

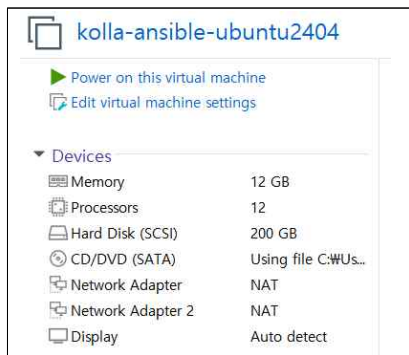
kolla-ansible을 이용한 오픈스택 1 노드 실습

Quick Start for deployment/evaluation

편의상 root 로 진행하였음

1. 사전 준비

- ubuntu 24.04 desktop (noble) > python 버전 3.10 확인해야 함(다른 버전 오류 많음)
- vmware workstation
- 실습 최소 사양 :
 - 2 network interfaces
 - 8 GB 메모리
 - 40 GB 디스크 공간



* 실습용 노트북에 여유가 있어 위와 같이 준비하였으며 CPU 는 VT-x/AMD-v 체크해 두었음

2. OS 설치 후 해야할 일

IP 구성하기

- ens32 은 api 및 관리용
- ens33 은 외부 연결용(bridge 로 구성되어 내부 인스턴스들과 연결됨)

```
root@openstack:~# cat /etc/netplan/01-network-manager-all.yaml
```

```
# Let NetworkManager manage all devices on this system
```

```
network:
```

```
  version: 2
```

```
  renderer: networkd
```

```
  ethernets:
```

```
    ens32:
```

```
      addresses:
```

- 211.183.3.100/24

```
      gateway4: 211.183.3.2
```

```
      nameservers:
```

```
        addresses:
```

- 8.8.8.8

```
ens33:
    dhcp4: no
root@openstack:~# netplan apply
root@openstack:~# systemctl enable systemd-networkd --now

root@openstack:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:6e:0f:3d brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 211.183.3.100/24 brd 211.183.3.255 scope global ens32
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe6e:f3d/64 scope link
        valid_lft forever preferred_lft forever
3: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:6e:0f:47 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet6 fe80::20c:29ff:fe6e:f47/64 scope link
        valid_lft forever preferred_lft forever
root@openstack:~#
```

방화벽과 ssh

```
root@openstack:~# systemctl daemon-reload
root@openstack:~# systemctl disable ufw --now
root@openstack:~#
root@openstack:~# vi /etc/ssh/sshd_config
41 #LoginGraceTime 2m
42 PermitRootLogin yes
43 #StrictModes yes

root@openstack:~# systemctl restart ssh
```

3. 의존성 패키지 설치

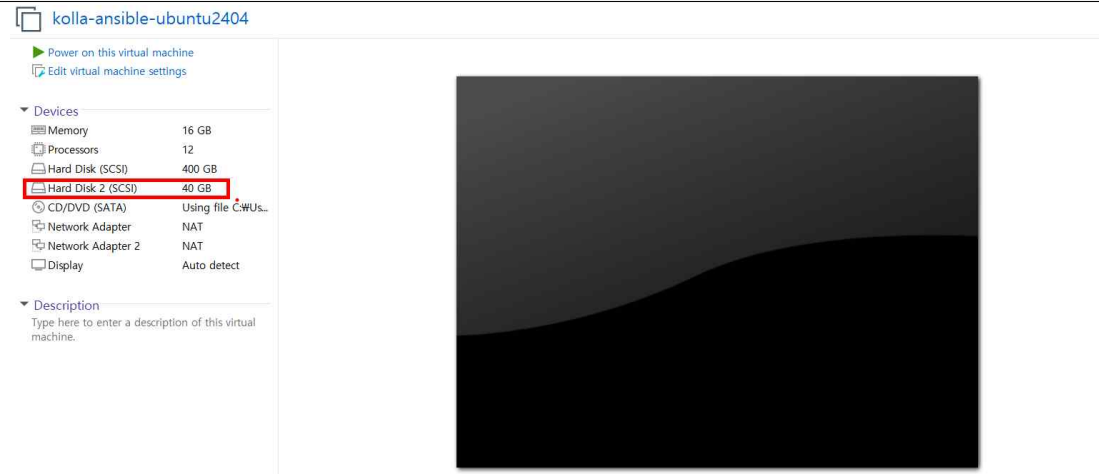
```
root@openstack:~# apt update
# 파이썬 빌드 의존성 패키지 설치
root@openstack:~# apt install git python3-dev libffi-dev gcc libssl-dev
libdbus-glib-1-dev python3-dbus libdbus-1-dev -y
```

