How to configure API & CLI access for mCloud

Vincent (Vinnie) Curle - 2026-01-07 - mCloud

Overview

This guide contains an overview of how you can use the OpenStack CLI client with a TOTP (Time-based One-Time Password) enabled user, or for a machine-user with the API services in your mCloud Dashboard.

CLI Access for a TOTP-Enabled User

For standard users, you can use the TOTP-enabled OpenStack client to authenticate. Download the client from the GitHub repository below and follow the instructions in the README file to complete the setup:

https://github.com/micron21/totp-capable-openstack-client-for-mcloud/

API Access for a Machine User

Prerequisite: As Machine users are unable to log into the mCloud Dashboard, you will need to contact the Micron21 Support Team to request a new machine user for mCloud API access, and supply the static public IP address the machine user will connect from and request a new machine user for mCloud API access. Once confirmed, you will be provided with a specific API-enabled user.

Method

Installing and configuring the OpenStack CLI Client

1. Using the link below, install the OpenStack CLI client on your preferred workstation
    1. https://docs.openstack.org/newton/user-guide/common/cli-install-openstack-command-line-clients.html
2. Log in to your mCloud Dashboard (https://mcloud.micron21.com)
3. At the top right, click your user, then click "OpenStack RC file" and save/copy the contents of the file to your workstation.
4. When you want to use the client, load the RC file by running "source admin-openrc.sh" or "source project-name-openrc.sh" as required and enter your user's password when you are prompted for credentials
5. You can now use the OpenStack CLI client via the API
    1. Note: Each time you create a new session (terminal, ssh, etc), you will need to re-run the source command to load the environment variables.

Issuing a custom keystone token

Sometimes, when making changes via CLI or Postman/curl and testing, you will need an authentication token.

Once you have set up your OpenStack client, you can issue a token, example below:

```
# openstack token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2024-11-14T10:02:55+0000         |
| id         | { HERE WILL BE YOUR TOKEN}       |
| project_id | cb58ecXXXXXXXXXXXXXXXXXXXX38e06  |
| user_id    | 5c9495XXXXXXXXXXXXXXXXXXXXfcd50  |
+------------+----------------------------------+
```

You can also curl directly against Keystone, with a few more arguments. More information from the official documentation: https://docs.openstack.org/keystone/latest/api_curl_examples.html

