OpenStack Infra 구축

<Ubuntu16.04에서 DevStack 이용

및 SUSE Linux에서 Step by Step 방식>

ver. 1.0

2018년 11월 30일

목차

[1. 프로젝트 개요 3](#_Toc1321163)

[1-1. 프로젝트 명 3](#_Toc1321164)

[1-2. 프로젝트 기간 3](#_Toc1321165)

[1-3. 프로젝트 목적 3](#_Toc1321166)

[2. 프로젝트 범위 3](#_Toc1321167)

[2-1. 작업 명세서 3](#_Toc1321168)

[3. 프로젝트 추진체계 4](#_Toc1321169)

[3-1. 프로젝트 수행 인원 4](#_Toc1321170)

[3-2. 인원 별 역할 4](#_Toc1321171)

[4. 세부일정 4](#_Toc1321172)

[4-1. 상세 일정 4](#_Toc1321173)

[5. 개요 4](#_Toc1321174)

[5-1. Private Cloud와 Public Cloud 4](#_Toc1321175)

[5-2. OpenStack 7](#_Toc1321176)

[6. OpenStack 설치 방법 9](#_Toc1321177)

[6-1. CentOS에서의 설치 방법 9](#_Toc1321178)

[6-2. Ubuntu에서의 설치 방법 10](#_Toc1321179)

[6-3. All-in-one (Single Node), Multi-Nodes 설치 10](#_Toc1321180)

[7. Ubuntu16.04에서 DevStack을 이용한 Multi-Node 설치 10](#_Toc1321181)

[7-1. 구성도 11](#_Toc1321182)

[7-2. 설치 과정 12](#_Toc1321183)

[7-3. 설치 완료 및 Dashboard 접속 16](#_Toc1321184)

[8. SUSE Linux에서 Step by Step 방식으로OpenStack 설치 22](#_Toc1321185)

[8-1. 구성도 22](#_Toc1321186)

[8-2. SUSE Linux란? 22](#_Toc1321187)

[8-3. 설치 과정 23](#_Toc1321188)

[8-4. 인스턴스 실행 45](#_Toc1321189)

[8-5. 문제점 및 해결방안 49](#_Toc1321190)

1. **프로젝트 개요**

**1-1. 프로젝트 명**

Ubuntu16.04에서 DevStack 이용 및 SUSE Linux에서 Step by Step 방식으로 OpenStack Infra 구축

**1-2. 프로젝트 기간**

2018년 11월 19일 ~ 2018년 11월 30일

**1-3. 프로젝트 목적**

▷ Private Cloud와 Public Cloud에 대한 이해

▷ Cloud Computing Platform을 직접 구축 및 관리함으로써 Cloud에 대한 이해도 향상

▷ 시스템 문제 해결력 향상

1. **프로젝트 범위**

**2-1. 작업 명세서**

|  |  |
| --- | --- |
| 업무 | 업무범위 |
| Infra 구축 | * 환경설정 * Identity Service - Keystone * Image Service - Glance * Compute Service – Nova * Networking Service – Neutron * Dashboard - Horizon |
| 인스턴스 실행 및 테스트 | * CLI 방식 * Dashboard 이용 * 인스턴스 생성 및 실행 |

1. **프로젝트 추진체계**

**3-1. 프로젝트 수행 인원**



**3-2. 인원 별 수행 업무**

각 인원 모두 공통 인프라 구축 후 비교 검토

1. **세부일정**

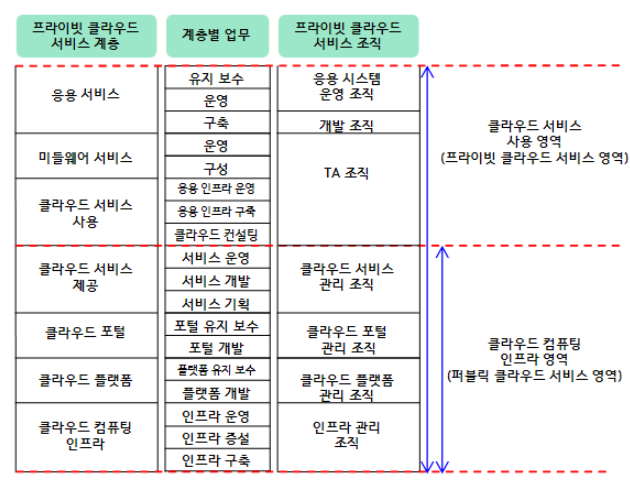
**4-1. 상세 일정**

상세 일정: 2018년 11월 19일 ~ 2018년 11월 30일

1. **개요**

**5-1. Private Cloud와 Public Cloud**

* **Private Cloud란?**

****

Private Cloud는 특정 사용자가 관리하는 조직 전용 시스템에서 제공하는 가상 리소스 풀로,

셀프 서비스 인터페이스를 통해 리소스를 자동으로 프로비저닝하고 할당할 수 있습니다.

Private Cloud는 위치나 소유권이 아니라, 개인 정보 설정과 관리 책임에 의해 정의됩니다. 독립

적인 액세스 권한을 갖는 단일 고객만 독점적으로 리소스를 사용할 수 있는 경우 온 사이트 또는

오프 사이트 인프라에서 Private Cloud를 지원할 수 있습니다.

Public Cloud와 Private Cloud에서 제공하는 서비스의 유형은 크게 다르지 않습니다. 클라우드

서비스를 제공하는 클라우드 컴퓨팅 인프라를 관리 운영하는 역할을 누가 관리하는가에 따라서

Public Cloud와 Private Cloud로 나뉩니다.

* **Private Cloud의 이점**

Private Cloud는 가상화 플랫폼의 확장이라는 수준을 넘어 다양한 컴퓨팅 리소스를 추상화하고

이러한 리소스에 대해 제어 가능한 셀프 서비스 액세스를 제공합니다. Private Cloud는 기존

가상화 기술을 능가하는 다음과 같은 이점을 제공합니다.

1. 인프라 용량이 증가되어 대량의 컴퓨팅 및 스토리지 요구 처리 가능
2. 셀프 서비스 사용자 인터페이스와 정책 기반 관리를 통한 온디맨드 서비스 제공
3. 사용자 요구에 따라 리소스를 효율적으로 할당
4. 인프라 전체의 리소스에 대한 가시성 향상

Private Cloud는 용량 활용률이 낮은 인스턴스를 줄입니다. 가상화 덕분에 리소스를 물리적으로

설치하는 데 있어 위치 제약이 없어지므로 기업은 원하는 방식으로 리소스를 자동으로 설정 및

재설정할 수 있습니다. 또한 Private Cloud는 조직의 보안 정책 및 적용 사례에 따라 다른

클라우드 옵션보다 강력한 보안을 제공할 수 있습니다.

* **Private Cloud 플랫폼 종류**

****

* **Public Cloud란?**

Public Cloud는 서비스 제공업체(벤더)가 구축한 서버, 스토리지 등의 IT 인프라를 기업들이

사용료를 내고 이용하는 방식을 의미합니다. 서비스 제공자가 자체 구축인지, 외부 인지에 따라

구별됩니다. Public Cloud는 아래와 같은 특징을 가집니다.

* 클라우드 서비스 제공자가 클라우드 서비스를 구축 및 관리하며 서비스 사용자를

대상 가입 식 서비스 방식을 사용.

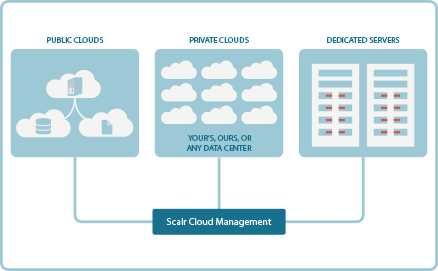
* 규모의 경제를 통한 서비스 액세스 비용 절감.
* 서비스 융통성 강화.
* 서비스 구현 속도 향상.
* **Hybrid Cloud란?**

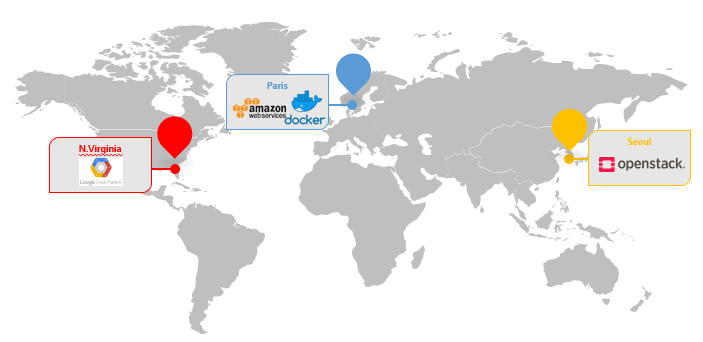
상호 운영 가능한 Public Cloud와 Private Cloud의 조합으로 구성되며, 일반적으로

사용자는 비즈니스에 중요하고 보안이 필요한 서비스와 데이터는 Private Cloud의 통제

하에 두고 덜 중요한 정보나 처리는 Public Cloud를 이용합니다. 기존 Public Cloud와

Private Cloud 서비스가 결합된 형태의 혼합형 Cloud 서비스를 말합니다.





<출처: <https://www.auro.io/business-cloud-hosting/hybrid-cloud>>

위 그림과 같이 Region별로 한쪽은 Google Cloud, 한쪽은 AWS/Docker, 다른 한쪽은

OpenStack 등을 이용하여 서로 다른 Region을 Hybrid 형태로 어떻게 잘 연계하면서 관리하는

그러한 요구사항에 대한 고민이 많이 필요합니다.

