------------+-------+
| cores                |    20 |
| instances            |    10 |
| ram                  | 51200 |
| volumes              |    10 |
| snapshots            |    10 |
| gigabytes            |  1000 |
| backups              |    10 |
| volumes___DEFAULT__  |    -1 |
| gigabytes___DEFAULT__ |    -1 |
| snapshots___DEFAULT__ |    -1 |
| groups               |    10 |
| trunk                |    -1 |
| networks             |   100 |
| ports                |   500 |
| rbac_policies        |    10 |
| routers              |    10 |
| subnets              |   100 |
| subnet_pools         |    -1 |
| fixed-ips            |    -1 |
| injected-file-size   | 10240 |
| injected-path-size   |   255 |
| injected-files       |     5 |
| key-pairs            |   100 |
| properties           |   128 |
| server-groups        |    10 |
| server-group-members |    10 |
| floating-ips         |    50 |
| secgroup-rules       |   100 |
| secgroups            |    10 |
| backup-gigabytes     |  1000 |
| per-volume-gigabytes |    -1 |
+----------------------+-------+
```

第 8 章 管理项目

作为 OpenStack 管理员，您可以创建和管理项目。项目是共享虚拟资源池，您可以为其分配 OpenStack 用户和组。您可以在每个项目中配置共享虚拟资源的配额。您可以创建多个不会干扰彼此的权限和资源的项目。用户可以与多个项目关联。每个用户都必须为为其分配的每个项目分配一个角色。

8.1. 创建一个项目

创建一个项目，将成员添加到项目中，并为项目设置资源限值。

1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Projects**。
3. 点击 **Create Project**。
4. 在 **Project Information** 选项卡中，输入项目的名称和描述。**Enabled** 复选框会被默认选中。
5. 在**项目成员**选项卡上，从 **All Users** 列表向项目添加成员。
6. 在 **Quotas** 选项卡中，指定项目的资源限值。
7. 点击 **Create Project**。

8.2. 编辑项目

您可以编辑项目以更改其名称或描述、启用或禁用它，或更新项目中的成员。

1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Projects**。
3. 在项目 **Actions** 列中，单击箭头，再单击 **Edit Project**。
4. 在 **Edit Project** 窗口中，您可以更新项目以更改其名称或描述，然后启用或禁用项目。
5. 在 **项目成员选项卡** 上，添加成员到项目中，或者根据需要删除它们。
6. 点击 **Save**。

注意

Enabled 复选框会被默认选中。若要临时禁用项目，可清除 **Enabled** 复选框。若要启用禁用的项目，可选中 **Enabled** 复选框。

8.3. 删除项目

1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Projects**。
3. 选择您要删除的项目。
4. 单击 **Delete Projects**。此时会显示 **Confirm Delete Projects** 窗口。

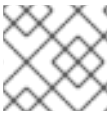
5. 单击 **Delete Projects** 以确认操作。

该项目被删除，任何用户对都解除关联。

8.4. 更新项目配额

配额是您为每个项目设置的操作限制，以优化云资源。您可以设置配额来防止项目资源被耗尽，而无需通知。您可以在项目和 project-user 级别强制实施配额。

1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Projects**。
3. 在项目 **Actions** 列中，单击箭头，再单击 **Modify Quotas**。
4. 在 **Quota** 选项卡中，根据需要修改项目配额。
5. 点击 **Save**。



注意

目前还不支持 *嵌套配额*。因此，您必须单独对项目和子项目管理配额。

8.5. 更改活跃的项目

将项目设置为活动项目，以便您可以使用控制面板与项目中的对象交互。要将项目设置为活跃项目，您必须是项目的成员。用户也需要成为多个项目的成员，才能启用 **Set as Active Project** 选项。您不能将禁用的项目设置为 active，除非它被重新启用。

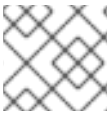
1. 以具有管理特权的用户身份登录控制面板。
2. 选择 **Identity > Projects**。
3. 在项目 **Actions** 列中，单击箭头，再单击 **Set as Active Project**。
4. 或者，以非管理员用户身份，在项目 **Actions** 列中单击 **Set as Active Project**，这将成为 列中的默认操作。

8.6. 项目层次结构

您可以使用 Identity 服务(keystone)中的多租户项目嵌套项目。多租户允许子项目从父项目继承角色分配。

8.6.1. 创建分层项目和子项目

您可以使用 keystone 域和项目实施层次结构多租户(HMT)。首先创建新域，然后在该域内创建项目。然后，您可以将子项目添加到该项目中。您还可以通过将用户添加到该子项目的 **admin** 角色，将用户提升为子项目的管理员。



注意

keystone 使用的 HMT 结构目前没有在仪表板中表示。

流程

1. 创建名为 **corp** 的新 keystone 域：

   示例命令

   ```
   $ openstack domain create corp
   ```

   输出示例

   ```
   +-------------+----------------------------------+
   | Field       | Value                            |
   +-------------+----------------------------------+
   | description |                                  |
   | enabled     | True                             |
   | id          | 6059b0b93b8d4ddb9d31ea47de93f0ab |
   | name        | corp                             |
   | options     | {}                               |
   | tags        | []                               |
   +-------------+----------------------------------+
   ```

2. 在 **corp** 域中创建父项目(**private-cloud**)：

   示例命令

   ```
   $ openstack project create private-cloud --domain corp
   ```

   输出示例

   ```
   +-------------+----------------------------------+
   | Field       | Value                            |
   +-------------+----------------------------------+
   | description |                                  |
   | domain_id   | 6059b0b93b8d4ddb9d31ea47de93f0ab |
   | enabled     | True                             |
   | id          | e86f182e16e24441b71c7296585c2e21 |
   | is_domain   | False                            |
   | name        | private-cloud                    |
   | options     | {}                               |
   | parent_id   | 6059b0b93b8d4ddb9d31ea47de93f0ab |
   | tags        | []                               |
   +-------------+----------------------------------+
   ```

3. 在 **私有云** 父项目中创建子项目(**dev**)，同时指定 **corp** 域：

   示例命令

   ```
   $ openstack project create dev --parent private-cloud --domain corp
   ```

   输出示例

   ```
   +-------------+----------------------------------+
   | Field       | Value                            |
   +-------------+----------------------------------+
   | description |                                  |
   ```

