4.2 자격증명(Identity) 서비스

4.2.1 자격증명 서비스 개요

OpenStack 자격증명(Identity) 서비스는 인증, 권한 관리 및 서비스 카탈로그 제공을 위한 통합된 서비스로 Keystone이라는 프로젝트 명을 갖는다. OpenStack의 다른 서비스들에서는 공통 API형태로 자격증명 서비스가 사용되며, OpenStack에서 직접 사용자를 생성하지 않고, LDAP과 같은 기존 시스템과의 통합을 통해 사용자 정보를 제공할 수 있다.
OpenStack 자격증명 서비스와 통합된 다른 서비스에서 사용자 요청이 발생하면 자격증명 서비스를 통해 사용자 권한 부여를 수행하고, 각 서비스들은 사용자 요청을 수행하게 된다.
자격증명 서비스는 다음의 구성 요소로 이루어져 있다.

- 서버
  중앙에서 RESTful 인터페이스를 통해 인증과 권한 관리를 제공한다.
- 드라이버
  드라이버(서비스 백엔드라고도 불린다)는 중앙 서버에 통합되어, 기존에 이미 구축되어 있는 SQL 데이터베이스나 LDAP 서버 등의 OpenStack 외부의 자격증명 정보 저장소 연결에 사용된다
- 모듈
  자격증명 서비스를 사용하는 OpenStack 구성 요소의 주소 공간에서 실행되는 미들웨어는 서비스 요청을 중간에서 가로채어 사용자 자격 증명(credentials)을 추출하여 중앙의 서버로 전송하여 권한을 부여한다. 미들웨어 모듈과 OpenStack 구성 요소 간 통합에는 파이썬 웹 서버 게이트웨어 인터페이스가 사용된다.

OpenStack 자격증명 서비스 설치 시, OpenStack의 각 서비스가 등록되어야 자격증명 서비스가 이들 각 서비스 들의 설치 여부 및 네트워크 상에서 어디에 설치되어 있는지를 추적할 수 있다.

4.2.2 자격증명 서비스 설치 및 구성

- 사전 준비 사항

OpenStack 자격증명 서비스를 설치하기에 앞서 데이터베이스와 관리 토큰을 생성해야 한다.

1. 다음의 명령을 실행하여 데이터베이스를 생성한다.

a. 컨트롤러 노드에서 데이터베이스 클라이언트를 사용하여, 데이터베이스 서버(컨트롤러 노드)에 root로 접속한다. 이 때, 앞에서 지정한 DB root 암호인 *dbpassword* 로 접속한다.

```
# mysql -u root -p
Enter password:
```

b. keystone 데이터베이스를 생성한다. 이 글에서 데이터베이스 명령어는 대문자료 표기한다.

```
MariaDB [(none)]> CREATE DATABASE keystone;
```

c. keystone 데이터베이스에 다음의 명령을 사용하여 권한을 부여한다. 이 때 *keystonedpass*를 암호로 사용한다. 모든 명령어는 줄바꿈없이 한줄에 입력한다.

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost'
IDENTIFIED BY 'keystonedbpass';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED
BY 'keystonedbpass';
```

d. 데이터베이스에서 빠져나온다.

```
MariaDB [(none)]> EXIT;
```

2. 초기 설정 과정에서 관리 토큰으로 사용할 랜덤 값을 생성한다. 이 글에서는 *a7de9d998cbf87932fe2* 로 생성되었다. 이 값은 다음의 설치 과정에서 사용되니, 기억해두기 바란다.

```
# openssl rand -hex 10
a7de9d998cbf87932fe2
```

- 구성요소 설치 및 구성

이 글에서 설명하는 설정 사항은 기본 구성 파일의 해당 섹션이나 옵션을 변경하기보다는 추가할 것을 권장한다. 설정 파일에서(...) 부분은 기존 설정파일에서 유지되어야할 디폴트 옵션을 나타낸다. 이 글에서는 Apache HTTP 서버와 mod_wsgi 패키지를 사용하여, 5000번 포트와 35357 포트를 통해 자격증명 서비스 요청을 처리하며, 기본적으로 keystone 서비스는 이 포트에 대해 응답대기(listen) 상태를 유지한다. 따라서, 이 글에서는 keystone 서비스를 수동으로 활성화 한다.

1. 컨트롤러 노드에서 다음의 명령어를 실행하여 패키지를 설치한다.