**5-2. OpenStack**

* **OpenStack이란?**

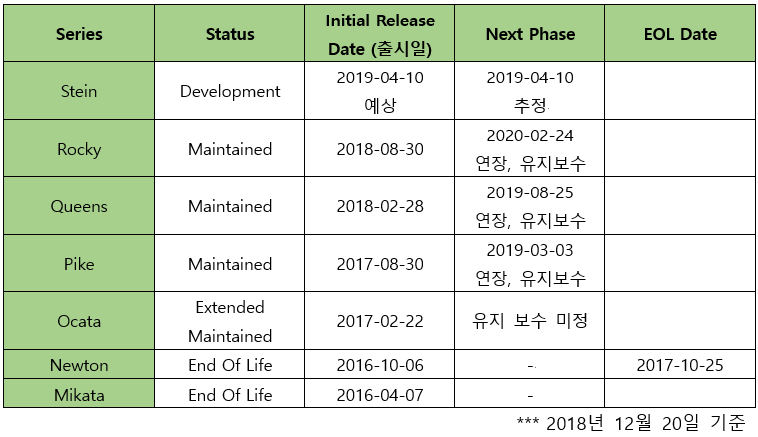
클라우드 컴퓨팅 플랫폼을 개발하고 관리할 수 있는 공개 소스 소프트웨어 기반의 클라우드

운영체제입니다. 서비스형 IaaS를 쉽게 구축할 수 있는 플랫폼으로, 관리자는 데이터센터의

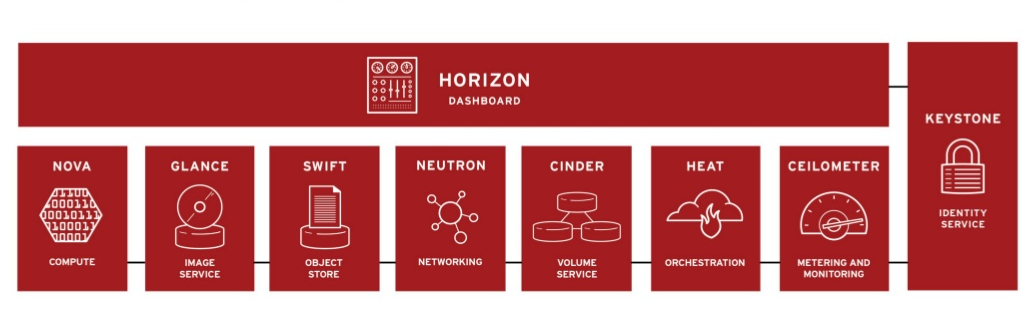
프로세싱, 스토리지, 네트워킹 자원들을 대시보드를 통해 제어할 수 있고, 사용자는 웹을 통해

필요한 기능을 사용할 수 있습니다.

**<OpenStack Release>**



* **OpenStack 구성요소**



<출처: https://sarc.io/index.php/create/406-openstack>

1. **Horizon (Dashboard)**

* OpenStack에서 일어나는 서비스에 대한 인터페이스를 웹 Base로 제공합니다.
* VM을 생성하거나 IP를 지정하는 등의 서비스를 CLI가 아닌 웹에서 사용자가 직접 실행할

수 있도록 도와주는 툴입니다.

1. **Keystone (Identity)**

* OpenStack에서 제공하는 모든 서비스에 대한 인증과 권한 부여를 담당합니다.
* 인증 외에도 사용자 관리, 보안 그룹 관리, 각종 서비스의 Endpoint URL (Rest API)을

관리하는 기능을 제공합니다.

* Keystone을 통하여 사용자가 해당 서비스를 실행할 권한이 있는지를 확인합니다.

1. **Nova (Compute)**

* 사용자의 요청을 VM 내부에서 변환시켜 Compute 서비스를 적용합니다.
* VM 생성, IP 연결 등의 작업을 실행하고 Compute 자원을 관리하는 역할을 합니다.
* 호스트 관리, 다수의 Hypervisor 지원

1. **Glance (Image)**

* VM 디스크 이미지를 저장하고 불러오는 역할을 합니다.
* Nova가 VM을 생성할 때 Glance에서 이미지를 불러와서 프로비저닝을 진행합니다.

1. **Neutron (Networking)**

* 네트워크를 가상화 시켜서 서비스를 제공하는 디바이스들을 연결합니다.
* 사용자가 독립적인 네트워크를 만들거나 원하는 디바이스만 선택해서 연결할 수도

있습니다.

1. **Cinder (Block Storage)**

* VM에 볼륨을 제공합니다. VM 생성 시 볼륨을 연결하거나 이미 생성된 VM에 볼륨을

연결할 수 있으며, 연결을 해제하여 다른 VM에 연결하는 것도 가능합니다.

1. **Ceilometer (Telemetry)**

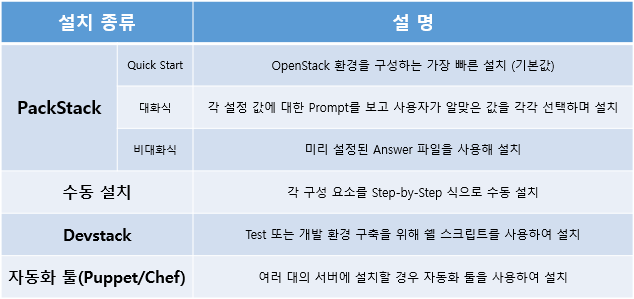
* OpenStack Cloud의 비용이나 통계정보를 모니터링/미터링하는 서비스입니다.

\*\*\* 이 외에도 용도에 따라 다양한 서비스를 제공하며, 사용자가 원하는 서비스를 선택해서

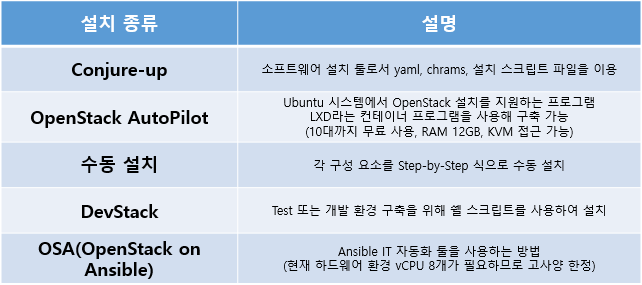
사용할 수 있습니다.

1. **OpenStack 설치 방법**

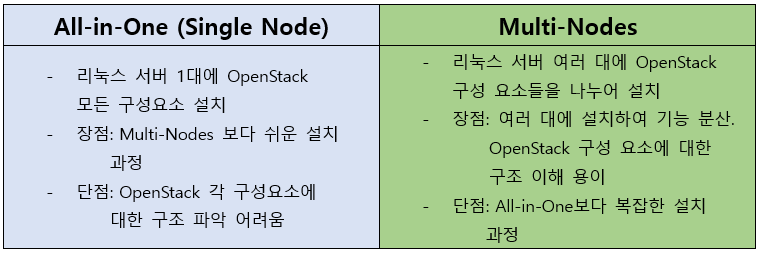
**6-1. CentOS에서의 설치 방법**



**6-2. Ubuntu에서의 설치 방법**



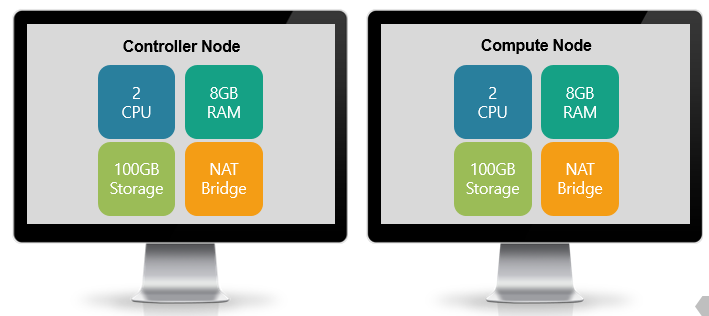
**6-3. All-in-one (Single Node), Multi-Nodes 설치**



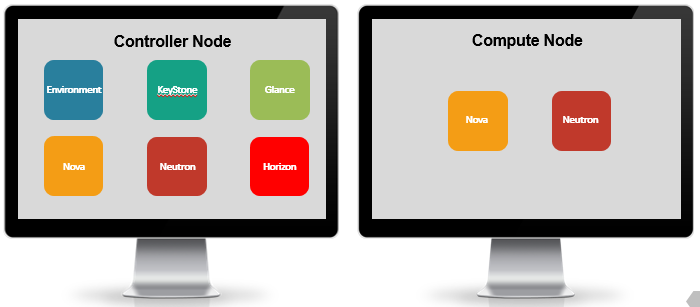
1. **Ubuntu16.04에서 DevStack을 이용한 Multi-Node 설치**

**7-1. 구성도**

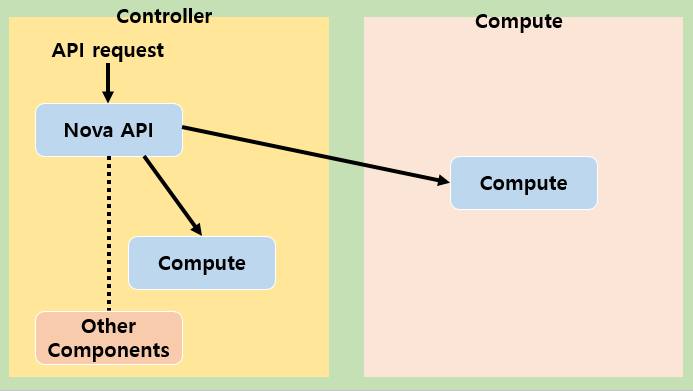
**< Controller Node / Compute Node 하드웨어 구성도 >**



**< Controller Node / Compute Node 서비스 >**



**< Installation Type >**

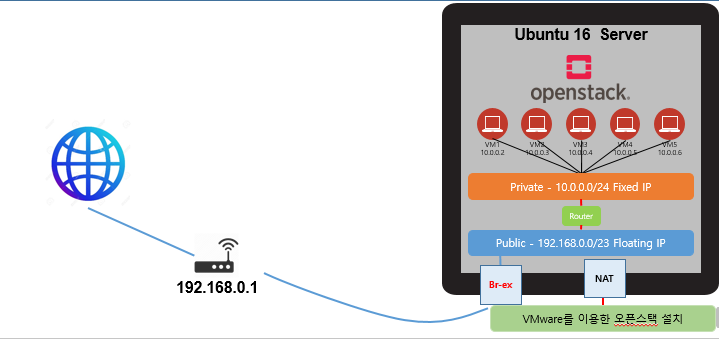


Controller Node가 작업에 대한 요청을 받았을 때 직접 Controller Node에서 작업을 수행할

수도 있지만 Compute Node에서 작업을 수행할 수도 있습니다.

