Red Hat OpenStack Platform 16.1

Users and Identity Management Guide

Managing users and keystone authentication

Last Updated: 2023-08-28

Red Hat OpenStack Platform 16.1 Users and Identity Management Guide

Managing users and keystone authentication

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

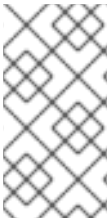
All other trademarks are the property of their respective owners.

Abstract

Manage application credentials, users, roles, projects, and quotas.

Table of Contents

PREFACE



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 .

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message . :leveloffset: +0

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.

CHAPTER 1. PREFACE

Identity service

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

Projects are organizational units containing a collection of resources. You can assign users to roles within 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 Red Hat OpenStack (RHOSP) deployment must include at least one user assigned to a role within a project. As a cloud administrator, you can:

- 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. The Identity service 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 the Identity service.

CHAPTER 2. MANAGING USERS

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.

2.1. 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**.

2.2. 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**.

2.3. ENABLING OR DISABLING A 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 into the dashboard.
- Get access to RHOSP 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**.

2.4. DELETING A USER WITH THE DASHBOARD

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

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 3. MANAGING ROLES

Red Hat OpenStack Platform (RHOSP) 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 to attach to a project.
- An administrative role to enable non-admin users to administer the environment.

NOTE

The Identity service (keystone) has also added the **reader** role that will show up in role listings. Do not use the **reader** role as it has not been integrated into other OpenStack projects, and provides inconsistent permissions across services.

You can also create custom roles specific to your environment.

3.1. UNDERSTANDING THE RED HAT OPENSTACK PLATFORM ADMIN ROLE

When you assign a user the role of **admin**, 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
```

3.2. 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
+----------------------------------+-----------------+
| ID                               | Name            |
+----------------------------------+-----------------+
| 01d92614cd224a589bdf3b171afc5488 | admin           |
| 034e4620ed3d45969dfe8992af001514 | member          |
| 0aa377a807df4149b0a8c69b9560b106 | ResellerAdmin   |
| 9369f2bf754443f199c6d6b96479b1fa | heat_stack_user |
```

```
| cfea5760d9c948e7b362abc1d06e557f | reader          |
| d5cb454559e44b47aaa8821df4e11af1 | swiftoperator   |
| ef3d3f510a474d6c860b4098ad658a29 | service         |
+----------------------------------+-----------------+
```

2. View details for a specified role:

```
$ openstack role show admin
```

Example

```
$ openstack role show admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 01d92614cd224a589bdf3b171afc5488 |
| name      | admin                            |
+-----------+----------------------------------+
```

NOTE

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

3.3. CREATING AND ASSIGNING ROLES WITH THE CLI

As an administrator, you can create and manage roles using the Identity service (keystone) client with the following set of commands. Each Red Hat OpenStack Platform deployment must include at least one project, one user, and one role, linked together.

You can assign users to more than one project. To assign users to multiple projects, create a role and assign that role to a user-project pair.

NOTE

You can use either the name or ID to specify users, roles, or projects.

Procedure

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

```
$ openstack role create <role_name>
```

2. To assign a user to a project, first find the user, role, and project names or IDs by using the following commands:
   - openstack user list
   - openstack role list
   - openstack project list

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

```
$ openstack role add <role_name> --user <user_name> --project <project_name>
```

The following example assigns the **admin** role to the **admin** user in the **demo** project:

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

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

```
$ openstack role assignment list --user <user_name> --project <project_name> --names
```

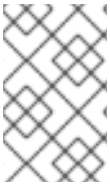
The following example verifies that the **admin** user is assigned to the **demo** project with the role of **admin**.

```
$ openstack role assignment list --user admin --project demo --names
+-------+---------------+-------+--------------+--------+--------+-----------+
| Role  | User          | Group | Project      | Domain | System | Inherited |
+-------+---------------+-------+--------------+--------+--------+-----------+
| admin | admin@Default |       | demo@Default |        |        | False     |
+-------+---------------+-------+--------------+--------+--------+-----------+
```

