Red Hat OpenStack Platform 16.1

Users and Identity Management Guide

Managing users and keystone authentication

Last Updated: 2020-10-18

Red Hat OpenStack Platform 16.1 Users and Identity Management Guide

Managing users and keystone authentication

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide explains how to manage application credentials, users, roles, projects, and quotas.

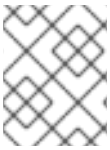
Table of Contents

PREFACE

As a cloud administrator, you can manage projects, users, and roles. Projects are organizational units in the cloud to which you can assign users. Projects are also known as tenants or accounts. Users can be members of one or more projects. Roles define the actions that users can perform.

Each OpenStack deployment must include at least one project, one user, and one role, linked together. As a cloud administrator, you can add, update, and delete projects and users, assign users to one or more projects, and change or remove these assignments. You can manage projects and users independently from each other.

You can also configure user authentication with the Keystone identity service to control access to services and endpoints. Keystone provides token-based authentication and can integrate with LDAP and Active Directory, so you can manage users and identities externally and synchronize the user data with Keystone.



NOTE

Keystone v2 was deprecated in Red Hat OpenStack Platform 11 (Ocata). It was removed in Red Hat OpenStack Platform 13 (Queens), leaving only Keystone v3 available.

CHAPTER 1. USER MANAGEMENT

1.1. USER MANAGEMENT

As a cloud 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.

1.1.1. Create a User

Use this procedure to create users in the dashboard. You can assign a primary project and role to the user. Note that users created in the dashboard are Keystone users by default. To integrate Active Directory users, you can configure the LDAP provider included in the Red Hat OpenStack Platform Identity service.

1. As an admin user in the dashboard, select **Identity > Users**.
2. Click **Create User**.
3. Enter a user name, email, and preliminary password for the user.
4. Select a project from the **Primary Project** list.
5. Select a role for the user from the **Role** list (the default role is **_member_**).
6. Click **Create User**.

1.1.2. Edit a User

Use this procedure to update the user's details, including the primary project.

1. As an admin user in the dashboard, select **Identity > Users**.
2. In the User's **Actions** column, click **Edit**.
3. In the **Update User** window, you can update the **User Name**, **Email**, and **Primary Project**.
4. Click **Update User**.

1.1.3. Enable or Disable a User

Use this procedure to enable or disable a user. You can disable or enable only one user at a time. A disabled user cannot log in to the dashboard, and does not have access to any OpenStack services. Also, a disabled user's primary project cannot be set as active. A disabled user can be enabled again, unlike deleting a user where the action cannot be reversed. A disabled user must be re-enabled for any user-project action in the dashboard.

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**.

1.1.4. Delete a User

As an admin user, use this procedure to delete a user using the dashboard. This action cannot be reversed, unlike disabling a user. Deleted users get delisted from a project's members' list for projects it belongs to. All roles associated with the user-project pair are also lost.

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 2. ROLE MANAGEMENT

2.1. ROLE MANAGEMENT

OpenStack uses a role-based access control (RBAC) mechanism to manage access to its resources. Roles define which actions users can perform. By default, there are two predefined roles: a member role that gets attached to a tenant, and an administrative role to enable non-admin users to administer the environment. Note that there are abstract levels of permission, and it is possible to create the roles the administrator needs, and configure services adequately.

2.1.1. View Roles

Use the following command to list the available predefined roles.

```
$ openstack role list
+----------------------------------+---------------+
| ID                               | Name          |
+----------------------------------+---------------+
| 4fd37c2c993a4acab8e1b5896afb8687 | SwiftOperator |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_      |
| a0f19c1381c54770ae068456c4411d82 | ResellerAdmin |
| ae49e2b796ea4820ac51637be27650d8 | admin         |
+----------------------------------+---------------+
```

To get details for a specified role, run:

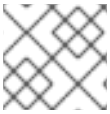
```
$ openstack role show admin
```

Example

```
$ openstack role show admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | ae49e2b796ea4820ac51637be27650d8 |
| name      | admin                            |
+-----------+----------------------------------+
```

2.1.2. Create and Assign a Role

As a cloud administrator, you can create and manage roles on the Keystone client using the following set of commands. Each OpenStack deployment must include at least one project, one user, and one role, linked together. However, users can be members of multiple projects. To assign users to multiple projects, create a role and assign that role to a user-project pair. Note that you can create a user and assign a primary project and default role in the dashboard.



NOTE

Either the name or ID can be used to specify users, roles, or projects.