Controller Node가 작업을 수행하도록 Controller Node에게 명령어를 보내서 수행하도록 합니다.

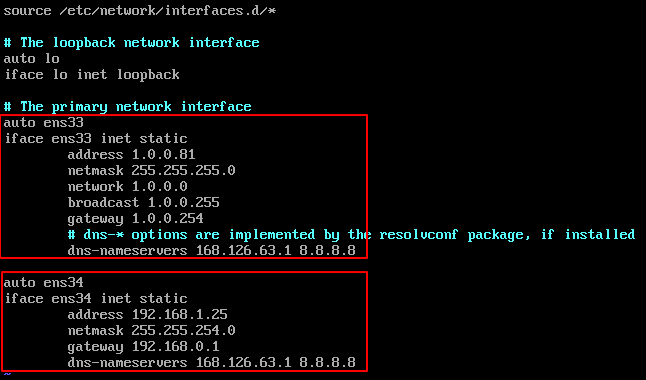
**< OpenStack 네트워크 구성도 >**



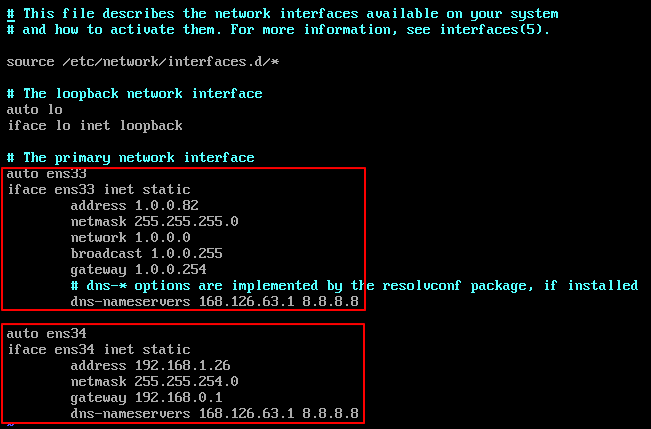
**7-2. 설치 과정**

* **Controller Node와 Compute Node 공통 설정**

1. **Network Setting**

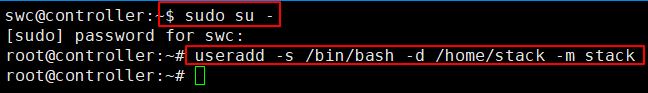


▷Controller Node에서 네트워크 설정을 해줍니다.



▷Compute Node에서 네트워크 설정을 해줍니다.

1. **Stack 계정 생성**



$ sudo su –

# useradd –s /bin/bash –d /home/stack –m stack

1. **Root 계정으로 암호 필요 없이 Stack 계정 권한 가져오게 하기**



# echo “stack ALL=(ALL) NOPASSWD: ALL” >> /etc/sudoers

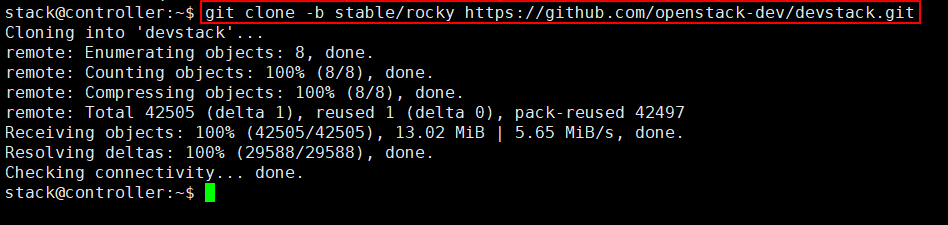
1. **Access를 위해 ssh 키를 이용하여 각 노드에서 stack 사용자를 설정**



# mkdir ~/.ssh; chmod 700 ~/.ssh

1. **DevStack Release별로 설치 파일 다운로드**

(만약 이 부분에서 Release를 지정하지 않으면 개발중인 Stein Release 다운로드)



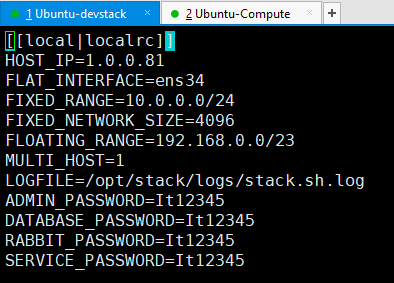
$ git clone –b stable/rocky https://github.com/openstack-dev/devstack.git

