Red Hat OpenStack Services on OpenShift 18.0

Performing security operations

Operating security services in a Red Hat OpenStack Services on OpenShift environment

Last Updated: 2024-08-28

Red Hat OpenStack Services on OpenShift 18.0 Performing security operations

Operating security services in a Red Hat OpenStack Services on OpenShift environment

OpenStack Team
rhos-docs@redhat.com

Abstract

Back up and restore secure information, rotate passwords, and operate secure services in a Red Hat OpenStack Services on OpenShift environment.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Providing documentation feedback in Jira

Use the Create Issue form to provide feedback on the documentation for Red Hat OpenStack Services on OpenShift (RHOSO) or earlier releases of Red Hat OpenStack Platform (RHOSP). When you create an issue for RHOSO or RHOSP documents, the issue is recorded in the RHOSO Jira project, where you can track the progress of your feedback.

To complete the Create Issue form, ensure that you are logged in to Jira. If you do not have a Red Hat Jira account, you can create an account at https://issues.redhat.com.

1. Click the following link to open a **Create Issue** page: Create Issue
2. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.
3. Click **Create**.

CHAPTER 1. ROTATING A USERNAME OR PASSWORD FOR DATABASE ACCOUNTS IN RHOSO

You can rotate the username and password used for database accounts in Red Hat OpenStack Services on OpenShift (RHOSO) by editing an **openstackcontrolplanes** custom resource.

1.1. GENERATING A NEW USERNAME AND PASSWORD FOR DATABASE ACCOUNTS

You can generate a new username and password for a database account by editing the YAML file that contains the templates for the database accounts. The database account corresponds to a MariaDBAccount YAML resource. The MariaDBAccount YAML refers to an actual database username and a Secret that contains a password. The name of the MariaDBAccount and the username that the MariaDBAccount refers to are different. The MariaDBAccount is a resource identifier, which uses dashes to seperate each word, and the MariaDB username uses underscores to seperate each word.

When you change the value of databaseAccount: parameter in the yaml file, this creates a new MariaDBAccount resource, that has a randomly-generated username and secret that contains a password. This resource creates a pod that creates an actual MariaDB database username in the target database.

IMPORTANT

To avoid interruptions to long-running migrations that cannot be interrupted, use a maintenance window. For example, block storage (Cinder) volume migrations or compute service (Nova) host migrations.

Procedure

1. To retrieve the name of the YAML file that contains the details of the database, use the **oc get openstackcontrolplanes** command.
2. To access the YAML file, use the **oc edit openstackcontrolplanes <YAML_File_Name>** command.
    - Replace <YAML_File_Name> with the name of the YAML file that you retrieved in step 1.
3. In the YAML file, locate the database account that you want to generate a new username and password for.
4. Next to the **databaseAccount:** label, change the name of the database account. A new database account and password is automatically generated, and the services automatically switch to the new database account.

    NOTE

    You must separate each word in the MariaDBAccount name with a dash.

5. Optional: To delete the old database account, use the **oc delete mariadbaccounts <mariadbaccount_name>** command. The MariaDBAccount remains until the resources are updated to use a new MariaDBAccount.
    - Replace <mariadbaccount_name> with the name of the database account that you want to delete.

CHAPTER 2. SECURE ROLE BASED ACCESS CONTROL IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

All users who authenticate to the Identity service (keystone) in Red Hat OpenStack Services on OpenShift (RHOSO) 18.0 use secure role-based access control (SRBAC). SRBAC is the updated role-based authentication schema, and is present in all deployments of RHOSO. All policies now use this schema.

2.1. SRBAC IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

Keystone implements secure roles using personas, which are scope-specific roles. You can assign roles to users or groups that are either project or system scoped.

2.1.1. Roles

Roles define which actions users can perform. There are three OpenStack roles that are available within the project and system scopes.

admin

Granting the admin role in any scope includes all create, read, update, or delete operations on resources and APIs.

member

The **member** role is allowed to create, read, update, and delete resources that are owned by the scope in which they are a member.

reader

The **reader** role is for read-only operations, regardless of the scope it is applied to. This role can view resources across the entirety of the scope to which it is applied.

2.1.2. Scope

The scope is the context in which operations are performed. All OpenStack resources are owned by a specific project, so access to those resources is granted by assigning a role to the project scope.

2.1.3. Personas

Project admin

This persona includes create, read, update and delete operations on resources across projects, which includes adding and removing users and other projects. **Granting a user admin in any scope grants full API access to all APIs in all available scopes.**

Project member

The project member persona is for users who are granted permission to consume resources within the project scope. This persona can create, list, update, and delete resources within the project to which they are assigned. This persona implies all permissions granted to project readers.

