Red Hat OpenStack Platform 13

使用 OpenStack Key Manager 管理 Secret

如何将 OpenStack Key Manager (Barbican)与 OpenStack 部署集成。

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 使用 OpenStack Key Manager 管理 Secret

如何将 OpenStack Key Manager (Barbican)与 OpenStack 部署集成。

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

如何将 OpenStack Key Manager (Barbican)与 OpenStack 部署集成。

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。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 概述

OpenStack Key Manager (barbican)是 Red Hat OpenStack Platform 的 secret Manager。您可以使用 barbican API 和命令行集中管理 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 Block Storage (cinder)卷加密、临时磁盘加密和 Object Storage (swift)加密等等。
- **非对称密钥和证书** - 用于 glance 镜像签名和验证。

在本发行版本中，barbican 提供与 Block Storage (cinder)和计算(nova)组件的集成。

第 2 章 选择后端

机密（如证书、API 密钥和密码）可以作为加密 Blob 存储在 barbican 数据库中，或者直接存储在安全存储系统中。

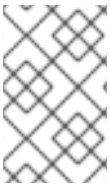
要将 secret 作为加密的 blob 存储在 barbican 数据库中，可以使用以下选项：

- **简单加密插件** - 默认启用简单的加密插件，并使用单个对称密钥加密 secret 的 Blob。这个密钥以纯文本形式存储在 **barbican.conf** 文件中。

注意

目前，仅支持简单的加密插件。

- **PKCS#11 crypto 插件** - PKCS#11 crypto 插件使用特定于项目的密钥加密密钥(KEK)加密 secret，这些密钥保存在 barbican 数据库中。这些项目特定的 KEK 由主 KEK 进行加密，存储在硬件安全模块(HSM)中。所有加密和解密操作都放置在 HSM 中，而不是进程内存中。PKCS#11 插件通过 PKCS#11 协议与 HSM 通信。由于加密是在安全硬件上完成的，并且每个项目使用不同的 KEK，所以此选项比简单的加密插件更安全。



注意

关于高可用性(HA)选项：barbican 服务在 Apache 中运行，由 director 配置，以使用 HAProxy 进行高可用性。后端层的 HA 选项取决于所使用的后端。例如，对于简单加密，所有 barbican 实例都在配置文件中具有相同的加密密钥，从而形成简单的 HA 配置。

2.1. 在后端间迁移

Barbican 允许您为项目定义不同的后端。如果没有项目映射，则 secret 会存储在全局默认后端中。这意味着可以配置多个后端，但必须定义至少一个全局后端。为不同后端提供的 heat 模板包含将每个后端设置为默认值的参数。

如果在特定后端中存储 secret，然后决定迁移到新的后端，您可以保留旧的后端，同时启用新的后端作为全局默认值（或作为项目特定的后端）。因此，旧的 secret 仍可通过旧后端使用。

第 3 章 安装 BARBICAN

Red Hat OpenStack Platform 中不默认启用 Barbican。此流程描述了如何在现有 OpenStack 部署中部署 barbican。Barbican 作为容器化服务运行，因此此步骤还描述了如何准备和上传新的容器镜像：

注意

此流程将 barbican 配置为使用 **simple_crypto** 后端。其他后端可用，如 **PKCS11** 和 DogTag，但本发行版本中不支持它们。

1. 在 undercloud 节点上，为 barbican 创建环境文件。这将指示 director 安装 barbican （在 *openstack overcloud deploy [...]* 中包含的时）

```
$ cat /home/stack/configure-barbican.yaml
 parameter_defaults:
   BarbicanSimpleCryptoGlobalDefault: true
```

   - **BarbicanSimpleCryptoGlobalDefault** - 将这个插件设置为全局默认插件。
   - 更多选项也可以配置：
     - **BarbicanPassword** - 为 barbican 服务帐户设置密码。
     - **BarbicanWorkers** - 设置 **barbican::wsgi::apache** 的 worker 数量。默认使用 **'%{::processorcount}'**。
     - **BarbicanDebug** - 启用调试。
     - **BarbicanPolicies** - 定义要为 barbican 配置的策略。使用 hash 值，例如： **{ barbican-context_is_admin: { key: context_is_admin, value: 'role:admin' } }**。然后，此条目会添加到 **/etc/barbican/policy.json** 中。后续章节中将详细介绍策略。
     - **BarbicanSimpleCryptoKek** - director 生成密钥加密密钥(KEK) （如果没有指定）。

