ISBN :

RealLab 관리자 매뉴얼

일 자: 2014년 11월 30일
부 서: 첨단연구망센터/첨단연구망개발팀
제출자: 권 윤 주

한국과학기술정보연구원
305-806 대전광역시 유성구 어은동 52번지
TEL (042)869-0676 / FAX (042)869-0679
www.kisti.re.kr

RealLab 관리자 매뉴얼

작성자: 권윤주

첨단연구망개발팀, 한국과학기술정보연구원
yulli@kisti.re.kr
최종수정일: 2014년 11월 30일

Abstract

RealLab은 KREONET/GLORIAD를 이용하는 응용연구자에게 네트워크와 더불어 좀 더 편리하게 이용할 수 있는 연구환경제공을 해주고자하는 취지로 제안되었다. 이에 RealLab은 컴퓨팅자원, 스토리지 자원등의 하드웨어적 자원과 다양한 운영체제 등의 소프트웨어적인 자원을 네트워크 자원과 더불어 통합적으로 제공할 수 있는 플랫폼이다. 이를 위해 RealLab은 OpenStack을 기반으로 구축되었다. OpenStack은 구축하고자하는 클라우드 요구사항에 따라 함께 운영될 수 있는 다수 개의 프로젝트로 이루어졌으므로, RealLab이 필요로하는 기능에 따라 취사선택하여 설치가 가능하다. 본 문서에서는 Ubuntu 12.04(LTS) 기반으로 하여 RealLab노드들에 OpenStack 컴포넌트를 설치하는 내용을 기술하였다. Devstack을 통하여 OpenStack 전체 컴포넌트들에 대한 일괄설치도 가능하지만 이 설치문서에서는 각각의 컴포넌트 별 설치와 설정내용에 대해서 기술할 것이다.

Topics

1. 선행 작업

1) 하드웨어 요구사항

OpenStack 컴포넌트들은 기본적인 하드웨어 상에서 운영된다고 가정한다. 최소수준으로 고려될 수 있는 하드웨어 설정은 다음과 같다.

Server	Recommended Hardware	Notes
Cloud Controller node (runs network, volume, API, scheduler and image services)	Processor: 64-bit x86 Memory : 12GB RAM Disk : 30GB(SATA, SAS or SSD) Volume : Compute 노드에 연결될 볼륨들을 위한 2TB(SATA)를 가진 2개의 디스크 Network : one 1Gbps	2개의 네트워크 인터페이스가 추천되지만 강제사항은 아니다. 12GB RAM에 4-core 서버정도면 cloud controller로 사용하기에 충분하다.
Compute nodes (runs virtual instances)	Processor : 64-bit x86 Memory : 32GB RAM	2GB RAM으로 하나의 m1.small instance 또는 3개의 m1.tiny 인스턴스를 메모리 스와핑없이 운영할 수 있다. 따라서 2GB 메모리는 Compute node의 최소 테스트환경이다.

2) RealLab 구성 환경

RealLab은 OpenStack 기반의 고성능 클라우드 플랫폼이다. RealLab의 구성은 1대의 Controller 노드와 다수의 Compute 노드, 그리고 스토리지 노드 1대로 되어있다. 이를 위해 다음과 같은 환경으로 RealLab 노드들은 설치된다.

- 운영체제 : Ubuntu 12.04 LTS
- 사용할 OpenStack 버전 : OpenStack Grizzly
- Controller에 사용될 OpenStack 컴포넌트 : keystone, glance, nova, cinder
- Compute에 사용될 OpenStack 컴포넌트 : nova

3) Grizzly Package

OpenStack은 6개월에 한 번씩 버전을 업그레이드하기 때문에 현재 상태에서 apt-get 으로 패키지들을 설치하게 되면 현재 배포되고 있는 버전으로 설치되게 된다. 현재 RealLab은 OpenStack Grizzly버전으로 구성되어 있기 때문에 OpenStack Grizzly버전으로 전체 시스템을 구성하기 위해서 다음과 같은 선작업들을 필요로한다.

```
# apt-get install ubuntu-cloud-keyring
```

/etc/apt/source.list.d/cloud-archive.list 를 편집한다.

```
# in /etc/apt/source.list.d/cloud-archive.list
deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/grizzly main
```

설치되어 있는 패키지들을 업그레이드한다.

```
# sudo apt-get update && apt-get upgrade
```

4) 공통 소프트웨어 설치

OpenStack 컴포넌트 공통으로 필요로 하는 소프트웨어를 설치한다.

- NTP (Network Time Protocol)
  각각의 서비스들이 다수개의 시스템에 설치되어 있으며 시간을 동기화시키기 위하여 NTP 서비스 설치가 필요하다.

  ```
  $ sudo    apt-get install -y ntp
  ```

  ntp.conf를 하고 서비스를 재시작함으로써 Controller노드 상의 NTP 서버를 셋업한다.

  ```
  # sed -i 's/server ntp.ubuntu.com\nserver 127.127.1.0\nfudge 127.127.1.0 stratum 10/g' /etc/ntp.conf
  ```

  이후 Controller 노드와 Compute 노드 사이에 시간동기화를 위하여 compute 노드에 NTP client를 셋업한다.

- Database, MySQL
  OpenStack에서 자료 저장을 위하여 DB를 많이 사용하고 있는데, 이를 위하여 하나의 DBMS를 설치하여야 한다. 이 매뉴얼에서는 주로 많이 쓰이고 있는 MySQL로 사용하는 것으로 가정한다. 이에 MySQL 서버 설치 및 설정이 필요하다. 다음과 같이 root계정으로 다음의 패키지들을 설치한다.

  ```
  #apt-get install python-mysqldb mysql-server
  ```

  그리고 localhost(127.0.0.1)로부터 any(0.0.0.0)까지로 bind_address를 바꾸기 위해 "sed"를 이용하여 /etc/mysql/my.cnf를 수정한다. 그리고 mysql 서비스를 재시작한다.

```
# sed -i 's/127.0.0.1/0.0.0.0/g' /etc/myql/my.cnf
# service mysql restart
```

- Message Queue, RabbitMQ

OpenStack을 구성하는 다양한 컴포넌트들간 상호 통신을 하기 위하여 메시지 큐(Message Queue)방식을 사용하고 있기 때문에 메시지 큐 서버 설치가 필요하다. 일반적으로 Qpid 또는 RabbitMQ를 이용하지만 ZeroMQ도 가능하다. 이 문서에서는 RabbitMQ 기반의 설치를 기술한다. 이를 위해 다음과 같이 메시지 큐 서버를 설치한다.

```
$ sudo apt-get install rabbitmq-server
```

2. OpenStack Terminology

1) 버전 네임(Version Name)

각각의 OpenStack 릴리즈는 알파벳 순서대로 증가하는 형태의 이름을 가지고 있다. 그리고 각 릴리즈들에 상응하는 버전 번호도 가지고 있다.(Cactus Release 이후부터 OpenStack은 6개월마다 릴리즈 스케줄을 채용했다.) 별도로, OpenStack 컴포넌트 중 Object Storage인 Swift는 릴리즈 주기 및 릴리즈 스케줄을 따로 관리하고 있다.

표 1 OpenStack version names

Release name	Release date	OpenStack version number for Block Storage, Compute, Identity, Image, and Networking	OpenStack Object Storage version number
Juno	2014.10	2014.2	2.1.0
Icehouse	2014.04	2014.1	1.13.0
Havana	2013.10	2013.3	1.9.1
Grizzly	2013.04	2013.1	1.7.6
Folsom	2012.10	2012.2	1.7.2
Essex	2012.04	2012.1	1.4.8
Diablo	2011.10	2011.3	1.4.3
Cactus	2011.04	2011.2	1.3.0
Bexar	2011.03	2011.1	1.2.0
Austin	2010.10	0.9.0	1.0.0

2) 코드 네임(Code Name)

OpenStack 서비스는 각각 코드 네임을 가지고 있다. 예를 들면 Image Service의 코드 네임은 "Glance"이다. 주요 서비스 및 코드 네임 리스트는 다음과 같다.

표 2 OpenStack 서비스 별 코드 네임

Service name	Code name
Identity	Keystone
Compute	Nova
Image	Glance
Dashboard	Horizon
Object Storage	Swift
Volumes	Cinder
Networking	Neutron

3. OpenStack Components

OpenStack은 클라우드 운영체제로써의 주요 기능이 각각 독립된 하부 프로젝트로 나뉘어 개발되고 있다. 초기에는 컴퓨팅 서비스인 Nova, 오브젝트 저장장치 서비스인 Swift, 이미지 관리 서비스인 Glance의 세 가지 하부 프로젝트로 시작되었으나, 배포판 버전이 업데이트 되면서 새로운 프로젝트가 지속적으로 추가 발표되어 점차 그 영역을 확장해 나가고 있다. 일례로 네트워킹 서비스와 볼륨 저장장치 서비스와 같이 Nova에서 함께 제공되던 서비스가 별도의 프로젝트로 분리되거나 인증 서비스 또는 사용자 인터페이스와 같이 독립된 프로젝트가 새롭게 추가되기도 한다.