```
| domain_id   | 6059b0b93b8d4ddb9d31ea47de93f0ab |
| enabled     | True                             |
| id          | 71f05cdd5b8a45d4b1a928bfe7c2e20d |
| is_domain   | False                            |
| name        | dev                              |
| options     | {}                               |
| parent_id   | e86f182e16e24441b71c7296585c2e21 |
| tags        | []                               |
+-------------+----------------------------------+
```

4. 创建名为 **qa** 的另一个子项目：

示例命令

```
$ openstack project create qa --parent private-cloud --domain corp
```

输出示例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | 6059b0b93b8d4ddb9d31ea47de93f0ab |
| enabled     | True                             |
| id          | 03801ad929b84c8fb091bf3248e6db68 |
| is_domain   | False                            |
| name        | qa                               |
| options     | {}                               |
| parent_id   | e86f182e16e24441b71c7296585c2e21 |
| tags        | []                               |
+-------------+----------------------------------+
```

注意

您可以使用 Identity API 查看项目层次结构。如需更多信息，请参阅 https://developer.openstack.org/api-ref/identity/v3/index.html?expanded=show-project-details-detail

8.6.2. 配置对分层项目的访问

默认情况下，新创建的项目没有分配的角色。当您为父项目分配角色权限时，您可以包含- **inherited** 标志，以指示子项目从父项目中继承分配的权限。例如，具有对父项目的 **admin** 角色的用户也具有对子项目的 **admin** 访问权限。

授予用户访问权限

1. 查看分配给项目的现有权限：

```
$ openstack role assignment list --project private-cloud
```

2. 查看现有角色：

示例命令

```
$ openstack role list
```

输出示例

```
+----------------------------------+----------------+
| ID                               | Name           |
+----------------------------------+----------------+
| 01d92614cd224a589bdf3b171afc5488 | admin          |
| 034e4620ed3d45969dfe8992af001514 | member         |
| 0aa377a807df4149b0a8c69b9560b106 | ResellerAdmin  |
| cfea5760d9c948e7b362abc1d06e557f | reader         |
| d5cb454559e44b47aaa8821df4e11af1 | swiftoperator  |
| ef3d3f510a474d6c860b4098ad658a29 | service        |
+----------------------------------+----------------+
```

3. 授予用户帐户 **user1** 对 **private-cloud** 项目的访问权限：

```
$ openstack role add --user user1 --user-domain corp --project private-cloud member
```

使用- **inherited** 标志重新运行此命令。因此，**user1** 还有权访问 **private-cloud** 子项目，它们继承了角色分配：

```
$ openstack role add --user user1 --user-domain corp --project private-cloud member --
inherited
```

4. 查看权限更新的结果：

示例命令

```
$ openstack role assignment list --effective --user user1 --user-domain corp
```

输出示例

```
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| Role                             | User                             | Group | Project                          | Domain |
Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| 034e4620ed3d45969dfe8992af001514 | 10b5b34df21d485ca044433818d134be |       |
c50d5cf4fe2e4929b98af5abdec3fd64 |        | False     |
| 034e4620ed3d45969dfe8992af001514 | 10b5b34df21d485ca044433818d134be |       |
11fccd8369824baa9fc87cf01023fd87 |        | True      |
| 034e4620ed3d45969dfe8992af001514 | 10b5b34df21d485ca044433818d134be |       |
b4f1d6f59ddf413fa040f062a0234871 |        | True      |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
```

user1 用户继承了对 **qa** 和 **dev** 项目的访问权限。此外，因为- **inherited** 标志应用于父项目，**user1** 也接收对稍后创建的任何子项目的访问。

删除用户的访问权限

必须单独删除显式和继承的权限。

1. 从显式分配的角色中删除用户：

```
$ openstack role remove --user user1 --project private-cloud member
```

2. 检查更改的结果。请注意，继承的权限仍然存在：

示例命令

```
$ openstack role assignment list --effective --user user1 --user-domain corp
```

输出示例

```
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| Role                             | User                             | Group | Project                          | Domain |
Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| 034e4620ed3d45969dfe8992af001514 | 10b5b34df21d485ca044433818d134be |       |
11fccd8369824baa9fc87cf01023fd87 |        | True      |
| 034e4620ed3d45969dfe8992af001514 | 10b5b34df21d485ca044433818d134be |       |
b4f1d6f59ddf413fa040f062a0234871 |        | True      |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
```

3. 删除继承的权限：

```
$ openstack role remove --user user1 --project private-cloud member --inherited
```

4. 检查更改的结果。继承的权限已被删除，生成的输出现在为空：

```
$ openstack role assignment list --effective --user user1 --user-domain corp
```

8.6.3. 经销商项目概述

使用 *Reseller* 项目，目标是拥有域的层次结构；这些域最终允许您考虑将云的部分重新销售，其子域代表完全启用的云。这个工作分为几个阶段，阶段 1 如下所示：

经销商的第 1 阶段

经销商（阶段 1）是层次结构多租户(HMT)的扩展，如下所述： 创建分层项目和子项目。在以前的版本中，keystone 域最初是存储用户和项目的容器，并在数据库后端中使用自己的表。现在，域不再存储在自己的表中，并已合并到项目表中：

- 域现在是项目的类型，可通过 **is_domain** 标志区分。
- 域表示项目层次结构中的顶级项目： domains 是项目层次结构中的根项
- API 已更新，以使用 **projects** 子路径创建和检索域：
  - 通过创建一个将 **is_domain** 标志设为 true 的项目来创建新域

- 列出属于域的项目：get 项目，包括 **is_domain** 查询参数。

8.7. 项目安全管理

安全组是一组 IP 过滤规则，可分配给项目实例，用于定义对实例的网络访问。安全组是特定于项目的；项目成员可以编辑其安全组的默认规则，并添加新的规则集。

所有项目都有一个默认安全组，它应用到没有其他定义的安全组的任何实例。除非更改默认值，否则此安全组拒绝所有传入的流量，并且只允许来自实例的传出流量。

您可以在实例创建过程中直接将安全组应用到实例，或应用到正在运行的实例上的端口。

注意

您无法在实例创建过程中将基于角色的访问控制 (RBAC) 共享安全组直接应用到实例。要将 RBAC 共享安全组应用到实例，您必须首先创建端口，将共享安全组应用到该端口，然后将该端口分配给实例。请参阅 *创建和管理实例* 中的 向端口添加安全组。