```
# yum -y install openstack-keystone httpd mod_wsgi
```

2. /etc/keystone/keystone.conf 파일을 다음과 같이 수정한다. 설정 파일에서 주석 처리되어 있는 항을 그대로 두고 바로 아래에 설정 내용을 추가할 것을 권장한다.

a. [Default] 섹션에 초기 관리 토큰인 admin_token을 지정한다. 여기에는 앞에서 생성한 *a7de9d998cbf87932fe2* 를 사용한다.

```
# vi /etc/keystone/keystone.conf
[DEFAULT]
...
admin_token = a7de9d998cbf87932fe2
```

b. [database] 섹션에 데이터베이스 접속을 설정한다. keystone DB 암호인 *keystonedbpass*를 사

용한다.

```
# vi /etc/keystone/keystone.conf
[database]
...
connection = mysql+pymysql://keystone:keystonedbpass@controller/keystone
```

c. [token] 섹션에서 Fernet 토큰 프로바이더를 설정한다.

```
# vi /etc/keystone/keystone.conf
[token]
...
provider = fernet
```

4. 다음의 명령어로 자격증명 서비스 데이터베이스를 채워준다.

```
# su –s /bin/sh –c “keystone-manage db_sync” keystone
```

5. Fernet 키를 초기화 한다.

```
# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
```

- Apache HTTP 서버 설정

1. /etc/httpd/conf/httpd.conf 파일에서 ServerName을 컨트롤러 노드를 가리키도록 수정한다.

```
# vi /etc/httpd/conf/httpd.conf
ServerName controller
```

2. 다음의 내용으로 /etc/httpd/conf/wsgi-keystone.conf 을 생성한다.

```
# vi /etc/httpd/conf/wsgi-keystone.conf
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
# 위 라인은 줄 바꿈없이 한 줄로 기입한다.
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
```

```
    ErrorLog /var/log/httpd/keystone-error.log
    CustomLog /var/log/httpd/keystone-access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
# 위 라인은 줄 바꿈없이 한 줄로 기입한다.
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/httpd/keystone-error.log
    CustomLog /var/log/httpd/keystone-access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>
```

3. Apache HTTP 서비스를 시작하고, 시스템 시작시에 자동으로 실행되도록 등록한다.

```
# systemctl enable httpd.service
# systemctl start httpd.service
```

4. keystone 서비스를 시작하고, 시스템 시작시에 자동으로 실행되도록 등록한다

```
# systemctl enable openstack-keystone.service
# systemctl start openstack-keystone.service
```

4.2.3 서비스 엔티티(Entity)와 API 엔드포인트 생성

OpenStack 자격증명 서비스를 설치하기에 앞서 데이터베이스와 관리 토큰을 생성해야 한다

● 사전 준비사항

자격증명 서비스 데이터베이스에는 인증과 카탈로그 서비스를 위한 정보가 기본적으로 포함되어 있지 않으므로, 앞절의 자격증명 서비스 설치 및 구성에서 생성한 임시 인증 토큰을 사용하여 자격증명 서비스를 위한 서비스 엔티티와 API 엔드포인트를 초기화하여야 한다.

인증 토큰을 openstack 명령어에 전달하기 위해서는 --os-token 파라메터를 사용하거나 OS_TOKEN 환경변수에 설정하여야 한다. 마찬가지로 openstack 명령어에 자격증명 URL 값을 전달하기 위해서는 --os-url 파라메터를 사용하거나, OS_URL 환경변수에 설정해야 한다. 이 글에서는 명령어가 길어지는 것을 방지하기 위하여 환경변수를 사용한다.

1. 다음을 실행하여 인증토큰을 환경 변수에 설정한다. 앞절에서 생성한 인증 토큰인 *a7de9d998cbf87932fe2* 를 사용한다.

```
# export OS_TOKEN=a7de9d998cbf87932fe2
```

2. 엔드포인트 URL을 설정한다. 컨트롤러 노드의 35357번 포트를 사용하며, 버전은 3을 의미한다.

```
# export OS_URL=http://controller:35357/v3
```

3. 자격증명 API 버전을 3으로 설정한다.