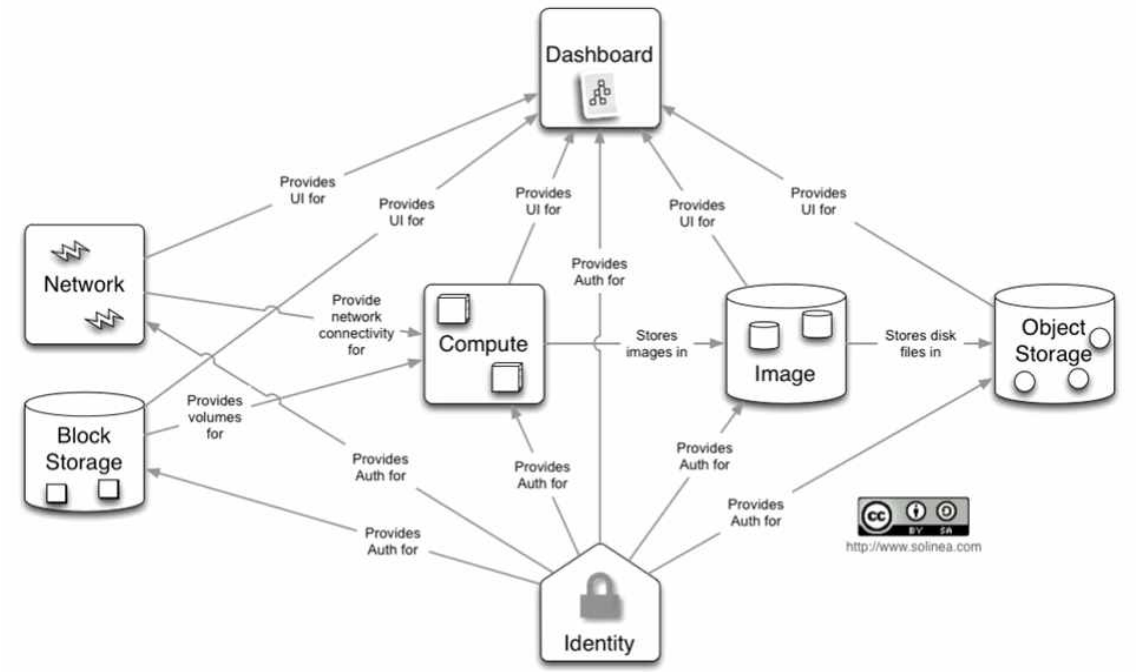

그림 1 OpenStack 구조

위의 그림은 Grizzly 배포판을 기준으로 OpenStack의 공식 하부 프로젝트간의 상호 관계를 도식화 한 것이다. OpenStack의 여러 하부 프로젝트는 기본적으로 가상머신을 생성하고 제어하는 Nova 프로젝트를 중심으로 상호 연계되어 있다. 가상머신에 대한 네트워킹 서비스는 Neutron 프로젝트를, 가상머신에서 사용하는 추가적인 볼륨 저장장치는 Cinder 프로젝트를 이용하여 서비스를 제공한다. 가상머신 이미지는 Glance 프로젝트를 통해 관리되고, Swift 프로젝트를 통해 저장할 수 있다. 또한 기본적으로 사용자가 OpenStack 서비스를 이용하기 위한 인터페이스인 dashboard는 Horizon 프로젝트로써, 모든 하부 프로젝트의 인터페이스가 Horizon를 통해 사용자에게 제공되며, 서비스를 이용하고자 하는 사용자를 인증하고 권한을 검증하는 서비스는 Keystone 프로젝트를 통해 제공한다.

1) 컴퓨팅 서비스 : Nova

Nova는 IaaS 시스템의 중심이 되는 구성 요소이다. 언어는 Python을 사용하며, 병렬 프로그래밍을 위한 Eventlet, AMQP 통신을 위한 Kombu, 데이터베이스 접근을 위한 SQL Alchemy와 같은 외부 라이브러리를 사용한다. Nova는 전용 하드웨어나 특별한 소프트웨어 요구 조건 없이 일반적인 하드웨어에서 사용 가능하며, 기존 시스템과 써드 파티 기술과 통합될 수 있도록 설계되어있다. 자동적으로 컴퓨터 자원의 풀을 가동하여 관리할 수 있으며, 베어메탈과 HPC (High-Performance Computing) 등의 여러 가상화 기술을 이용할 수 있도록 되어있다. KVM, XenServer, Hyper-V, 리눅스의 LXC 등의 하이퍼바이저를 사용할 수 있다. OpenStack의 ARM 위에서 여러 종류의 하이퍼바이저를 동작시킬 수 있다.

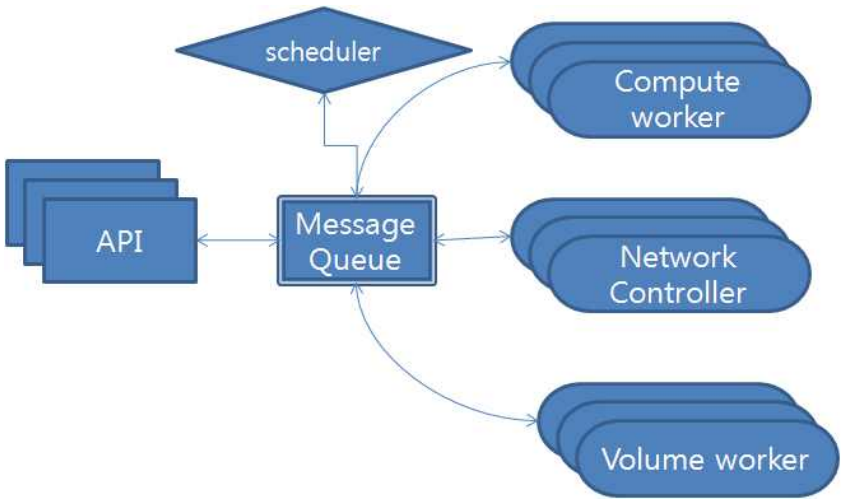

그림 2 nova 구조

nova는 위의 그림에서 보는 것처럼 구성되어 있으며 각각 요소들의 역할은 다음과 같다.

- API : public facing interface(compute API call들에 대한 수용)
- Message Queue: nova 컴포넌트간 메시지 전달을 위한 허브역할, (ex. RabbitMQ, ZeroMQ)
- Scheduler: MQ로부터의 가상머신할당 요청에 대하여, 언제 가상머신을 running 시켜야 할지 결정
- Compute Worker: 가상 머신을 호스팅하며, 메시지큐에서 명령을 받을 때마다 하이퍼바이저와 가상머신을 제어
- Volume: 영구적인 볼륨 관리

2) 오브젝트 저장장치 서비스 : Swift

Swift는 확장성과 용장성이 있는 오브젝트 스토리지 시스템이다. 오브젝트와 파일들은 데이터 센터를 중심으로 널리 퍼져있는 서버의 디스크 드라이브에 적재된다. OpenStack에서 제공하는 소프트웨어는 클러스터 간의 데이터 복제와 무결성을 보장

한다. 스토리지 클러스터는 새로운 서버들을 추가하면서 확장된다. 만약 서버나 하드 드라이브에서 실패가 일어나면, OpenStack은 그 내용을 활성 중인 다른 노드로부터 클러스터 안의 새로운 곳으로 복제한다. OpenStack은 소프트웨어 로직을 사용하여 디바이스들로의 데이터 복제나 분배를 보장할 수 있기 때문에 비싸지 않은 일반적인 하드 드라이브와 서버들을 사용할 수 있다

3) 볼륨 저장장치 서비스 : Cinder

블록 스토리지 시스템인 Cinder는 OpenStack의 인스턴스가 블록 레벨 스토리지 디바이스를 안정적으로 이용할 수 있도록 해준다. 블록 스토리지 시스템은 블록 디바이스를 서버에 만들고, 마운트하고 마운트를 해제하는 것을 관리한다. 블록 스토리지의 볼륨은 OpenStack의 Compute로 완전히 통합되며, Horizon을 통해 클라우드 사용자가 필요에 따라 관리할 수 있도록 해준다.

블록 스토리지는 데이터베이스 스토리지, 확장 가능한 파일 시스템, 로우 블록 레벨 스토리지에 접근할 수 있는 서버와 같이 성능에 민감한 시나리오에 적합하다. 스냅샷은 블록 스토리지 볼륨에 저장된 데이터를 백업하기 위한 강력한 기능을 제공한다. 스냅샷은 새로운 블록 스토리지 볼륨을 생성하거나 복구할 수 있다.

4) 네트워킹 서비스 : Neutron

Neutron(이전의 Quantum)은 네트워크와 IP 주소들을 관리하는 시스템이다. 다른 클라우드 운영체제와 마찬가지로, 네트워크 시스템을 통해서 관리자와 사용자가 데이터센터에 접근할 수 있다. OpenStack의 Neutron은 병목 현상이 일어나지 않게 해주며, 사용자가 네트워크 설정을 통해 셀프 서비스를 할 수 있도록 해준다.