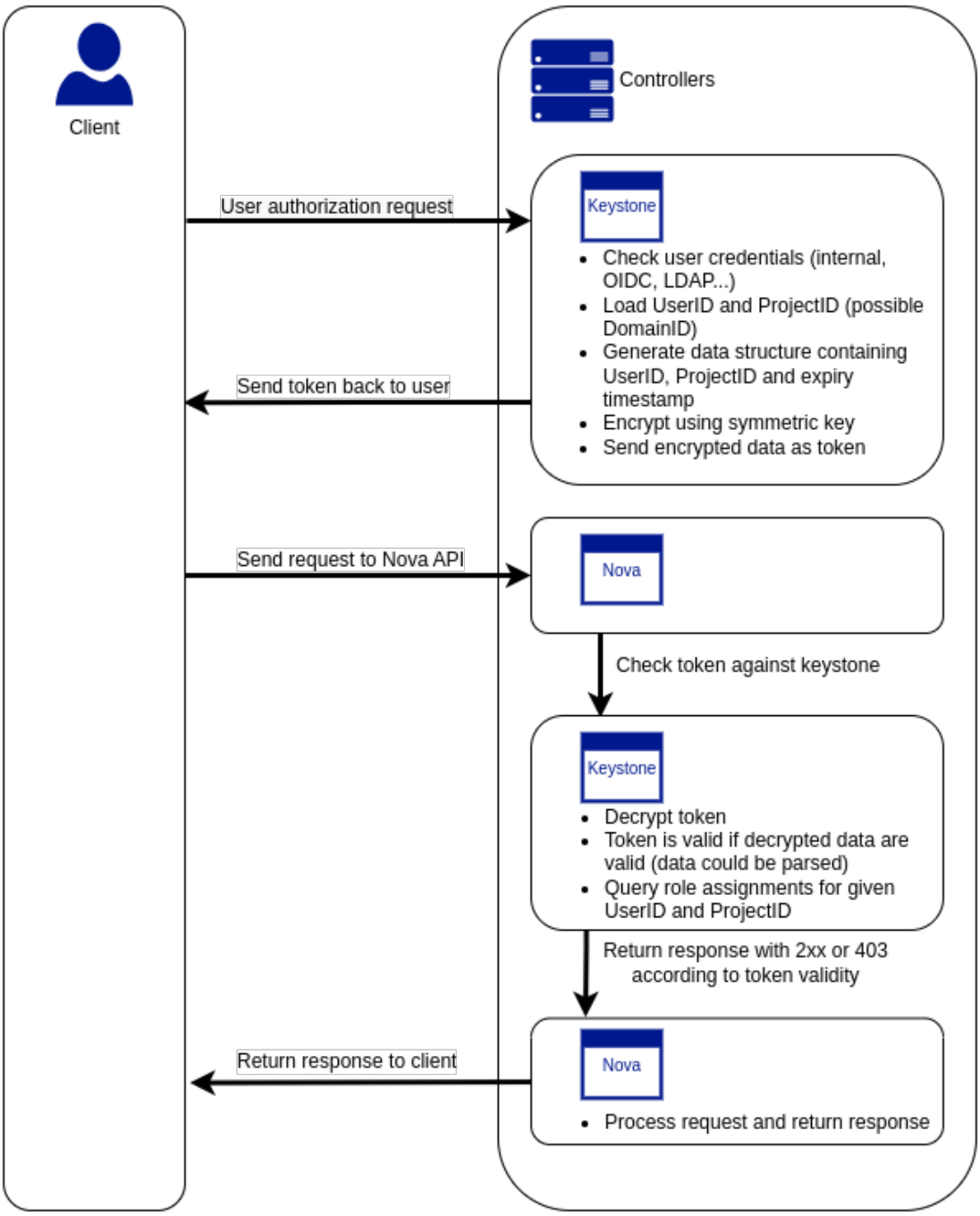
Once added, you can use that token when issuing Curl commands.

Using a custom keystone token against an API endpoint

Here's an example of using a token against the Octavia Load Balancer Service API endpoint.

```
curl -g -i -X GET https://mcloud.micron21.com:9876/v2.0/lbaas/loadbalancers -H
"Accept: application/json" -H "User-Agent: openstacksdk/3.0.0 keystoneauth1/5.6.0
python-requests/2.31.0 CPython/3.12.3" -H "X-Auth-Token: { HERE PUT YOUR TOKEN }"

e.g.

curl -g -i -X GET https://mcloud.micron21.com:9876/v2.0/lbaas/loadbalancers -H
"Accept: application/json" -H "User-Agent: openstacksdk/3.0.0 keystoneauth1/5.6.0
python-requests/2.31.0 CPython/3.12.3" -H "X-Auth-Token:
gAAAXXXXXXXXXXXXXXXXXXXXXXXatbjOj8ZRoWhsK-g416Ky7zXJfiPQqDdYje6y-Y2nPQBefh3Rfezlk3zc0-
CFwFZfUHvfuWXXXXXXXXXXXXXXXXXXXXXuvCpqHhmtvXWpnr-x0ygulB_sWXXXXXXXXXXXXXXXoS-
Q6yfZSTncXMYjxraTN0"

returns:

HTTP/2 200
date: Wed, 13 Nov 2024 23:20:56 GMT
server: Apache/2.4.52 (Ubuntu)
x-openstack-request-id: req-7998fd45-e91e-4b6e-8c4c-8759152821cd
content-type: application/json
content-length: 48

{"loadbalancers": [], "loadbalancers_links": []}
```

Keystone and API flowchart image example


Client
Controllers
User authorization request
Keystone
Check user credentials (internal, OIDC, LDAP...)
Load UserID and ProjectID (possible DomainID)
Generate data structure containing UserID, ProjectID and expiry timestamp
Encrypt using symmetric key
Send encrypted data as token
Send token back to user
Send request to Nova API
Nova
Check token against keystone
Keystone
Decrypt token
Token is valid if decrypted data are valid (data could be parsed)
Query role assignments for given UserID and ProjectID
Return response with 2xx or 403 according to token validity
Nova
Process request and return response
Return response to client