```
# export OS_IDENTITY_API_VERSION=3
```

● 서비스 엔티티와 API 엔드포인트 생성

1. 자격증명 서비스는 OpenStack 환경에서 카탈로그 서비스를 관리한다. OpenStack 서비스들은 이 카탈로그를 통해 다른 서비스들이 사용가능한지를 결정한다.

다음의 명령어로 자격증명 서비스를 위한 keystone 서비스 엔티티를 생성한다. 모든 명령어는 줄바꿈 없이 한줄에 입력한다. 결과 값에서 id 는 다른 값을 가질 수 있다.

```
# openstack service create --name keystone --description “OpenStack Identity”
identity
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack Identity               |
| enabled     | True                             |
| id          | 22a9a574c424411dbd4dd6a081eec6e2 |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
```

2. 자격증명 서비스는 OpenStack 환경에서 서비스와 관련된 API 엔드포인트 카탈로그를 관리한다. OpenStack 서비스들은 이 카탈로그를 통해 다른 서비스들과 어떻게 통신할 것인지를 결정한다.
OpenStack 은 각 서비스에서 admin, internal, public 등 3 개의 API 엔드포인트를 사용한다. admin API 엔드포인트는 기본적으로 사용자와 테넌트의 수정에 사용되며 public 과 intelnal 는 작업이 허용되지 않는다. 관리 작업은 admin API 엔드포인트를 통해 이루어진다.
실제 운영 환경에서는 보안을 위해 서로 다른 유형의 사용자 별로 분리된 네트워크에서 이들 API 가 동작하도록 구성한다. 예를 들어, 고객이 클라우드를 관리할 수 있도록 public API 네트워크는 인터넷을 통해 접속이 가능하도록 구성할 수 있다. admin API 네트워크는 운영자가 조직 내부에서만 클라우드 인프라를 관리하도록 제한할 수 있다. internal API 는 OpenStack 서비스가 구동되는 호스트 서버들 사이에서만 허용되도록 구성할 수 있다. 한편, OpenStack 은 확장성을 위해 복수의 region 을 지원한다.
이 글에서는 모든 API 엔드포인트가 관리 네트워크를 사용하도록 구성하며, 기본 region 으로 RegionOne 을 사용한다.

a. public API 엔드포인트를 생성한다.

```
# openstack endpoint create --region RegionOne identity public
http://controller:5000/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 3f5ebb554b8c4345b9c0622673d8263c |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 22a9a574c424411dbd4dd6a081eec6e2 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+
```

b. internal API 엔드포인트를 생성한다.

```
# openstack endpoint create --region RegionOne identity internal
http://controller:5000/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
```

```
| id           | 0e2347e756744d6489e867280f601fd4 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 22a9a574c424411dbd4dd6a081eec6e2 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:5000/v3        |
+--------------+----------------------------------+
```

c. admin API 엔드포인트를 생성한다.