그리고 Neutron은 여러 어플리케이션과 사용자 그룹들에게 네트워킹 모델을 제공한다. 표준 모델들은 플랫 네트워크나 VLAN을 쓴다. OpenStack 네트워킹은 전용 IP나 DHCP를 두어 IP 주소를 관리한다.

Neutron을 통해 사용자들은 스스로 네트워크를 생성하고, 트래픽 조절을 할 수 있으며, 서버와 디바이스를 하나 이상의 네트워크에 연결시킬 수 있다. 관리자들은 대용량과 여러 tenancy들을 지원 할 수 있는 OpenFlow 같은 Software-defined networking (SDN) 기술을 이용할 수 있다. OpenStack 네트워킹은 프레임워크를 확장하여 intrusion extension systems (IDS), 로드 밸런싱, firewalls, private networks(VPN) 등의 네트워크 서비스를 추가로 이용할 수 있다.

5) 사용자 인터페이스 : Horizon

Horizon은 관리자와 사용자가 클라우드 자원에 접근할 수 있는 그래픽 인터페이스를 제공한다. 이 인터페이스를 통해 추가의 관리 툴과 같은 써드 파티 프로덕트와 서비스를 제공받을 수 있다. 개발자들은 OpenStack API나 EC2 API를 이용하여 자동적으

로 접속하고 자원을 관리하는 툴들을 만들 수 있다.

6) 인증 서비스 : Keystone
Keystone은 OpenStack의 사용자 인증 시스템으로써 LDAP와 같은 백엔드 디렉토리 서비스들과 통합될 수 있다. Keystone은 OAuth2.0 형태의 사용자 인증을 제공한다.

7) 이미지 서비스 : Glance
OpenStack의 Glance는 디스크와 서버의 이미지를 탐색하고, 등록하고, 전송하는 제공한다. 저장된 이미지는 템플릿으로 사용될 수 있다. 개수에 제한받지 않고 백업을 저장하고 카탈로그할 수 있다. Glance는 OpenStack 오브젝스 스토리지를 포함하여 디스크와 서버의 이미지를 다양한 백엔드에 저장할 수 있다. 이 이미지 서비스 API가 제공하는 표준 REST 인터페이스는 디스크 이미지에 대한 쿼리 정보를 포함하고 있어 클라이언트는 새로운 서버로 이미지를 스트리밍할 수 있다.

8) 기타 프로젝트
현재까지 공식 배포판에 포함된 상기 프로젝트 외에도 다양한 실험적 프로젝트가 진행 중에 있다. 일례로 과금을 위한 사용자의 서비스 사용 정보를 계측하고 모니터링하기 위한 Ceilometer 프로젝트와, 템플릿 기반으로 클라우드 서비스를 제공하기 위해 필요한 다양한 자원을 조율(orchestration)하여 제공하는 Heat 프로젝트를 꼽을 수 있다.

4. Controller 설치 및 설정

1) Keystone(인증서비스)

Keystone은 인증서(Credential)와 토큰(Token)을 이용하여 OpenStack의 전체 컴포넌트들에 대한 인증을 수행해주는 컴포넌트이다. 여기서 인증서는 사용자가 그들이 누구인지 증명할 수 있도록 표현하는 데이터로서, 예를 들면, username/password, username/API key, 자신과 사진이 담긴 운전면허증 등을 의미하고 토큰은 OpenStack의 각 리소스에 접근하는 데 사용되는 데이터로서, Keystone을 통해 발급받는다. 각 토큰은 그 토큰을 가지고 어떤 자원에 접근할 수 있는 지 그 범위를 기술해준다. 토큰은 정해진 기간동안 유효하다. 이러한 형태로 OpenStack의 인증서비스를 수행하며 OAuth2.0을 따른다.

가. 기본 개념
Keystone 서비스는 두 가지 주된 기능을 제공한다.
- User Management : 사용자와 사용자가 허가받고 행하는 것들을 파악
- Service Catalog : 가능한 서비스와 그 서비스들의 API 엔드포인트 목록화

User Management는 다음의 3가지 중요한 개념을 가지고 있다.

- User : OpenStack 클라우드 서비스를 사용하고 있는 사람, 시스템 또는 서비스를 대표하는 개념
- Tenant : 자원을 그룹핑하거나 고립시킬 때 사용하는 하나의 컨테이너. 서비스 운영자에 따라 tenant는 고객, 계정, 기관 또는 프로젝트와 매핑될 수 있다.
- Roles : role은 권리(right)와 권한을 하나의 집합으로 포함하는 개념.

user는 실제 사용자를 대표하는 것으로 사용자이름, 패스워드, 이메일과 같은 정보와 연계되어 있다. 다음의 예는 “alice”라는 이름의 사용자를 생성하는 것이다.

```
$ keystone user-create --name=alice --pass=mypassword123 --email=alice@example.com
```

tenant는 프로젝트, 그룹 또는 단체로서 생각할 수 있다. OpenStack 서비스는 tenant 단위로 제공될 수 있다. Tenant를 생성하는 방법은 다음과 같다.

```
$ keystone tenant-create --name=acme
```

Role은 주어진 tenant에 user가 수행할 수 있는, 또는 허가될 수 있는 행위를 정의한다.

```
$ keystone role-create --name=compute-user
```

한 user는 다른 tenant에서 다른 role을 할당받을 수도 있다. 그리고 한 user는 하나의 tenant에서 다수개의 role을 할당받을 수도 있다. 기본적으로 Compute(Nova), Identity(Keystone), Image(Glance)서비스에 있는 policy.json은 단지 "admin" role에 대해서만 기술되어 있다. 따라서 admin role이 요구되지 않는 모든 오퍼레이션에 대해서 한 tenant내의 그 외의 role을 가진 user에 의해서 접근될 수 있다.

cf) /etc/[SERVICE_CODENAME]/policy.json은 user가 주어진 서비스에 어떤 행위를 할 수 있는 지 기술할 수 있다. 예를 들면, /etc/nova/policy.json은 Compute서비스에 대한 접근 정책을 정의하고 있다. 오페레이션을 실행하는 데 있어서 user를 제한하고 싶다면, 각 서비스의 policy.json을 수정하면 된다.

Keystone에서, user에 대한 토큰은 user가 책임질 수 있는 role 리스트를 포함한다. 해당 user에 의해 불려지는 서비스들은 user가 가지고 있는 role의 집합을 해석하는 방법과 각 role이 접급할 수 있는 오퍼레이션 또는 자원들을 결정한다.

Service Management는 다음의 두 가지 개념을 가지고 있다.

- Service : Compute(Nova), Object Storage(Swift) 등의 OpenStack 서비스 의미함. 하나의 서비스는 사용자가 자원을 접근할 수 있고 오퍼레이션을 실행할 수 있는 하나 이상의 엔드포인트를 제공한다.
- Endpoints : 서비스가 접근될 수 있는 네트워크 접근가능한 주소로서 대체로 URL로 기술될 수 있는 주소를 위미한다.

Keystone은 각 서비스에 대응하는 user(예를 들면, Compute Service를 한 'nova')와 service 라는 이름의 특별한 service tenant를 유지관리한다.

나. Keystone 설치 및 설정

OpenStack 서비스를 제공하는 다른 서버들이 접근가능한 서버에 "root"로서 Keystone을 설치한다.

```
# apt-get install keystone
```

설치 후에, keystone에서 데이터베이스로 mysql을 사용하려면 디폴트로 생성된 sqlite 데이터베이스를 삭제하고 MySQL데이터베이스로 설정을 변경한다.

```
# rm /var/lib/keystone/keystone.db
```

그리고 MySQL을 실행시켜서 "keystone"이라는 이름의 데이터베이스를 생성하고

“keystone” MySQL 데이터베이스에 대한 full access를 가진 “keystone”이라는 이름의 MySQL user를 생성한다. 그 명령어는 아래와 같다.

```
$ mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> grant all on keystone.* to 'keystone'@'%' identified by [keystone_db_passwd];
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on keystone.* to 'keystone'@'localhost' identified by [your_keystone_db_passwd];
Query OK, 0 rows affected (0.00 sec)
mysql> grant all on keystone.* to 'keystone'@[your_controller_ip] identified by [your_keystone_db_passwd];
Query OK, 0 rows affected (0.00 sec)
```

Keystone이 설치되고 나면 keystone의 기본 설정파일은 /etc/keystone/에 위치한 keystone.conf이다. Keystone이 사용하는 데이터베이스를 mysql로 사용하기 위해서 설정파일 내에 “connection” 부분에 해당 내용을 반영하여 변경해야 한다

```
# in /etc/keystone/keystone.conf
...
connection = mysql://keystone:[your_keystone_db_passwd]@[your_controller_ip] /keystone
...
```

기본적으로, Keystone은 Keystone과 그 외 다른 모든 서비스 사이에 ssl encryption을 사용하므로 Encryption certificates를 생성하기 위해서 다음과 같은 명령을 실행한다.

```
# keystone-manage pki_setup
# chown -R keystone:keystone /etc/keystone/*
```

새로운 데이터베이스 설정을 반영하기 위해 Keystone서비스를 재시작한다.

```
# service keystone restart
```

마지막으로, “root”계정으로 Keystone 데이터베이스를 초기화한다.

```
# keystone-manage db_sync
```

다. keystone과 타서비스들 연동 설정

keystone은 OpenStack의 각 컴포넌트들에 대한 인증 및 인가를 담당하는 서비스이

다. 이에 따라 keystone을 통해 위의 기능을 위임하고자 하는 서비스들은 keystone에 등록이 되어야 한다. Keystone에 서비스를 등록할 때에는 해당 서비스를 접근할 수 있는 사용자(user), 프로젝트(tenant), 사용자의 역할(role)의 집합으로 기술하게 되므로, 그에 따른 정의가 먼저 이루어져야 한다.