4. 가상환경을 위한 의존성 패키지 설치 및 cinder 볼륨 준비

```
root@openstack:~# apt install python3-venv -y
root@openstack:~# python3 -m venv /root/venv --system-site-packages
root@openstack:~# source /root/venv/bin/activate
(venv) root@openstack:~#
```

```
# 최신 버전의 pip 설치
(venv) root@openstack:~# pip install -U pip
```

cinder 볼륨을 위한 추가 디스크 부착



```
(venv) root@openstack:~# apt update && apt install -y lvm2
(venv) root@openstack:~# ls /dev/sd*
/dev/sda /dev/sda1 /dev/sda2 /dev/sdb
(venv) root@openstack:~#
(venv) root@openstack:~# pvcreate /dev/sdb
Physical volume "/dev/sdb" successfully created.
(venv) root@openstack:~# vgcreate cinder-volumes /dev/sdb
Volume group "cinder-volumes" successfully created
(venv) root@openstack:~#
(venv) root@openstack:~# vgdisplay -s
"cinder-volumes" <40.00 GiB [0          used / <40.00 GiB free]
(venv) root@openstack:~#
```

5. kolla-ansible 설치

```
(venv) root@openstack:~# pip install \
git+https://opendev.org/openstack/kolla-ansible@master
(venv) root@openstack:~# mkdir -p /etc/kolla
(venv) root@openstack:~# chown $USER:$USER /etc/kolla
(venv) root@openstack:~#

# globals.yml, passwords.yml 파일을 /etc/kolla 디렉토리로 복사하기
(venv) root@openstack:~# cp -r \
/root/venv/share/kolla-ansible/etc_examples/kolla/* /etc/kolla
(venv) root@openstack:~# ls /etc/kolla
globals.yml passwords.yml
```

```
(venv) root@openstack:~#
```

1 노드 설치를 위해 'all-in-one' 인벤토리 파일을 현재 위치로 복사하기

```
(venv) root@openstack:~# ls /root/venv/share/kolla-ansible/ansible/inventory/  
all-in-one  multinode
```

```
(venv) root@openstack:~# cp \  
/root/venv/share/kolla-ansible/ansible/inventory/all-in-one .
```

```
(venv) root@openstack:~#
```

```
# globals.yml
```

- 오픈스택 전체 배포에 영향을 주는 전역 구성 변수(global configuration variables)를 정의

- kolla_base_distro : 사용하는 리눅스 배포판
- kolla_install_type : 설치 방식(source, binary)
- openstack_release : 배포할 오픈스택 버전 (2024.1, main)
- network_interface : api 통신 등에 사용할 인터페이스 이름 (ens32)
- neutron_external_interface : 외부 네트워크로 연결할 NIC
- kolla_internal_vip_address : api 접근할 가상 IP
- enable_서비스명 : 특정 서비스 활성화 여부 (enable_horizon: yes)
- docker_registry : 도커 이미지 저장소 주소

```
# passwords.yml
```

- 오픈스택 서비스 별 사용자 계정, 데이터베이스, 메시지 큐, 인증서 등에 필요한 패스워드 및 시크릿 정보

- keystone_admin_password : 오픈스택 관리자 계정 패스워드
- glance_database_password : glance 서비스 DB 패스워드
- nova_keystone_password : nova 서비스의 keystone 인증용 패스워드
- rabbitmq_password : RabbitMQ 패스워드
- horizon_secret_key : Horizon 웹 UI 용 Django 시크릿 키