Project reader

The project reader persona is for users who are granted permission to view non-sensitive resources in the project. On projects, assign the reader role to end users who need to inspect or view resources, or to auditors, who only need to view project-specific resources within a single project for the purposes of an audit The project-reader persona will not address all auditing use cases.

System admin

System administrators are typically cloud administrators who can affect the behavior of the deployment. **Granting a user admin in any scope grants full API access to all APIs in all available scopes.**

System members and system readers

System members and system readers have the same authorization and can view all resources within keystone. The **system reader** persona is useful for auditors if the audit does not require access to sensitive information.

NOTE

Additional personas based on the **domain** scope are not available for use.

2.2. ASSIGNING ROLES IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

With SRBAC on Red Hat OpenStack Services on OpenShift (RHOSO), you can assign users to the role of admin, member, or reader within either the project or system scope.

Procedure

- Use syntax similar to the following to provide a user a role with a project scope:

  Example command

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

- Use syntax similar to the following to provide a user a role with a system scope:

  Example command

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

  Replace <role> with either reader, member, or admin.

CHAPTER 3. INTRODUCTION TO THE IDENTITY SERVICE (KEYSTONE)

As a cloud administrator, you can manage projects, users, and roles.

Projects within the Red Hat OpenStack Services on OpenShift (RHOSO) data plane are organizational units containing a collection of resources. You can assign OpenStack users to roles within those projects. Roles define the actions that those users can perform on the resources within a given project. Users can be assigned roles in multiple projects.

Each RHOSO data plane deployment must include at least one user assigned to a role within a project. As a cloud administrator, a user can perform the following actions within OpenStack:

- Add, update, and delete projects and users.
- Assign users to one or more roles, and change or remove these assignments.
- Manage projects and users independently from each other.

You can also configure user authentication with the Identity service (keystone)to control access to services and endpoints.

3.1. AUTHENTICATE WITH CLOUDS.YAML

Prerequisites

- The administrator has created a project for you and they have provided you with a **clouds.yaml** file for you to access the cloud.
- You have installed the **python-openstackclient** package.

NOTE

To execute **openstack** client commands on the cloud you must specify the name of the cloud detailed in your **clouds.yaml** file. You can specify the name of the cloud in one of two ways:

Use the **--os-cloud** option with each command:

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

Use this option if you access more than one cloud.

Create an environment variable for the cloud name in your **bashrc** file:

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

CHAPTER 4. MANAGING USERS

As an OpenStack administrator, you can add, modify, and delete users in the dashboard. Users can be members of one or more projects. You can manage projects and users independently from each other.

4.1. CREATING USERS WITH THE DASHBOARD

You can assign a primary project and role to the user. Users that you create with OpenStack Dashboard (horizon) are Identity service users by default. You can integrate Active Directory users by configuring the LDAP provider included with the Identity service.

Procedure

1. Log in to the Dashboard as an admin user.
2. Select **Identity > Users**.
3. Click **Create User**.
4. Enter a user name, email, and preliminary password for the user.
5. Select a project from the **Primary Project** list.
6. Select a role for the user from the **Role** list. The default role is **member**.
7. Click **Create User**.

4.2. EDITING USERS WITH THE DASHBOARD

You can update user details, including the primary project.

Procedure

1. Log in to the dashboard as an admin user.
2. Select **Identity > Users**.
3. In the **Actions** column, click **Edit**.
4. In the **Update User** window, you can update the **User Name**, **Email**, and **Primary Project**.
5. Click **Update User**.

4.3. ENABLING OR DISABLING USERS WITH THE DASHBOARD

You can disable a user with the dashboard. This action is reversible, unlike deleting a user.

Limitations:

- You cannot disable or enable more than one user at a time.
- You cannot set the primary project of a user to active.

The result is that the user you have disabled cannot:

- Log in to the dashboard.
- Get access to OpenStack services.
- Do any user-project action in the dashboard.

Procedure

1. As an admin user in the dashboard, select **Identity > Users**.
2. In the **Actions** column, click the arrow, and select **Enable User** or **Disable User**. In the **Enabled** column, the value then updates to either **True** or **False**.

4.4. DELETING A USER WITH THE DASHBOARD

You must be a user with an administrative role to delete other users. This action cannot be reversed.

Procedure

1. As an admin user in the dashboard, select **Identity > Users**.
2. Select the users you want to delete.
3. Click **Delete Users**. The **Confirm Delete Users** window is displayed.
4. Click **Delete Users** to confirm the action.