예를 들어, "demo"라는 이름의 디폴트 tenant를 생성해보자.

```
$ keystone tenant-create    -name demo -description "Default Tenant"
$ keystone tenant-list
+----------------------------------------+----------+-------------+
|       id                               | name     | enabled     |
+----------------------------------------+----------+-------------+
|    1d69e6c45ebf48b4b30677d63d3bcaa3    | demo     |    True     |
+----------------------------------------+----------+-------------+
```

그리고 "admin"이라는 이름의 user를 생성한다.

```
$ keystone user-create    --tenant-id 1d69e6c45ebf48b4b30677d63d3bcaa3 --name
admin --pass adminPass
+-------------+----------------------------------------------------------+
| Property    |    Value                                                 |
+-------------+----------------------------------------------------------+
|  email      |                                                          |
| enabled     | True                                                     |
|    id       | 4e93d0fed61b4b47a34d4dcd2b0491b2                         |
|  name       |  admin                                                   |
| tenantId    | 1d69e6c45ebf48b4b30677d63d3bcaa3                         |
+-------------+----------------------------------------------------------+
```

디폴트 policy.json파일의 기반이 되는 관리 역할인, "admin" role을 생성한다.

```
$ keystone role-create    --name admin
+-----------+-------------------------------------------+
| Property  |              value                        |
+-----------+-------------------------------------------+
|    id     |  d2e2722f4b4d48b080d90b08a9c3922c         |
|   name    |        admin                              |
+-----------+-------------------------------------------+
```

"user-add-role"이라는 서브 커맨드로 "admin" user 에게 "admin" role을 부여한다.

```
$ keystone user-role-add  \
 --user-id     4e93d0fed61b4b47a34d4dcd2b0491b2 \
 --tenant-id    1d69e6c45ebf48b4b30677d63d3bcaa3 \
 --role-id     d2e2722f4b4d48b080d90b08a9c3922c
```

다음으로 "service" tenant를 생성한다. 이 tenant는 OpenStack에서 service catalog에 등록될 service들을 관리하는 tenant가 된다.

```
$ keystone tenant-create    --name service --description "Service Tenant"
+-------------+----------------------------------------------+
| Property     |        Value                                 |
+-------------+----------------------------------------------+
| description   |       Service    Tenant                      |
| enabled       |          True                                |
|       id      |    df6e4007f95542ea9bbbb23696d6c6b9       |
| name          |                service                       |
+--------------+----------------------------------------------+
```

그리고 Keystone service catalog에 추가되어야 할 서비스들을 하나씩 'Service' Tenant에 생성한다. 다음은 'service' tenant에 추가될 user들이다.

✓ Glance 서비스 user 추가

```
$ keystone user-create --tenant-id df6e4007f95542ea9bbbb23696d6c6b9 --name
glance –pass [your_glance_passwd]
+------------+----------------------------------------------+
| Property    |     Value                                    |
+------------+----------------------------------------------+
| email        |                                             |
| enabled     |           True                              |
|     id        |     4220d8bae75043e6b8da52904f16e13d     |
|    name     |                    glance                    |
| tenantId    |   df6e4007f95542ea9bbbb23696d6c6b9        |
+------------+----------------------------------------------+
```

- "glance" user에 대하여 "admin"role 설정

```
$ keystone user-role-add \
 --user-id 4220d8bae75043e6b8da52904f16e13d \
 --tenant-id    df6e4007f95542ea9bbbb23696d6c6b9 \
 --role-id     d2e2722f4b4d48b080d90b08a9c3922c
```

✓ Nova 서비스 user 추가

```
$ keystone user-create --tenant-id df6e4007f95542ea9bbbb23696d6c6b9 --name
nova –pass [your_nova_passwd]
+-----------+-----------------------------------------+
| Property   |      Value                              |
+-----------+-----------------------------------------+
| email      |                                         |
| enabled    |         True                            |
|     id     |      8ae0c152284e40c78f25017a0369e573     |
|   name     |          nova                           |
| tenantId   |        df6e4007f95542ea9bbbb23696d6c6b9   |
+-----------+-----------------------------------------+
```

- “nova” user에 대하여 “admin” role 설정

```
$ keystone user-role-add \
  --user-id     8ae0c152284e40c78f25017a0369e573 \
  --tenant-id    df6e4007f95542ea9bbbb23696d6c6b9 \
  --role-id     d2e2722f4b4d48b080d90b08a9c3922c
```

✓ Cinder 서비스 user 추가

```
$ keystone user-create --tenant-id df6e4007f95542ea9bbbb23696d6c6b9 --name
cinder –pass [your_cinder_passwd]
+-------------+-----------------------------------------+
| Property     |   Value                                 |
+-------------+-----------------------------------------+
|  email       |                                         |
| enabled      |              True                       |
|     id       |     9ae0c152284e40c78f25017a0369e574    |
|   name       |            cinder                       |
| tenantId     |     df6e4007f95542ea9bbbb23696d6c6b9    |
+--------------+-----------------------------------------+
```

✓ “cinder” user에 대하여 “admin” role 설정

```
$ keystone user-role-add --user-id  9ae0c152284e40c78f25017a0369e574 \
 --tenant-id df6e4007f95542ea9bbbb23696d6c6b9 \
 --role-id d2e2722f4b4d48b080d90b08a9c3922c
```

라. 서비스 정의

Keystone은 또한 service catalog로서의 역할도 한다. 그것을 통해 OpenStack의 다른 시스템들이 OpenStack 서비스의 API endpoint가 어디에 위치하는 지를 알려준다. 특히, OpenStack Dashboard의 경우 이 service catalog에 대한 의존도가 높다. Keystone에 서비스를 정의하는 방법은 두 가지가 있다. 한 가지는 template파일을 이용하는 것이고, 또 한 가지는 database backend를 이용하는 것이다. template 파일을 이용하는 것이 더 간편하긴 하지만 DevStack같은 개발환경을 위해서만 권고된다. Database backend를 이용하면 reliability, availability, 그리고 data redundancy를 제공할 수 있다. 서비스를 정의하는 방식은 /etc/keystone/keystone.conf에 포함되어야 하며 database backend를 이용하는 방식으로의 설정은 다음과 같다.

```
# in /etc/keystone/keystone.conf
...
[catalog]
driver =    keystone.catalog.backends.sql.Catalog
...
```

▶ Keystone service catalog entry를 위한 요소들

catalog에 있는 각 서비스에 대하여, catalog에 서비스를 생성시키는 오퍼레이션(keystone service-create)과 서비스의 엔드포인트를 생성시키는 오퍼레이션(keystone endpoint-create)이 수행되어야 한다. keystone service-create은 다음의 속성들과 함께 수행되어야 한다.

--name : 서비스의 이름(e.g., nova, ec2, glance, keystone)
--type : 서비스의 타입(e.g., compute, ec2, image, identity)
--description : 서비스의 설명(e.g., “Nova Compute Service”)

keystone endpoint-create은 해당 서비스로 연결할 수 있는 클라이언트들의 종류를 기술하는 데이터베이스 엔트리를 생성하는 것으로서, 다음의 속성들과 함께 수행되어야 한다.

--region : 사용하고 있는 OpenStack cloud 주어진 region 이름
--service-id : keystone service-create에 의해 반환되는 ID 필드
--publicurl : 해당 service에 외부 접근가능한 URL
--internalurl : 해당 service에 내부적으로 접근가능한 URL. 이것은 일반적으로 publicurl과 같은 값을 갖는다.
--adminurl : 해당 서비스의 admin에 대한 URL. Keystone과 EC2서비스는 publicurl과 adminurl을 다르게 사용하지만 다른 서비스들의 경우에는 이들 둘의 값은 같다.