不要在不创建允许所需出口的组的情况下删除默认安全组。例如，如果您的实例使用 DHCP 和元数据，您的实例需要安全组规则来允许到 DHCP 服务器和元数据代理的出口。

8.7.1. 创建安全组

创建安全组，以便您可以配置安全规则。例如，您可以启用 ICMP 流量，或者禁用 HTTP 请求。

流程

1. 在控制面板中，选择 **Project > Compute > Access & Security**。
2. 在 **Security Groups** 选项卡中，单击 **Create Security Group**。
3. 输入组的名称和描述，然后单击 **创建安全组**。

8.7.2. 添加安全组规则

默认情况下，新组的规则仅提供传出访问。您必须添加新规则以提供额外的访问权限。

流程

1. 在控制面板中，选择 **Project > Compute > Access & Security**。
2. 在 **Security Groups** 选项卡中，点您要编辑的安全组的**管理规则**。
3. 单击 **Add Rule** 以添加新规则。
4. 指定规则值，然后单击 **Add**。
   需要以下规则字段：

   规则

   节点类型。如果您指定了规则模板（如 *SSH*），则会自动填写其字段：

   - TCP：通常用于在系统之间交换数据，以及用于最终用户通信。
   - UDP：通常用于在系统间交换数据，特别是在应用程序级别。

- ICMP：网络设备（如路由器）使用来发送错误或监控消息。

方向

Ingress （入站）或 Egress （出站）。

打开端口

对于 TCP 或 UDP 规则，打开 **Port** 或 **Port Range**（单个端口或端口范围）：

- 对于一系列端口，在 **From Port** 和 **To Port** 字段中输入 port 值。
- 对于单个端口，在 **Port** 字段中输入端口值。

类型

ICMP 规则的类型；必须在范围-1: *255 中*。

代码

ICMP 规则的代码；必须在范围-1: *255 中*。

远程

此规则的流量源：

- CIDR (Classless Inter-Domain Routing): IP 地址块，它限制对块中的 IP 的访问。在 Source 字段中输入 CIDR。
- 安全组：允许组中的任何实例访问任何其他组实例的 Source 组。

8.7.3. 删除安全组规则

删除您不再需要的安全组规则。

流程

1. 在控制面板中，选择 **Project > Compute > Access & Security**。
2. 在 **Security Groups** 选项卡中，点安全组的**管理规则**。
3. 选择安全组规则，然后点**删除规则**。
4. 再次单击 **Delete Rule**。

注意

您无法撤销 delete 操作。

8.7.4. 删除安全组

删除您不再需要的安全组。

流程

1. 在控制面板中，选择 **Project > Compute > Access & Security**。
2. 在 **Security Groups** 选项卡中，选择组，再单击 **Delete Security Groups**。

3. 单击 **Delete Security Groups**。

注意

您无法撤销 delete 操作。

第 9 章 管理域

Identity Service (keystone)域是您可以在 keystone 中创建的额外命名空间。

注意

Identity Service 包括一个名为 **Default** 的内置域。建议您只为服务帐户保留这个域，并为用户帐户创建单独的域。

9.1. 查看域列表

您可以使用 **openstack domain list** 命令查看域列表：

示例命令

```
$ openstack domain list
```

输出示例

```
+----------------------------------+-----------------+---------+--------------------+
| ID                               | Name            | Enabled | Description        |
+----------------------------------+-----------------+---------+--------------------+
| 3abefa6f32c14db9a9703bf5ce6863e1 | TestDomain      | True    |                    |
| 69436408fdcb44ab9e111691f8e9216d | corp            | True    |                    |
| a4f61a8feb8d4253b260054c6aa41adb | federated_domain | True    |                    |
| default                          | Default         | True    | The default domain |
+----------------------------------+-----------------+---------+--------------------+
```

9.2. 创建新域

您可以使用 **openstack domain create** 命令创建新域：

示例命令

```
$ openstack domain create TestDomain
```

输出示例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| enabled     | True                             |
| id          | 3abefa6f32c14db9a9703bf5ce6863e1 |
| name        | TestDomain                       |
+-------------+----------------------------------+
```

9.3. 查看域的详情

您可以使用 **openstack domain show** 命令查看域的详情：

示例命令

```
$ openstack domain show TestDomain
```

输出示例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| enabled     | True                             |
| id          | 3abefa6f32c14db9a9703bf5ce6863e1 |
| name        | TestDomain                       |
+-------------+----------------------------------+
```

9.4. 禁用域

您可以根据您的要求禁用并启用域。

流程

1. 使用 the- **disable** 选项**禁用** 域：

   示例命令

   ```
   $ openstack domain set TestDomain --disable
   ```

2. 确认域已被禁用：

   示例命令

   ```
   $ openstack domain show TestDomain
   ```

   输出示例

   ```
   +-------------+----------------------------------+
   | Field       | Value                            |
   +-------------+----------------------------------+
   | description |                                  |
   | enabled     | False                            |
   | id          | 3abefa6f32c14db9a9703bf5ce6863e1 |
   | name        | TestDomain                       |
   +-------------+----------------------------------+
   ```

3. 如果需要，**使用--enable** 选项重新启用域：

   示例命令

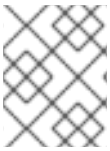
   ```
   $ openstack domain set TestDomain --enable
   ```

第 10 章 应用程序凭证

使用 *应用凭据* 来避免在配置文件中嵌入用户帐户凭据的做法。相反，用户会创建一个 Application Credential，它接收对单个项目的委派访问权限，并具有自己的不同 secret。用户也可以将委派的特权限制为该项目中的单个角色。这可让您采用最小特权的原则，其中经过身份验证的用户只能获得一个项目的访问权限，以及它正常工作的角色，而不是所有项目和角色。