* **Controller Node**

1. **DevStack이 정상적으로 설치됐는지 확인**



1. **Local.conf 파일 수정**

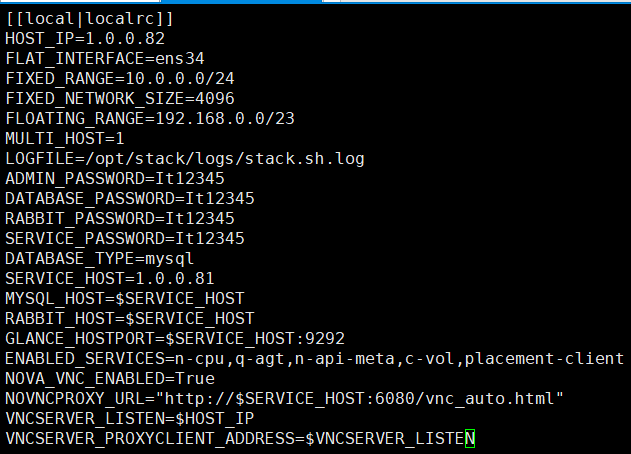


1. **./stack.sh 쉘 스크립트 실행**



* **Compute Node**

1. **Local.conf 파일**

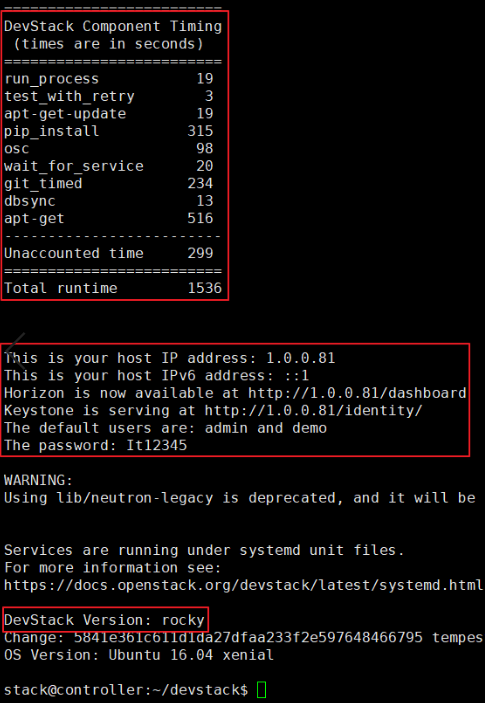


1. **./stack.sh 쉘 스크립트 실행**

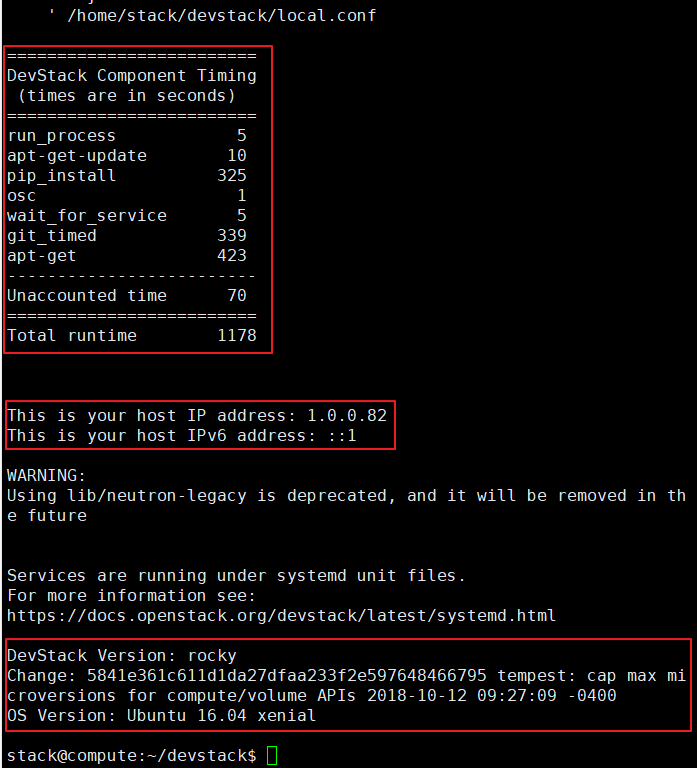


**7-3. 설치 완료 및 Dashboard 접속**

1. **설치 완료 후 Controller Node 화면**



1. **설치 완료 후 Compute Node 화면**

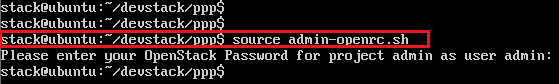


1. **설치 확인**

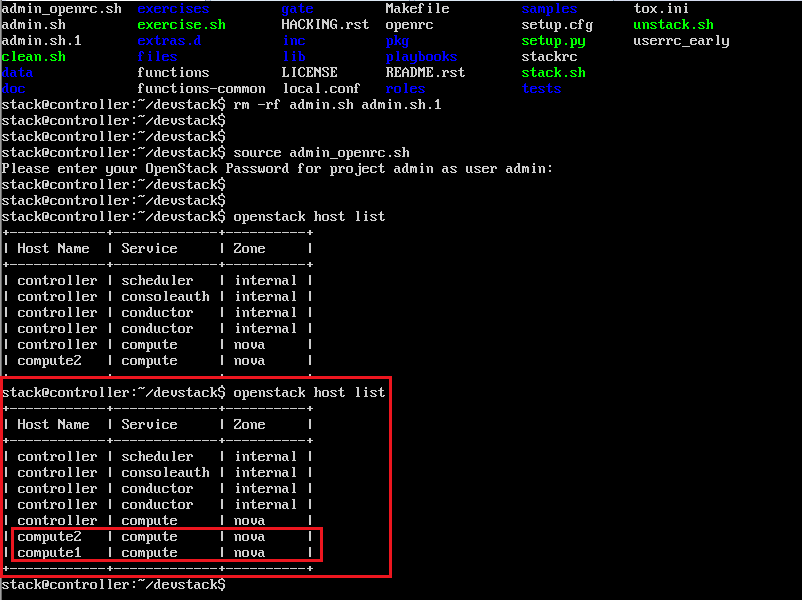


▷대시보드에서 admin 계정으로 접속하여 우측 상단의 계정 명을 클릭하여 [OpenStack RC File

v3] 파일을 다운받아서 OpenStack이 설치된 Ubuntu 가상 머신 안으로 옮겨줍니다.



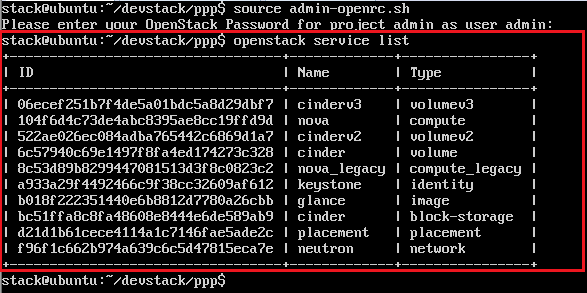
$ source admin-openrc.sh 명령어로 옮긴 스크립트 파일을 적용시킵니다.



▷Controller Node와 Compute Node가 모두 설치되었으면

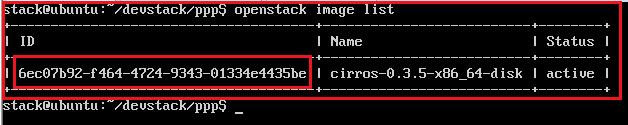
Controller Node에서 Compute Node가 잘 설치되었는지 확인합니다.

$ openstack host list



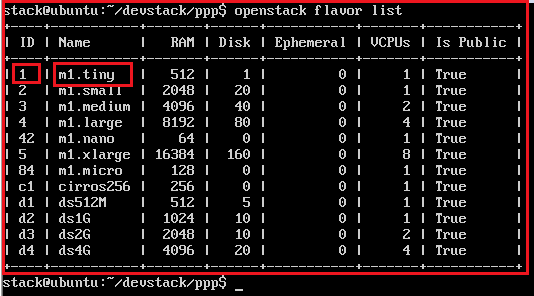
▷다음 명령어로 설치된 서비스를 확인할 수 있습니다.

$ openstack service list: 설치된 서비스 목록



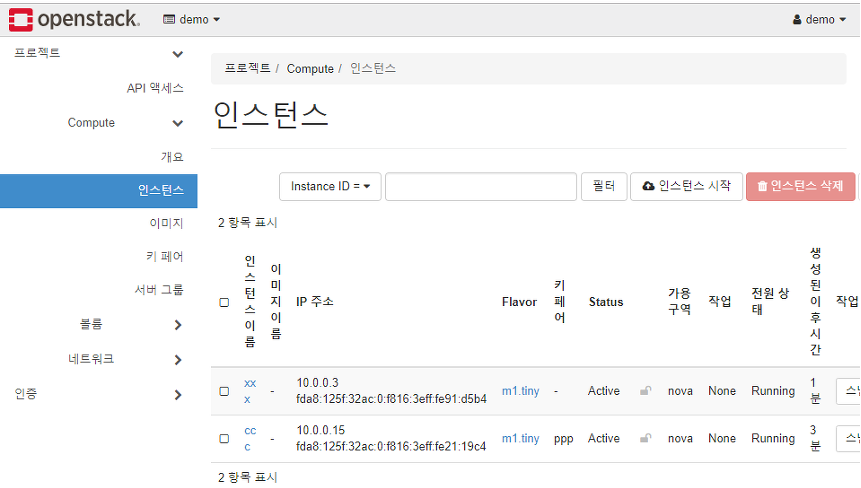
▷다음 명령어로 사용 가능한 image의 ID를 확인합니다.

$ openstack image list: 이미지 목록



▷다음 명령어로 flavor의 ID를 확인합니다.

$ openstack flavor list: flavor 목록



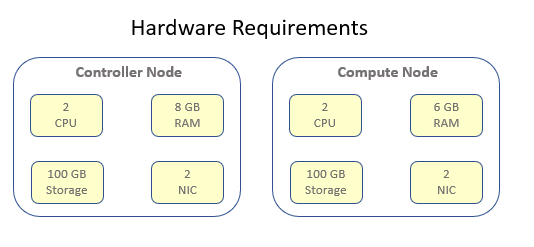
▷Controller Node에서 대시보드로 접근하여 인스턴스가 제대로 생성된다면 이상없이

설치되었다고 할 수 있습니다.

1. **SUSE Linux에서 Step by Step 방식으로OpenStack 설치**

**8-1. 구성도**

**< Controller Node / Compute Node 하드웨어 구성도 >**



Controller Node에 Identity 서비스, 이미지 서비스, Compute 관리, 네트워킹 관리 부분, 다양한

네트워킹 에이전트, 대시보드를 실행합니다.

Compute Node에는 인스턴스를 동작시키는 Hypervisor를 실행합니다.

**8-2. SUSE Linux란?**

* OpenSUSE

SUSE에서 판매하는 제품들인 SUSE Linux Enterprise Server와 SUSE Linux Enterprise Desktop의

기반으로 사용되고 있습니다. SUSE Linux의 선행 프로젝트로서 SUSE Linux의 개발사인 노벨 사의

후원을 받아 제작되었으며 상당히 안정적인 리눅스입니다. 긴 역사, 높은 안정성, 변화와 안정

사이의 균형 등의 특징으로 유럽에서 특히 높은 인기를 누리고 있습니다.

수세의 장점으로는 중앙집중관리인 YaST라는 관리 프로그램이 탑재되어 있으며, 타 리눅스에서

터미널에서만 가능하던 옵션을 GUI로 간단하게 설정할 수 있다는 점이 있습니다.

**8-3. 설치 과정**

* **환경 설정**

1. **보안**



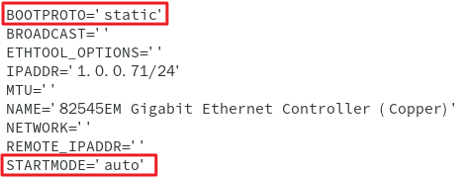
▷OpenStack 서비스에 대한 암호, 정책, 암호화를 포함한 다양한 보안 방법을 지원하고,

추가적으로 데이터베이스 서버와 암호 보안을 지원하는 브로커를 포함한 서비스를 지원합니다.

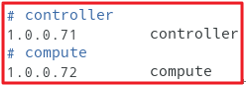
1. **호스트 네트워킹**

첫 번째 인터페이스(NAT)를 관리 인터페이스로 설정하고, 두 번째 인터페이스(Bridge)를

Provider 인터페이스로 사용합니다.



▷/etc/sysconfig/network/ifcfg-eth0 파일이 위의 표시된 내용을 포함하도록 설정합니다.



▷몇몇 배포 판은 127.0.1.1 과 같이 실제 호스트 이름을 다른 루프 백 IP주소로 변환하는

부가적인 항목을 추가합니다. 이름 변환 문제를 방지하기 위해 해당 항목을 삭제합니다.

127.0.0.1 항목은 삭제하면 안 됩니다.

▷설정을 완료하였다면, 인터넷과 Node들 간의 접근 테스트를 시행합니다.

1. **Network Time Protocol(NTP)**

Node 사이에 서비스들을 적절히 동기화하기 위해 NTP구현하는 Chrony를 설치합니다. Controller

Node가 보다 정확한 서버를 가리키도록 하고 다른 Node들은 Controller Node를 가리키도록

구성할 것을 권장합니다.

Controller Node와 Compute Node에서 # zypper install chrony 명령어로 설치해줍니다.



▷Controller Node의 /etc/chrony.conf 파일에 다른 Node가 Controller Node에서의 chrony 데몬에

접속하도록 활성화하기 위해 “allow 1.0.0.0/24”를 추가해줍니다.