다음은 각각의 서비스와 서비스 endpoints들을 정의하는 방법을 보여준다.

```
$ keystone service-create --name=Keystone --type=Identity --description="Identity Service"
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+----------------------+----------------------------------------------------------------+
|       Property       |                          Value                                 |
+----------------------+----------------------------------------------------------------+
| description          |                  Identity Service                              |
|          id          |    7f9dfe1b73914feaa042e3f32d1429e3                            |
|          name        |                          Keystone                              |
|          type        |                           Identity                             |
+----------------------+----------------------------------------------------------------+
```

```
$ keystone endpoint-create \
> --region Region One \
> --service-id=7f9dfe1b73914feaa042e3f32d1429e    \
> --publicurl=http://[ControllerIP]:5000/v2.0 \
> --internalurl=\
> --adminurl=
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+----------------------+----------------------------------------------------------------+
|       Property       |                          Value                                 |
+----------------------+----------------------------------------------------------------+
|       adminurl       |                                                                |
|          id          |   99050bb5bc3247a39d8edaf802bd5856                             |
| internalurl          |                                                                |
|      publicurl       |                                                                |
|        region        |                  RegionOne                                     |
|      service_id      |   7f9dfe1b73914feaa042e3f32d1429e3                             |
+----------------------+----------------------------------------------------------------+
```

```
$ keystone service-create –name=Nova --type=compute --description="Compute Service"
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+----------------------+----------------------------------------------------------------+
|       Property       |                             Value                              |
+----------------------+----------------------------------------------------------------+
|  description         |                  Compute Service                               |
|          id          |     7999361469974f59ab7f178c8e98f070                           |
|          name        |                       Nova                                     |
|          type        |                       compute                                  |
+----------------------+----------------------------------------------------------------+
```

```
$ keystone endpoint-create \
>--region RegionOne \
>--service-id=7999361469974f59ab7f178c8e98f070    \
>--publicurl=' \
>--internalurl=' \
>--adminurl='http://[ControllerIP]:8774/v2/%(tenant_id)s'
WARNING: Bypassing authentication using a token & endpoint (authentication credentials are being ignored).
+----------------------+----------------------------------------------------------------+
|       Property       |                Value                                           |
+----------------------+----------------------------------------------------------------+
|       adminurl       |                                                                |
|          id          |          8cfc49a3f38b4a7ea523aa1ca421cf12                      |
|  internalurl         |                                                               |
|      publicurl       |                                                                |
|         region       |                RegionOne                                       |
|      service_id      |     7999361469974f59ab7f178c8e98f070                           |
+----------------------+----------------------------------------------------------------+
```

```
$ keystone service-create --name=Glance --type=image --description="Image Service"
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+----------------------+----------------------------------------------------------+
|       Property       |                  Value                                   |
+----------------------+----------------------------------------------------------+
| description          |                  Image Service                           |
|          id          |  3c91be3d25ad454fa5b71a3bb4637512                        |
|          name        |                        Glance                            |
|          type        |                        image                             |
+----------------------+----------------------------------------------------------+
```

```
$ keystone --endpoint endpoint-create --region RegionOne \
>--service-id=3c91be3d25ad454fa5b71a3bb4637512    \
>--publicurl= \
>--internalurl= \
>--adminurl=
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+----------------------+----------------------------------------------------------+
|       Property       |                  Value                                   |
+----------------------+----------------------------------------------------------+
|       adminurl       |                                                          |
|          id          |          739dcd5774a441fd826965856ba78fcb                |
|       internalurl    |                                                          |
|    publicurl         |                                                          |
|         region       |                  RegionOne                               |
|     service_id       |    3c91be3d25ad454fa5b71a3bb4637512                      |
+----------------------+----------------------------------------------------------+
```

```
$ keystone service-create --name=Cinder --type=volume --description="Volume Service"
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+---------------------+----------------------------------------------------------+
|       Property      |                          Value                           |
+---------------------+----------------------------------------------------------+
| description         |                    Volume Service                        |
|          id         |     057b6c10678f4323bdc208323c6c78ec      |
|          name       |                          Cinder                          |
|          type       |                          volume                          |
+---------------------+----------------------------------------------------------+
```

```
$ keystone endpoint-create --region RegionOne \
>--service-id=057b6c10678f4323bdc208323c6c78ec    \
>--publicurl=' \
>--internalurl=' \
> --adminurl='
WARNING: Bypassing authentication  using a token & endpoint (authentication credentials are being ignored).
+--------------------+-----------------------------------------------------------------------+
|       Property     |                          Value                                        |
+--------------------+-----------------------------------------------------------------------+
|      adminurl      |                                                                       |
|          id        |          f9395f5901a94b87ab579217c4582ad3                             |
| internalurl        |                                                                       |
|      publicurl     |                                                                       |
|   region           |                          RegionOne                                    |
|      service_id    |              057b6c10678f4323bdc208323c6c78ec                         |
+--------------------+-----------------------------------------------------------------------+
```

※ 설치팁

keystone_basic.sh, keystone_endpoints_basic.sh 스크립트를 이용하여 데이터를 일괄 업로드 한다.(이때, 스크립트 내에 'host_ip' 수정 필요함)

2) Glance(이미지 서비스)

Glance는 OpenStack에서 Image들의 등록, 검색, 관리등의 역할을 담당한다.

가. Glance 설치 및 설정

Glance를 “root”계정으로 설치한다.

```
# apt-get install glance
```

MySQL을 사용한다면, /var/lib/glance/ 디렉토리안에 있는 glance.sqlite 파일을 삭제한다.

```
# rm /var/lib/glance/glance.sqlite
```

그리고 Glance database backend를 설정하기 위해서 MySQL을 사용하여 “glance” MySQL데이터베이스와 “glance” MySQL user를 생성한다. 그리고 “glance” user에게는 “glance” MySQL 데이터베이스 접근에 대한 전권을 준다. 이를 위한 MySQL 명령은 다음과 같다.

```
$ mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> grant all on glance.* to 'glance'@'%' identified by '[your_glance_db_passwd]';
Query OK, 0 rows affected (0.00 sec)
mysql> grant all on glance.* to 'glance'@'localhost' identified by '[your_glance_db_passwd]';
Query OK, 0 rows affected (0.00 sec)
```

▶ Glance 설정 파일 수정

Glance는 Glance API server, Glance Registry server, 그리고 Glance가 이미지들을 저장할 수 있게 하는 다양한 storage backends를 설정하는 데 사용하는 다양한 옵션들을 가지고 있다. 디폴트로, glace-api.conf config file에 [DEFAULT]라는 섹션에 명시되어 있다.

Glance-api 서비스는 OpenStack Images API 버전1과 버전2를 구현하였다. 기본적으로 두 가지 모두 glance-api.conf 파일에 True로 설정되어 있다.

enable_v1_api=True
enable_v2_api=True

glance-api.conf 파일에서 한 가지 옵션은 “false’로 만들어야 한다.

그리고 Glance에서는 인증(authentication)을 위해서 keystone을 사용하기 때문에 /etc/glance/glance-api.conf에 그에 대한 설정이 필요하다.

```
# in /etc/glance/glance-api.conf
...
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
auth_tenant_name = service
admin_user = glance
admin_password = [your_glance_passwd]

[paste_deploy]
config_file = /etc/glance/glance-api-paste.ini
flavor = keystone
```

그리고 데이터베이스로서 sqlite보다는 MySQL로 지정하려면 다음과 같은 설정이 /etc/glance/glance-api.conf 필요하다.

```
# in /etc/glance/glance-api.conf
...
sql_connection = mysql://glance:[your_glance_db_passwd]@[MYSQL-SERVER-IP]/glance
```

변경된 세팅을 반영시키기 위하여 glance-api를 재시작한다.

```
# service glance-api restart
or
# restart glance-api
```

앞서 설정한 admin user와 service tenant, db에 대한 내용을 반영하기 위하여, 그리고 identity service를 활성화시키기 위하여 /etc/glance/glance-registry.conf의 마지막 부분을 업데이트 한다.

```
# in /etc/glance/glance-registry.conf
...
sql_connection                                                                  =
mysql://glance:[your_glance_db_passwd]@[MYSQL-SERVER-IP]/glance
...
[keystone_authtoken]
auth_host =    127.0.0.1
auth_port =    35357
auth_protocol    = http
auth_tenant_name    = service
admin_user =    glance
admin_password    = [your_glance_passwd]

[paste_deploy]
config_file =    /etc/glance/glance-registry-paste.ini
flavor =    keystone
```

그리고 /etc/glance/glance-registry-paste.ini도 Identity service인 keystone을 활성화시키기 위해서 업데이트 되어야 한다.

```
# Use this    pipeline for keystone auth
[pipeline:glance-registry-keystone]
pipeline = authtoken context registryapp
```

Database backends로서 MySQL 데이터베이스를 이용하기 위해서 위의 glance-registry.conf 설정파일에서 보는 것처럼 'sql_connection'의 내용을 수정해야 한다. 그리고 glance-registry데몬을 재구동시킨다.

```
# service    glance-registry restart
or
# restart    glance-registry
```

Ubuntu 12.04에서는, 데이터베이스 테이블들이 버전 컨트롤되고 있기 때문에 아래와 같은 단계를 반드시 수행해야 한다.

```
# glance-manage version_control 0
```