使用应用凭据，您可以在不显示用户凭据的情况下使用 API，并且应用可以在不需要嵌入式用户凭据的情况下向 Keystone 进行身份验证。

您可以使用应用程序凭证生成令牌并为应用程序配置 **keystone_authtoken** 设置。在以下部分中描述了这些用例。



注意

Application Credential 依赖于创建它的用户帐户，因此当该帐户被删除或无法访问相关角色时，它将终止。

10.1. 使用应用凭证生成令牌

在仪表板中，应用凭据作为自助服务功能供用户使用。本例演示了用户如何创建应用凭据，然后使用它来生成令牌。

1. 创建测试项目并测试用户帐户：

   a. 创建名为 **AppCreds** 的项目：

   ```
   $ openstack project create AppCreds
   ```

   b. 创建名为 **AppCredsUser** 的用户：

   ```
   $ openstack user create --project AppCreds --password-prompt AppCredsUser
   ```

   c. 为 **AppCreds** 项目授予 **member** 角色的 **AppCredsUser** 访问权限：

   ```
   $ openstack role add --user AppCredsUser --project AppCreds member
   ```

2. 以 **AppCredsUser** 用户身份登录到仪表板并创建应用程序凭证：
   Overview → Identity → Application Credentials → +Create Application Credential。

   注意

   确保您下载 **clouds.yaml** 文件内容，因为在关闭了 **Your Application Credential** 的弹出窗口后，您无法再次访问它。

3. 使用 CLI 创建名为 **/home/stack/.config/openstack/clouds.yaml** 的文件，并粘贴 **clouds.yaml** 文件的内容。

   ```
   # This is a clouds.yaml file, which can be used by OpenStack tools as a source
   # of configuration on how to connect to a cloud. If this is your only cloud,
   # just put this file in ~/.config/openstack/clouds.yaml and tools like
   # python-openstackclient will just work with no further config. (You will need
   # to add your password to the auth section)
   ```

```
# If you have more than one cloud account, add the cloud entry to the clouds
# section of your existing file and you can refer to them by name with
# OS_CLOUD=openstack or --os-cloud=openstack
clouds:
  openstack:
    auth:
      auth_url: http://10.0.0.10:5000/v3
      application_credential_id: "6d141f23732b498e99db8186136c611b"
      application_credential_secret: "<example secret value>"
    region_name: "regionOne"
    interface: "public"
    identity_api_version: 3
    auth_type: "v3applicationcredential"
```

注意

这些值将为您的部署有所不同。

4. 使用 Application Credential 生成令牌。使用以下命令时，不得作为任何特定用户提供，且必须与 **clouds.yaml** 文件位于同一个目录中。

```
$ openstack --os-cloud=openstack token issue
+------------+-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------+
| Field      | Value
|
+------------+-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------+
| expires    | 2018-08-29T05:37:29+0000
|
| id         | gAAAAABbhiMJ4TxxFlTMdsYJpfStsGotPrns0lnpvJq9ILtdi-
NKqisWBeNiJlUXwmnoGQDh2CMyK9OeTsuEXnJNmFfKjxiHWmcQVYzAhMKo6_QMUtu_Qm
6mtpzYYHBrUGboa_Ay0LBuFDtsjtgtvJ-r8G3TsJMowbKF-yo--
O_XLhERU_QQVl3hl8zmMRdmLh_P9Cbhuolt |
| project_id | 1a74eabbf05c41baadd716179bb9e1da
|
| user_id    | ef679eeddfd14f8b86becfd7e1dc84f2
|
+------------+-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------+
```

注意

如果您收到与 **__init__（）类似的错误，则有一个意外的关键字参数 'application_credential_secret'**，则您可能仍然 source 到之前的凭证。对于新环境，请运行 **sudo su - stack**。

10.2. 将应用凭证与应用程序集成

应用凭据可用于对应用程序进行 keystone 身份验证。使用 Application Credentials 时，**keystone_authtoken** 设置使用 **v3applicationcredential** 作为身份验证类型，并包含您在凭证创建过程中接收的凭证。输入以下值：

- **application_credential_secret** ：应用程序凭证 secret。

- **application_credential_id** ：应用程序凭证 ID。
- （可选） **application_credential_name** ：如果使用命名的应用程序凭证而不是 ID，您可以使用此参数。

例如：

```
[keystone_authtoken]
auth_url = http://10.0.0.10:5000/v3
auth_type = v3applicationcredential
application_credential_id = "6cb5fa6a13184e6fab65ba2108adf50c"
application_credential_secret = "<example password>"
```

10.3. 管理应用程序凭证

您可以使用命令行来创建和删除应用凭证。

create 子命令基于当前源的帐户创建应用凭据。例如，当以 **admin** 用户身份提供时创建凭证会将相同的角色授予应用程序凭证：

```
$ openstack application credential create --description "App Creds - All roles" AppCredsUser
+--------------+-----------------------------------------------------------------------------------------+
| Field        | Value                                                                                   |
+--------------+-----------------------------------------------------------------------------------------+
| description  | App Creds - All roles                                                                   |
| expires_at   | None                                                                                    |
| id           | fc17651c2c114fd6813f86fdbb430053                                                        |
| name         | AppCredsUser                                                                            |
| project_id   | 507663d0cfe244f8bc0694e6ed54d886                                                        |
| roles        | member reader admin                                                                     |
| secret       | fVnqa6I_XeRDDkmQnB5lx361W1jHtOtw3ci_mf_tOID-09MrPAzkU7mv-
by8ykEhEa1QLPFJLNV4cS2Roo9lOg |
| unrestricted | False                                                                                   |
+--------------+-----------------------------------------------------------------------------------------+
```

警告

使用- **unrestricted** 参数可让应用程序凭证创建和删除其他应用程序凭证和信任。这是潜在的危险行为，默认是禁用的。您不能将- **-unrestricted** 参数与其他访问规则结合使用。

默认情况下，生成的角色成员资格包含分配给创建凭据的帐户的所有角色。您可以通过只委派访问特定角色来限制角色成员资格：

```
$ openstack application credential create --description "App Creds - Member" --role member
AppCredsUser
+--------------+-----------------------------------------------------------------------------------------+
| Field        | Value                                                                                   |
+--------------+-----------------------------------------------------------------------------------------+
```

```
| description  | App Creds - Member                                   |
| expires_at   | None                                   |
| id           | e21e7f4b578240f79814085a169c9a44                                   |
| name         | AppCredsUser                                   |
| project_id   | 507663d0cfe244f8bc0694e6ed54d886                                   |
| roles        | member                                   |
| secret       |
XCLVUTYIreFhpMqLVB5XXovs_z9JdoZWpdwrkaG1qi5GQcmBMUFG7cN2htzMlFe5T5mdPsnf5JMNb
u0lh-4aCg |
| unrestricted | False                                   |
+--------------+----------------------------------------------------------------------------------------+
```