2. 此步骤为 barbican 准备新的容器镜像。您需要包含 **configure-barbican.yaml** 和所有相关模板文件。更改以下示例以适合您的部署：

```
$ openstack overcloud container image prepare \
 --namespace example.lab.local:5000/rhosp13 \
 --tag 2018-06-06.1 \
 --push-destination 192.168.100.1:8787 \
 --output-images-file ~/container-images-with-barbican.yaml \
 -e /home/stack/virt/config_lvm.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
 -e /home/stack/virt/network/network-environment.yaml \
 -e /home/stack/virt/hostnames.yml \
 -e /home/stack/virt/nodes_data.yaml \
 -e /home/stack/virt/extra_templates.yaml \
 -e /home/stack/virt/docker-images.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml \
 -e /home/stack/configure-barbican.yaml
```

3. 将新容器镜像上传到 undercloud registry：

```
$ openstack overcloud container image upload --debug --config-file container-images-with-barbican.yaml
```

4. 准备新的环境文件：

```
$ openstack overcloud container image prepare \
  --tag 2018-06-06.1 \
  --namespace 192.168.100.1:8787/rhosp13 \
  --output-env-file ~/container-parameters-with-barbican.yaml \
  -e /home/stack/virt/config_lvm.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/virt/network/network-environment.yaml \
  -e /home/stack/virt/hostnames.yml \
  -e /home/stack/virt/nodes_data.yaml \
  -e /home/stack/virt/extra_templates.yaml \
  -e /home/stack/virt/docker-images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml \
  -e /home/stack/configure-barbican.yaml
```

5. 要将这些更改应用到部署：更新 overcloud 并指定您在以前的 *openstack overcloud deploy [...]* 中使用的所有 heat 模板文件。例如：

```
$ openstack overcloud deploy \
    --timeout 100 \
    --templates /usr/share/openstack-tripleo-heat-templates \
    --stack overcloud \
    --libvirt-type kvm \
    --ntp-server clock.redhat.com \
    -e /home/stack/virt/config_lvm.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
    -e /home/stack/virt/network/network-environment.yaml \
    -e /home/stack/virt/hostnames.yml \
    -e /home/stack/virt/nodes_data.yaml \
    -e /home/stack/virt/extra_templates.yaml \
    -e /home/stack/container-parameters-with-barbican.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/barbican-backend-simple-crypto.yaml \
    -e /home/stack/configure-barbican.yaml \
    --log-file overcloud_deployment_38.log
```

3.1. 将用户添加到 OVERCLOUD 的创建者角色

用户必须是 **创建者** 角色的成员，才能创建和编辑 barbican secret，或者创建加密的卷将机密存储在 barbican 中。

1. 检索 **创建者** 角色的 **id** ：

```
openstack role show creator
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
```

```
| domain_id | None                             |
| id        | 4e9c560c6f104608948450fbf316f9d7 |
| name      | creator                          |
+-----------+----------------------------------+
```

注意

除非安装了 OpenStack Key Manager (barbican)，否则您不会看到 **创建** 者角色。

2. 将用户分配到 **创建** 者角色，并指定相关的项目。在本例中，**project_a** 项目中名为 **user1** 的用户被添加到 **创建者角色中** ：

```
openstack role add --user user1 --project project_a 4e9c560c6f104608948450fbf316f9d7
```

3.1.1. 测试 barbican 功能

这部分论述了如何测试 barbican 是否正常工作。

1. 创建测试 secret。例如：

```
$ openstack secret store --name testSecret --payload 'TestPayload'
+---------------+------------------------------------------------------------------------------------+
| Field         | Value                                                                              |
+---------------+------------------------------------------------------------------------------------+
| Secret href   | https://192.168.123.163/key-manager/v1/secrets/4cc5ffe0-eea2-449d-9e64-
b664d574be53 |
| Name          | testSecret                                                                         |
| Created       | None                                                                               |
| Status        | None                                                                               |
| Content types | None                                                                               |
| Algorithm     | aes                                                                                |
| Bit length    | 256                                                                                |
| Secret type   | opaque                                                                             |
| Mode          | cbc                                                                                |
| Expiration    | None                                                                               |
+---------------+------------------------------------------------------------------------------------+
```

2. 检索您刚才创建的 secret 的有效负载：

```
openstack secret get https://192.168.123.163/key-manager/v1/secrets/4cc5ffe0-eea2-449d-
9e64-b664d574be53 --payload
+---------+------------+
| Field   | Value      |
+---------+------------+
| Payload | TestPayload |
+---------+------------+
```