그리고 나서 데이터베이스를 생성하거나 마이그레이션할 수 있다.

```
# glance-manage db_sync
```

마지막으로 Glance 서비스들을 재구동 시킨다.

```
# service glance-registry    restart
# service glance-api    restart
```

나. Glance 설치 검증

Glance가 제대로 설치되었는지 검증하기 위해서 다음의 명령을 수행해본다.

```
$ glance index
```

이 명령에 대하여 아무 결과도 반환하지 않으면, Glance는 keystone과 제대로 연동되어 설치된 것으로 볼 수 있다. 그리고 다음으로 이미지를 업로드해 볼 수 있는데, Ubuntu UEC 이미지를 예로 사용하기 위하여 먼저 Controller 시스템에 다운받는다.

```
$wget \
http://uec-images.ubuntu.com/releases/12.04/release/ubuntu-12.04-server-cloudimg-amd64-disk1.img
```

그리고 다음의 명령으로 Glance에 해당 이미지를 추가시킨다.

```
$ glance add    name="Ubuntu 12.04 cloudimg amd64" \
 is_public=true    container_format=ovf \
 disk_format=qcow2 < ubuntu-12.04-server-cloudimg-amd64-disk1.img
```

이후 다음의 명령어를 한번 더 수행시키면 추가된 이미지를 볼 수 있다.

```
glance index
```

3) nova

Controller에서 nova는 compute 자원을 할당해주는 역할을 하는 컴포넌트이다.

가. nova를 위한 SQL 데이터베이스 설정

MySQL데이터베이스를 설정하기 위해서, nova 데이터베이스를 생성해야 한다. 그리고 해당 데이터베이스를 접근할 수 있도록 “nova” MySQL user를 생성하고 이 user에게는 “nova” 데이터베이스에 대한 전권을 준다. 그에 대한 MySQL명령은 다음과 같다.

```
# mysql -u root -p
mysql> create database nova;
Query OK, 1 row affected (0.00 sec)
mysql> grant all on nova.* to 'nova'@'localhost' identified by '[your_nova_db_passwd]';
Query OK, 0 rows affected (0.00 sec)
mysql> grant all on nova.* to  'nova'@'%' identified by '[your_nova_db_passwd]';
Query OK, 0 rows affected (0.00 sec)
```

nova-package에 관련된 모듈들을 인스톨한다.

```
# apt-get install nova-novncproxy novnc nova-api nova-ajax-console-proxy nova-cert nova-conductor nova-consoleauth nova-doc nova-scheduler
```

▶ nova 설정

nova의 설정을 위해서는 /etc/nova에 디폴트로 설치되어 있는 nova.conf를 이용한다. 패키지들은 자동적으로 nova라는 이름의 user로 설치과정을 진행하게 되지만, 만약 다른 유저로 설치하고 있다면 이후 nova.conf파일의 소유자는 nova:nova로 되어야 하고, 허가모드는 0640로 되어야 한다. 왜냐하면 이 파일에는 MySQL 서버의 username과 password를 포함하고 있기 때문이다.

```
# chown -R nova:nova /etc/nova
# chmod 640 /etc/nova/nova.conf
```

cf) Variable substitution

설정파일(configuration file)은 변수 대체를 지원한다. 하나의 설정 변수가 사용되었다면 그 후의 설정 값에서 $을 이용하여 레퍼런스 될 수 있다. 다음의 예에서와 같이 my_ip 정의되고 나면 $my_ip는 변수로서 사용될 수 있다.

Controller노드는 nova-api, nova-scheduler, nova-cert, nova-consoleauth, nova-conductor를 운영할 것이고 선택적으로 nova-network, nova-compute도 운영될 수 있다. nova-network의 설치 및 설정에 대해서는 뒤에 더 다루기로 한다.

nova 설정을 수정하기 전에 nova-service들을 정지시켜야 한다. 따라서 다음과 같이 명령을 수행한다.

```
# stop nova-api
# stop nova-conductor
# stop nova-scheduler
# stop nova-novncproxy
```

nova의 설정파일에는 설치할 서비스들에 대한 정보가 모두 기술되어 있어야 한다.

nova는 각 서비스들 간의 중심에서 서로를 연결하는 역할을 하고 있기 때문이다. 아래 nova.conf 예에서 보듯이, Cinder(volume)에 대한 정보, Glance에 대한 정보, Keystone에 대한 정보 등이 포함되어 있다.

```
# in /etc/nova/nova.conf

[DEFAULT]
my_ip=[ControllerIP]
...
#VOLUMES
volume_api_class=nova.volume.cinder.API

#DATABASE
sql_connection=mysql://nova:[your_nova_db_passwd]@[ControllerIP]/nova
...
#RABBITMQ
rpc_backend = nova.openstack.common.rpc.impl_kombu
rabbit_host=[ControllerIP]
...
#GLANCE
image_service=nova.image.glance.GlanceImageService
glance_api_server=[ControllerIP]:9292
auth_strategy=keystone
...
[keystone_authtoken]
auth_host=[ControllerIP]
auth_port=35357
auth_protocol=http
admin_tenant_name=service
admin_user=nova
admin_password=[your_nova_passwd]
signing_dirname=/tmp/keystone-signing-nova
enabled_apis=metadata
```

그리고 또한 /etc/nova/api-paste.ini도 아래와 같은 형태로 설정되어야 한다.

```
# in /etc/nova/api-paste.ini
...
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = [Controller-IP]
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = [your_nova_password]
```

▶ Database 설정하기

다음의 커맨드를 실행시킬 backend data store에 테이블을 생성하기 위해서 다음과 같이 명령한다.

```
$ sudo nova-manage db sync
```

위 명령에 대해서 문제가 있는지 없는 지 확인하고자 한다면, /var/log/nova/nova-manage.log로 볼 수 있다. 내용이 없다면 해당 명령이 정확하게 수행되었고 nova database가 이제 생성되었다는 것을 의미한다. 전체적으로 모든 서비스를 재구동 한다.

```
# start    nova-api
# start    nova-conductor
# start    nova-scheduler
# start    nova-novncproxy
```

만약 "start"라는 명령으로 nova 서비스들이 실행되지 않는다면, nova 서비스들이 제대로 운영되지 않을 수도 있다. 문제점을 찾기 위해 /var/log/nova의 로그들을 잘 살펴보자.

※ nova-network
single-host mode로 운영하게 될 경우, Controller 노드에만 nova-network가 설치되면 된다. 그러나 multi-host mode로 운영하게 될 경우, Controller노드와는 상관없이 각 Compute 노드에 nova-network, nova-api-metadata 서비스가 설치되어야 한다.

4) Cinder

OpenStack에서 블록 스토리지를 연동하는 컴포넌트로 cinder가 있다. 블록 스토리지란 하나의 block device로서, 사용자들이 VM 인스턴스에 연동(attach)시켜 사용할 수 있는 저장공간을 의미한다. OpenStack에서 이러한 블록 스토리지 연동을 위해 Cinder라는 프로젝트로 구현되었으며, 이것은 다양한 형태의 드라이버들을 지원한다.

표 4 Open-source file-level shared storage solutions

	object	block	File-level(live migration support)
Swift	✓		
LVM		✓	
Ceph	✓	✓	Experimental
Gluster	✓		✓
NFS		✓	✓
ZFS		✓	
sheepdog		experimental	

표 4는 볼륨드라이버들로서, 파일시스템을 사용하기 위한 실험적인 지원은 Folsom릴리즈에서 시작되었다. 이것은 초기에 Cinder에서 NFS를 사용하기 위한 레퍼런스 드라이버로 시작되었는데, Grizzly릴리즈에 와서 GlusterFS드라이버 뿐만이 아니라 NFS 드라이버까지 완전히 확장되었다. 따라서 드라이버에 따라 조금의 차이는 있겠지만 이러한 드라이버들은 "Block" storage 드라이버로서 역할을 하는 것이므로 NFS 또는 GlusterFS 파일 시스템 상에서 하나의 파일 시스템은 해당 인스턴스에 "virtual" volume으로 생성되고 매핑된다.(본 문서에서는 올해 Cinder의 volume 드라이버로 사용되었던 NFS를 기준으로 기술한다.)

Cinder를 설치하기 위해서는 우선적으로 NFS가 해당 시스템에 설치되어 있어야 한다.

가. NFS 드라이버 올리기

▶ 스토리지 서버로 작동할 리눅스 시스템에 NFS server 설치

Linux OS가 설치(본 문서에서는 Ubuntu 12.04)된 서버에 NFS관련 패키지들을 인스톨한다.

```
$ sudo apt-get install nfs-kernel-server nfs-common portmap
```

패키지 설치가 완료된 후에는 /etc/exports를 수정한다. 작성 내용으로는 export 할 디렉토리(예, /home/nfsTest))와 접근 가능한 네트워크 대역(예, 1*92.168.1.1/24*)과 접근허가들(예, rw, no_root_squash, async)이다.

```
# in /etc/exports
 /home/nfsTest 192.168.1.1/24(rw, no_root_squash, async)
```

위의 설정이 되어 있지 않으면 다음과 같은 에러가 발생한다.