删除应用程序凭证：

```
$ openstack application credential delete AppCredsUser
```

10.4. 替换应用程序凭证

应用凭据绑定到创建它们的用户帐户，如果用户帐户被删除，或者用户无法访问委派的角色，就会变得无效。因此，您应该准备生成新应用凭证。

为配置文件替换现有应用程序凭证

更新分配给应用程序的应用程序凭证（使用配置文件）：

1. 创建一组新的应用凭据。
2. 将新凭据添加到应用配置文件中，替换现有的凭据。如需更多信息，请参阅将应用程序凭证与应用程序集成。
3. 重启应用程序服务以应用更改。
4. 删除旧的应用凭据（如果适用）。有关命令行选项的更多信息，请参阅管理应用程序凭证。

替换 clouds.yaml 中的现有应用程序凭证

当替换 **clouds.yaml** 使用的应用程序凭证时，您必须使用 OpenStack 用户凭证创建替换凭证。默认情况下，您无法使用应用程序凭据来创建另一组应用凭据。**openstack application credential create** 命令根据当前源的帐户创建应用程序凭证。

1. 以最初创建即将到期的身份验证凭据的 OpenStack 用户进行身份验证。例如，如果您使用 使用应用凭证生成令牌 的流程，则必须作为 **AppCredsUser** 重新登录。
2. 创建名为 **AppCred2** 的应用凭据。这可以通过 OpenStack Dashboard 或 **openstack** CLI 界面完成：

   ```
   openstack application credential create --description "App Creds 2 - Member" --role member
   AppCred2
   ```

3. 复制上一命令输出中的 **id** 和 **secret** 参数。无法再次访问 **secret** 参数值。
4. 将 **${HOME}/.config/openstack/clouds.yaml** 文件中的 **application_credential_id** 和 **application_credential_secret** 参数值替换为您复制的 **secret** 和 **id** 值。

验证

1. 使用 clouds.yaml 生成令牌，以确认凭证按预期工作。使用以下命令时，不得作为任何特定用户提供，且必须与 **clouds.yaml** 文件位于同一个目录中：

```
$ openstack --os-cloud=openstack token issue
```

输出示例：

```
+------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Field      | Value                                                                                                                                                                 |
+------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| expires    | 2018-08-29T05:37:29+0000                                                                                                                                              |
| id         | gAAAAABbhiMJ4TxxFlTMdsYJpfStsGotPrns0lnpvJq9ILtdi-NKqisWBeNiJlUXwmnoGQDh2CMyK9OeTsuEXnJNmFfKjxiHWmcQVYzAhMKo6_QMUtu_Qm6mtpzYYHBrUGboa_Ay0LBuFDtsjtgtvJ-r8G3TsJMowbKF-yo--O_XLhERU_QQVl3hl8zmMRdmLh_P9Cbhuolt |
| project_id | 1a74eabbf05c41baadd716179bb9e1da                                                                                                                                      |
| user_id    | ef679eeddfd14f8b86becfd7e1dc84f2                                                                                                                                      |
+------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

第 11 章 使用 OPENSTACK KEY MANAGER (BARBICAN)管理 SECRET 和密钥

Red Hat OpenStack Services on OpenShift (RHOSO)的 OpenStack Key Manager (barbican)服务安全存储、置备和管理您的 secret 数据。您可以使用 OpenStack Key Manager 创建、更新和删除 secret 和加密密钥。

11.1. 关于 OPENSHIFT KEY MANAGER OPERATOR (BARBICAN)上的 RED HAT OPENSTACK SERVICES

OpenStack Key Manager (barbican)服务是 OpenShift (RHOSO)上 Red Hat OpenStack Services 的 secret Manager。这个 Operator 被默认安装并激活。OpenStack 管理员可以使用 Key Manager API 和命令行集中管理 OpenStack 使用的证书、密钥和密码。RHOSO Key Manager 服务与 Block Storage (cinder)、Networking (neutron)和 Compute (nova)服务集成。

证书、API 密钥和密码等机密存储在 barbican 数据库中，或者直接存储在安全存储系统中。在 RHOSO 18.0 中，简单的加密插件可用于加密 secret。

OpenStack Key Manager 支持以下用例：

- **对称加密密钥**
  - Block Storage (cinder)卷加密
  - Object Storage (swift)加密
  - 临时磁盘加密
- **非对称密钥和证书**
  - 镜像服务(glance)镜像签名
  - 镜像服务(glance)镜像验证

11.2. 查看 SECRET

要查看 secret 列表，请运行 **openstack secret list** 命令。列表中包含有关 secret 的 URI、名称、类型和其他信息。

流程

- 查看 secret 列表：

  示例命令

  ```
  $ openstack secret list
  ```

  输出示例

  ```
  +------------------------------------------------------------------------------+------+------------------
  ---------+--------+--------------------------------------+----------+------------+-------------+------
  +------------+
  | Secret href                                                    | Name | Created                | Status
  ```

```
| Content types                          | Algorithm | Bit length | Secret type | Mode | Expiration |
+----------------------------------------------------------------------------------+------+----------
---------+--------+-----------------------------------------+----------+-----------+------------+------
+------------+
| https://192.168.123.169:9311/v1/secrets/24845e6d-64a5-4071-ba99-0fdd1046172e       |
None | 2018-01-22T02:23:15+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes
|        256 | symmetric   | None | None       |
+----------------------------------------------------------------------------------+------+----------
---------+--------+-----------------------------------------+----------+-----------+------------+------
+------------+
```

11.3. 创建 SECRET

要创建 secret，请运行 **openstack secret store** 命令，并指定 secret 的名称以及可选的 secret 有效负载。

流程

- 创建 secret：

  示例命令

  ```
  $ openstack secret store --name testSecret --payload 'TestPayload'
  ```

  输出示例