3.2. 了解策略

Barbican 使用策略来确定允许哪些用户对 secret 执行操作，如添加或删除密钥。要实施这些控件，keystone 项目角色（如您之前创建的 **创建者** ）映射到 barbican 内部权限。因此，分配给这些项目角色的用户会收到对应的 barbican 权限。

3.2.1. 查看默认策略

默认策略在代码中定义，通常不需要任何修改。您可以通过从 **barbican** 源代码生成来查看默认策略：

1. 在非生产环境的系统上执行以下步骤，因为可以下载并安装其他组件。本例切换到 **queens** 分支，因此在使用不同版本时必须调整此功能。

```
git clone https://github.com/openstack/barbican
cd /home/stack/barbican
git checkout origin/stable/queens
tox -e genpolicy
```

   这会在包含默认设置的子目录中生成策略文件： **etc/barbican/policy.yaml.sample**。请注意，此路径引用仓库中的子目录，而不是系统的 **/etc** 目录。下方的步骤中阐述了此文件的内容。

2. 您生成的 **policy.yaml.sample** 文件描述了 barbican 使用的策略。该策略由四个不同的角色实施，用于定义用户如何与 secret 和 secret 元数据交互。用户通过分配给特定角色来接收这些权限：

   - **admin** - 可以删除、创建/编辑和读取 secret。
   - **创建者** - 可以创建/编辑和读取机密。无法删除 secret。
   - **observer** - 只能读取数据。
   - **Audit** - 只能读取元数据。不能读取 secret。
     例如，以下条目列出每个项目的 **admin**、**watch** r 和 **creator** Keystone 角色：在右侧，注意它们被分配了 **role:admin**、**role:observer** 和 **role:creator** 权限：

```
#
#"admin": "role:admin"

#
#"observer": "role:observer"

#
#"creator": "role:creator"
```

     这些角色也可通过 barbican 分组在一起。例如，指定 **admin_or_creator** 的规则可以应用到 **rule:admin** 或 **rule:creator** 的成员。

3. 也可以在文件中进一步向下，有 **secret:put** 和 **secret:delete** 操作。因此，请注意哪些角色具有执行这些操作的权限。在以下示例中，**secret:delete** 表示只有 **admin** 和 **creator** 角色成员才能删除 secret 条目。此外，规则也指出该项目的 **admin** 或 **creator** 角色中的用户可以删除该项目中的机密。项目匹配由 **secret_project_match** 规则定义，该规则也在策略中定义。

```
secret:delete": "rule:admin_or_creator and rule:secret_project_match"
```

第 4 章 在 BARBICAN 中管理 SECRET

4.1. 列出 SECRET

secret 由 URI 标识，以 href 值表示。本例显示了您在上一步中创建的 secret：

```
$ openstack secret list
+------------------------------------------------------------------------------+------+-------------------------
+--------+-----------------------------------------+----------+-----------+------------+------+------------+
| 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       |
+------------------------------------------------------------------------------+------+-------------------------
+--------+-----------------------------------------+----------+-----------+------------+------+------------+
```

4.2. 添加新 SECRET

创建测试 secret。例如：

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

4.3. 更新 SECRET

您无法更改 secret 的有效负载（而不是删除 secret），但如果您最初创建了 secret 而不指定有效负载，则之后可以使用 **更新** 功能为它添加有效负载。例如：

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

4.4. 删除 SECRET

您可以通过指定 URI 来删除 secret。例如：

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

4.5. 生成对称密钥

对称密钥用于某些任务，如 nova 磁盘加密和 swift 对象加密。

1. 使用 **创建顺序** 生成一个新的 256 位密钥，并将它存储在 barbican 中。例如：

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

- **--mode** - 生成的密钥可以配置为使用特定的模式，如 **ctr** 或 **cbc**。如需更多信息，请参阅 *NIST SP 800-38A*。

2. 查看订购的详情，标识生成的键的位置，**如下所示** ：

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

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

4.6. 备份和恢复密钥

备份和恢复加密密钥的过程因后端的类型而异：

4.6.1. 备份和恢复简单的加密后端

对于 *简单的* 加密后端，需要备份两个独立的组件：KEK 和数据库。建议您定期测试备份和恢复过程。

4.6.1.1. 备**份和恢复 KEK**

对于 *简单的加密* 后端，您需要备份包含主 KEK 的 **barbican.conf** 文件。此文件必须备份至安全强化的位置。实际数据存储在 Barbican 数据库中，该数据库采用加密状态，具体参见下一节。