"mount.nfs: access denied by server while mounting"

설정이 완료된 후, nfs 서버를 재구동한다.

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

▶ Controller에 NFS client 설치

OpenStack Controller 시스템에 NFS client 관련 패키지들을 설치한다.

```
$ sudo apt-get install portmap nfs-common
```

NFS 서버에 대한 마운트 포인트를 설정하기 위해 디렉토리(예, /home/nfsClient)를 생성한다.

```
$ sudo mkdir /home/nfsClient
```

재부팅시 자동 마운팅을 위해서 다음과 같은 방식으로 /etc/fstab을 변경할 수도

있다.

```
# in /etc/fstab
...
[NFSserver]:/home/nfsTest /home/nfsClient nfs rsize=8192,wsize=8192,timeo=14,intr
...
```

만약 방화벽이 있다면 NFS를 이용하기 위해서는 NFS관련 포트인 32771, 111, 2049를 열어줘야 한다.

NFS Server와 Client간 마운트가 잘 되는 지 확인을 위해서 다음과 같이 테스트 해볼 수 있다.

```
$ sudo mount [NFSserver]:/home/nfsTest    /home/nfsClient
```

나. Cinder 패키지 설치 및 설정

Cinder는 외부 인터페이스를 담당하는 api, 외부 요청을 스케줄링하는 scheduler, 그리고 block storage제공을 담당하는 volume으로 구성되어 있어 다음과 같이 세 가지 패키지를 인스톨해야 한다.

```
# apt-get install cinder-api cinder-volume cinder-scheduler
```

그리고 볼륨 드라이버를 NFS로 설정하기 위하여 /etc/cinder/cinder.conf를 수정하여야 한다. 설정해야 하는 내용은 볼륨 드라이버와 NFS의 네 가지 설정 플래그 중 cinder에서 사용하고자 하는 플래그이다. 본 문서에서는 "nfs_shares_config" flag를 선택하는 설정으로 기술한다.

```
# in    /etc/cinder/cinder.conf
...
volume_driver    = cinder.volume.drivers.nfs.NfsDriver
nfs_shrares_config=/etc/cinder/shares.txt
...
```

"nfs_shares_config"는 "설정 파일에서 nfs 서버 정보를 추출한다"는 플래그이므로, nfs서버 정보가 담긴 설정 파일의 경로를 적어준다. 그리고 shares.txt는 다음과 같은 형태로 작성한다.

```
# /etc/cinder/shares.txt
[NFSserverIP]:/home/nfsTest
```

Cinder의 database backend를 생성하기 위해 MySQL을 이용한다. 앞서 다른 컴포넌트들이 생성한 것처럼, cinder 데이터베이스와, cinder user를 생성하고 해당 user

에게 cinder 데이터베이스에 접근하여 조작할 수 있는 전권을 준다.

```
$ mysql -u root -p
mysql>  create database cinder;
Query  OK, 1 row affected (0.00 sec)
mysql>  grant all on cinder.* to 'cinder'@'%' identified by
'[your_cinder_db_passwd]';
Query  OK, 0 rows affected (0.00 sec)
mysql>  grant all on cinder.* to 'cinder'@'localhost' identified by
'[your_cinder_db_passwd]';
Query  OK, 0 rows affected (0.00 sec)
```

그리고 cinder 데이터베이스에 테이블들을 생성하기 위해서 "db sync"를 수행하여야 하는 데, 그 전에 cinder 모든 서버들의 운영을 중지시킨다.

```
$ sudo stop  cinder-api
$ sudo stop  cinder-volume
$ sudo stop  cinder-scheduler
```

이후 다음과 같이 cinder 데이터베이스에 대한 sync를 수행한다.

```
$ sudo cinder-manage db sync
```

위의 과정이 모두 완료되면 설정된 내용들을 반영하여 cinder의 configuration 파일인 /etc/cinder/cinder.conf와 /etc/cinder/api-paste.ini파일을 기술한다.

```
#in /etc/cinder/cinder.conf
 ...
[DEFAULT]
rootwrap_config   = /etc/cinder/rootwrap.conf
api_paste_confg   = /etc/cinder/api-paste.ini
sql_connection   = mysql://cinder:[your_cinder_db_passwd]@[ControllerIP]/cinder
volume_driver = cinder.volume.drivers.nfs.NfsDriver
nfs_shares_config = /etc/cinder/shares.txt
volume_name_template   = volume-%s
volume_group   = cinder-volumes
verbose =   True
auth_strategy   = keystone
state_path =   /var/lib/cinder
lock_path =   /var/lock/cinder
volumes_dir =   /var/lib/cinder/volumes

#RABBITMQ
rabbit_host=[ControllerIP]
...
```

```
#/etc/cinder/api-paste.ini
  ...
[filter:authtoken]
paste.filter_factory    = keystoneclient.middleware.auth_token:filter_factory
service_protocol    = http
service_host    = 127.0.0.1
service_port    = 5000
auth_host =    127.0.0.1
auth_port =    35357
auth_protocol    = http
admin_tenant_name    = service
admin_user =    cinder
admin_password    = [your_cinder_passwd]
signing_dir =    /var/lib/cinder
...
```

db sync가 완료되고 나면 cinder 서버들을 재구동시킨다.

```
#    service cinder-api start
#    service cinder-scheduler start
#    service cinder-volume start
```

다. Cinder 설치 검증

"cinder create" 명령어를 통해서 cinder가 block storage를 생성해본다.

```
$    cinder create --display-name=vol 1
+---------------------+--------------------------------------+
|          Property      |                Value                 |
+---------------------+--------------------------------------+
|       attachments     |                  []                  |
|   availability_zone   |                 nova                 |
|          bootable      |                false                 |
|         created_at      |      2013-08-27T03:50:01.220946      |
|   display_description|                 None                 |
|       display_name   |                 vol                  |
|              id           |  c22d2ba4-e61b-44e2-a021-9f33bc029d5f  |
|          metadata      |                  {}                  |
|             size          |                  1                   |
|       snapshot_id      |                 None                 |
|       source_volid      |                 None                 |
|           status         |               creating               |
|       volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

"cinder list"를 통해서 볼륨 생성 결과를 확인한다.

```
$    cinder list
+--------------------------------------+-------+----------------+-----+---------------+----------+--------------+
|                  ID                  |Status|Display Name|Size|VolumeType|Bootable|Attachedto|
+--------------------------------------+-------+----------------+------+---------------+----------+-------------+
|c22d2ba4-e61b-44e2-a021-9f33bc029d5f|Active| vol          | 1   | None        |    false|             |
+--------------------------------------+--------+---------------+------+---------------+----------+-------------+
```

위와 같이 cinder를 통해 볼륨을 생성하고 결과를 확인함으로써 cinder가 제대로 설치되었는지 검증할 수 있다.

5) Horizon

Horizon은 사용자들이 웹을 통해서 OpenStack 서비스를 접근할 수 있는 Dashboard로서, 앞서 정의했던 OpenStack 서비스들에 대한 사용자 인터페이스들이다.

가. Horizon 설치

"root" 계정으로, 다음의 패키지들을 설치한다.

```
# apt-get install -y memcached libapache2-mod-wsgi openstack-dashboard
```

나. Horizon 설정

▶ Simple Deployment (HTTP)

/etc/openstack-dashboard/local_settings.py 파일내에 OPENSTACK_HOST, Keystone(Identity Service) endpoint를 명시한다.

```
# in /etc/openstack-dashboard/local_settings.py
...
OPENSTACK_HOST = "127.0.0.1"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v2.0" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "Member"
...
```

▶ Secured Deployment (HTTPS)

기본 설치는 non-encrypted channel(HTTP)를 이용하지만, OpenStack Dashboard을 위해 SSL을 적용할 수도 있다. 적용 방법은 다음과 같다.

a. /etc/openstack-dashboard/local_settings.py내에 있는 다음의 변수를 활성화시킨다.

```
# in /etc/openstack-dashboard/local_settings.py
...
USE_SSL = True
...
```

b. /etc/apache2/ports.conf에 다음과 같은 라인을 추가한다.