  ```
  +---------------+-----------------------------------------------------------------------------------+
  | Field         | Value                                                                             |
  +---------------+-----------------------------------------------------------------------------------+
  | Secret href   | https://192.168.123.163:9311/v1/secrets/ecc7b2a4-f0b0-47ba-b451-
  0f7d42bc1746       |
  | Name          | testSecret                                                                          |
  | Created       | None                                                                              |
  | Status        | None                                                                              |
  | Content types | None                                                                                |
  | Algorithm     | aes                                                                               |
  | Bit length    | 256                                                                               |
  | Secret type   | opaque                                                                              |
  | Mode          | cbc                                                                               |
  | Expiration    | None                                                                              |
  +---------------+-----------------------------------------------------------------------------------+
  ```

11.4. 在 SECRET 中添加有效负载

您无法更改 secret 的有效负载，您只能删除它。如果您在没有指定有效负载的情况下创建 secret，则稍后您可以使用 **openstack secret update** 命令向其添加一个有效负载。

流程

- 在 secret 中添加有效负载：

  示例命令

```
$ openstack secret update https://192.168.123.163:9311/v1/secrets/ca34a264-fd09-44a1-8856-c6e7116c3b16 'TestPayload-updated'
```

11.5. 删除 SECRET

要删除 secret，请运行 **openstack secret delete** 命令并指定 secret URI。

流程

- 使用指定的 URI 删除 secret：

  示例命令

```
$ openstack secret delete https://192.168.123.163:9311/v1/secrets/ecc7b2a4-f0b0-47ba-b451-0f7d42bc1746
```

11.6. 生成对称密钥

您可以为特定的任务生成对称密钥，如计算服务磁盘加密和对象存储服务加密。

流程

1. 使用 **顺序** 创建并存储在 barbican 中生成新的 256 位密钥。例如：

   示例命令

```
$ openstack secret order create --name swift_key --algorithm aes --mode ctr --bit-length 256 --payload-content-type=application/octet-stream key
```

   输出示例

```
+----------------+-------------------------------------------------------------------------------------+
| Field          | Value                                                                               |
+----------------+-------------------------------------------------------------------------------------+
| Order href     | https://192.168.123.173:9311/v1/orders/043383fe-d504-42cf-a9b1-bc328d0b4832       |
| Type           | Key                                                                                 |
| Container href | N/A                                                                                 |
| Secret href    | None                                                                                |
| Created        | None                                                                                |
| Status         | None                                                                                |
| Error code     | None                                                                                |
| Error message  | None                                                                                |
+----------------+-------------------------------------------------------------------------------------+
```

   您还可以使用- **mode** 选项**将** 生成的密钥配置为使用特定模式，如 **ctr** 或 **cbc**。如需更多信息，请参阅 *NIST SP 800-38A*。

2. 查看识别生成的密钥位置的顺序详情，这里显示为 **Secret href** 值：

   示例命令

```
$ openstack secret order get https://192.168.123.173:9311/v1/orders/043383fe-d504-42cf-a9b1-bc328d0b4832
```

输出示例

```
+---------------+-------------------------------------------------------------------------------+
| Field         | Value                                                                         |
+---------------+-------------------------------------------------------------------------------+
| Order href    | https://192.168.123.173:9311/v1/orders/043383fe-d504-42cf-a9b1-bc328d0b4832   |
| Type          | Key                                                                           |
| Container href | N/A                                                                          |
| Secret href   | https://192.168.123.173:9311/v1/secrets/efcfec49-b9a3-4425-a9b6-5ba69cb18719  |
| Created       | 2018-01-24T04:24:33+00:00                                                     |
| Status        | ACTIVE                                                                        |
| Error code    | None                                                                          |
| Error message | None                                                                          |
+---------------+-------------------------------------------------------------------------------+
```

3. 检索 secret 的详细信息：

示例命令

```
$ openstack secret get https://192.168.123.173:9311/v1/secrets/efcfec49-b9a3-4425-a9b6-5ba69cb18719
```

输出示例

```
+--------------+-------------------------------------------------------------------------------+
| Field        | Value                                                                         |
+--------------+-------------------------------------------------------------------------------+
| Secret href  | https://192.168.123.173:9311/v1/secrets/efcfec49-b9a3-4425-a9b6-5ba69cb18719  |
| Name         | swift_key                                                                     |
| Created      | 2018-01-24T04:24:33+00:00                                                     |
| Status       | ACTIVE                                                                        |
| Content types | {u'default': u'application/octet-stream'}                                     |
| Algorithm    | aes                                                                           |
| Bit length   | 256                                                                           |
| Secret type  | symmetric                                                                     |
| Mode         | ctr                                                                           |
| Expiration   | None                                                                          |
+--------------+-------------------------------------------------------------------------------+
```

11.7. 加密 BLOCK STORAGE (CINDER)卷

您可以使用 barbican 管理块存储服务(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。密钥管理对用户是透明的；当您使用 **luks** 作为加密类型创建新卷时，cinder 为卷生成对称密钥 secret，并将其存储在 barbican 中。引导实例或附加加密卷时，计算服务从密钥管理器服务检索密钥，并将 secret 存储在本地 Compute 节点上的 libvirt secret。

流程

1. 创建使用加密的卷模板。当您创建新卷时，可以根据您在此处定义的设置进行建模：

示例命令

```
$ openstack volume type create --encryption-provider
nova.volume.encryptors.luks.LuksEncryptor --encryption-cipher aes-xts-plain64 --encryption-
key-size 256 --encryption-control-location front-end LuksEncryptor-Template-256
```

输出示例

```
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| Field       | Value
|
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| description | None
|
| encryption  | cipher='aes-xts-plain64', control_location='front-end', encryption_id='9df604d0-
8584-4ce8-b450-e13e6316c4d3', key_size='256',
provider='nova.volume.encryptors.luks.LuksEncryptor' |
| id          | 78898a82-8f4c-44b2-a460-40a5da9e4d59
|
| is_public   | True
|
| name        | LuksEncryptor-Template-256
|
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
```

2. 创建一个新卷，并指定它使用 **LuksEncryptor-Template-256** 设置：

示例命令

```
$ openstack volume create --size 1 --type LuksEncryptor-Template-256 'Encrypted-Test-
Volume'
```

输出示例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2018-01-22T00:19:06.000000           |
| description         | None                                 |
| encrypted           | True                                 |
| id                  | a361fd0b-882a-46cc-a669-c633630b5c93 |
| migration_status    | None                                 |
| multiattach         | False                                |
| name                | Encrypted-Test-Volume                |
| properties          |                                      |
```

```
| replication_status  | None                                 |
| size                | 1                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | LuksEncryptor-Template-256           |
| updated_at          | None                                 |
| user_id             | 0e73cb3111614365a144e7f8f1a972af     |
+---------------------+--------------------------------------+
```