CHAPTER 5. MANAGING ROLES

Red Hat OpenStack Services on OpenShift (RHOSO) uses secure role-based access control (SRBAC) to manage access to its resources. Roles define which actions users can perform. By default, there are three predefined roles:

- A reader role for viewing permissions on specificed projects.
- A member role for creating and managing resources within specified projects.
- An administrative role to enable non-admin users to administer the environment.

You can also create custom roles specific to your environment.

5.1. UNDERSTANDING OPENSTACK ADMIN ROLE

When you assign a user the role of **admin** in an OpenStack project, this user has permissions to view, change, create, or delete any resource on any project. This user can create shared resources that are accessible across projects, such as publicly available glance images, or provider networks. Additionally, a user with the **admin** role can create or delete users and manage roles.

The project to which you assign a user the **admin** role is the default project in which **openstack** commands are executed. For example, if an **admin** user in a project named **development** runs the following command, a network called **internal-network** is created in the **development** project:

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

The **admin** user can create an **internal-network** in any project by using the **--project** parameter:

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

5.2. VIEWING ROLES WITH THE CLI

As an administrator, you can view the details of existing roles.

Procedure

1. List the available predefined roles:

   ```
   $ openstack role list
   ```

1. View details for a specified role:

   ```
   $ openstack role show admin
   ```

NOTE

To get detailed information on the permissions associated with each role, you must audit its access to each API call.

5.3. ASSIGNING ROLES IN RED HAT OPENSTACK SERVICES ON OPENSHIFT

With SRBAC on Red Hat OpenStack Services on OpenShift (RHOSO), you can assign users to the role of admin, member, or reader within either the project or system scope.

Procedure

- Use syntax similar to the following to provide a user a role with a project scope:

  Example command

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

- Use syntax similar to the following to provide a user a role with a system scope:

  Example command

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

  Replace <role> with either reader, member, or admin.

CHAPTER 6. MANAGING GROUPS

You can use Identity Service (keystone) groups to assign consistent permissions to multiple user accounts.

6.1. CONFIGURING GROUPS WITH THE DASHBOARD

You can use the dashboard to manage the membership of keystone groups. However, you must use the command-line to assign role permissions to a group.

6.1.1. Creating a group

1. Log in to the dashboard as a user with administrative privileges.
2. Select **Identity > Groups**.
3. Click **+Create Group**.
4. Enter a name and description for the group.
5. Click **Create Group**.

6.1.2. Managing Group membership

You can use the dashboard to manage the membership of keystone groups.

1. Log in to the dashboard as a user with administrative privileges.
2. Select **Identity > Groups**.
3. Click **Manage Members** for the group that you want to edit.
4. Use **Add users** to add a user to the group. If you want to remove a user, mark its checkbox and click **Remove users**.

CHAPTER 7. QUOTA MANAGEMENT

As a cloud administrator, you can set and manage quotas for a project. Each project is allocated resources, and project users are granted access to consume these resources. This enables multiple projects to use a single cloud without interfering with each other's permissions and resources. A set of resource quotas are preconfigured when a new project is created. The quotas include the amount of VCPUs, instances, RAM, and floating IPs that can be assigned to projects. You can set or modify Compute and Block Storage quotas for new and existing projects using the dashboard. For more information, see Managing projects.

7.1. VIEWING COMPUTE QUOTAS FOR A PROJECT

Run the following command to list the currently set quota values.

Procedure

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

CHAPTER 8. MANAGING PROJECTS

As an OpenStack administrator, you can create and manage projects. A project is a pool of shared virtual resources, to which you can assign OpenStack users and groups. You can configure the quota of shared virtual resources in each project. You can create multiple projects that will not interfere with each other's permissions and resources. Users can be associated with more than one project. Each user must have a role assigned for each project to which they are assigned.

8.1. CREATING A PROJECT

Create a project, add members to the project and set resource limits for the project.

1. Log in to the Dashboard as a user with administrative privileges.
2. Select **Identity > Projects**.
3. Click **Create Project**.
4. On the **Project Information** tab, enter a name and description for the project. The **Enabled** check box is selected by default.
5. On the **Project Members** tab, add members to the project from the **All Users** list.
6. On the **Quotas** tab, specify resource limits for the project.
7. Click **Create Project**.

8.2. EDITING A PROJECT

You can edit a project to change its name or description, enable or temporarily disable it, or update the members in the project.

1. Log in to the Dashboard as a user with administrative privileges.
2. Select **Identity > Projects**.
3. In the project **Actions** column, click the arrow, and click **Edit Project**.
4. In the **Edit Project** window, you can update a project to change its name or description, and enable or temporarily disable the project.
5. On the **Project Members** tab, add members to the project, or remove them as needed.
6. Click **Save**.