- 要从备份中恢复密钥，您需要将恢复的 **barbican.conf** 复制到现有的 **barbican.conf**。

4.6.1.2. 备**份和恢复后端数据**库

这个步骤描述了如何为简单加密后端备份和恢复 barbican 数据库。为演示这一点，您将生成一个密钥，并将 secret 上传到 barbican。然后，您将备份 barbican 数据库，并删除您创建的 secret。然后，您将恢复数据库，并确认您之前创建的 secret 已恢复。

注意

请确定您还备份 KEK，因为这也是一个重要的要求。这是在上一节中介绍的。

4.6.1.2.1. 创建 test secret

1. 在 overcloud 上，使用 **创建的顺序** 生成一个新的 256 位密钥，并将它存储在 barbican 中。例如：

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret order create --name swift_key --
algorithm aes --mode ctr --bit-length 256 --payload-content-type=application/octet-stream key
+----------------+-----------------------------------------------------------------------+
| Field          | Value                                                                 |
+----------------+-----------------------------------------------------------------------+
| Order href     | http://10.0.0.104:9311/v1/orders/2a11584d-851c-4bc2-83b7-35d04d3bae86 |
| Type           | Key                                                                   |
| Container href | N/A                                                                   |
| Secret href    | None                                                                  |
| Created        | None                                                                  |
```

```
| Status        | None                                                                 |
| Error code    | None                                                                 |
| Error message | None                                                                 |
+---------------+----------------------------------------------------------------------+
```

2. 创建测试 secret：

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret store --name testSecret --payload
'TestPayload'
+---------------+----------------------------------------------------------------------+
| Field         | Value                                                                |
+---------------+----------------------------------------------------------------------+
| Secret href   | http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a |
| Name          | testSecret                                                           |
| Created       | None                                                                 |
| Status        | None                                                                 |
| Content types | None                                                                 |
| Algorithm     | aes                                                                  |
| Bit length    | 256                                                                  |
| Secret type   | opaque                                                               |
| Mode          | cbc                                                                  |
| Expiration    | None                                                                 |
+---------------+----------------------------------------------------------------------+
```

3. 确认创建了 secret：

```
(overcloud) [stack@undercloud-0 ~]$ 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       |
+------------------------------------------------------------------------+-----------+-----------------------
---+--------+----------------------------------------+----------+-----------+------------+------+------
------+
```

4.6.1.2.2. 备份 barbican 数据库

登录 **controller-0** 节点时运行这些步骤。

注意

只有 *barbican* 用户有权访问 *barbican* 数据库。因此，需要 barbican 用户密码来备份或恢复数据库。

1. 检索 *barbican* 用户密码。例如：

```
[heat-admin@controller-0 ~]$ sudo grep -r "barbican::db::mysql::password"
/etc/puppet/hieradata
/etc/puppet/hieradata/service_configs.json:    "barbican::db::mysql::password":
"seDJRsMNRrBdFryCmNUEFPPev",
```

2. 备份 *barbican* 数据库：

```
[heat-admin@controller-0 ~]$ mysqldump -u barbican -p"seDJRsMNRrBdFryCmNUEFPPev"
barbican > barbican_db_backup.sql
```

3. 数据库备份保存在 */home/heat-admin*中

```
[heat-admin@controller-0 ~]$ ll
total 36
-rw-rw-r--. 1 heat-admin heat-admin 36715 Jun 19 18:31 barbican_db_backup.sql
```

4.6.1.2.3. 删除测试 secret

1. 在 overcloud 上，删除之前创建的 secret，并验证它们不再存在。例如：

```
(overcloud) [stack@undercloud-0 ~]$ openstack secret delete
http://10.0.0.104:9311/v1/secrets/93f62cfd-e008-401f-be74-bf057c88b04a
(overcloud) [stack@undercloud-0 ~]$ openstack secret delete
http://10.0.0.104:9311/v1/secrets/f664b5cf-5221-47e5-9887-608972a5fefb
(overcloud) [stack@undercloud-0 ~]$ openstack secret list

(overcloud) [stack@undercloud-0 ~]$
```

4.6.1.2.4. 恢复数据库

登录 **controller-0** 节点时运行这些步骤。

1. 确保控制器上的 *barbican* 数据库授予对 **barbican** 用户以进行数据库恢复的访问权限：

```
[heat-admin@controller-0 ~]$ mysql -u barbican -p"seDJRsMNRrBdFryCmNUEFPPev"
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3799
Server version: 10.1.20-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| barbican           |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

```
MariaDB [(none)]> exit
Bye
[heat-admin@controller-0 ~]$
```

9）将备份文件恢复到 **barbican** 数据库：