生成的 secret 会自动上传到 barbican 后端。

3. 确保您可以看到 barbican secret UUID。这个值显示在 **encryption_key_id** 字段中。

示例命令

```
$ cinder --os-volume-api-version 3.64 volume show Encrypted-Test-Volume
```

输出示例

```
+------------------------------+--------------------------------------+
|Property                      |Value                                 |
+------------------------------+--------------------------------------+
|attached_servers              |[]                                    |
|attachment_ids                |[]                                    |
|availability_zone             |nova                                  |
|bootable                      |false                                 |
|cluster_name                  |None                                  |
|consistencygroup_id           |None                                  |
|created_at                    |2022-07-28T17:35:26.000000            |
|description                   |None                                  |
|encrypted                     |True                                  |
|encryption_key_id             |0944b8a8-de09-4413-b2ed-38f6c4591dd4 |
|group_id                      |None                                  |
|id                            |a0b51b97-0392-460a-abfa-093022a120f3 |
|metadata                      |                                      |
|migration_status              |None                                  |
|multiattach                   |False                                 |
|name                          |vol                                   |
|os-vol-host-attr:host         |hostgroup@tripleo_iscsi#tripleo_iscsi|
|os-vol-mig-status-attr:migstat|None                                  |
|os-vol-mig-status-attr:name_id|None                                  |
|os-vol-tenant-attr:tenant_id  |a2071ece39b3440aa82395ff7707996f     |
|provider_id                   |None                                  |
|replication_status            |None                                  |
|service_uuid                  |471f0805-072e-4256-b447-c7dd10ceb807 |
|shared_targets                |False                                 |
|size                          |1                                     |
|snapshot_id                   |None                                  |
|source_volid                  |None                                  |
|status                        |available                             |
|updated_at                    |2022-07-28T17:35:26.000000            |
|user_id                       |ba311b5c2b8e438c951d1137333669d4      |
```

```
|volume_type                |LUKS                                  |
|volume_type_id             |cc188ace-f73d-4af5-bf5a-d70ccc5a401c |
+---------------------------+--------------------------------------+
```

注意

您必须使用带有 Cinder CLI 的 **--os-volume-api-version 3.64** 参数来显示 **encryption_key_id** 值。没有等同的 OpenStack CLI 命令。

4. 使用 barbican 确认存在磁盘加密。在本例中，时间戳与 LUKS 卷创建时间匹配：

示例命令

```
$ openstack secret list
```

输出示例

```
+------------------------------------------------------------------------------+------+------------------
---------+--------+-----------------------------------------+----------+------------+-------------+------
+------------+
| Secret href                                                                  | Name | Created                  | Status
| Content types                        | Algorithm | Bit length | Secret type | Mode | Expiration |
+------------------------------------------------------------------------------+------+------------------
---------+--------+-----------------------------------------+----------+------------+-------------+------
+------------+
| https://192.168.123.169:9311/v1/secrets/0944b8a8-de09-4413-b2ed-38f6c4591dd4 | None |
2018-01-22T02:23:15+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes       |
256 | symmetric   | None | None       |
+------------------------------------------------------------------------------+------+------------------
---------+--------+-----------------------------------------+----------+------------+-------------+------
+------------+
```

5. 将新卷附加到现有实例：

```
$ openstack server add volume testInstance Encrypted-Test-Volume
```

该卷现在可用于挂载到客户机文件系统。

11.8. 签名镜像服务(GLANCE)镜像

在配置 Image Service (glance)以验证镜像时，您必须对这些镜像进行签名，然后才能上传这些镜像。使用 **openssl** 命令，为镜像签名存储在 Key Manager 服务(barbican)中的密钥，然后使用附带的镜像属性将镜像上传到镜像服务。在每次使用前验证镜像的签名，如果签名不匹配，实例构建过程会失败。

限制：

- 签名验证基于 **openSSL _rsa_sig_verify （）**，如果密钥长度不是 1024，它目前会在镜像服务中引发 **InvalidSignature** 异常。出现这个问题的原因是 OpenSSL 3.0.7 中的回归。签名验证可以正常工作，密钥长度为 2048 和 OpenSSL 3.0.2。
- OpenSSL 不支持 SHA256 算法。您必须使用 SHA512 或更高版本来创建签名的镜像。

流程

1. 使用 **openssl** 命令创建密钥和证书：

   - 创建私钥：

     ```
     $ openssl genrsa -out private_key.pem 1024
     ```

   - 创建公钥：

     ```
     $ openssl rsa -pubout -in private_key.pem -out public_key.pem
     ```

   - 创建证书：

     ```
     $ openssl req -new -key private_key.pem -out cert_request.csr
     ```

   - 提示时，请输入要包含在您的证书请求中的详情：

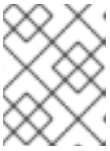
     ```
     $ openssl x509 -req -days 14 -in cert_request.csr -signkey private_key.pem -out
     new_cert.crt
     ```

2. 将证书上传到 Key Manager 服务，该服务存储在 **secret 存储中** ：

   ```
   $ openstack secret store --name test --algorithm RSA --secret-type certificate \
     --payload-content-type "application/octet-stream" \
     --payload-content-encoding base64 \
     --payload "$(base64 new_cert.crt)"
   ```

   输出示例：

   ```
   +---------------+------------------------------------------------------------------------+
   | Field         | Value                                                                  |
   +---------------+------------------------------------------------------------------------+
   | Secret href   | http://127.0.0.1:9311/v1/secrets/cd7cc675-e573-419c-8fff-33a72734a243 |
   +---------------+------------------------------------------------------------------------+
   ```

   注意

   记录证书 UUID，以便在稍后的步骤中使用。在本例中，UUID 是 **cd7cc675-e573-419c-8fff-33a72734a243**。

3. 检索镜像并创建签名：

   - 检索镜像：

     ```
     $ echo <This is my image> > <myimage>
     ```

   - 使用 **private_key.pem** 为镜像签名并生成 **.signature** 文件。例如：

     ```
     $ openssl dgst -sha512 -sign private_key.pem -sigopt rsa_padding_mode:pss -out
     myimage.signature myimage
     ```