6. ansible galaxy 요구사항 설치

```
(venv) root@openstack:~# kolla-ansible install-deps
```

- kolla-ansible 이 원활하게 작동하기 위해 필요한 파이썬 및 시스템 패키지들을 로컬에 설치

- 주요 설치 항목

- python 관련
- ansible-core
- docker
- Jinja2
- netaddr

- cryptography
- jsonschema
- 시스템 명령 관련
- docker
- pip
- virtualenv
- python3-venv

6. kolla passwords.yml 파일 구성

```
(venv) root@openstack:~# kolla-genpwd
(venv) root@openstack:~# kolla-genpwd
WARNING: Passwords file "/etc/kolla/passwords.yml" is world-readable. The permissions will be changed.
(venv) root@openstack:~# head /etc/kolla/passwords.yml
aodh_database_password: aNHYi7yyuVcyNkCbZC1BEa48mkyGld3V4DuELBRD
aodh_keystone_password: 602B5L2vKay9Y04amI9qrBeAhIt62woznseCzYr
barbican_crypto_key: ADW9mEw0RkSp8WPn1Ii3cn805kuE-HcEGeHL2BS05wY=
barbican_database_password: nZvpKwd9Zndib2P8Eil5L9NoEI4n45gY3EFKPDLh
barbican_keystone_password: 4JzbDogJs7nVZuPp9XN7Qdpv6GALBok5fc0mFeWI
barbican_p11_password: Qfcqnyd6c1p7Z7zLg13kBc2KcpQhtdW6g22XUrso
bifrost_ssh_key:
  private_key: '-----BEGIN PRIVATE KEY-----

MIIJQgIBADANBgkqhkiG9w0BAQEFAASCCSwggkoAgEAAoICAQCrMYmpsUBs3qJm
(venv) root@openstack:~#
```

7. kolla globals.yml 파일 구성

```
(venv) root@openstack:~# vi /etc/kolla/globals.yml
45 # valid options are [ centos , debian , rocky , ubuntu ]
46 kolla_base_distro: "ubuntu"
47
48 # Do not override this unless you know what you are doing.
49 openstack_release: "master"
50
51 # followed for other types of interfaces.
136 network_interface: "ens32"
100 # addresses for that reason.
161 neutron_external_interface: "ens33"
162 # network_interface as set in the networking section below.
65 kolla_internal_vip_address: "211.183.3.40"
66
75 # internal and external requests between two VIPs.
76 kolla_external_vip_address: "{{ kolla_internal_vip_address }}"
77
139 # the network_interface. These interfaces must contain an IP address.
140 kolla_external_vip_interface: "{{ network_interface }}"
141 #api_interface: "if network interface is"
167 neutron_plugin_agent: "openvswitch"
168
313 # valid options are [ true, false ]
314 openstack_logging_debug: "True"
315
317 # glance, keystone, neutron, nova, heat, and horizon.
318 enable_openstack_core: "yes"
319
32
enable_openvswitch: "{{ enable_neutron | bool and neutron_plugin_agent != 'linux
bridge' }}"
#enable_svs: "if enable_neutron | bool and neutron_plugin_agent == 'open' is"
```

#enable_neutron: "{{ enable_openstack_core | bool }}" 과 같이 기본 서비스는 "enable_openstack_core" 변수에 정의되어 있다.

```
316 # Enable core OpenStack services. This includes:
317 # glance, keystone, neutron, nova, heat, and horizon.
318 #enable_openstack_core: "yes"
319
```