3.4. IMPLIED ROLES

The Identity service (keystone) enforces access control confirming that a user is assigned to a specific role. The Identity service uses implied role assignments. If you assign a user to a role explicitly, then the user can also be assigned to additional roles implicitly. You can view the default implied roles in Red Hat OpenStack Platform:

```
$ openstack implied role list
+----------------------------------+----------------+----------------------------------+------------------+
| Prior Role ID                    | Prior Role Name | Implied Role ID                  | Implied Role Name |
+----------------------------------+----------------+----------------------------------+------------------+
| 54454217f38247e5a2131c8a47138d32 | admin          | b59703369e194123b5c77dad60d11a25 | member           |
| b59703369e194123b5c77dad60d11a25 | member         | 382761de4a9c4414b6f8950f8580897c | reader           |
+----------------------------------+----------------+----------------------------------+------------------+
```



NOTE

The Identity service (keystone) has also added the **reader** role that will show up in role listings. Do not use the **reader** role as it has not been integrated into other OpenStack services, and provides inconsistent permissions across services.

The role with higher permissions imply permissions associated with the role with fewer permissions. In the default implied roles above, admin implies member, and member implies reader. With implied roles, role assignments of a user are processed cumulatively, so that the user inherits the subordinate roles.

3.4.1. Creating implied roles

If you use custom roles, you can create implied associations.

NOTE

When you create a new role, it will have the same access policies as the **member** role by default. For information on creating unique policies for custom roles, see Using policy files for access control.

Procedure

- Use the following command to specify the role that implies another role:

```
$ openstack implied role create manager --implied-role poweruser
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| implies    | ab0b966e0e5e411f8d8b0cc6c26fefd1 |
| prior_role | 880761f64bff4e4a8923efda73923b7a |
+------------+----------------------------------+
```

Verification

- List all implied roles:

```
$ openstack implied role list
+----------------------------------+----------------+----------------------------------+------------------+
| Prior Role ID                    | Prior Role Name | Implied Role ID                  | Implied Role Name
|
+----------------------------------+----------------+----------------------------------+------------------+
| 54454217f38247e5a2131c8a47138d32 | admin          |
b59703369e194123b5c77dad60d11a25 | member           |
| 880761f64bff4e4a8923efda73923b7a | manager        |
ab0b966e0e5e411f8d8b0cc6c26fefd1 | poweruser        |
| b59703369e194123b5c77dad60d11a25 | member         |
382761de4a9c4414b6f8950f8580897c | reader           |
+----------------------------------+----------------+----------------------------------+------------------+
```

If the implied association is made in error, you can undo your changes:

```
openstack implied role delete manager --implied-role poweruser
```

CHAPTER 4. MANAGING GROUPS

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

4.1. USING THE COMMAND-LINE

Create a group and assign permissions to the group. Members of the group inherit the same permissions that you assign to the group:

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

4.2. USING 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. For more information, see Using the Command-line.

4.2.1. 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**.

4.2.2. 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 5. 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. Quotas can be enforced at both the project and the project-user level. You can set or modify Compute and Block Storage quotas for new and existing projects using the dashboard. For more information, see Chapter 6, *Project management*.

5.1. VIEWING 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    |
+-----------------------------+-------+
```

5.2. UPDATING COMPUTE QUOTAS FOR A USER

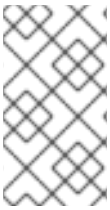
Run the following commands to update a particular quota value:

```
$ 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
```

5.3. SETTING 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
```

CHAPTER 6. PROJECT MANAGEMENT

6.1. PROJECT MANAGEMENT

As a cloud 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 with Red Hat OpenStack Platform 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.

6.1.1. 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**.

6.1.2. 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.

6.1.3. 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.

6.1.4. 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**.

6.1.5. 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.

6.2. PROJECT HIERARCHIES

6.2.1. Hierarchical Multitenancy (HMT) in the Identity Service

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

6.2.1.1. Creating the project and subprojects

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.