NOTE

The **Enabled** check box is selected by default. To temporarily disable the project, clear the **Enabled** check box. To enable a disabled project, select the **Enabled** check box.

8.3. DELETING A PROJECT

1. Log in to the Dashboard as a user with administrative privileges.
2. Select **Identity > Projects**.

3. Select the project that you want to delete.

4. Click **Delete Projects**. The **Confirm Delete Projects** window is displayed.

5. Click **Delete Projects** to confirm the action.

The project is deleted and any user pairing is disassociated.

8.4. UPDATING PROJECT QUOTAS

Quotas are operational limits that you set for each project to optimize cloud resources. You can set quotas to prevent project resources from being exhausted without notification. You can enforce quotas at both the project and the project-user level.

1. Log in to the Dashboard as a user with administrative privileges.

2. Select **Identity > Projects**.

3. In the project **Actions** column, click the arrow, and click **Modify Quotas**.

4. In the **Quota** tab, modify project quotas as needed.

5. Click **Save**.

NOTE

At present, *nested quotas* are not yet supported. As such, you must manage quotas individually against projects and subprojects.

8.5. CHANGING THE ACTIVE PROJECT

Set a project as the active project so that you can use the dashboard to interact with objects in the project. To set a project as the active project, you must be a member of the project. It is also necessary for the user to be a member of more than one project to have the **Set as Active Project** option be enabled. You cannot set a disabled project as active, unless it is re-enabled.

1. Log in to the Dashboard as a user with administrative privileges.

2. Select **Identity > Projects**.

3. In the project **Actions** column, click the arrow, and click **Set as Active Project**.

4. Alternatively, as a non-admin user, in the project **Actions** column, click **Set as Active Project** which becomes the default action in the column.

8.6. PROJECT HIERARCHIES

You can nest projects using multitenancy in the Identity service (keystone). Multitenancy allows subprojects to inherit role assignments from a parent project.

8.6.1. Creating hierarchical projects and sub-projects

You can implement Hierarchical Multitenancy (HMT) using keystone domains and projects. First create a new domain and then create a project within that domain. You can then add subprojects to that project. You can also promote a user to administrator of a subproject by adding the user to the **admin** role for that subproject.

NOTE

The HMT structure used by keystone is not currently represented in the dashboard.

Procedure

1. Create a new keystone domain called **corp**:

   Example command

   ```
   $ openstack domain create corp
   ```

   Example output

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

2. Create the parent project (**private-cloud**) within the **corp** domain:

   Example command

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

   Example output

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

3. Create a subproject (**dev**) within the **private-cloud** parent project, while also specifying the **corp** domain:

   Example command

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

Example output

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

4. Create another subproject called **qa**:

Example command

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

Example output

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

NOTE

You can use the Identity API to view the project hierarchy. For more information, see https://developer.openstack.org/api-ref/identity/v3/index.html?expanded=show-project-details-detail

8.6.2. Configuring access to hierarchical projects

By default, a newly-created project has no assigned roles. When you assign role permissions to the parent project, you can include the **--inherited** flag to instruct the subprojects to inherit the assigned permissions from the parent project. For example, a user with **admin** role access to the parent project also has **admin** access to the subprojects.

Granting access to users

1. View the existing permissions assigned to a project:

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

2. View the existing roles:

Example command

```
$ openstack role list
```

Example output

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

3. Grant the user account **user1** access to the **private-cloud** project:

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

Re-run this command using the **--inherited** flag. As a result, **user1** also has access to the **private-cloud** subprojects, which have inherited the role assignment:

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

4. Review the result of the permissions update:

Example command

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

Example output

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

The **user1** user has inherited access to the **qa** and **dev** projects. In addition, because the **--inherited** flag was applied to the parent project, **user1** also receives access to any subprojects that are created later.

Removing access from users

Explicit and inherited permissions must be separately removed.

1. Remove a user from an explicitly assigned role:

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

2. Review the result of the change. Notice that the inherited permissions are still present:

Example command

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

Example output

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

3. Remove the inherited permissions:

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

4. Review the result of the change. The inherited permissions have been removed, and the resulting output is now empty:

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

8.6.3. Reseller project overview

With the *Reseller* project, the goal is to have a hierarchy of domains; these domains will eventually allow you to consider reselling portions of the cloud, with a subdomain representing a fully-enabled cloud. This work has been split into phases, with phase 1 described below:

Phase 1 of reseller

Reseller (phase 1) is an extension of Hierarchical Multitenancy (HMT), described here: Creating hierarchical projects and sub-projects. Previously, keystone domains were originally intended to be containers that stored users and projects, with their own table in the database back-end. As a result, domains are now no longer stored in their own table, and have been merged into the project table:

- A domain is now a type of project, distinguished by the **is_domain** flag.
- A domain represents a top-level project in the project hierarchy: domains are roots in the project hierarchy
- APIs have been updated to create and retrieve domains using the **projects** subpath:
  - Create a new domain by creating a project with the **is_domain** flag set to true
  - List projects that are domains: get projects including the **is_domain** query parameter.