따라서 위와 같은 기본 서비스는 별도의 설정이 없더라도 기본 배포 서비스에 포함된다.

```
342 #enable_ceph_rgw_loadbalancer: "{{ enable_ceph_rgw | bool }}"
343 #enable_cinder: "no"
344 #enable_cinder_backup: "yes"
```

cinder 는 기본 서비스에 포함되어 있지 않고 기본 불리언이 no 이므로 배포가 되지 않는다. 필요하다면 관련된 항목을 yes 로 변경한 뒤, 배포해야 사용할 수 있다.

```
342 #enable_ceph_rgw_loadbalancer: "{{ enable_ceph_rgw | bool }}"
343 enable_cinder: "yes"
344 enable_cinder_backup: "yes"
345 enable_cinder_backend_iscsi: "{{ enable_cinder_backend_lvm | bool }}"
346 enable_cinder_backend_lvm: "yes"
347 #enable_cinder_backend_rdf: "no"
348 #enable_cinder_backend_rdf: "no"
389 enable_iscsid: "{{ enable_cinder | bool and enable_cinder_backend_iscsi | bool }}"
390 #enable_kubernetes: "no"
```

```
143 #dns_interface: "{{ network_interface }}"
144 octavia_network_interface: "{{ api_interface }}"
145
146 # The network interface. These interfaces must contain an IP address.
147 kolla_external_vip_interface: "{{ network_interface }}"
148 api_interface: "{{ network_interface }}"
149 #tunnel_interface: "{{ network_interface }}"
150 #dns_interface: "{{ network_interface }}"
151 octavia_network_interface: "{{ api_interface }}"
152
153 # Configure the address family (AF) per network.
154 # Valid options are [ ipv4, ipv6 ]
155 network_address_family: "ipv4"
156 api_address_family: "{{ network_address_family }}"
157 #storage_address_family: "{{ network_address_family }}"
158 #migration_address_family: "{{ api_address_family }}"
159 #tunnel_address_family: "{{ network_address_family }}"
160 octavia_network_address_family: "{{ api_address_family }}"
161 #interface_network_neutron_vpnaas: "{{ enable_neutron_vpnaas | bool }}"
162 enable_horizon_octavia: "{{ enable_octavia | bool }}"
163 #enable_horizon_tacker: "{{ enable_tacker | bool }}"
164 #enable_neutron_l3_agent: "no"
165 enable_neutron_provider_networks: "yes"
166 #enable_nova_ssh: "yes"
167 enable_octavia: "yes"
168 #enable_octavia_driver_agents: "off enable octavia | bool and neutron al"
169 # and keep your other octavia config like before.
170 octavia_auto_configure: yes
171
```



```
# Octavia security groups. lb-mgmt-sec-grp is for amphorae.
octavia_amp_security_groups:
  mgmt-sec-grp:
    name: "lb-mgmt-sec-grp"
    enabled: true
    rules:
      - protocol: icmp
      - protocol: tcp
        src_port: 9443
        dst_port: 9443
      - protocol: tcp
        src_port: 443
        dst_port: 443
      - protocol: tcp
        src_port: 22
        dst_port: 22
      - protocol: tcp
        src_port: "{{ octavia_amp_listen_port }}"
        dst_port: "{{ octavia_amp_listen_port }}"
```

```
# - ipv6_ra_mode (optional)
octavia_amp_network:
  name: lb-mgmt-net
  shared: false
  subnet:
    name: lb-mgmt-subnet
    cidr: "{{ octavia_amp_network_cidr }}"
    gateway_ip: "10.1.0.1"
    enable_dhcp: yes
    allocation_pool_start: "10.1.0.101"
    allocation_pool_end: "10.1.0.199"

# Octavia management network subnet CIDR.
octavia_amp_network_cidr: 10.1.0.0/24

octavia_amp_image_tag: "amphora"

# Load balancer topology options are [ SINGLE, ACTIVE_STANDBY ]
octavia_loadbalancer_topology: "SINGLE"
```

```
# - extra_specs (optional)
octavia_amp_flavor:
  name: "amphora"
  is_public: no
  vcpus: 1
  ram: 1024
  disk: 5
```

```
334 #enable_aodn: "no"
335 enable_barbican: "yes"
336 #enable_barbican_ssl: "yes"
431 #enable_proxyssl: "yes"
432 enable_redis: "yes"
433 #enable_skylake: "no"
```

octavia 인증서 생성

```
(venv) root@openstack:~# kolla-ansible octavia-certificates -i ./all-in-one
```

```
# (venv) root@openstack:~# ls /etc/kolla/config/octavia/에서
client.cert-and-key.pem  server_ca.key.pem
client_ca.cert.pem  server_ca.cert.pem
```

위와 같은 4개의 파일을 확인할 수 있다.