+

```
[heat-admin@controller-0 ~]$ sudo mysql -u barbican -p"seDJRsMNRrBdFryCmNUEFPPev"
barbican < barbican_db_backup.sql
[heat-admin@controller-0 ~]$
```

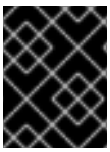
4.6.1.2.5. 验证恢复过程

1. 在 overcloud 上，验证测试 secret 是否已成功恢复：

```
(overcloud) [stack@undercloud-0 ~]$ 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       |
+------------------------------------------------------------------------+-----------+----------------------
---+--------+-----------------------------------------+----------+-----------+------------+------+------
------+
(overcloud) [stack@undercloud-0 ~]$
```

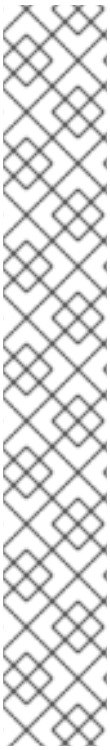
第 5 章 加密 CINDER 卷

您可以使用 barbican 来管理 Block Storage (cinder)加密密钥。此配置使用 LUKS 加密附加到您的实例的磁盘，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，并将它存储在 barbican 中。引导实例时（或附加加密卷）时，nova 将密钥从 barbican 检索，并在本地存储为 Compute 节点上的 Libvirt secret。



重要

Nova 会在其第一次使用时格式化加密卷（如果它们未加密）。然后，生成的块设备会提供给 Compute 节点。



注意

如果您打算更新任何配置文件，请注意某些 OpenStack 服务现在在容器内运行；这适用于 keystone、nova 和 cinder 等。因此，有一些管理实践需要考虑：

- 不要更新您在物理节点的主机操作系统上找到的任何配置文件，例如：**/etc/cinder/cinder.conf**。容器化服务不引用此文件。
- 不要更新容器中运行的配置文件。重新启动容器后，更改将会丢失。相反，如果您必须更改容器化服务，请更新 **/var/lib/config-data/puppet-generated/** 中的配置文件，该配置文件用于生成容器。

  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**

  在重启容器后，会应用更改。

1. 在运行 **cinder-volume** 和 **nova-compute** 服务的节点上，确认 nova 和 cinder 都配置为使用 barbican 进行密钥管理：

```
$ crudini --get /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf
key_manager backend
castellan.key_manager.barbican_key_manager.BarbicanKeyManager

$ crudini --get /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf
key_manager backend
castellan.key_manager.barbican_key_manager.BarbicanKeyManager
```

2. 创建使用加密的卷模板。当您创建新卷时，它们可以从您在这里定义的设置中建模：

```
$ 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
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| 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
|
+-------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
```

3. 创建新卷，并指定它使用 **LuksEncryptor-Template-256** 设置：

注意

确保创建加密的卷的用户在项目上具有 **创建者** barbican 角色。如需更多信息，请参阅 **Grant 用户访问创建者角色** 部分。

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

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：

```
$ openstack secret list
```

```
+------------------------------------------------------------------------------+------+----------------
---------+--------+-----------------------------------------+----------+------------+-------------+------
+------------+
| 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       |
+------------------------------------------------------------------------------+------+----------------
---------+--------+-----------------------------------------+----------+------------+-------------+------
+------------+
```

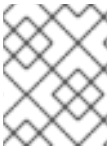
5. 将新卷附加到现有实例。例如：

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

然后，卷会呈现给客户端操作系统，并可使用内置工具挂载。

5.1. 将现有卷密钥迁移到 BARBICAN

在以前的版本中，部署可能已使用 **ConfKeyManager** 来管理磁盘加密密钥。这意味着生成了固定密钥，然后存储在 nova 和 cinder 配置文件中。可以使用以下步骤将密钥 ID 迁移到 barbican。这个工具适用于通过扫描范围中的 **encryption_key_id** 条目迁移到 barbican 来实现。每个条目获得一个新的 barbican 密钥 ID，并且保留现有的 **ConfKeyManager** secret。



注意

在以前的版本中，您可以使用 **ConfKeyManager** 来为加密的卷重新分配所有权。对于具有由 barbican 管理的密钥的卷来说，这不可能实现。

注意

激活 barbican 不会破坏现有的 **keymgr** 卷。

启用后，迁移过程会自动运行，但需要一些配置，具体如下一节中所述。实际迁移在 **cinder-volume** 和 **cinder-backup** 进程中运行，您可以在 cinder 日志文件中跟踪进度。

- **cinder-volume** - 迁移存储在 Cinder 卷和快照表中的密钥。
- **cinder-backup** - 在备份表中迁移密钥。

5.1.1. 迁移步骤概述

1. 部署 barbican 服务。
2. 将 **创建者** 角色添加到 cinder 服务。例如：