```
# openstack endpoint create --region RegionOne identity admin
http://controller:35357/v3
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 80d0a4f8f9e44de5b31e4d76c499f3cf |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 22a9a574c424411dbd4dd6a081eec6e2 |
| service_name | keystone                         |
| service_type | identity                         |
| url          | http://controller:35357/v3       |
+--------------+----------------------------------+
```

참고) OpenStack에서 추가되는 각각의 서비스는 자격증명 서비스에서 하나 이상의 서비스 엔티티와 3개의 API 엔드포인트가 필요하다.(이 절에서는 keystone 서비스 엔티티와 internal, public, admin API 엔드포인트를 생성하였다.)

● 도메인, 프로젝트, 사용자 및 역할 생성

자격증명 서비스는 OpenStack 서비스에 대한 인증 서비스를 제공한다. 인증 서비스는 도메인(domain), 프로젝트(project), 사용자(user) 및 역할(role)의 조합으로 제공되는데, 여기에서 프로젝트가 테넌트(tenant)에 해당된다. 각각에 대한 설명은 다음과 같다.

■ 도메인(domain)

도메인은 자격증명 API v3 엔티티로, OpenStack 자격증명 엔티티 관리가 수행되는 프로젝트, 그룹 및 사용자가 모두 포함된 관리 영역을 의미한다.

- 프로젝트(project)
  프로젝트는 OpenStack 에서 "소유권(ownership)"을 구분하는 기본 단위로 OpenStack 의 자원들은 특정 프로젝트에 속한다. OpenStack 자격증명에서 프로젝트는 특정 도메인에 속해야 한다.
- 사용자(user)
  OpenStack 자격증명에서 특정 도메인에 속한 각각의 API 의 소유자를 의미한다. OpenStack 컴퓨트에서 사용자는 역할이나 프로젝트 또는 두가지와 결합된다.
- 역할(role)
  특정 작업들을 수행할 수 있는 사용자의 특성을 의미하며, 권한 집합을 포함한다. 역할은 사용자에게 상속되며, 프로젝트나 도메인 사용자에게 할당된다.

Figure 10 에 도메인, 프로젝트, 사용자 및 역할에 대한 예시가 나타나 있다. GotoCloud KR 도메인에는 Development 와 Management 라는 프로젝트가 포함되며, Development 프로젝트에는 사용자 Tom, Jane 및 Testing VM 과 Development VM 이라는 자원이 속해있다. 그리고, Management 프로젝트에는 Manager VM 이 속해있다. 사용자 Bob 은 GotoCloud KR 도메인에는 속해 있으나, 프로젝트에는 속해 있지 않다. Tester 역할은 Tom 에게 할당되어 있으며, Testing VM 에 접속 가능하다. Developer 역할은 Tom, Jane 및 Bob 에게 할당되어 있으며, Manager VM 에 접속할 수 있다.

이 때, Tom 과 Jane 은 Development 프로젝트 내의 자원에 대해서만 접속이 가능하나, Bob 은 Development 프로젝트와 Management 프로젝트에 속해있는 역할을 할당 받았으므로, Development VM 과 Management VM 에 접속이 가능하게 된다.

GotoCloud US 도메인에는 Marketing 프로젝트와 Alice 사용자 및 Marketing VM 이 속해 있으며, Marketer 역할은 Alice 에게 할당되어 있다. Alice 는 Marketing 프로젝트 자원만 접속할 수 있다.

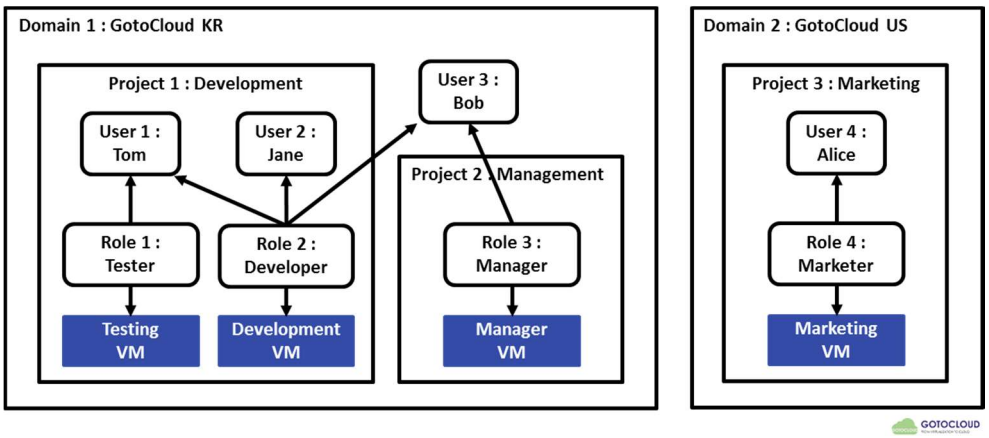

Figure 10 Domain, Project, User and Role

1. 다음의 명령어를 사용하여 도메인(domain)을 생성한다. 이 글에서는 default 도메인을 사용한다.

```
# openstack domain create --description "Default Domain" default
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Default Domain                   |
| enabled     | True                             |
| id          | dfff203e4310474ab2ff3f8de8c63a98 |
| name        | default                          |
+-------------+----------------------------------+
```

2. OpenStack 관리를 위한 관리 프로젝트, 사용자 및 역할을 생성한다.

a. admin 프로젝트를 생성한다.

```
# openstack project create --domain default --description "Admin Project" admin
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Admin Project                    |
| domain_id   | dfff203e4310474ab2ff3f8de8c63a98 |
| enabled     | True                             |
| id          | 0fdf2c3a7a5c466e8ffd4fd73332ecc0 |
| is_domain   | False                            |
| name        | admin                            |
| parent_id   | dfff203e4310474ab2ff3f8de8c63a98 |
+-------------+----------------------------------+
```

b. admin 사용자를 생성한다. admin 사용자 생성시에 admin 암호는 앞에서 설정한 *gotocloud_admin* 을 사용한다.