8.7. PROJECT SECURITY MANAGEMENT

Security groups are sets of IP filter rules that can be assigned to project instances, and which define networking access to the instance. Security groups are project specific; project members can edit the default rules for their security group and add new rule sets.

All projects have a default security group that is applied to any instance that has no other defined security group. Unless you change the default values, this security group denies all incoming traffic and allows only outgoing traffic from your instance.

You can apply a security group directly to an instance during instance creation, or to a port on the running instance.

NOTE

You cannot apply a role-based access control (RBAC)-shared security group directly to an instance during instance creation. To apply an RBAC-shared security group to an instance you must first create the port, apply the shared security group to that port, and then assign that port to the instance. See Adding a security group to a port in *Creating and managing instances*.

Do not delete the default security group without creating groups that allow required egress. For example, if your instances use DHCP and metadata, your instance requires security group rules that allow egress to the DHCP server and metadata agent.

8.7.1. Creating a security group

Create a security group so that you can configure security rules. For example, you can enable ICMP traffic, or disable HTTP requests.

Procedure

1. In the dashboard, select **Project > Compute > Access & Security**
2. On the **Security Groups** tab, click **Create Security Group**.
3. Enter a name and description for the group, and click **Create Security Group**.

8.7.2. Adding a security group rule

By default, rules for a new group only provide outgoing access. You must add new rules to provide additional access.

Procedure

1. In the dashboard, select **Project > Compute > Access & Security**
2. On the **Security Groups** tab, click **Manage Rules** for the security group that you want to edit.
3. Click **Add Rule** to add a new rule.
4. Specify the rule values, and click **Add**.
   The following rule fields are required:

   Rule

   Rule type. If you specify a rule template (for example, *SSH*), its fields are automatically filled in:

   - TCP: Typically used to exchange data between systems, and for end-user communication.
   - UDP: Typically used to exchange data between systems, particularly at the application level.
   - ICMP: Typically used by network devices, such as routers, to send error or monitoring messages.

   Direction

   Ingress (inbound) or Egress (outbound).

   Open Port

   For TCP or UDP rules, the **Port** or **Port Range** (single port or range of ports) to open:

   - For a range of ports, enter port values in the **From Port** and **To Port** fields.
   - For a single port, enter the port value in the **Port** field.

   Type

   The type for ICMP rules; must be in the range *-1:255*.

   Code

   The code for ICMP rules; must be in the range *-1:255*.

   Remote

   The traffic source for this rule:

   - CIDR (Classless Inter-Domain Routing): IP address block, which limits access to IPs within the block. Enter the CIDR in the Source field.
   - Security Group: Source group that enables any instance in the group to access any other group instance.

8.7.3. Deleting a security group rule

Delete security group rules that you no longer require.

Procedure

1. In the dashboard, select **Project > Compute > Access & Security**
2. On the **Security Groups** tab, click **Manage Rules** for the security group.
3. Select the security group rule, and click **Delete Rule**.
4. Click **Delete Rule** again.

NOTE

You cannot undo the delete action.

8.7.4. Deleting a security group

Delete security groups that you no longer require.

Procedure

1. In the dashboard, select **Project > Compute > Access & Security**
2. On the **Security Groups** tab, select the group, and click **Delete Security Groups**.
3. Click **Delete Security Groups**.

NOTE

You cannot undo the delete action.

CHAPTER 9. MANAGING DOMAINS

Identity Service (keystone) domains are additional namespaces that you can create in keystone.

NOTE

Identity Service includes a built-in domain called **Default**. It is suggested you reserve this domain only for service accounts, and create a separate domain for user accounts.

9.1. VIEWING A LIST OF DOMAINS

You can view a list of domains with the **openstack domain list** command:

Example command

```
$ openstack domain list
```

Example output

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

9.2. CREATING A NEW DOMAIN

You can create a new domain with the **openstack domain create** command:

Example command

```
$ openstack domain create TestDomain
```

Example output

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

9.3. VIEWING THE DETAILS OF A DOMAIN

You can view the details of a domain with the **openstack domain show** command:

Example command

```
$ openstack domain show TestDomain
```

Example output

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

9.4. DISABLING A DOMAIN

You can disable and enable domains according to your requirements.