```
#openstack role create creator
#openstack role add --user cinder creator  --project service
```

3. 重启 **cinder-volume** 和 **cinder-backup** 服务。
4. **cinder-volume** 和 **cinder-backup** 会自动开始迁移过程。
5. 监控日志中的显示显示迁移完成的消息，并检查更多的卷是否使用 **ConfKeyManager** all-zeros 加密密钥 ID。
6. 从 **cinder.conf** 和 **nova.conf** 删除 **fixed_key** 选项。您必须确定哪个节点已经配置有此设置。
7. 从 cinder 服务中删除 **创建** 者角色。

5.1.2. 行为差异

Barbican 管理的加密卷的行为与使用 **ConfKeyManager** 的卷不同：

- 您无法转让加密卷的所有权，因为目前无法传输 barbican secret 的所有权。
- Barbican 对允许读取和删除 secret 的限制更为严格的，可能会影响一些 Cinder 卷操作。例如，用户无法附加、分离或删除不同用户的卷。

5.1.3. 查看迁移过程

本节论述了如何查看迁移任务的状态。启动进程后，日志中会显示这些条目之一。这表明迁移是否正确启动，或者它识别了它遇到的问题：

- **没有迁移加密密钥，因为 ConfKeyManager 仍在使用。**
- **没有迁移加密密钥，因为 ConfKeyManager 的 fixed_key 没有被使用。**
- **不支持迁移加密密钥，因为迁移到 'XXX' key_manager 后端不被支持。** - 此消息不太可能出现 - 这个信息。它是一种安全检查，可以处理使用 barbican 以外的其他密钥管理器后端代码。这是因为代码只支持一个迁移场景： *From* ConfKeyManager *to* barbican。
- **没有迁移加密密钥，因为没有与此主机关联的卷。** - 当 **cinder-volume** 在多个主机上运行时，特定主机没有关联任何卷。这是因为每个主机负责处理自己的卷。
- **开始迁移 ConfKeyManager 键。**
- **将卷 <UUID> 加密密钥迁移到 Barbican** - 迁移期间，将检查所有主机的卷；如果卷仍在使用 ConfKeyManager 的密钥 ID （识别它的所有零事实(**00000000-0000-0000-0000-0000-0000-00000000000000000**0000000000)，则会出现这个消息。
  - 对于 **cinder-backup**，这个消息使用略有不同的大写： **Migrating Volume [...]** 或 **Migrating Backup [...]**
- 每个主机检查其所有卷后，主机会显示摘要状态信息：

  > `No volumes are using the ConfKeyManager's encryption_key_id.`
  > `No backups are known to be using the ConfKeyManager's encryption_key_id.`

  您可能还会看到以下条目：

  使用 ConfKeyManager 的全零加密密钥 ID仍然有 %d 卷。仍然会使用 ConfKeyManager 的全零加密密钥 ID 来 %d 备份。请注意，这两个消息都可能会出现在 **cinder-volume** 和 **cinder-backup** 日志中。每个服务只处理其自身条目的迁移，但该服务知道了另一个状态。因

此，**cinder-volume** 知道 **cinder-backup** 仍然具有要迁移的备份，**cinder-backup** 知道 **cinder-volume** 服务是否具有要迁移的卷。

虽然每个主机仅迁移自己的卷，但摘要信息基于全局评估，确定是否有卷是否还需要迁移。收到确认后，从 **cinder.conf** 和 **nova.conf** 中删除 **fixed_key** 设置。如需更多信息，请参阅 *清理下面的固定密钥* 部分。

5.1.4. 对迁移过程进行故障排除

5.1.4.1. 角色分配

barbican secret 只能在 requestor 具有创建者角色时才会 **创建**。这意味着 cinder 服务本身需要创建者角色，否则会出现类似以下内容的日志序列：

1. **开始迁移 ConfKeyManager 键。**
2. **将卷 <UUID> 加密密钥迁移到 Barbican**
3. **迁移加密密钥时出错： Forbidden: Secret creationtries not allowed - 请检查您的用户/项目权限**
4. **ConfKeyManager 的 all-zeros 加密密钥 ID 仍存在 %d 卷。**

关键信息是第三条： **Secret 创建尝试没有被允许。**要解决这个问题，更新 **cinder** 帐户的权限：

1. 运行 **openstack role add --project service --user cinder creator**
2. 重启 **cinder-volume** 和 **cinder-backup** 服务。

因此，迁移的下一次尝试应该会成功。

5.1.5. 清理固定密钥

重要

encryption_key_id 最近仅添加到 **Backup** 表中，作为 Queens 发行版本的一部分。因此，已存在的加密卷备份可能会存在。all-zeros **encryption_key_id** 存储在备份本身中，但它不会出现在 **备份** 数据库中。因此，迁移过程无法了解某些加密卷的备份是否仍依赖于全零的 **ConfKeyMgr** 密钥 ID。

将密钥 ID 迁移到 barbican 后，固定的密钥将保留在配置文件中。这可能会对某些用户造成安全问题，因为 **fixed_key** 值没有在 **.conf** 文件中加密。要解决这个问题，您可以从 nova 和 cinder 配置中手动删除 **fixed_key** 值。但是，在进行前，首先完成测试并检查日志文件的输出，因为仍然依赖于此值的磁盘无法访问。

1. 检查现有的 **fixed_key** 值。值必须与这两个服务匹配。