```
# openstack user create --domain default --password-prompt admin
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | dfff203e4310474ab2ff3f8de8c63a98 |
```

```
| enabled   | True                             |
| id        | 491ad98dcac34cdeb764bea3c17e23b3 |
| name      | admin                            |
+-----------+----------------------------------+
```

c. admin 역할을 생성한다.

```
# openstack role create admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | a7ab6e681a8f45399a80d9d134a5c541 |
| name      | admin                            |
+-----------+----------------------------------+
```

d. admin 역할을 admin 프로젝트와 사용자에게 할당한다.

```
# openstack role add --project admin --user admin admin
```

3. 다음으로 OpenStack 환경에 추가하는 각 서비스에 대한 고유 사용자를 갖는 service 프로젝트를 생성한다.

```
# openstack project create --domain default --description “Service Project” service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | dfff203e4310474ab2ff3f8de8c63a98 |
| enabled     | True                             |
| id          | 80a9d3bacfaf4ca2a9cd3cfda06fc0b7 |
| is_domain   | False                            |
| name        | service                          |
| parent_id   | dfff203e4310474ab2ff3f8de8c63a98 |
+-------------+----------------------------------+
```

4. 관리 작업이 아닌 일반 작업은 관리 권한이 없는 프로젝트와 사용자로 수행되어야 한다. 이를 위해 demo 프로젝트와 사용자를 생성한다.

a. demo 프로젝트를 생성한다.

```
# openstack project create --domain default --description “Demo Project” demo
+-------------+----------------------------------+
```

```
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | dfff203e4310474ab2ff3f8de8c63a98 |
| enabled     | True                             |
| id          | aa0545a0308946d6bff4cacf9004e0da |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | dfff203e4310474ab2ff3f8de8c63a98 |
+-------------+----------------------------------+
```

b. demo 사용자를 생성한다. demo 사용자 생성시에 demo 암호는 앞에서 설정한 *gotocloud_demo* 를 사용한다.

```
# openstack user create --domain default --password-prompt demo
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | dfff203e4310474ab2ff3f8de8c63a98 |
| enabled   | True                             |
| id        | 1a677d0d22e34aa088675989b6788bce |
| name      | demo                             |
+-----------+----------------------------------+
```

c. user 역할을 생성한다.

```
# openstack role create user
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | cd1aed1c94eb43c2b9a4da5d4662988a |
| name      | user                             |
+-----------+----------------------------------+
```

d. user 역할을 demo 프로젝트와 사용자에 할당한다.

```
# openstack role add --project demo --user demo user
```

새로운 프로젝트와 사용자 생성은 위의 과정을 반복한다.(이 과정은 대시보드 설치 후에 대시보드에서 수행할 수 있다.)

4.2.4 동작 확인

다른 서비스를 설치하기에 앞서 자격증명 서비스의 동작을 검증한다. 이 과정은 컨트롤러 노드에서 수행한다.

1. 보안상 이유로, /etc/keystone/keystone-paste.ini 파일을 열어서 [pipeline:public_api], [pipeline:admin_api] 및 [pipeline:api_v3] 섹션을 찾아서 아래와 같이 admin_token_auth 를 삭제한다.

```
# vi /etc/keystone/keystone-paste.ini
...
[pipeline:public_api]
pipeline = cors sizelimit url_normalize request_id ~~admin_token_auth~~ ...
...
[pipeline:admin_api]
pipeline = cors sizelimit url_normalize request_id ~~admin_token_auth~~ ...
...
[pipeline:api_v3]
pipeline = cors sizelimit url_normalize request_id ~~admin_token_auth~~ ...
```

2. 임시로 설정한 OS_TOKEN 과 OS_URL 환경 변수를 해제한다.

```
# unset OS_TOKEN OS_URL
```

3. 앞에서 생성한 admin 사용자로 인증 토큰을 요청한다. 이 때 admin 사용자 암호(이글에서는 *gotocloud_admin*)를 입력한다. admin 사용자는 자격증명 서비스 API 로 admin API 엔드포인트(포트 35537)를 사용하여 접속한다.