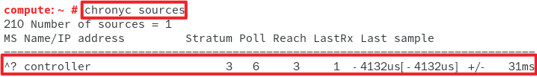
# systemctl enable chronyd.service

# systemctl start chronyd.service NTP Service를 시작합니다.



▷Compute Node의 /etc/chrony.conf 파일에 시계 동기화를 위해 Controller Node를 가리키도록

설정 후 NTP Service를 시작합니다.



▷검증을 위해 Controller Node와 Compute Node에 # chronyc sources 를 입력합니다.

1. **OpenStack Packages**

# zypper addrepo –f obs://Cloud.OpenStack:Rocky/openSUSE\_Leap\_42.3 Rocky

# zypper refresh && zypper dist-upgrade

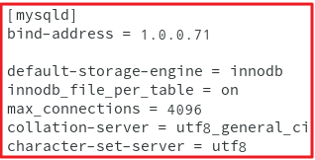
OpenStack 버전에 기반한 Open Build 서비스 저장소를 활성화하고 패키지를 업그레이드합니다.

# zypper install python-openstackclient

1. **SQL Database**

Controller Node에서 실행합니다.

# zypper install mariadb-client mariadb python-PyMySQL 패키지를 설치합니다.



▷/etc/my.cnf.d/openstack.cnf 파일 생성하여 편집합니다.

[mysqld] 섹션을 생성하고, 다른 Node들이 관리 네트워크를 통한 액세스를 활성화하기 위해

Controller Node의 관리 IP주소를 bind-address 키로 설정합니다. 유용한 옵션 및 UTF-8 문자셋을

활성화하기 위해 부가적인 키를 설정합니다.

# systemctl enable mysql.service

# systemctl start mysql.service

데이터베이스 서비스를 시작하고 시스템 부팅 시 시작하도록 설정합니다.

mysql\_secure\_installation 스크립트를 실행하여 데이터베이스 서비스 보안을 강화하고

데이터베이스 root 계정에 대해 알맞은 암호를 선택합니다.

1. **Message Queue**

Controller Node에서 실행합니다.

# zypper install rabbitmq-server

# systemctl enable rabbitmq-server.service

# systemctl start rabbitmq-server.service

서비스 간 작업과 상태 정보에 대한 상호 교환 및 조정을 위해 message queue를 사용합니다.

메시지 큐 서비스를 시작하고 시스템이 부팅될 때 시작하도록 구성합니다.





▷openstack 사용자를 추가하고, 구성, 쓰기와 읽기 접근을 허용합니다.

1. **Memcached**

서비스를 위한 Identity 서비스 인증 메커니즘에서는 토큰을 캐싱하기 위해 Memcached를 사용

합니다. Controller Node에서 실행합니다.

# zypper install memcached python-python-memcached



▷/etc/sysconfig/memcached 파일에 관리 IP 주소를 사용하는 서비스를 구성합니다.

관리 네트워크를 통해 다른 노드에 대한 액세스를 활성화하도록 구성합니다.

# systemctl enable memcached.service

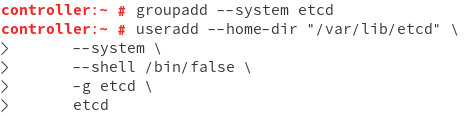
# systemctl start memcached.service

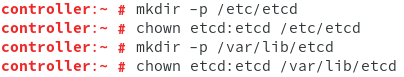
Memcached 서비스를 시작하고 시스템 부팅 시 시작하도록 설정합니다.

1. **Etcd**

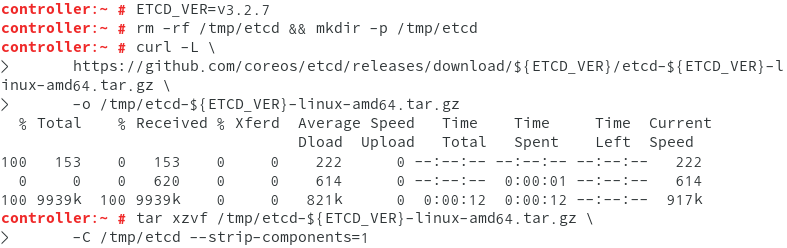
OpenStack 서비스들은 분산 키 잠금 관리, 구성 저장, 서비스가 살아있는지 및 다른 시나리오에

대한 지속적인 추적을 위한 안정적인 분산 키-값 저장소인 Etcd 사용합니다.





▷Controller Node에서 etcd 사용자를 생성하고 필요한 디렉토리를 생성합니다.



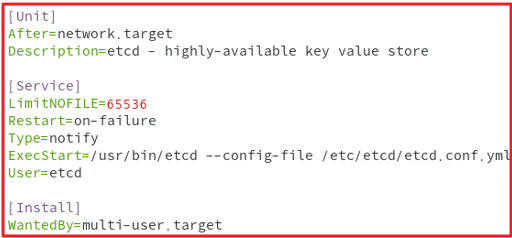


▷etcd tarball 다운로드 및 설치합니다.



▷/etc/etcd/etcd.conf.yml 생성하여 관리 네트워크를 통해 다른 Node에 대한 액세스를 활성화

하도록 Controller Node에 대한 관리 IP 주소로 설정합니다.



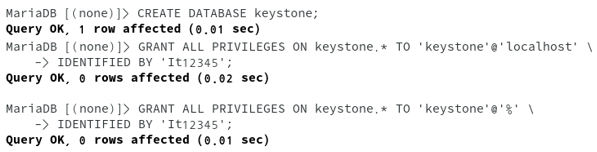
▷/usr/lib/systemd/system/etcd.service 생성 및 편집합니다.

# systemctl daemon-reload && systemctl enable etcd && systemctl start etcd

* **Identity Service – Keystone**

1. **전제 조건**

Identity 서비스를 설치하고 구성하기 전에 데이터베이스를 만들어야 합니다.

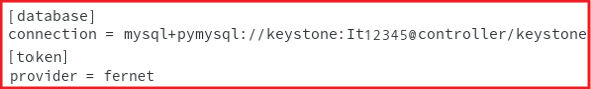


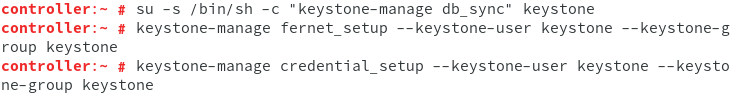
$ mysql –u root –p 명령어를 사용하여 root 사용자로 데이터베이스 서버에 연결합니다.

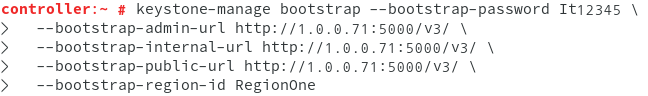
Keystone 데이터베이스를 생성하고 적절한 액세스 권한을 부여합니다.

1. **구성 요소 설치 및 구성**

# zipper install openstack-keystone 명령어를 실행하여 패키지를 설치합니다.



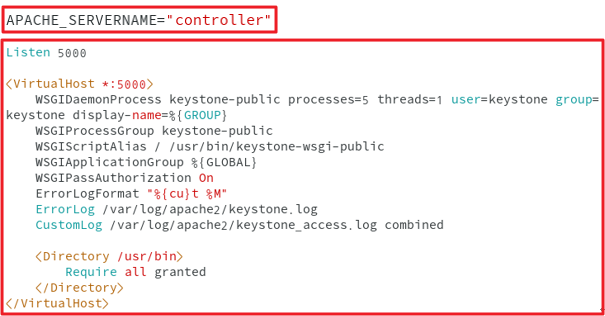




▷/etc/keystone/keystone.conf 파일을 편집하고 위와 같이 구성합니다.

1. **Apache HTTP 서버 구성**

# zypper install apache && zypper install apache2-mod\_wsgi 패키지를 설치해줍니다.

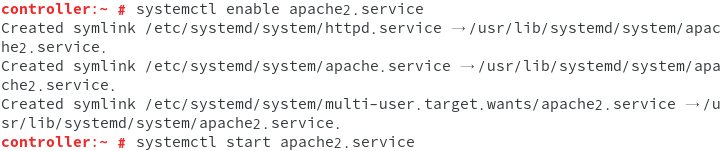




▷/etc/sysconfig/apache2 파일과 /etc/apache2/conf.d/wsgi-keystone.conf 파일을 각각 편집하고,

/etc/keystone 디렉토리의 소유권을 변경합니다.

1. **설치**



▷Apache HTTP 서비스를 시작하고 시스템이 부팅될 때 시작되도록 구성합니다.

1. **도메인, 프로젝트, 사용자 및 역할 만들기**

사용자 환경에 추가하는 각 서비스에 대해 고유한 사용자를 포함하는 프로젝트(service)와,

권한이 없는 프로젝트와 사용자를 사용해야 하는 일반 작업을 위한 프로젝트(myproject) 및

사용자(myuser)를 생성합니다.

$ openstack project create –domain default –description “Service Project” service

$ openstack project create –domain default –description “Demo Project” myproject

$ openstack user create –domain default –password-prompt myuser

$ openstack role create myrole

$ openstack role add –project myproject --user myuser myrole

1. **작동 확인**

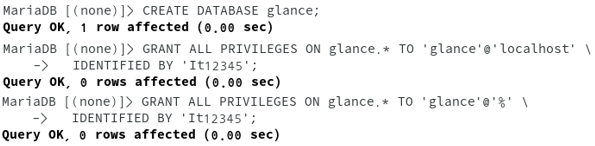


▷임시 변수 OS\_AUTH\_URL 및 OS\_PASSWORD 환경 변수 설정을 해제합니다.

admin 사용자로서 인증 토큰을 요청하고, myuser 사용자로서 인증 토큰을 요청합니다.

* **Image Service - glance**

1. **전제 조건**



▷glance 데이터베이스를 만들고 적절한 액세스 권한을 부여합니다.