```
crudini --get /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf keymgr
fixed_key
crudini --get /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keymgr
fixed_key
```

2. 重要信息：备份现有的 **fixed_key** 值。这可让您在出现问题时恢复该值，或者需要恢复使用旧加密密钥的备份。

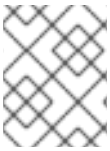
3. 删除 **fixed_key** 值：

```
crudini --del /var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf keymgr
fixed_key
crudini --del /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keymgr
fixed_key
```

第 6 章 加密 AT-REST SWIFT 对象

默认情况下，上传到对象存储的对象以未加密的形式存储。因此，可以从文件系统直接访问对象。如果磁盘在放弃前无法正确擦除，则可能会造成安全隐患。当您启用了 barbican 时，对象存储服务(swift)可以透明地加密和解密您的存储(at-rest)对象。at-rest 加密与 in-transit 加密不同的是它在存储在磁盘上时被加密的对象。

Swift 以透明方式执行这些加密任务，在上传到 swift 时自动加密对象，然后在提供给用户时自动解密。这个加密和解密使用相同的(ymmetric)密钥进行，密钥保存在 barbican 中。



注意

在启用加密并添加到 swift 集群后，您无法禁用加密功能，因为数据现在存储在加密状态中。因此，如果禁用加密，数据将无法被读取，直到您使用相同密钥重新启用加密为止。

6.1. 为 SWIFT 启用 AT-REST 加密

1. 您可以通过在环境文件中包括 **SwiftEncryptionEnabled: True** 来启用 swift 加密功能，然后使用 **/home/stack/overcloud_deploy.sh** 重新运行 **openstack overcloud deploy**。请注意，您仍需要启用 barbican，如 *Install Barbican* 章节中所述。

2. 确认 swift 已配置为使用 at-rest 加密：

```
$ crudini --get /var/lib/config-data/puppet-generated/swift/etc/swift/proxy-server.conf pipeline-main pipeline

pipeline = catch_errors healthcheck proxy-logging cache ratelimit bulk tempurl formpost authtoken keystone staticweb copy container_quotas account_quotas slo dlo versioned_writes kms_keymaster encryption proxy-logging proxy-server
```

其结果应包括 **用于加密** 的条目。

第 7 章 验证 GLANCE 镜像

启用 Barbican 后，您可以配置镜像服务(glance)，以验证已上传的镜像没有被篡改。在这种实现中，镜像首先使用存储在 barbican 中的密钥进行签名。然后，镜像会上传到 glance，以及附带的签名信息。因此，在每次使用前会验证镜像的签名，如果签名不匹配，实例构建过程会失败。

Barbican 与 glance 的集成意味着，您可以使用 **openssl** 命令和私钥为 glance 镜像签名，然后再上传它们。

7.1. 启用 GLANCE 镜像验证

在您的环境文件中，使用 **VerifyGlanceSignatures: True** 设置启用镜像验证。您必须重新运行 **openstack overcloud deploy** 命令，以便此设置生效。

要验证是否已启用 glance 镜像验证，请在 overcloud Compute 节点上运行以下命令：

```
$ sudo crudini --get /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf glance
verify_glance_signatures
```

注意

如果将 Ceph 用作镜像和计算服务的后端，则创建一个 CoW 克隆。因此，无法执行镜像签名验证。

7.2. 验证镜像

要配置 glance 镜像进行验证，请完成以下步骤：

1. 确认 glance 已配置为使用 barbican：

```
$ sudo crudini --get /var/lib/config-data/puppet-generated/glance_api/etc/glance/glance-
api.conf key_manager backend
castellan.key_manager.barbican_key_manager.BarbicanKeyManager
```

2. 生成证书：