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

6.2.1.2. 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 also has **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            |
+----------------------------------+-----------------+
| 01d92614cd224a589bdf3b171afc5488 | admin           |
| 034e4620ed3d45969dfe8992af001514 | member          |
| 0aa377a807df4149b0a8c69b9560b106 | ResellerAdmin   |
| 9369f2bf754443f199c6d6b96479b1fa | heat_stack_user |
| 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:

-

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

6.2.2. Removing access

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:

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

6.2.3. Nested quotas

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

6.2.4. 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:

6.2.4.1. Phase 1 of reseller

Reseller (phase 1) is an extension of Hierarchical Multitenancy (HMT), described here: Section 6.2.1, "Hierarchical Multitenancy (HMT) in the 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.

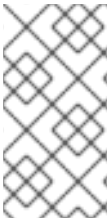
~

6.3. 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 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 .

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

6.3.1. Creating a security group

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**.

6.3.2. 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.

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

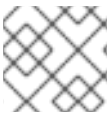
   The traffic source for this rule:

   - 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.

6.3.3. Deleting 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.

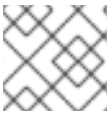


NOTE

You cannot undo the delete action.

6.3.4. Deleting 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 7. MANAGING DOMAINS

Identity Service (keystone) domains are additional namespaces that you can create in keystone. Use keystone domains to partition users, groups, and projects. You can also configure these separate domains 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.

7.1. VIEWING A LIST OF DOMAINS

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

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

7.2. CREATING A NEW DOMAIN

You can create a new domain using **openstack domain create**:

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

7.3. VIEWNG THE DETAILS OF A DOMAIN

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

```
$ openstack domain show TestDomain
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| enabled     | True                             |
```

```
| id          | 3abefa6f32c14db9a9703bf5ce6863e1 |
| name        | TestDomain                       |
+-------------+----------------------------------+
```

7.4. DISABLING A DOMAIN

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

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

2. Confirm that 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 8. IDENTITY MANAGEMENT

8.1. 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 by using a CA certificate.

You must obtain the CA certificate from Active Directory, convert the CA certificate file into Privacy Enhanced Mail (PEM) file format, and configure secure LDAP communication for the Identity service. You can perform this configuration with one of three methods, depending on where and how the CA trust is configured.

8.1.1. Obtaining the CA Certificate from Active Directory

Use the following example code to query Active Directory to obtain the CA certificate. The CA_NAME is the name of the certificate and the rest of the parameters can be changed according to your configuration:

```
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
```

8.1.2. 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:

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

For troubleshooting, you can execute the following query to check if LDAP is working, and to ensure that 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'
```

8.1.3. Methods for configuring secure LDAP communication for the Identity Service

8.1.3.1. 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 horizon container:

   ```
   # systemctl restart tripleo_horizon
   ```

8.1.3.2. 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 horizon container:

   ```
   # systemctl restart tripleo_horizon
   ```

-

8.1.3.3. 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 secures LDAP communication only 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 **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** file.

1. Enable TLS:

```
# openstack-config --set /var/lib/config-data/puppet-generated/keystone/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 /var/lib/config-data/puppet-generated/keystone/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 /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf ldap tls_req_cert [CERT_BEHAVIOR]
```

4. Restart the keystone and horizon containers:

```
# systemctl restart tripleo_keystone
# systemctl restart tripleo_horizon
```

CHAPTER 9. 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.

You can use this methodology to 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.

9.1. 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,
```

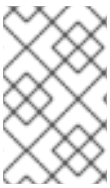
```
# just put this file in ~/.config/openstack/clouds.yaml and tools like
# python-openstackclient will just work with no further config. (You will need
# to add your password to the auth section)
# If you have more than one cloud account, add the cloud entry to the clouds
# section of your existing 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.

```
[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_P9CbhuoIt |
| 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**.

9.2. INTEGRATE 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>"
```

9.3. USE THE COMMAND LINE TO MANAGE 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
```

9.4. OPERATIONAL TASKS

9.4.1. Replace an existing Application Credential

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.

9.4.2. 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 Section 9.2, “Integrate 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 Section 9.3, “Use the command line to manage Application Credentials”.

9.4.3. For clouds.yaml files

Replace an existing Application Credential used by **clouds.yaml**:

For example, if your **clouds.yaml** contains an Application Credential called **AppCred1** that 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 9.1, "Using Application Credentials to generate tokens" for more information.