$ . admin-openrc

$ openstack user create --domain default --password-prompt glance

$ openstack role add –project service --user glance admin

$ openstack service create --name glance --description “OpenStack Image” image

서비스 자격 증명을 위해 glance 사용자를 생성하고 admin 역할을 추가 후 entity를 생성합니다.



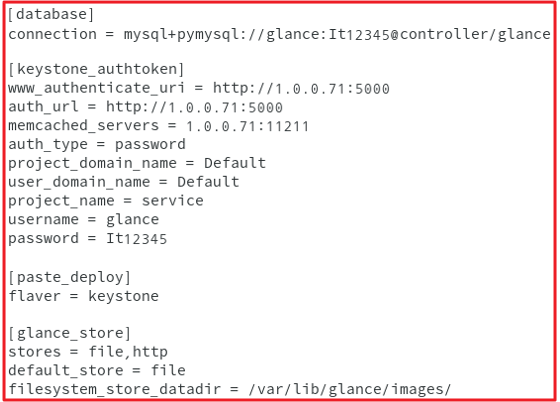
▷위와 같이 Image Service API Endpoint를 작성합니다.

$ openstack endpoint create --region RegionOne image internal http://controller:9292

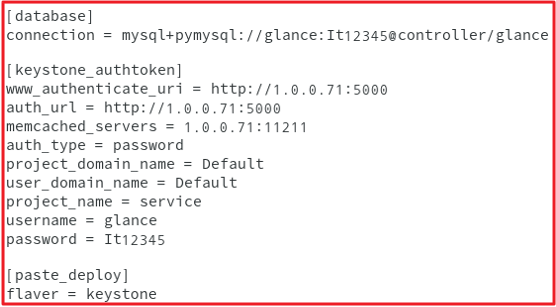
$ openstack endpoint create --region RegionOne image admin http://controller:9292

1. **구성 요소 설치 및 구성**

# zypper install openstack-glance openstack-glance-api openstack-glance-registry



▷/etc/glance/glance-api.conf 파일을 편집합니다.



▷/etc/glance/glance-registry.conf 파일을 편집합니다.



▷Image Service를 시작하고 시스템이 부팅될 때 시작되도록 구성합니다.

1. **작동 확인**

cirros를 사용하여 Image Service의 작동을 확인합니다.

$ . admin-openrc

$ wget <http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img>

$ openstack image create “cirros” --file cirros-0.4.0-x86\_64-disk.img \

--disk-format qcow2 --container-format bare --public

모든 프로젝트가 접근할 수 있도록 qcow2 디스크 포맷, 노출된 컨테이너 형식 등을 사용하여

Image Service에 Image를 업로드합니다.

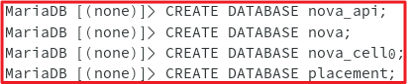


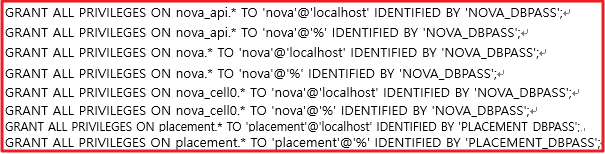
▷Image 업로드 확인 및 속성 유효성 검사를 합니다.

* **Compute Service - nova**

<Controller node 설치 및 구성>

1. **Controller node 전제 조건**





▷데이터베이스를 생성하고 적절한 액세스 권한을 부여합니다.

$ . admin-openrc

$ openstack user create --domain default --password-prompt nova

$ openstack role add --project service --user nova admin

$ openstack service create --name nova --description "OpenStack Compute" compute

Compute 서비스 자격 증명을 만듭니다.



▷위와 같이 Compute API 끝점을 만듭니다.

$ openstack endpoint create --region RegionOne compute internal <http://controller:8774/v2.1>

$ openstack endpoint create --region RegionOne compute admin <http://controller:8774/v2.1>

$ openstack user create --domain default –password-prompt placement

$ openstack role add --project service –user placement admin

사용자를 만들고 관리 역할이 있는 서비스 프로젝트에 사용자를 추가합니다.

$ openstack service create --name placement --description “Placement API” placement

$ openstack endpoint create --region RegionOne placement public http://controller:8780

$ openstack endpoint create --region RegionOne placement internal http://controller:8780

$ openstack endpoint create --region RegionOne placement admin http://controller:8780

서비스 카탈로그에 Placement API 항목을 만들고 Placement API 서비스 끝점을 생성합니다.

1. **Controller node 구성 요소 설치 및 구성**

# zypper install openstack-nova-api openstack-nova-scheduler openstack-nova-conductor \

openstack-nova-consoleauth openstack-nova-novnproxy openstack-nova-placement-api iptables

패키지를 설치합니다.



▷위와 같이 /etc/nova/nova.conf 파일을 편집합니다.

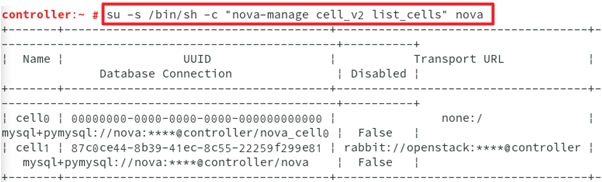
# su -s /bin/sh -c “nova-manage api\_db sync” nova

# su -s /bin/sh -c “nova-manage cell\_v2 map\_cell0” nova

# su -s /bin/sh -c “nova-manage cell\_v2 create\_cell --name=cell1 --verbose” nova [UUID]

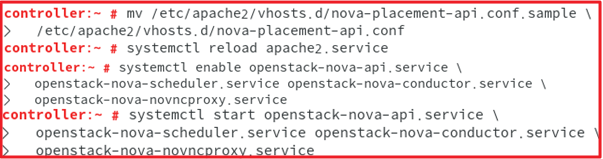
# su -s /bin/sh -c “nova-manage db sync” nova

nova-api와 placement 데이터베이스를 채우고 cell0, cell1 데이터베이스 등록 및 생성합니다.



▷nova cell0 및 cell1이 올바르게 등록되어 있는지 확인합니다.

1. **Controller node 설치**



▷Compute 서비스를 시작하고 시스템이 부팅될 때 시작되도록 구성합니다.

<Compute node 설치 및 구성>

1. **Compute node 구성 요소 설치 및 구성**

# zypper install openstack-nova-compute genisoimage qemu-kvm libvirt

패키지를 설치합니다.



▷위와 같이 /etc/nova/nova.conf 파일을 편집하고

# modprobe nbd로 커널 모듈 nbd가 로드 되었는지 확인합니다.

1. **Compute node 설치**



▷Compute node가 가상 컴퓨터의 하드웨어 가속을 지원하는지 여부를 결정합니다.

만약 0을 반환하면 /etc/nova/nova.conf 파일의 [libvirt] 섹션에서 virt\_type을 qemu로 수정합니다.

Compute 서비스를 시작하고 시스템 부팅 시 자동으로 시작되도록 구성합니다.

1. **Cell 데이터베이스에 Compute node 추가**

$ openstack compute service list --service nova-compute

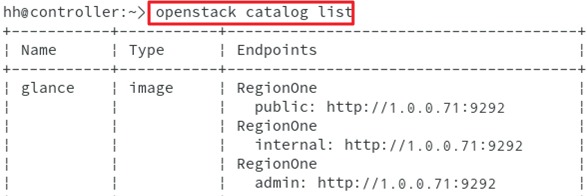
# su -s /bin/sh -c “nova-manage cell\_v2 discover\_hosts --verbose” nova

데이터베이스에 컴퓨팅 호스트가 있는지 확인하고 컴퓨팅 호스트를 찾습니다.

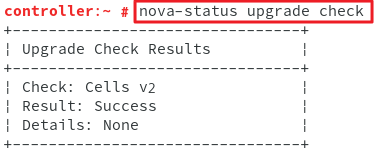
1. **Controller node에서 작동 확인**

$ openstack compute service list

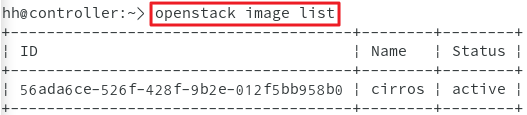
consoleauth, scheduler, conductor 및 compute 서비스 구성 요소를 나열하여 확인합니다.



▷Identity Service의 API endpoint를 나열하여 Identity Service와의 연결을 확인합니다.



▷Cell 및 Placement API가 제대로 작동하는지 확인합니다.

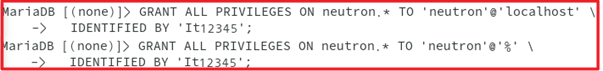


▷image service의 image를 나열하여 image service와의 연결을 확인합니다.

* **Networking Service - neutron**

<Controller node 설치 및 구성>

1. **Controller node 전제 조건**



▷neutron 데이터베이스를 생성하고 적절한 액세스 권한을 부여합니다.

또한 neutron 사용자를 생성하고 admin 역할을 추가합니다.

$ openstack service create --name neutron --description “OpenStack Networking” network

neutron service entity를 생성합니다.