```
# in /etc/apache2/ports.conf
...
NameVirtualHost *:443
...
```

c. /etc/apache2/conf.d/openstack-dashboard.conf를 다음과 같이 수정한다.
수정 전:

```
WSGIScriptAlias /horizon /usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi
WSGIDaemonProcess horizon user=www-data group=www-data processes=3 threads=10
Alias /static /usr/share/openstack-dashboard/openstack_dashboard/static/
<Directory /usr/share/openstack-dashboard/openstack_dashboard/wsgi>
        Order allow,deny
        Allow from all
</Directory>
```

수정 후:

```
<VirtualHost *:80>
        ServerName reallab.kreonet.net
        RedirectPermanent / https://reallab.kreonet.net/
</VirtualHost>
<VirtualHost *:443>
        ServerName reallab.kreonet.net
        SSLEngine On
        SSLCertificateFile /etc/apache2/SSL/cert.pem
        SSLCACertificateFile /etc/apache2/SSL/Global-Chain.pem
        SSLCertificateKeyFile /etc/apache2/SSL/key.pem
        SetEnvIf User-Agent ".*MISE.*" nokeepalive ssl-unclean-shutdown

        WSGIScriptAlias /horizon /usr/share/openstack-dashboard/openstack_dashboard/wsgi/django.wsgi
        WSGIDaemonProcess horizon user=www-data group=www-data processes=3 threads=10
        Alias /static /usr/share/openstack-dashboard/openstack_dashboard/static/
        <Directory /usr/share/openstack-dashboard/openstack_dashboard/wsgi>
                Order allow,deny
                Allow from all
        </Directory>
</VirtualHost>
```

위의 설정에서는 추가적으로 private key, public key, certificate를 정의하였다. 이로써 Apache는 443포트로의 트래픽을 수신하며 non-secured 요청에 대해서 HTTPS 프로토콜로 리다이렉트하게 한다.

d. apache와 memcached 재시작한다.

```
# service apache2 restart
# service memcached restart
```

다. Horizon 설치 검증

Horizon이 잘 설치되었는지 확인하기 위해서, 웹브라우저에서 Horizon이 설치된 IP로 페이지요청을 해본다. 위의 설정상 URL은 다음과 같다.

```
https://[horizon-IP]/horizon
```

일단 해당 URL로 접근이 되어 로그인화면이 보이면 Identity Service인 Keystone에서 생성했던 사용자의 정보로 접근한다.(id/password)

라. Horizon에 RealLab Logo 및 Site Branding 삽입(Optional)

OpenStack Dashboard에 사용자가 원하는 로고 및 Site branding을 넣기 위해서는 다음과 같은 작업이 필요하다.

a. site branding을 위해서 /etc/openstack-dashboard/local_settings.py에 다음 라인을 추가한다.

```
# in /etc/openstack-dashboard/local_settings.py
...
USE_SSL = True
SITE_BRANDING = 'RealLab Dashboard'
...
```

b. Horizon의 기본 포맷에 들어갈 수 있는 로고 제작. 로고는 두 가지 형태로 제작하여, 하나는 로그인 페이지에서 사용하고 다른 하나는 일반 페이지에서 사용할 수 있도록 한다. 그리고 되도록 로고 배경은 투명하게 제작한다.(파일은 .png로 제작)

c. 두 개의 .png 파일을 다음 디렉토리에 위치시킨다.
/usr/share/openstack-dashboard/openstack-dashboard/static/dashboard/img

d. /usr/share/openstack-dashboard/openstack-dashboard/template/_stysheet.html에 다음 내용 추가한다.

```
<style type = "text/css">
h1.brand a{
    background : url({{STATIC_URL}} dashboard/img/RealLab-rect.png) no-repeat;
    display : block;
    float : left;
    width : 116px;
    height : 123px;
    text-indent : -9999px;
    margin-left : 53px;
    margin-top : 30px;
    margin-bottom : 10px;
}

#splash .login{
    backgroud : #fff url({{STATIC_URL}} dashboard/img/RealLab-login.png) no-repeat
     center : 35px;
}
</style>
```

5. Compute 노드 설치 및 설정

OpenStack Compute 노드에 대한 설치 작업은 OpenStack Controller를 설치하고 설정하는 데 비하여 간단히 진행될 수 있다. 그러나 Controller 노드와 Compute 노드가 연동되어 RealLab이 잘 구축되기 위해서는 Compute 노드의 네트워크 설정이 매우 중요하다.
본 문서에서는 OpenStack Compute 노드의 기본적인 패키지 설치 과정과 OpenStack의 네트워킹 방식과 그에 대한 설정과정을 기술한다.

1) Controller 노드와 시간 맞추기
다수의 시스템을 통해서 서비스가 이루어지기 때문에, 노드간 시스템 동기화가 필요하다. 이를 위해 Controller 노드와 주기적으로 시간을 동기화 시킬 수 있도록 아래와 같이 cron 파일(/etc/cron.daily/ntpdate)을 작성한다.

```
# in /etc/cron.daily/ntpdate
ntpdate 203.253.235.131
hwclock -w
```

2) 하드웨어 지원 설정
OpenStack Compute 패키지를 설치하기 전에 시스템 바이오스에서 "Virtualization Technology"가 활성화되어있는 지 확인할 필요가 있다.

a. 하드웨어적으로 가상화를 지원하는 지 여부 확인

```
user@nova-compute2:~$ sudo apt-get install cpu
user@nova-compute2:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

b. kvm 활성화

```
root@nova-compute2:/home/useri# lsmod | grep kvm
kvm_intel               132759  0
kvm                     414070  1 kvm_intel
root@nova-compute2:/home/user# modprobe    kvm_intel
root@nova-compute2:/home/user# modprobe    kvm
```

3) OpenStack Compute 패키지 설치
가. compute service : nova-compute

```
sudo apt-get install    nova-compute
```

나. network service : nova-network
Single-host mode로 운영하게 될 경우, Compute 노드에 nova-compute만 설치하

면 된다. 그러나 mult-host mode로 운영하게 될 경우, 각 Compute 노드에 nova-compute외에 nova-network, nova-api-metadata 서비스도 설치되어야 한다.

a. Compute 노드의 네트워크 인터페이스 설정

브릿지 인터페이스인 br100을 추가하기 위해서 /etc/network/interface을 수정한다.

```
# This file describes the    network interfaces available on your system
# and how to activate them.    For more information, see interfaces(5).

# The loopback network    interface
auto lo
iface lo inet loopback

# The primary network    interface
auto eth4
iface eth4 inet static
        address 134.75.114.133
        netmask 255.255.255.128
        network 134.75.114.128
        broadcast 134.75.114.255
        gateway 134.75.114.129
        dns-nameservers 150.183.95.96

# The management network interface
auto eth0
iface eth0 inet static
        address 203.253.235.153
        netmask 255.255.255.192
        network 203.253.235.128
        broadcast 203.253.235.191
        gateway 203.253.235.129
        # dns-* options are implemented by    the resolvconf package, if installed
        dns-nameservers 150.183.95.96

#Bridge network interface for VM networks
auto br100
iface br100 inet static
        address 10.0.0.1
        netmask 255.255.255.0
        bridge_stp off
        bridge_fd 0
        bridge_ports eth1

iface eth1 inet manual
        up ifconfig $IFACE 0.0.0.0    up
        up ifconfig $IFACE promisc

~
```

그리고 네트워킹 서비스 재시작을 수행한다.
/etc/init.d/networking restart

b. 운영모드에 따라 nova-network를 설치한다.

```
# apt-get install nova-network
```

그리고, 네트워크 관련 정보들을 설정해주기 위하여 /etc/nova/nova.conf를 수정한다.

```
# NETWORK
multi_host=True      //default는 single-host
send_arp_for_ha=True //multi-host 모드의 경우 arp 정보 브로드캐스트를 위해 활성화
update_dns_entries=True //multi-host모드의 경우, dns 정보 업데이트 활성화

network_manager=nova.network.manager.FlatDHCPManager //네트워크 매니저 설정

# Change my_ip to match each host
public_interface=eth4 //해당 노드의 인스턴스가 외부와 통신할 때 사용할 인터페이스
flat_network_bridge=br100 // 인스턴스들이 사용할 브릿지
flat_interface=eth1 // 인스턴스간 통신에 사용될 인터페이스
fixed_range='' //인스턴스가 생성될 때 할당받는 사설아이피대역(추후 설정가능)
```

c. 인스턴스들에 의해 사용될 사설 네트워크 대역 설정하기

```
# nova network-create private -fixed-range-v4=192.168.100.0/24 -bridge-interface=br100
```

'nova network-create' 이라는 명령어를 통해 인스턴스가 사용할 사설 네트워크 대역을 설정하게 되는 데, 설정하게 되면 앞서 nova.conf에서 공백으로 작성되었던 fixed_range가 결정되는 것이다. 이것은 디폴트로 single-host 모드가 적용되기 때문에, multi-host 모드를 사용하기 위해서는 옵션으로 '--multi-host=1'을 명령행에 적어줘야 한다.

참고문헌

[1]http://www.hastexo.com/resources/docs/installing-openstack-essex-4-ubuntu-1204-precise-pangolin/step-4-install-and-configure

[2] NFS 구축 : http://surai.tistory.com/entry/우분투에 NFS 구축

[3] OpenStack NFS 드라이버 : http://docs.openstack.org/trunk/openstack-block-storage/admin/content/NFS-driver.html

[4] OpenStack 기술문서 : OpenStack Intall and Deploy Manual-Ubuntu, 2013.1

[5] OpenStack 기술문서 : System Administration for OpenStack Block Storage

[6] OpenStack 기술문서 : OpenStack Virtual Machine Image Guilde, 2013.6

[7] OpenStack 관련 Q&A : https://lists.launchpad.net/openstack/msg20073.html

ISBN :
ISBN :