```
# openstack --os-auth-url http://controller:35357/v3 --os-project-domain-name
default --os-user-domain-name default --os-project-name admin --os-username admin
token issue
Password:
+------------+----------------------------------------------------------------
| Field      | Value
+------------+----------------------------------------------------------------
| expires    | 2016-08-10T04:52:33.662152Z
| id         | gAAAAABXqqUCiH6t0SUkKp02yf4KDz9TM1rnA347L6XL0QU6_QrghJiqeUTfop0jL047
```

```
|            | 4bS2ZKzmfN_F6YWxJylOcHY_WeEPasJTy9HycVa8pmjHdId1oCkzihpKTsnMcuHXJyp7
| project_id | 0fdf2c3a7a5c466e8ffd4fd73332ecc0
| user_id    | 491ad98dcac34cdeb764bea3c17e23b3
+------------+----------------------------------------------------------------
```

4. 앞에서 생성한 demo 사용자로 인증 토큰을 요청한다. 이 때 demo 사용자 암호(이글에서는 *gotocloud_demo*)를 입력한다. 일반 사용자는 자격증명 서비스 API로 public API 엔드포인트(포트 5000)를 사용하여 접속한다.

```
# openstack --os-auth-url http://controller:5000/v3 --os-project-domain-name
default --os-user-domain-name default --os-project-name demo --os-username demo
token issue
Password:
+------------+----------------------------------------------------------------
| Field      | Value
+------------+----------------------------------------------------------------
| expires    | 2016-08-10T04:55:57.227628Z
| id         | gAAAAABXqqXNsolZ1ENqZWl7jgOcYCb3hT3SeQFXieRLD2a7v3YxmLbksWTS8pHezopj
|            | 58tTq6kka96htTThpUIE8AEpuGDASiarb78zWwWc4BwAtYbiV0fO_gXCscquf4cTIfaX
| project_id | aa0545a0308946d6bff4cacf9004e0da
| user_id    | 1a677d0d22e34aa088675989b6788bce
+------------+----------------------------------------------------------------
```

4.2.5 OpenStack 클라이언트 환경 스크립트

앞에서는 openstack 클라이언트를 사용하여 자격증명 서비스와 상호작용하고자 할 때, 환경 변수와 명령어 옵션을 사용하였다. 클라이언트 작업의 효율을 위해 OpenStack에서는 OpenRC 파일로 알려져 있는 간단한 클라이언트 환경 스크립트를 지원한다. 이 스크립트에는 모든 클라이언트에서 사용 가능한 공통적인 옵션과 함께 고유한 옵션이 포함되어 있다. 이 스크립트에 대한 더 자세한 정보는 OpenStack End User Guide를 참조하기 바란다.

- OpenStack 클라이언트 환경 스크립트 생성

admin과 demo 프로젝트와 사용자 용 클라이언트 환경 스크립트를 생성한다. 이 가이드에서는 클라이언트 작업 시 필요한 인증을 위해 이 스크립트를 계속 사용할 예정이다.

1. admin-openrc 파일을 다음과 같이 생성한다.(이 파일은 /root/ 아래에 생성한다) OS_PASSWORD에 admin 암호인 *gotocloud_admin*을 사용한다.

```
# vi /root/admin-openrc
```

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=gotocloud_admin
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

2. demo-openrc 파일을 다음과 같이 생성한다.(이 파일은 /root/ 아래에 생성한다) OS_PASSWORD에 demo 암호인 *gotocloud_demo*를 사용한다.

```
# vi /root/demo-openrc
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=gotocloud_demo
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

- 스크립트 사용

특정 프로젝트와 사용자로 클라이언트를 실행하기 전에, 이 스크립트를 다음과 같이 로드한다.

1. admin-openrc 파일을 로드하여 자격증명 서비스의 위치와 admin 프로젝트 및 사용자의 암호를 포함하는 환경변수를 설정한다.

```
# cd /root
# source admin-openrc
```

2. 인증 토큰을 요청한다.

```
# openstack token issue
+------------+----------------------------------------------------------------------
| Field      | Value
+------------+----------------------------------------------------------------------
| expires    | 2016-08-10T05:13:44.905780Z
| id         | gAAAAABXqqn420AEpG3I5fyaBRFc016gPWi8wyQweOZkUimjKPYTNvnVu1lreHtEbtMg
```

```
|            | viqSkeYMvr70Snbt0bVgZk59lIDtHW8r3orOs98HiAHCfteNR9iO05SgpbbFvK6vxcqM
| project_id | 0fdf2c3a7a5c466e8ffd4fd73332ecc0
| user_id    | 491ad98dcac34cdeb764bea3c17e23b3
+------------+----------------------------------------------------------------
```