Procedure

1. Disable a domain using the **--disable** option:

   Example command

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

2. Confirm that the domain has been disabled:

   Example command

   ```
   $ openstack domain show TestDomain
   ```

   Example output

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

3. Use the **--enable** option to re-enable the domain, if required:

   Example command

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

CHAPTER 10. APPLICATION CREDENTIALS

Use *Application Credentials* to avoid the practice of embedding user account credentials in configuration files. Instead, the user creates an Application Credential that receives delegated access to a single project and has its own distinct secret. The user can also limit the delegated privileges to a single role in that project. This allows you to adopt the principle of least privilege, where the authenticated service gains access only to the one project and role that it needs to function, rather than all projects and roles.

With Application Credentials, you can consume an API without revealing your user credentials, and applications can authenticate to Keystone without requiring embedded user credentials.

You can use Application Credentials to generate tokens and configure **keystone_authtoken** settings for applications. These use cases are described in the following sections.

NOTE

The Application Credential is dependent on the user account that created it, so it will terminate if that account is ever deleted, or loses access to the relevant role.

10.1. USING APPLICATION CREDENTIALS TO GENERATE TOKENS

Application Credentials are available to users as a self-service function in the dashboard. This example demonstrates how a user can create an Application Credential and then use it to generate a token.

1. Create a test project, and test user accounts:

   a. Create a project called **AppCreds**:

      ```
      $ openstack project create AppCreds
      ```

   b. Create a user called **AppCredsUser**:

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

   c. Grant **AppCredsUser** access to the **member** role for the **AppCreds** project:

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

2. Log in to the dashboard as **AppCredsUser** and create an Application Credential: **Overview → Identity → Application Credentials → +Create Application Credential**.

   NOTE

   Ensure that you download the **clouds.yaml** file contents, because you cannot access it again after you close the pop-up window titled **Your Application Credential**.

3. Create a file named **/home/stack/.config/openstack/clouds.yaml** using the CLI and paste the contents of the **clouds.yaml** file.

   ```
   # 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"
```

NOTE

These values will be different for your deployment.

4. Use the Application Credential to generate a token. You must not be sourced as any specific user when using the following command, and you must be in the same directory as your **clouds.yaml** file.

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

NOTE

If you receive an error similar to **__init__() got an unexpected keyword argument 'application_credential_secret'**, then you might still be sourced to the previous credentials. For a fresh environment, run **sudo su - stack**.

10.2. INTEGRATING APPLICATION CREDENTIALS WITH APPLICATIONS

Application Credentials can be used to authenticate applications to keystone. When you use Application Credentials, the **keystone_authtoken** settings use **v3applicationcredential** as the authentication type and contain the credentials that you receive during the credential creation process. Enter the following values:

- **application_credential_secret**: The Application Credential secret.
- **application_credential_id**: The Application Credential id.
- (Optional) **application_credential_name**: You might use this parameter if you use a named application credential, rather than an ID.

For example:

```
[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. MANAGING APPLICATION CREDENTIALS

You can use the command line to create and delete Application Credentials.

The **create** subcommand creates an application credential based on the currently sourced account. For example, creating the credential when sourced as an **admin** user will grant the same roles to the Application Credential:

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

WARNING

Using the **--unrestricted** parameter enables the application credential to create and delete other application credentials and trusts. This is potentially dangerous behavior and is disabled by default. You cannot use the **--unrestricted** parameter in combination with other access rules.

By default, the resulting role membership includes all the roles assigned to the account that created the credentials. You can limit the role membership by delegating access only to a specific role:

```
$ 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
u0Ih-4aCg |
| unrestricted | False                                                                                  |
+--------------+----------------------------------------------------------------------------------------+
```

To delete an Application Credential:

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

10.4. REPLACING APPLICATION CREDENTIALS

Application credentials are bound to the user account that created them and become invalid if the user account is ever deleted, or if the user loses access to the delegated role. As a result, you should be prepared to generate a new application credential as needed.

Replacing existing application credentials for configuration files

Update the application credentials assigned to an application (using a configuration file):

1. Create a new set of application credentials.
2. Add the new credentials to the application configuration file, replacing the existing credentials. For more information, see Integrating Application Credentials with applications.
3. Restart the application service to apply the change.
4. Delete the old application credential, if appropriate. For more information about the command line options, see Managing Application Credentials.

Replacing the existing application credentials in clouds.yaml

When you replace an application credential used by **clouds.yaml**, you must create the replacement credentials using OpenStack user credentials. By default, you cannot use application credentials to create another set of application credentials. The **openstack application credential create** command creates an application credential based on the currently sourced account.

1. Authenticate as the OpenStack user that originally created the authentication credentials that are about to expire. For example, if you used the procedure Using Application Credentials to generate tokens, you must log in again as **AppCredsUser**.

2. Create an Application Credential called **AppCred2**. This can be done using the OpenStack Dashboard, or the **openstack** CLI interface:

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

3. Copy the **id** and **secret** parameters from the output of the previous command. The **secret** parameter value cannot be accessed again.

4. Replace the **application_credential_id** and **application_credential_secret** parameter values in the **${HOME}/.config/openstack/clouds.yaml** file with the **secret** and **id** values that you copied.

Verification

1. Generate a token with clouds.yaml to confirm that the credentials are working as expected. You must not be sourced as any specific user when using the following command, and you must be in the same directory as your **clouds.yaml** file:

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

   Example output:

   ```
   +------------+---------------------------------------------------------------------------------------------------
   -----------------------------------------------------------------------------------------------------------+
   | Field      | Value
   |
   +------------+---------------------------------------------------------------------------------------------------
   -----------------------------------------------------------------------------------------------------------+
   | expires    | 2018-08-29T05:37:29+0000
   |
   | id         | gAAAAABbhiMJ4TxxFITMdsYJpfStsGotPrns0lnpvJq9ILtdi-
   NKqisWBeNiJlUXwmnoGQDh2CMyK9OeTsuEXnJNmFfKjxiHWmcQVYzAhMKo6_QMUtu_Qm
   6mtpzYYHBrUGboa_Ay0LBuFDtsjtgtvJ-r8G3TsJMowbKF-yo--
   O_XLhERU_QQVl3hl8zmMRdmLh_P9Cbhuolt |
   | project_id | 1a74eabbf05c41baadd716179bb9e1da
   |
   | user_id    | ef679eeddfd14f8b86becfd7e1dc84f2
   |
   +------------+---------------------------------------------------------------------------------------------------
   -----------------------------------------------------------------------------------------------------------+
   ```

CHAPTER 11. MANAGING SECRETS AND KEYS WITH OPENSTACK KEY MANAGER (BARBICAN)

The OpenStack Key Manager (barbican) service for Red Hat OpenStack Services on OpenShift (RHOSO) securely stores, provisions and manages your secret data. You use OpenStack Key Manager to create, update, and delete secrets and encryption keys.

11.1. ABOUT THE RED HAT OPENSTACK SERVICES ON OPENSHIFT KEY MANAGER OPERATOR (BARBICAN)

The OpenStack Key Manager (barbican) service is the secrets manager for Red Hat OpenStack Services on OpenShift (RHOSO). This Operator is installed and active by default. An OpenStack administrator can use the the Key Manager API and command line to centrally manage the certificates, keys, and passwords used by OpenStack. The RHOSO Key Manager service integrates with Block Storage (cinder), Networking (neutron), and Compute (nova) services.

Secrets such as certificates, API keys, and passwords, are stored in the barbican database or directly in a secure storage system. In RHOSO 18.0. the simple crypto plug-in is available to encrypt secrets.

OpenStack Key Manager supports the following use cases:

- **Symmetric encryption keys**
  - Block Storage (cinder) volume encryption
  - Object Storage (swift) encryption
  - Ephemeral disk encryption
- **Asymmetric keys and certificates**
  - Image service (glance) image signing
  - Image service (glance) image verification

11.2. VIEWING SECRETS

To view the list of secrets, run the **openstack secret list** command. The list includes the URI, name, type, and other information about the secrets.

Procedure

- View the list of secrets:

  Example command

  ```
  $ openstack secret list
  ```

  Example output

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

```
| 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. CREATING A SECRET

To create a secret, run the **openstack secret store** command and specify the name of the secret and optionally the payload for the secret.

Procedure

- Create a secret:

  #### Example command

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

  #### Example Output

  ```
  +---------------+-----------------------------------------------------------------------------------+
  | 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. ADDING A PAYLOAD TO A SECRET

You cannot change the payload of a secret, you can only delete it. If you create a secret without specifying a payload, then you can later add a payload to it by using the **openstack secret update** command.

Procedure

- Add a payload to a secret:

Example command

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

11.5. DELETING A SECRET

To delete a secret, run the **openstack secret delete** command and specify the secret URI.

Procedure

- Delete a secret with the specified URI:

  Example command

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

11.6. GENERATING A SYMMETRIC KEY

You can generate a symmetric key for specific tasks such as Compute service disk encryption and Object Storage service encryption.

Procedure

1. Generate a new 256-bit key using **order create** and store it in barbican. For example:

   Example command

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

   Example output

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

   You can also use the **--mode** option to configure generated keys to use a particular mode, such as **ctr** or **cbc**. For more information, see *NIST SP 800-38A*.

2. View the details of the order to identify the location of the generated key, shown here as the **Secret href** value:

   Example command

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

   Example output

   ```
   +----------------+-----------------------------------------------------------------------------------+
   | 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. Retrieve the details of the secret:

   Example command

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

   Example output

   ```
   +---------------+-----------------------------------------------------------------------------------+
   | 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. ENCRYPTING BLOCK STORAGE (CINDER) VOLUMES

You can use barbican to manage your Block Storage service (cinder) encryption keys. This configuration uses LUKS to encrypt the disks attached to your instances, including boot disks. Key management is transparent to the user; when you create a new volume using **luks** as the encryption type, cinder generates a symmetric key secret for the volume and stores it in barbican. When booting the instance or attaching an encrypted volume, the Compute service retrieves the key from the Key Manager service and stores the secret locally as a libvirt secret on the Compute node.

Procedure

1. Create a volume template that uses encryption. When you create new volumes, they can be modeled off the settings you define here:

   Example command

   ```
   $ 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
   ```

   Example output

   ```
   +-------------+--------------------------------------------------------------------------------------------------
   ------------------------------------------------------------------------------------+
   | 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. Create a new volume and specify that it uses the **LuksEncryptor-Template-256** settings:

   Example command

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

   Example output

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

The resulting secret is automatically uploaded to the barbican back end.

3. Ensure that you can see the barbican secret UUID. This value is displayed in the **encryption_key_id** field.

Example command

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

Example output

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

NOTE

You must use the **--os-volume-api-version 3.64** parameter with the Cinder CLI to display the **encryption_key_id** value. There is no equivalent OpenStack CLI command.

4. Use barbican to confirm that the disk encryption key is present. In this example, the timestamp matches the LUKS volume creation time:

   Example command

   ```
   $ openstack secret list
   ```

   Example output

   ```
   +------------------------------------------------------------------------------------+------+------------------
   ---------+--------+-----------------------------------------+-----------+------------+-------------+------
   +------------+
   | 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. Attach the new volume to an existing instance:

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

The volume is now available for you to mount to the guest file system.

11.8. SIGNING IMAGE SERVICE (GLANCE) IMAGES

When you configure the Image Service (glance) to validate images, you must sign those images before uploading them. Use the **openssl** command to sign an image with a key that is stored in the Key Manager service (barbican), and then upload the image to the Image service with the accompanying image properties. The signature of the image is verified before each use, and the instance build process fails if the signature does not match.

Limitations

- Signature verification is based on **openSSL _rsa_sig_verify()**, which currently raises an **InvalidSignature** exception in the Image service if the key length is not 1024. This issue occurs because of a regression in OpenSSL 3.0.7. Signature verification works correctly with key length 2048 and OpenSSL 3.0.2.
- OpenSSL does not support the SHA256 algorithm. You must use SHA512 or greater to create a signed image.

Procedure

1. Use the **openssl** command to create the keys and the certificate:
   - Create the private key:

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

   - Create the public key:

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

   - Create the certificate:

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

   - When prompted, enter details to include in your certificate request:

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

2. Upload the certificate to the Key Manager service, where it is stored in the **secret store**:

   ```
   $ 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)"
   ```

   Example output:

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

   NOTE

   Record the certificate’s UUID for use in a later step. In this example, the UUID is **cd7cc675-e573-419c-8fff-33a72734a243**.

3. Retrieve an image and create a signature:
   - Retrieve an image:

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

   - Use **private_key.pem** to sign the image and generate the **.signature** file. For example:

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

   - Convert the **.signature** file to Base64 format:

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

   - Load the Base64 value into a variable to use when you upload the image:

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

4. Upload an image with valid signature properties to the Image service:

   ```
   $ 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. VALIDATING SNAPSHOTS

Snapshots are saved as Image service (glance) images. If you configure the Compute service (nova) to check for signed images, then snapshots must by signed, even if they were created from an instance with a signed image.

Procedure

1. Download the snapshot from the Image service:

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

   - Replace **<local_file_name>** with the file name of the downloaded image.
   - Replace **<snapshot_image_name>** with the name of your snapshot.

2. Generate a signature to validate the snapshot. Use the same process that you use to generate a signature to validate an image. For more information, see Signing Image service (glance) images.

3. Update the snapshot image properties:

   ```
   $ 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>
```

- Replace **<cd7cc675-e573-419c-8fff-33a72734a243>** with the UUID of the certificate for the image.
- Replace **<snapshot_image_id>** with the ID of your snapshot.

4. Optional: Remove the downloaded Image service image from the filesystem:

```
$ rm <local_file_name>
```