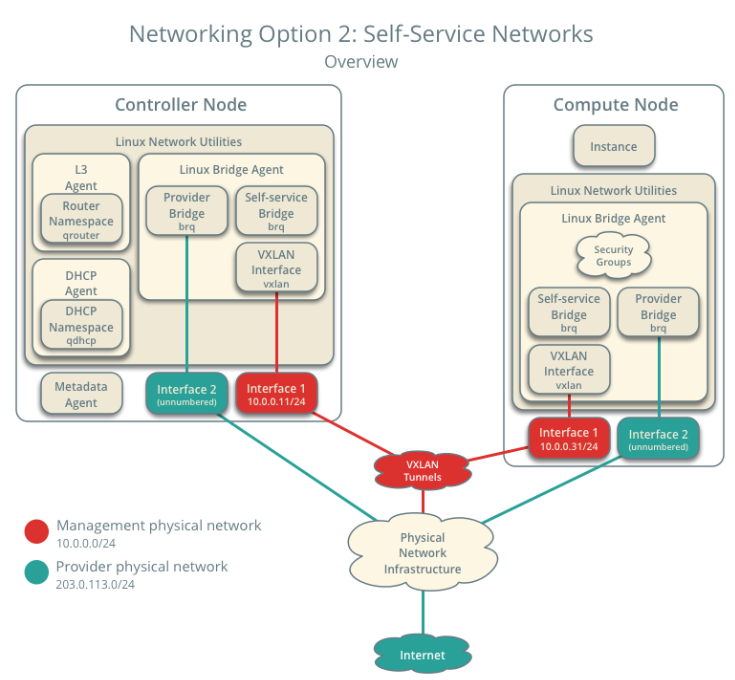
▷위와 같이 Networking Service API endpoint를 작성합니다.

$ openstack endpoint create --region RegionOne network internal http://controller :9696

$ openstack endpoint create --region RegionOne network admin http://controller :9696

1. **Controller node Networking Option 2: Self-service networks**

<**Self-Service Networks>**



XVLAN과 같은 overlay segmentation 방식을 사용하여 self-service 네트워크를 가능하게 하는

Layer-3 (routing) 서비스를 포함하는 provider 네트워크를 확장합니다.

근본적으로는, 가상 네트워크를 NAT를 사용하여 물리 네트워크로 라우팅합니다.

/etc/neutron/neutron.conf 파일을 편집하여 서버 구성 요소를 구성합니다.

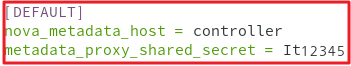
/etc/neutron/plugins/ml2/ml2\_conf.ini 파일을 편집하여Modular Layer 2 플러그 인을 구성합니다.

/etc/neutron/plugins/ml2/linuxbridge\_agent.ini 파일을 편집하여 Linux Bridge agent를 구성합니다.

/etc/neutron/l3\_agent.ini 파일을 편집하여 Layer-3 agent를 구성하고,

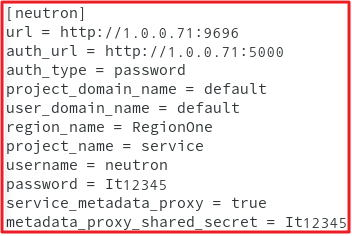
마지막으로 /etc/neutron/dhcp\_agent.ini 파일을 편집하여 DHCP agent를 구성합니다.

1. **Controller node metadata agent 구성**



▷/etc/neutron/metadata\_agent.ini 파일을 위와 같이 편집합니다.

1. **Networking Service를 사용하도록 Compute Service 구성**



▷/etc/nova/nova.conf 파일에서 액세스 매개 변수를 구성하고 metadata proxy를 활성화합니다.

1. **설치**



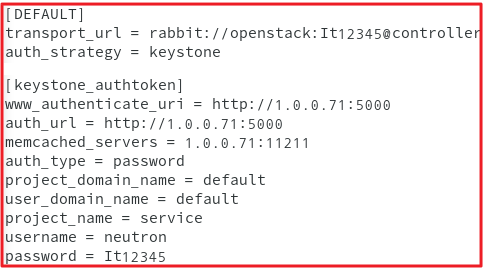
▷SUSE는 기본적으로 apparmor를 활성화하고 dnsmasq를 제한합니다. apparmor를 완전히

비활성화하거나 dnsmasq 프로필만 비활성화해야 합니다.

마지막으로 Compute API Service, Networking Service, Layer-3 Service를 시작합니다.

<Compute node 설치 및 구성>

1. **Compute node 공통 구성 요소 구성**



# zypper install --no-recommends openstack-neutron-linuxbridge-agent bridge-utils

먼저 구성 요소를 설치합니다.

그리고 /etc/neutron/neutron.conf 파일을 위와 같이 편집하고

추가로 [oslo\_concurrency] 섹션에서 lock\_path = /var/lib/neutron/tmp로 수정합니다.

1. **Compute node Networking Option 2: Self-service networks**

/etc/neutron/plugins/ml2/linuxbridge\_agent.ini 파일을 편집하여 Linux Bridge agent를 구성합니다.



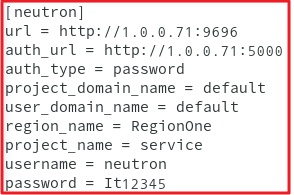
▷위의 sysctl 값이 1로 모두 설정되어 있는지 확인하여 Linux 운영 체제 커널이 Network Bridge

filter를 지원하는지 확인합니다.

네트워킹 브리지 지원을 사용하려면 br\_netfilter 커널 모듈을 로드해야 합니다.

# modprobe br\_netfilter 명령어로 커널 모듈을 로드하고 # sysctl -a 명령어로 확인합니다.

1. **Networking Service를 사용하도록 Compute Service 구성**



▷위와 같이 /etc/nova/nova.conf 파일을 편집합니다.

1. **설치**

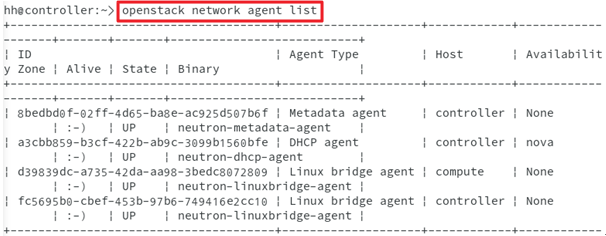


▷/etc/sysconfig/neutron 파일에 위의 내용이 포함되어 있는지 확인하고,

Compute Service와 Linux Bridge agent를 시작합니다.

1. **Controller node에서 작동 확인**

$ openstack extension list --network 명령어로 neutron-server 프로세스가 실행됐는지 확인합니다.



▷neutron agent가 성공적으로 시작되었는지 확인합니다.

* **Dashboard – horizon**

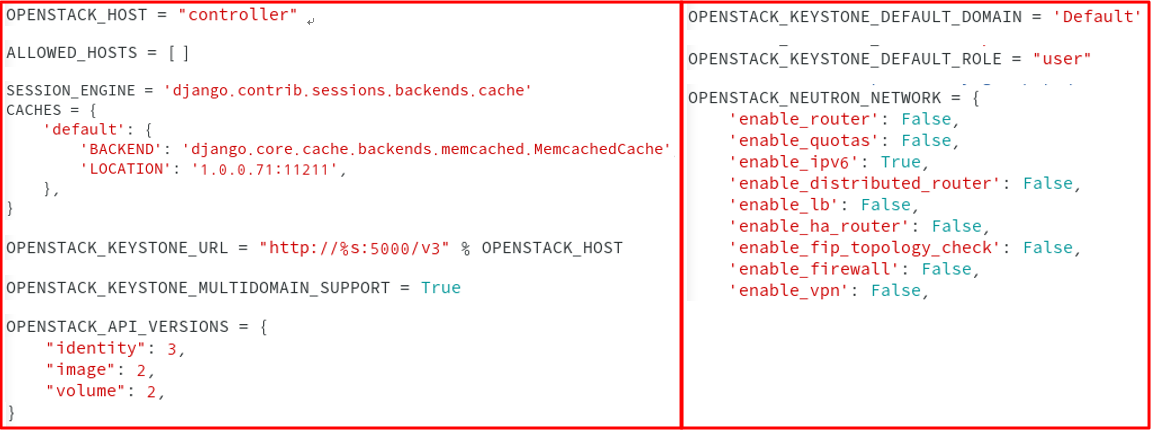
1. **구성 요소 설치 및 구성**

# zipper install openstack-dashboard

# cp /etc/apache2/conf.d/openstack-dashboard.conf.sample \

/etc/apache2/conf.d/openstack-dashboard.conf

패키지를 설치하고 웹 서버를 구성합니다.



▷/srv/www/openstack-dashboard/openstack\_dashboard/local/local\_settings.py 파일을 편집합니다.

1. **설치**

# systemctl restart apache2.service memcached.service

웹 서버 및 세션 저장 서비스를 다시 시작합니다.

1. **동작 확인**

<http://controller/> 웹 브라우저를 사용하여 대시보드에 액세스합니다.

Admin 또는 demo 사용자 및 default 도메인 자격 증명을 사용하여 인증합니다.

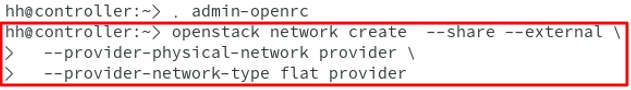
**8-4. 인스턴스 실행**

인스턴스를 실행하기 위해서는 가상 네트워크, Flavor, Key Pair, 보안 그룹을 먼저 생성해야

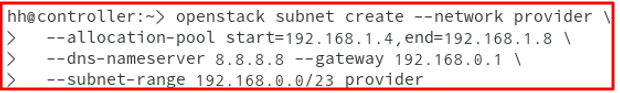
합니다. CLI 환경과 대시 보드에서 모두 생성 가능합니다.

1. **가상 네트워크 생성**

Networking Option 2를 선택한 경우 Provider와 Self-service Network를 모두 생성해야 합니다.



▷먼저 Provider 네트워크를 생성합니다.

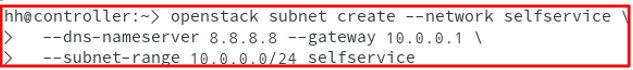


▷네트워크에 Subnet을 생성합니다.