1. Create the **new-role** role:

```
$ openstack role create [ROLE_NAME]
```

Example

```
$ openstack role create new-role
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 880c116b6a55464b99ca8d8d8fe26743 |
| name      | new-role                         |
+-----------+----------------------------------+
```

2. To assign a user to a project, you must assign the role to a user-project pair. To do this, obtain the user, role, and project names or IDs:

   a. List users:

   ```
   $ openstack user list
   ```

   b. List roles:

   ```
   $ openstack role list
   ```

   c. List projects:

   ```
   $ openstack project list
   ```

3. Assign a role to a user-project pair.

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

   Example

   In this example, you assign the **admin** role to the **admin** user in the **demo** project:

   ```
   $ openstack role add --project demo --user 895e43465b9643b9aa29df0073572bb2
   ae49e2b796ea4820ac51637be27650d8
   ```

4. Verify the role assignment for the user **admin**:

   ```
   $ openstack role assignment list --user [USER_ID]  --project [PROJECT_ID]
   ```

   Example

   ```
   $ openstack role assignment list --user 895e43465b9643b9aa29df0073572bb2 --project
   demo
   +----------------------------------+----------------------------------+-------+----------------------------------
   --+--------+-----------+
   | Role                             | User                             | Group | Project                          | Domain |
   Inherited |
   +----------------------------------+----------------------------------+-------+----------------------------------
   ```

```
--+--------+-----------+
| ae49e2b796ea4820ac51637be27650d8 | 895e43465b9643b9aa29df0073572bb2 |       |
7efbdc8b4ab448b8b5aeb9fa5898ce23 |        | False     |
+----------------------------------+----------------------------------+-------+-------------------------------
--+--------+-----------+
```

2.2. IMPLIED ROLES AND DOMAIN-SPECIFIC ROLES

2.2.1. Implied roles

In OpenStack, access control is enforced by confirming that a user is assigned to a specific role. Until recently, those roles had to be explicitly assigned to either a user, or to a group in which the user was a member. Identity Service (keystone) has now added the concept of implied role assignments: If a user is explicitly assigned to a role, then the user could be implicitly assigned to additional roles as well.

2.2.2. Inference Rules

Implied assignment is managed by role inference rules. An inference rule is written in the form **superior implies subordinate**. For example, a rule might state that the **admin** role implies the **_member_** role. As a result, a user assigned to **admin** for a project would implicitly be assigned to the **_member_** role as well.

With *implied roles*, a user's role assignments are processed cumulatively, allowing the user to inherit the subordinate roles. This result is dependent on an inference rule being created that specifies this outcome.

2.2.2.1. Keystone Configuration

For keystone to observe implied roles, the **infer_roles** setting must be enabled in */etc/keystone/keystone.conf*:

```
[token]
infer_roles = true
```

Implied roles are governed by a defined set of inference rules. These rules determine how a role assignment can result in the implied membership of another role. See Section 2.2.3.1, "Demonstration of Implied Roles" for an example.

2.2.3. Prevent Certain Roles From Being Implied

You can prevent certain roles from being implied onto a user. For example, in */etc/keystone/keystone.conf*, you can add a **ListOpt** of roles:

```
[assignment]
prohibited_implied_role = admin
```

This will prevent a user from ever being assigned a role implicitly. Instead, the user will need to be explicitly granted access to that role.

2.2.3.1. Demonstration of Implied Roles

This section describes how to create an inference rule, resulting in an implied role. These rules control how one role can imply membership of another. The example rule used in the following procedure will imply that members of the **admin** role also have **_member_** access:

2.2.3.1.1. Assign a Role to the User

1. Retrieve the ID of a user that will have the **_member_** role implied. For example:

```
$ openstack user show User1
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | ce803dd127c9489199c89ce3b68d39b4 |
| name                | User1                            |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

2. Retrieve the ID of the **demo** project:

```
$ openstack project show demo
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | default tenant                   |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 2717ebc905e449b5975449c370edac69 |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | default                          |
+-------------+----------------------------------+
```

3. Retrieve the ID of the **admin** role:

```
$ openstack role show admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 9b821b2920544be7a4d8f71fa99fcd35 |
| name      | admin                            |
+-----------+----------------------------------+
```

4. Give the **User1** user **admin** privileges to the **demo** project:

```
$ openstack role add --user User1 --project demo admin
```

5. Confirm the **admin** role assignment:

```
$ openstack role assignment list --user User1 --project demo --effective
+----------------------------------+----------------------------------+-------+------------------------------
```

```
--+--------+-----------+
| Role                             | User                             | Group | Project                          | Domain |
Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| 9b821b2920544be7a4d8f71fa99fcd35 | ce803dd127c9489199c89ce3b68d39b4 |       |
2717ebc905e449b5975449c370edac69 |        | False     |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
```

2.2.3.1.2. Create the Inference Rule

Now that you have granted the **admin** role to *User1*, run the following steps to create the inference rule:

1. First, confirm User1's current role membership:

```
$ openstack role assignment list --user User1 --project demo --effective
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| Role                             | User                             | Group | Project                          | Domain |
Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| 9b821b2920544be7a4d8f71fa99fcd35 | ce803dd127c9489199c89ce3b68d39b4 |       |
2717ebc905e449b5975449c370edac69 |        | False     |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
```

2. Retrieve the list of role IDs:

```
$ openstack role list
+----------------------------------+--------------+
| ID                               | Name         |
+----------------------------------+--------------+
| 9b821b2920544be7a4d8f71fa99fcd35 | admin        |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_     |
| ea199fe4293745719c2afd3402ed7b95 | ResellerAdmin |
| fe8eba5dfd1e4f4a854ad20a150d995e | SwiftOperator |
+----------------------------------+--------------+
```

3. Create the inference rule. These are currently created using **curl**. This example uses the IDs of the roles returned in the previous step. It also runs the command using the **admin_token** in *keystone.conf*:

```
source overcloudrc
export OS_TOKEN=`grep ^admin_token /etc/keystone/keystone.conf | awk -F'=' '{print $2}'`
curl -X PUT  -H "X-Auth-Token: $OS_TOKEN" -H "Content-type: application/json"
$OS_AUTH_URL/roles/9b821b2920544be7a4d8f71fa99fcd35/implies/9fe2ff9ee4384b1894a90
878d3e92bab
```

4. Review the results using the CLI. In this example, User1 has received implied access to the **_member_** role, as indicated by ID **9fe2ff9ee4384b1894a90878d3e92bab**:

```
source overcloudrc
```

```
# openstack role assignment list --user User1 --project demo --effective
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| Role                             | User                             | Group | Project                          | Domain |
Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| 9b821b2920544be7a4d8f71fa99fcd35 | ce803dd127c9489199c89ce3b68d39b4 |       |
2717ebc905e449b5975449c370edac69 |        | False     |
| 9fe2ff9ee4384b1894a90878d3e92bab | ce803dd127c9489199c89ce3b68d39b4 |       |
2717ebc905e449b5975449c370edac69 |        | False     |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
```

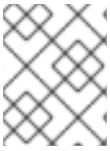
5. Review your inference rules using *curl*:

```
source overcloudrc
export OS_TOKEN=`grep ^admin_token /etc/keystone/keystone.conf | awk -F'=' '{print $2}'`
curl -s -H "X-Auth-Token: $OS_TOKEN" $OS_AUTH_URL/role_inferences | python -
mjson.tool
{
    "role_inferences": [
        {
            "implies": [
                {
                    "id": "9fe2ff9ee4384b1894a90878d3e92bab",
                    "links": {
                        "self":
"https://osp.lab.local:5000/v3/roles/9fe2ff9ee4384b1894a90878d3e92bab"
                    },
                    "name": "_member_"
                }
            ],
            "prior_role": {
                "id": "9b821b2920544be7a4d8f71fa99fcd35",
                "links": {
                    "self": "https://osp.lab.local:5000/v3/roles/9b821b2920544be7a4d8f71fa99fcd35"
                },
                "name": "admin"
            }
        }
    ]
}
```

2.2.4. Domain-Specific Roles

Domain-specific roles grant you more granular control when defining rules for roles, allowing the roles to act as aliases for the existing *prior* roles. Note that you cannot have a global role implying a domain-specific role. As a result, if you list the effective role assignments of a user in a project, the domain-specific roles will not be present.

Domain-specific roles can be created by a user who administers their keystone domain; they do not have to be administrators of the OpenStack deployment. This means that a domain-specific role definition can be limited to a specific domain.



NOTE

Domain-specific roles cannot be used to scope a token. This can only be done with global roles.

2.2.4.1. Using Domain-Specific Roles

This example describes how to create a domain specific role and review its effect.

1. Create a domain:

```
$ openstack domain create corp01
```

2. Create a role that specifies a domain (note that this parameter is distinct from **--domain**):

```
$ openstack role create operators --role-domain domain-corp01
```

CHAPTER 3. GROUP MANAGEMENT

3.1. MANAGE KEYSTONE GROUPS

3.1.1. Using the Command-line

You can use Identity Service (keystone) groups to assign consistent permissions to multiple user accounts. This example creates a group and then assigns permissions to the group. As a result, members of the group will inherit the same permissions that were assigned to the group:

NOTE

The **openstack group** subcommands require keystone **v3**.

1. Create the group **grp-Auditors**:

```
$ openstack group create grp-Auditors
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| id          | 2a4856fc242142a4aa7c02d28edfdfff |
| name        | grp-Auditors                     |
+-------------+----------------------------------+
```

2. View a list of keystone groups:

```
$ openstack group list --long
+----------------------------------+--------------+-----------+-------------+
| ID                               | Name         | Domain ID | Description |
+----------------------------------+--------------+-----------+-------------+
| 2a4856fc242142a4aa7c02d28edfdfff | grp-Auditors | default   |             |
+----------------------------------+--------------+-----------+-------------+
```

3. Grant the **grp-Auditors** group permission to access the **demo** project, while using the **_member_** role:

```
$ openstack role add _member_ --group grp-Auditors --project demo
```

4. Add the existing user **user1** to the **grp-Auditors** group:

```
$ openstack group add user grp-Auditors user1
user1 added to group grp-Auditors
```

5. Confirm that **user1** is a member of **grp-Auditors**:

```
$ openstack group contains user grp-Auditors user1
user1 in group grp-Auditors
```

6. Review the effective permissions that have been assigned to **user1**:

```
$ openstack role assignment list --effective --user user1
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| Role                             | User                             | Group | Project                          | Domain |
Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | 3fefe5b4f6c948e6959d1feaef4822f2 |       |
0ce36252e2fb4ea8983bed2a568fa832 |        | False     |
+----------------------------------+----------------------------------+-------+----------------------------------
--+--------+-----------+
```

3.1.2. Using Dashboard

You can use the dashboard to manage the membership of keystone groups. You will need to use the command-line to assign role permissions to a group, as covered in the previous example.

3.1.2.1. Create a Group

1. As an admin user in the dashboard, select **Identity > Groups**.
2. Click **+Create Group**.
3. Enter a name and description for the group.
4. Click **Create Group**.

3.1.2.2. Manage Group Membership

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

1. As an admin user in the dashboard, select **Identity > Groups**.
2. Click **Manage Members** for the group you need to edit.
3. Use **Add users** to add a user to the group. If you need to remove a user, mark its checkbox and click or **Remove users**.

CHAPTER 4. QUOTA MANAGEMENT

4.1. 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 tenant is created. The quotas include the amount of VCPUs, instances, RAM, floating IPs, that can be assigned to tenants. Quotas can be enforced at both the tenant (or project) and the tenant-user level. Note that you can set or modify Compute and Block Storage quotas for new and existing tenants using the dashboard. See Chapter 5, *Project Management* for the procedure on how to set and update project quotas within the dashboard.

4.1.1. View Compute Quotas for a User

Run the following command to list the currently set quota values for a user:

```
$ nova quota-show --user [USER] --tenant [TENANT]
```

Example

```
$ nova quota-show --user demoUser --tenant demo
+-----------------------------+-------+
| Quota                       | Limit |
+-----------------------------+-------+
| instances                   | 10    |
| cores                       | 20    |
| ram                         | 51200 |
| floating_ips                | 5     |
| fixed_ips                   | -1    |
| metadata_items              | 128   |
| injected_files              | 5     |
| injected_file_content_bytes | 10240 |
| injected_file_path_bytes    | 255   |
| key_pairs                   | 100   |
| security_groups             | 10    |
| security_group_rules        | 20    |
| server_groups               | 10    |
| server_group_members        | 10    |
+-----------------------------+-------+
```

4.1.2. Update Compute Quotas for a User

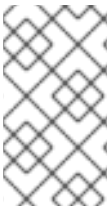
Run the following commands to update a particular quota value:

```
$ nova quota-update --user [USER] --[QUOTA_NAME] [QUOTA_VALUE] [TENANT]
$ nova quota-show --user [USER] --tenant [TENANT]
```

Example

```
$ nova quota-update --user demoUser --floating-ips 10 demo
$ nova quota-show --user demoUser --tenant demo
```

```
+-----------------------------+-------+
| Quota                       | Limit |
+-----------------------------+-------+
| instances                   | 10    |
| cores                       | 20    |
| ram                         | 51200 |
| floating_ips                | 10    |
| ...                         |       |
+-----------------------------+-------+
```



NOTE

To view a list of options for the quota-update command, run:

```
$ nova help quota-update
```

4.1.3. Set Object Storage Quotas for a User

Object Storage quotas can be classified under the following categories:

- Container quotas - Limits the total size (in bytes) or number of objects that can be stored in a single container.
- Account quotas - Limits the total size (in bytes) that a user has available in the Object Storage service.

To set either container quotas or the account quotas, the Object Storage proxy server must have the parameters **container_quotas** or **account_quotas** (or both) added to the **[pipeline:main]** section of the **proxy-server.conf** file:

```
[pipeline:main]
pipeline = catch_errors [...] tempauth container-quotas \
account-quotas slo dlo proxy-logging proxy-server