- 将 **.signature** 文件转换为 Base64 格式：

```
$ base64 -w 0 myimage.signature > myimage.signature.b64
```

- 将 Base64 值加载到上传镜像时使用的变量中：

```
$ image_signature=$(cat myimage.signature.b64)
```

4. 将具有有效签名属性的镜像上传到镜像服务：

```
$ openstack image-create \
  --name <my_signed_image> \
  --container-format bare \
  --disk-format qcow2 \
  --property img_signature="$image_signature" \
  --property img_signature_certificate_uuid="$cert_uuid" \
  --property img_signature_hash_method='SHA-512' \
  --property img_signature_key_type='RSA-PSS' < myimage
```

11.9. 验证快照

快照保存为 Image 服务(glance)镜像。如果您将 Compute 服务(nova)配置为检查已签名的镜像，则必须由签名快照，即使它们是从带有签名镜像的实例创建的。

流程

1. 从镜像服务下载快照：

```
$ openstack image save --file <local_file_name> <snapshot_image_name>
```

- 将 **<local_file_name>** 替换为下载的镜像的文件名。
- 将 **<snapshot_image_name>** 替换为快照的名称。

2. 生成签名以验证快照。使用您用来生成签名来验证镜像的同一进程。如需更多信息，请参阅签名镜像 服务(glance)镜像。

3. 更新快照镜像属性：

```
$ openstack image set \
  --property img_signature="$image_signature" \
  --property img_signature_certificate_uuid="<cd7cc675-e573-419c-8fff-33a72734a243>" \
  --property img_signature_hash_method="SHA-512" \
  --property img_signature_key_type="RSA-PSS" \
  <snapshot_image_id>
```

- 将 **<cd7cc675-e573-419c-8fff-33a72734a243>** 替换为镜像的证书 UUID。
- 将 **<snapshot_image_id>** 替换为您的快照 ID。

4. 可选：从文件系统中删除下载的镜像服务镜像：

```
$ rm <local_file_name>
```

11.10. 备份简单的加密密钥

要备份 **简单的** 加密密钥，您必须备份密钥管理器(barbican)数据库和 **BarbicanSimpleCryptoKek** 密钥，用于加密和从数据库解密值。将密钥和密钥管理器数据库存储在安全强化的位置。

先决条件

- 您有一个安全强化的位置，用于 **BarbicanSimpleCryptoKEK** 密钥和密钥管理器数据库。

流程

1. 从 **osp-secret** 检索 base64 解码的 **BarbicanSimpleCryptoKEK** 值，并将它保存到名为 kek.txt 的文件中：

```
oc get secret osp-secret -o yaml | grep BarbicanSimpleCryptoKEK | awk '{print $2}' | base64 -d > kek.txt
```

2. 检索 Key Manager 数据库 root 密码：

```
DB_PASS=$(oc get secret osp-secret -o yaml | grep DbRootPassword | awk '{print $2}' | base64 -d)
```

3. 使用 Key Manager 数据库 root 密码备份 *barbican* 数据库：

```
$ oc exec -it openstack-galera-0 -- mysqldump -u root -p"${DB_PASS}" barbican > barbican_db_backup.sql
```

4. 检查数据库备份是否已存储在当前工作目录中：

```
$ ll
total 36
-rw-rw-r--. 1 tripleo-admin tripleo-admin 36715 Jun 19 18:31 barbican_db_backup.sql
```

5. 将 **barbican_db_backup.sql** 和 **kek.txt** 文件备份到安全强化的位置。

11.11. 恢复简单的加密加密密钥

从备份中恢复 barbican 数据库和 **BarbicanSimpleCryptoKek** 密码，以恢复简单的加密加密密钥。

流程

1. 使用 base64 对 **BarbicanSimpleCryptoKek** 键进行编码，并将结果值写入 osp-secret ：

```
$ echo <barbican_key> | base64
YmFyYmljYW5fc2ltcGxlX2N5cHRvX2tlawo=
```

将 <barbican_key> 替换为您要恢复的 **BarbicanSimpleCryptoKek** 键

2. 打开 osp-secret secret 对象以进行编辑：

```
$ oc edit secret osp-secret
```

3. 添加第 1 步中的值。编写并保存您的更改：

```
....
  BarbicanDatabasePassword: cGFzc3dvcmQK
  BarbicanPassword: cGFzc3dvcmQK
  BarbicanSimpleCryptoKEK: YmFyYmljYW5fc2ltcGxlX2N5cHRvX2tleawo=
  CeilometerPassword: cGFzc3dvcmQK
  CinderDatabasePassword: cGFzc3dvcmQK
  CinderPassword: cGFzc3dvcmQK
  DatabasePassword: cGFzc3dvcmQK
....
```

4. 检索 barbican 数据库 root 密码：

```
DB_PASS=$(oc get secret osp-secret -o yaml | grep DbRootPassword | awk '{print $2}' |
base64 -d)
```

5. 将备份文件恢复到 **barbican** 数据库：

```
$ oc exec openstack-galera-0 --  mysql -u root -p"${DB_PASS}" barbican < <sql_backup>
```

- 将 <sql_backup> 替换为 mysql 数据库备份文件的名称。

  输出示例

  ```
  Defaulted container "galera" out of: galera, mysql-bootstrap (init)
  ```

验证

- 您可以使用 openstackclient pod 验证测试 secret 是否已成功恢复：

```
$ oc exec -n openstack -t openstackclient -- openstack secret list
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+----------------------------------------+----------+-----------+------------+------+------
------+
| Secret href                                                            | Name    | Created               | Status |
Content types                        | Algorithm | Bit length | Secret type | Mode | Expiration |
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+----------------------------------------+----------+-----------+------------+------+------
------+
| http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a | testSecret |
2018-06-19T18:25:25+00:00 | ACTIVE | {u'default': u'text/plain'}            | aes    |     256 |
opaque   | cbc  | None    |
| http://10.0.0.104:9311/v1/secrets/f664b5cf-5221-47e5-9887-608972a5fefb | swift_key  |
2018-06-19T18:24:40+00:00 | ACTIVE | {u'default': u'application/octet-stream'} | aes    |
256 | symmetric   | ctr  | None    |
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+----------------------------------------+----------+-----------+------------+------+------
------+
```