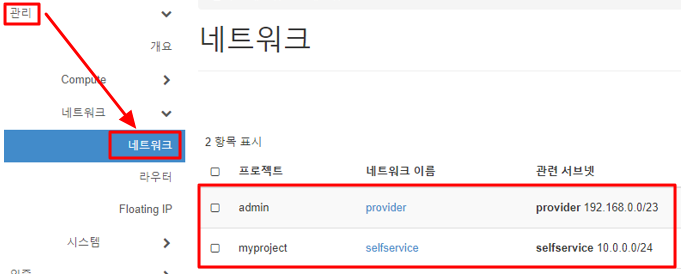
이제 NAT를 통해 실제 네트워크 인프라에 연결하는 Self-service Network를 생성합니다.



▷selfservice 네트워크를 생성합니다.



▷네트워크에 Subnet을 생성합니다.



▷Dashboard의 관리 > 네트워크 항목에 2개의 네트워크가 생성된 것을 확인할 수 있습니다.



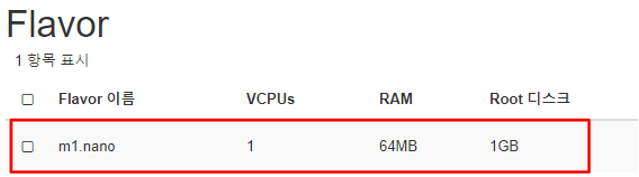
▷selfservice 네트워크를 공유 상태로 편집합니다.

1. **Router 생성**

외부 네트워크를 provider 네트워크로 지정하여 router를 생성합니다.



1. **Flavor 생성**



▷Flavor가 저장 공간에 비해 크게 설정될 경우 인스턴스 생성이 불가합니다.

1. **Key Pair 생성**

대부분의 Cloud Image는 기존 암호 인증보다는 공개 키 인증을 지원합니다.

1. **보안 그룹 규칙 추가**

기본적으로 default 보안 그룹은 모든 인스턴스에 적용되며 인스턴스에 대한 원격 액세스를

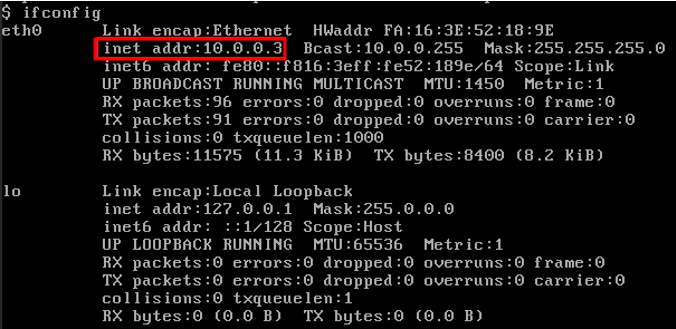
거부하는 방화벽 규칙을 포함합니다.

1. **Floating IP 할당**

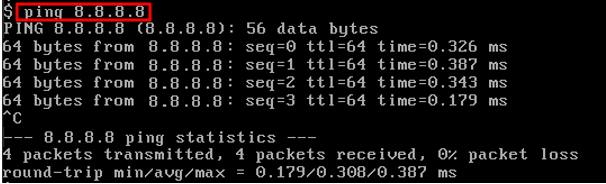


▷지정한 Subnet 범위 내에서 Floating IP가 할당됩니다.

1. **인스턴스 생성 및 실행**



▷생성된 인스턴스 콘솔로 접속합니다.



▷정상적으로 외부와 통신 되는지 확인합니다.

**8-5. 문제점 및 해결방안**

1. **RabbitMQ Server 실행 오류**



Message Queue Service인 RabbitMQ 설치 후 RabbitMQ Server가 실행되지 않는 오류입니다.

* **해결방안**

/etc/hosts에서 Loopback Interface Name과 Hostname이 일치하지 않을 경우 RabbitMQ Server가

실행되지 않습니다. Loopback Interface와 Hostname을 일치시켜 주면 정상적으로 실행됩니다.

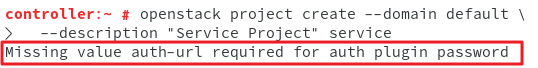
1. **Apache2 설치 오류**

Apache2 패키지를 설치할 때 mod\_wsgi 패키지가 같이 설치되지 않아서 오류가 발생합니다.

* **해결방안**

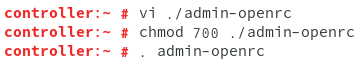
zypper install apache2-mod\_wsgi

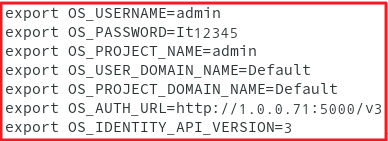
1. **환경변수 오류**



auth plugin password에 필요한 값인 auth-url 값이 누락되어 발생하는 오류입니다.

* **해결방안**







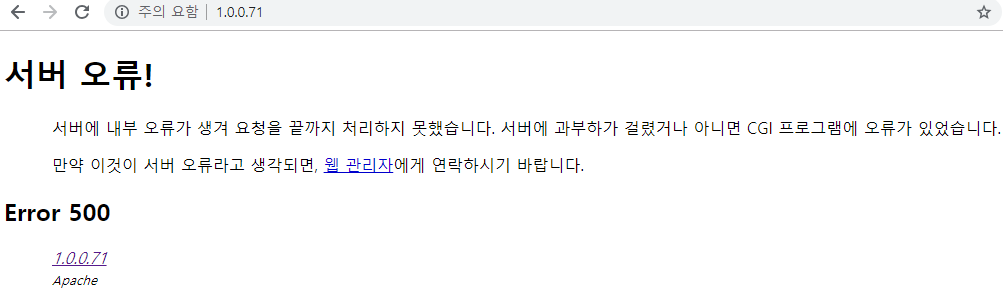
계속 쓰게 될 환경변수 이므로 스크립트로 만들어 실행시켜줍니다.

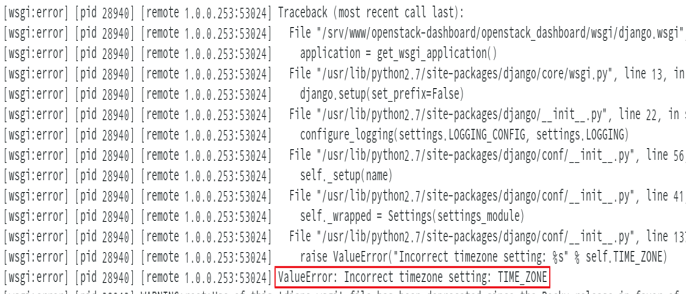
1. **Database 변경사항 적용**

Database에 값을 넣었지만 아무 값이 안 들어가는 경우

MariaDB [keystone]> flush privileges; 명령어를 실행하여 적용시켜 줍니다.

1. **Dashboard 접속 오류\_(1)**



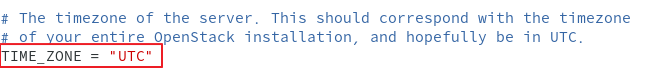


Horizon 실행 후 대시보드로 접속이 안 되는 오류입니다.

vi /var/log/apache2/openstack-dashboard-error\_log

로그 파일에서 에러 확인이 가능하며, Time zone setting이 맞지 않아 발생하였습니다.

* **해결방안**



vi /srv/www/openstack-dashboard/openstack\_dashboard/local/local\_settings.py

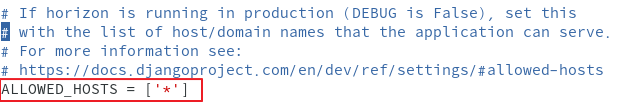
협정 세계 표준시인 UTC로 변경해줍니다.

1. **Dashboard 접속 오류\_(2)**



잘못된 HTTP\_Host header로 인해 Horizon 실행 후 대시보드 접속이 안 되는 오류입니다.

* **해결방안**

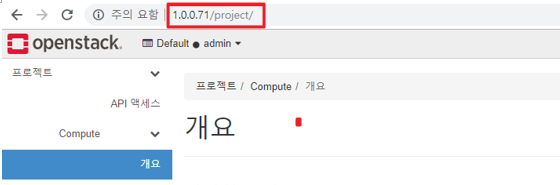


vi /srv/www/openstack-dashboard/openstack\_dashboard/local/local\_settings.py

모든 HOST가 접속 가능하도록 설정하고,

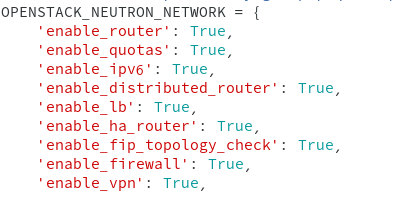
만약 특정 호스트만 접속이 가능하도록 하고 싶다면 IP를 지정해주어도 됩니다.

1. **Router 활성화**



대시보드 접속은 가능하나, 네트워크 라우터 설정 탭이 보이지 않는 오류입니다.

* **해결방안**



vi /srv/www/openstack-dashboard/openstack\_dashboard/local/local\_settings.py

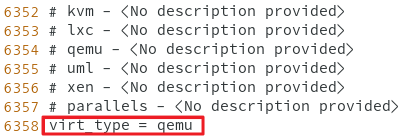
위의 설정들을 모두 True로 변경하고,

설정 변경 후엔 apache2.service, memcached.service 서비스들을 재 시작합니다.

1. **인스턴스 부팅 오류**

인스턴스 생성 후 가상 머신 타입이 kvm을 지원하지 않아, 인스턴스 부팅이 안 되는 오류입니다.

* **해결방안**



virt\_type = qemu로 변경합니다.