```
openssl genrsa -out private_key.pem 1024
openssl rsa -pubout -in private_key.pem -out public_key.pem
openssl req -new -key private_key.pem -out cert_request.csr
openssl x509 -req -days 14 -in cert_request.csr -signkey private_key.pem -out
x509_signing_cert.crt
```

3. 将证书添加到 barbican secret 存储中：

```
$ source ~/overcloudrc
$ openstack secret store --name signing-cert --algorithm RSA --secret-type certificate --
payload-content-type "application/octet-stream" --payload-content-encoding base64  --
payload "$(base64 x509_signing_cert.crt)" -c 'Secret href' -f value
https://192.168.123.170:9311/v1/secrets/5df14c2b-f221-4a02-948e-48a61edd3f5b
```

注意

记录生成的 UUID，以便在后续步骤中使用。在本例中，证书的 UUID 是 **5df14c2b-f221-4a02-948e-48a61edd3f5b**。

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

```
$ openssl dgst -sha256 -sign private_key.pem -sigopt rsa_padding_mode:pss -out cirros-0.4.0.signature cirros-0.4.0-x86_64-disk.img
```

5. 将生成的 **.signature** 文件转换为 *base64* 格式：

```
$ base64 -w 0 cirros-0.4.0.signature  > cirros-0.4.0.signature.b64
```

6. 将 *base64* 值加载到变量中，以便在后续命令中使用它：

```
$ cirros_signature_b64=$(cat cirros-0.4.0.signature.b64)
```

7. 将已签名的镜像上传到 glance。对于 **img_signature_certificate_uuid**，您必须指定您之前上传到 barbican 的签名密钥的 UUID：

```
 openstack image create \
 --container-format bare --disk-format qcow2 \
 --property img_signature="$cirros_signature_b64" \
 --property img_signature_certificate_uuid="5df14c2b-f221-4a02-948e-48a61edd3f5b" \
 --property img_signature_hash_method="SHA-256" \
 --property img_signature_key_type="RSA-PSS" cirros_0_4_0_signed \
 --file cirros-0.4.0-x86_64-disk.img
+---------------------------------+------------------------------------------------------------------------------
----+
| Property                        | Value                                                                        |
+---------------------------------+------------------------------------------------------------------------------
----+
| checksum                        | 25c96942084f61e008d593b6c2cfda00
|
| container_format                | bare                                                                         |
| created_at                      | 2018-01-23T05:37:31Z                                                         |
| disk_format                     | qcow2                                                                        |
| id                              | d3396fa0-2ea2-4832-8a77-d36fa3f2ab27                                         |
| img_signature                   |
lcI7nGgoKxnCyOcsJ4abbEZEpzXByFPIgiPeiT+Otjz0yvW00KNN3fI0AA6tn9EXrp7fb2xBDE4UaO3v |
|                                 |
lFquV/s3mU4LcCiGdBAl3pGsMlmZZIQFVNcUPOaayS1kQYKY7kxYmU9iq/AZYyPw37KQI52smC/zoO54 |
|                                 | zZ+JpnfwIsM=                                                                 |
| img_signature_certificate_uuid | ba3641c2-6a3d-445a-8543-851a68110eab
|
| img_signature_hash_method       | SHA-256                                                                      |
| img_signature_key_type          | RSA-PSS                                                                      |
| min_disk                        | 0                                                                            |
| min_ram                         | 0                                                                            |
| name                            | cirros_0_4_0_signed                                                          |
| owner                           | 9f812310df904e6ea01e1bacb84c9f1a
```

```
|
| protected                    | False                                                                                |
| size                         | None                                                                                 |
| status                       | active                                                                               |
| tags                         | []                                                                                   |
| updated_at                   | 2018-01-23T05:37:31Z                                                                 |
| virtual_size                 | None                                                                                 |
| visibility                   | shared                                                                               |
+------------------------------+--------------------------------------------------------------------------------------
----+
```

8. 您可以查看 Compute 日志中 glance 的镜像验证活动： **/var/log/containers/nova/nova-compute.log**。例如，您可以在实例引导时收到以下条目：

```
2018-05-24 12:48:35.256 1 INFO nova.image.glance [req-7c271904-4975-4771-9d26-
cbea6c0ade31 b464b2fd2a2140e9a88bbdacf67bdd8c a3db2f2beaee454182c95b646fa7331f
- default default] Image signature verification succeeded for image d3396fa0-2ea2-4832-
8a77-d36fa3f2ab27
```