[filter:account_quotas]
use = egg:swift#account_quotas

[filter:container_quotas]
use = egg:swift#container_quotas
```

Use the following command to view and update the Object Storage quotas. All users included in a project can view the quotas placed on the project. To update the Object Storage quotas on a project, you must have the role of a ResellerAdmin in the project.

To view account quotas:

```
# swift stat

Account: AUTH_b36ed2d326034beba0a9dd1fb19b70f9
Containers: 0
Objects: 0
Bytes: 0
Meta Quota-Bytes: 214748364800
```

```
X-Timestamp: 1351050521.29419
Content-Type: text/plain; charset=utf-8
Accept-Ranges: bytes
```

To update quotas:

```
# swift post -m quota-bytes:<BYTES>
```

For example, to place a 5 GB quota on an account:

```
# swift post -m quota-bytes:5368709120
```

To verify the quota, run the **swift stat** command again:

```
# swift stat

Account: AUTH_b36ed2d326034beba0a9dd1fb19b70f9
Containers: 0
Objects: 0
Bytes: 0
Meta Quota-Bytes: 5368709120
X-Timestamp: 1351541410.38328
Content-Type: text/plain; charset=utf-8
Accept-Ranges: bytes
```

CHAPTER 5. PROJECT MANAGEMENT

5.1. PROJECT MANAGEMENT

As a cloud administrator, you can create and manage projects (tenants). A tenant describes a project with an assigned number of OpenStack users and resources. It is possible to set up quotas for each tenant. This enables multiple projects to use a single cloud without interfering with each other's permissions and resources. The words project and tenant are used interchangeably. Users can be associated with more than one project. Each user-project pairing must have a role associated with it.

5.1.1. Create a Project

Use this procedure to create projects, add members to the project, and set resource limits for the project.

1. As an admin user in the dashboard, select **Identity > Projects**.
2. Click **Create Project**.
3. On the **Project Information** tab, enter a name and description for the project (the **Enabled** check box is selected by default).
4. On the **Project Members** tab, add members to the project from the **All Users** list.
5. On the **Quotas** tab, specify resource limits for the project.
6. Click **Create Project**.

5.1.2. Edit a Project

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

1. As an admin user in the dashboard, select **Identity > Projects**.
2. In the project's **Actions** column, click the arrow, and click **Edit Project**.
3. In the **Edit Project** window, you can update a project to change its name or description, and enable or temporarily disable the project.
4. On the **Project Members** tab, add members to the project, or remove them as needed.
5. 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.

5.1.3. Delete a Project

1. As an admin user in the dashboard, select **Identity > Projects**.
2. Select the project you want to delete.

3. Click **Delete Projects**. The **Confirm Delete Projects** window is displayed.
4. Click **Delete Projects** to confirm the action.

The project gets deleted and any user pairing will be disassociated.

5.1.4. Update Project Quotas

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

1. As an admin user in the dashboard, select **Identity > Projects**.
2. In the project's **Actions** column, click the arrow, and click **Modify Quotas**.
3. In the **Quota** tab, modify project quotas as needed.
4. Click **Save**.

5.1.5. Change Active Project

A user can set a project as the active project only of which they are a member. 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. Setting a project as an active project enables you to access objects in the dashboard for the active project. Note that a disabled project cannot be set as active, unless it is re-enabled.

1. As an admin user in the dashboard, select **Identity > Projects**.
2. In the project's **Actions** column, click the arrow, and click **Set as Active Project**
3. Alternatively, as a non-admin user, in the project's **Actions** column, click **Set as Active Project** which becomes the default action in the column.

5.2. PROJECT HIERARCHIES

5.2.1. Hierarchical Multitenancy (HMT) in Identity Service

Projects can be nested using multitenancy in keystone. Multitenancy allows subprojects to inherit role assignments from a parent project.

5.2.1.1. Create the Project and Subprojects

You can implement Hierarchical Multitenancy (HMT) using keystone domains and projects. Begin by creating a new domain and then creating 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.

For example:

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

```
$ openstack domain create corp
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| enabled     | True                             |
| id          | 69436408fdcb44ab9e111691f8e9216d |
| name        | corp                             |
+-------------+----------------------------------+
```

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

```
$ openstack project create private-cloud --domain corp
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | 69436408fdcb44ab9e111691f8e9216d |
| enabled     | True                             |
| id          | c50d5cf4fe2e4929b98af5abdec3fd64 |
| is_domain   | False                            |
| name        | private-cloud                    |
| parent_id   | 69436408fdcb44ab9e111691f8e9216d |
+-------------+----------------------------------+
```

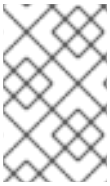
3. Create a subproject (**dev**) within the **private-cloud** parent project, while also specifying the **corp** domain:

```
$ openstack project create dev --parent private-cloud --domain corp
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | 69436408fdcb44ab9e111691f8e9216d |
| enabled     | True                             |
| id          | 11fccd8369824baa9fc87cf01023fd87 |
| is_domain   | False                            |
| name        | dev                              |
| parent_id   | c50d5cf4fe2e4929b98af5abdec3fd64 |
+-------------+----------------------------------+
```

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

```
$ openstack project create qa --parent private-cloud --domain corp
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | 69436408fdcb44ab9e111691f8e9216d |
| enabled     | True                             |
| id          | b4f1d6f59ddf413fa040f062a0234871 |
| is_domain   | False                            |
```

```
| name        | qa                               |
| parent_id   | c50d5cf4fe2e4929b98af5abdec3fd64 |
+-------------+----------------------------------+
```



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

5.2.1.2. Granting Access to Users

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 will also have **admin** access to the subprojects.

1. View the existing permissions assigned to a project:

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

2. View the existing roles:

```
$ openstack role list
+----------------------------------+---------------+
| ID                               | Name          |
+----------------------------------+---------------+
| 3a5137e4b620489791df1152ac013bfa | ResellerAdmin |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_      |
| cf4f87df933b455f957cf03b6d3784d2 | admin         |
| eef5cea6ff9549aa98cccc208c370d80 | SwiftOperator |
+----------------------------------+---------------+
```

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 the command above 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:

```
$ openstack role assignment list --effective --user user1 --user-domain corp
+----------------------------------+----------------------------------+-------+----------------------------------+------
--+-----------+
| Role                             | User                             | Group | Project                          | Domain | Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------+------
--+-----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | 10b5b34df21d485ca044433818d134be |       |
c50d5cf4fe2e4929b98af5abdec3fd64 |        | False     |
| 9fe2ff9ee4384b1894a90878d3e92bab | 10b5b34df21d485ca044433818d134be |       |
```

```
11fccd8369824baa9fc87cf01023fd87 |        | True      |
| 9fe2ff9ee4384b1894a90878d3e92bab | 10b5b34df21d485ca044433818d134be |       |
b4f1d6f59ddf413fa040f062a0234871 |        | True      |
+----------------------------------+----------------------------------+-------+----------------------------------+------
--+-----------+
```

You will see in the results that **user1** has inherited access to the **qa** and **dev** projects. In addition, because the **--inherited** flag was applied to the parent project, **user1** will also automatically get access to any subprojects that are created later.

5.2.2. Removing access

Explicit and inherited permissions must be separately removed. For example:

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:

```
$ openstack role assignment list --effective --user user1 --user-domain corp
+----------------------------------+----------------------------------+-------+----------------------------------+------
--+-----------+
| Role                             | User                             | Group | Project                          | Domain | Inherited |
+----------------------------------+----------------------------------+-------+----------------------------------+------
--+-----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | 10b5b34df21d485ca044433818d134be |       |
11fccd8369824baa9fc87cf01023fd87 |        | True      |
| 9fe2ff9ee4384b1894a90878d3e92bab | 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
```

5.2.3. Nested Quotas

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

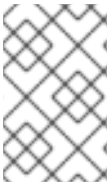
5.2.4. Reseller Overview

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:

5.2.4.1. Phase 1 of Reseller

Reseller (phase 1) is an extension of Hierarchical Multitenancy (HMT), described here: Section 5.2.1, "Hierarchical Multitenancy (HMT) in Identity Service". 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.



NOTE

Phase 1 does not allow you to create a hierarchy of domains, meaning that the subdomains are not yet available. In addition, this does not change the scope of tokens, neither does it include the hierarchy support necessary for projects other than keystone.

5.3. 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 to your instance.

5.3.1. Create a Security Group

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

5.3.2. Add a Security Group Rule

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

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*.

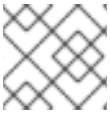
Remote

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.

5.3.3. Delete a Security Group Rule

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.

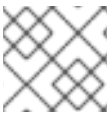


NOTE

You cannot undo the delete action.

5.3.4. Delete a Security Group

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 6. DOMAIN MANAGEMENT

Identity Service (keystone) domains are additional namespaces you can create in keystone. You would use keystone domains to partition users, groups, and projects. These separate domains can also be configured to authenticate users in different LDAP or Active Directory environments. For more information see the *Integrate with Identity Service* guide.

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.

6.1. VIEW A LIST OF DOMAINS

You can view a list of domains using **openstack domain list**. For example:

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

NOTE

If this command is not available, check you have enabled keystone v3 for your command line session.

6.2. CREATE A NEW DOMAIN

You can create a new domain using **openstack domain create**. For example:

```
$ openstack domain create TestDomain
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| enabled     | True                             |
| id          | 3abefa6f32c14db9a9703bf5ce6863e1 |
| name        | TestDomain                       |
+-------------+----------------------------------+
```

6.3. VIEW THE DETAILS OF A DOMAIN

You can view the details of a domain using **openstack domain show**. For example:

```
$ openstack domain show TestDomain
+-------------+----------------------------------+
| Field       | Value                            |
```

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

6.4. DISABLE A DOMAIN

1. You can disable a domain using **--disable**. For example:

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

2. Confirm the domain has been disabled:

```
$ openstack domain show TestDomain
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| enabled     | False                            |
| id          | 3abefa6f32c14db9a9703bf5ce6863e1 |
| name        | TestDomain                       |
+-------------+----------------------------------+
```

3. You can then re-enable the domain, if required:

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

CHAPTER 7. IDENTITY MANAGEMENT

7.1. SECURE LDAP COMMUNICATION

If you have configured the Identity service (keystone) to authenticate against or to retrieve identity information from an LDAP server, you can secure LDAP communication for the Identity service using a CA certificate.

This section outlines how to obtain the CA certificate from Active Directory, how to convert the CA certificate file into Privacy Enhanced Mail (PEM) file format, and the three methods for configuring secure LDAP communication for the Identity service. The procedure in each method must be performed depending on where and how the CA trust is configured.

7.1.1. Obtaining the CA Certificate from Active Directory

The following code shows an example of how to query Active Directory to obtain the CA certificate. The CA_NAME is the name of the certificate (you can see it in mmc.exe) and the rest of the parameters can be changed according to your setup:

```
CA_NAME="WIN2012DOM-WIN2012-CA"
AD_SUFFIX="dc=win2012dom,dc=com" LDAPURL="ldap://win2012.win2012dom.com"
ADMIN_DN="cn=Administrator,cn=Users,$AD_SUFFIX"
ADMINPASSWORD="MyPassword"

CA_CERT_DN="cn=latexmath:[$CA_NAME,cn=certification authorities,cn=public key
services,cn=services,cn=configuration,$]AD_SUFFIX"

TMP_CACERT=/tmp/cacert.`date +'%Y%m%d%H%M%S'`.$$.pem

ldapsearch -xLLL -H
latexmath:[$LDAPURL -D `echo \"$]ADMIN_DN"`-W -s base -b`echo
"$CA_CERT_DN"` objectclass=* cACertificate
```

7.1.2. Converting the CA Certificate into PEM file format

Create a file called /path/cacert.pem and include the contents of the LDAP query – that obtained the CA certificate from Active Directory, within the header and footer, as shown in the example below:

```
-----BEGIN CERTIFICATE-----
MIIDbzCCAlegAwIBAgIQQD14hh1Yz7tPFLXCkKUOszANB... -----END
CERTIFICATE-----
```

For troubleshooting, you can execute the following query to check if LDAP is working, and to ensure the PEM certificate file was created correctly.

```
LDAPTLS_CACERT=/path/cacert.pem ldapsearch -xLLL -ZZ -H $LDAPURL -s base -b ""
"objectclass=*" currenttime
```

The query should return a result similar to:

```
dn: currentTime:
20141022050611.0Z
```

You can run the following command to get a CA certificate if it was hosted by a web server.

Example

- $HOST=redhat.com
- $PORT=443

```
# echo Q | openssl s_client -connect $HOST:$PORT | sed -n -e
'/BEGIN CERTIFICATE/,/END CERTIFICATE/ p'
```

7.1.3. Methods for Configuring Secure LDAP Communication for the Identity Service

7.1.3.1. Method 1

Use this method if the CA trust is configured at the LDAP level using a PEM file. Manually specify the location of a CA certificate file. The following procedure secures LDAP communication not only for the Identity service, but for all applications that use the OpenLDAP libraries.

1. Copy the file containing your CA certificate chain in PEM format to the **/etc/openldap/certs** directory.
2. Edit **/etc/openldap/ldap.conf** and add the following directive, replacing [CA_FILE] with the location and name of the CA certificate file:

   ```
   TLS_CACERT /etc/openldap/certs/[CA_FILE]
   ```

3. Restart the *httpd* service:

   ```
   # systemctl restart httpd.service
   ```

7.1.3.2. Method 2

Use this method if the CA trust is configured at the LDAP library level using a Network Security Services (NSS) database. Use the **certutil** command to import and trust a CA certificate into the NSS certificate database used by the OpenLDAP libraries. The following procedure secures LDAP communication not only for the Identity service, but for all applications that use the OpenLDAP libraries.

1. Import and trust the certificate, replacing [CA_FILE] with the location and name of the CA certificate file:

   ```
   # certutil -d /etc/openldap/certs -A -n "My CA" -t CT,, -a -i [CA_FILE]
   # certutil -d /etc/openldap/certs -A -n "My CA" -t CT,, -a -i [CA_FILE]
   ```

2. Confirm the CA certificate was imported correctly:

   ```
   # certutil -d /etc/openldap/certs -L
   ```

   Your CA certificate is listed, and the trust attributes are set to **CT,,**.

3. Restart the *httpd* service:

   ```
   # systemctl restart httpd.service
   ```

-

7.1.3.3. Method 3

Use this method if the CA trust is configured at the Keystone level using a PEM file. The final method of securing communication between the Identity service and an LDAP server is to configure TLS for the Identity service.

However, unlike the two methods above, this method only secures LDAP communication for the Identity service and does not secure LDAP communication for other applications that use the OpenLDAP libraries.

The following procedure uses the **openstack-config** command to edit values in the **/etc/keystone/keystone.conf** file.

1. Enable TLS:

```
# openstack-config --set /etc/keystone/keystone.conf ldap use_tls True
```

2. Specify the location of the certificate, replacing [CA_FILE] with the name of the CA certificate:

```
# openstack-config --set /etc/keystone/keystone.conf ldap tls_cacertfile [CA_FILE]
```

3. Specify the client certificate checks performed on incoming TLS sessions from the LDAP server, replacing [CERT_BEHAVIOR] with one of the behaviors listed below:

   demand

   a certificate will always be requested from the LDAP server. The session will be terminated if no certificate is provided, or if the certificate provided cannot be verified against the existing certificate authorities file.

   allow

   a certificate will always be requested from the LDAP server. The session will proceed as normal even if a certificate is not provided. If a certificate is provided but it cannot be verified against the existing certificate authorities file, the certificate will be ignored and the session will proceed as normal.

   never

   a certificate will never be requested.

```
# openstack-config --set /etc/keystone/keystone.conf ldap tls_req_cert [CERT_BEHAVIOR]
```

4. Restart the *httpd* service:

```
# systemctl restart httpd.service
```

CHAPTER 8. APPLICATION CREDENTIALS

Application Credentials help you 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 only gains access to the one project and role that it needs to function, rather than all of them.

This approach allows you to consume an API with revealing your user credentials, and lets applications 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.

8.1. USE 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**. For example:

```
$ openstack project create AppCreds
```

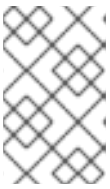
   b. Create a user called **AppCredsUser**. For example:

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

   c. Grant **AppCredsUser** access to the **_member_** role for the **AppCreds** project. For example:

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

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



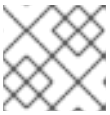
NOTE

Be sure to download the **clouds.yaml** file contents, as you will not be able to access it again once 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. For example:

```
# 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 exact 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.

```
[stack@undercloud-0 openstack]$ 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**.

8.2. INTEGRATE APPLICATION CREDENTIALS WITH APPLICATIONS

Application Credentials can be used to authenticate applications to keystone. When using Application Credentials, the **keystone_authtoken** settings use **v3applicationcredential** as the authentication type and will contain the credentials you received during the credential creation process. You will need to enter the following values:

- **application_credential_secret**: The Application Credential secret.
- **application_credential_id**: The Application Credential id.
- **application_credential_name**: (Optional) - You might use this if using 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>"
```

8.3. USE THE COMMAND LINE TO MANAGE APPLICATION CREDENTIALS

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

The **create** subcommand will create 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                                                                                   |
+--------------+-----------------------------------------------------------------------------------------+
```

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 only delegating access to a specific role. For example:

```
$ 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
```

8.4. OPERATIONAL TASKS

8.4.1. Replace an existing Application Credential

Application Credentials are bound to the user account that created them and will 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.

8.4.1.1. For configuration files

To 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's configuration file, replacing the existing credentials. This is described in Section 8.2, "Integrate Application Credentials with applications".
3. Restart the application's service to apply the change.
4. Delete the old Application Credential, if appropriate. For more information on the command line options, see Section 8.3, "Use the command line to manage Application Credentials".

8.4.1.2. For clouds.yaml files

To replace an existing Application Credential used by **clouds.yaml**:

For example, if your **clouds.yaml** contains an Application Credential called **AppCred1**, and it is due to expire:

1. Create an Application Credential called *AppCred2*.
2. Add the new **AppCred2** to the **clouds.yaml** file, while removing the **AppCred1** configuration.
3. Generate a token with **clouds.yaml** to confirm that the credentials are working as expected. See step 4 of Section 8.1, "Use Application Credentials to generate tokens" for more information.