파일명	기능
client.cert-and-key.pem	<p>**Octavia controller(예: worker/health-manager)**가 **Amphora(로드 밸런서 VM)**와 통신할 때 사용하는 클라이언트 인증서 + 개인키입니다.</p> <p>즉, Octavia 컨트롤러가 amphora-agent에 접근 시 자신을 인증하는 데 사용됩니다.</p>
server_ca.cert.pem	<p>**Amphora 내부의 서버 측 인증서(예: agent)**가 사용할 인증서들을 검증하는 데 사용되는 서버용 CA 인증서입니다.</p> <p>즉, client가 서버를 검증할 때 사용하는 루트 인증서입니다.</p>
server_ca.key.pem	<p>위의 server_ca.cert.pem을 생성할 때 사용된 **비밀 키(private key)**입니다.</p> <p>보안 유지가 필요하며 외부에 공개되면 안 됩니다.</p>
client_ca.cert.pem	<p>amphora 내부의 agent가 Octavia 컨트롤러로부터 받은 클라이언트 인증서(즉, client.cert-and-key.pem)를 검증할 때 사용하는 CA 인증서입니다.</p> <p>즉, Amphora는 이 인증서를 이용해 요청자가 신뢰할 수 있는 컨트롤러 인지 확인합니다.</p>

생성시 초기에 오류가 발생하거나 octavia_api 와 amphora 간 통신에 필요하므로 해당 파일은 octavia 관련 인스턴트들이 모두 배포되면 octavia_api 내부에 옮겨두어야 한다.

8. Deployment

```
# 도커, 오픈스택 클라이언트 패키지 준비
(venv) root@openstack:~# pip install docker python-openstackclient

# openvswitch 클라이언트 설치
(venv) root@openstack:~# apt install -y openvswitch-switch

# 이벤트리 파일에 기반하여 오픈스택을 설치할 대상 서버(노드)들에 대해 사전 준비
작업 수행
(venv) root@openstack:~# kolla-ansible bootstrap-servers -i ./all-in-one

# 호스트를 위한 pre-deployment 체크
(venv) root@openstack:~# kolla-ansible prechecks -i ./all-in-one

# 배포하기(오랜 시간 필요)
(venv) root@openstack:~# kolla-ansible deploy -i ./all-in-one
```


9. 오픈스택 사용하기

admin 사용자 설정을 위한 clouds.yaml 파일 생성하기

```
(venv) root@openstack:~# kolla-ansible post-deploy -i ./all-in-one
```

```
(venv) root@openstack:~# ls /etc/kolla/
admin-openrc.sh      keystone-fernet      nova-conductor
admin-openrc-system.sh keystone-ssh          nova-libvirt
clouds.yaml          kolla-toolbox        nova-metadata
cron                 mariadb               nova-novncproxy
```

```
(venv) root@openstack:~# source /etc/kolla/admin-openrc.sh
(venv) root@openstack:~# openstack service list
```

ID	Name	Type
00fc543d814f4e2eb85d5ac5fbd5b0a2	nova	compute
01569cf582e14a34a8f32cc87608b478	heat-cfn	cloudformation
11dfa54a4018462fa79ce5d7154d414f	heat	orchestration
32f434997259477597466f95e940d015	cinder	block-storage
571cdd89e11840978333e7b3e64ea1c2	neutron	network
60670b493fee4fc2b7055ea6c556e17b	placement	placement
6b180375e8ce416ead07d664f2248855	barbican	key-manager
ab5ce9c8b8bd4fedafddd4f4b1fa22cc	keystone	identity
c697db73af4d4785ad8e1f8b21e9cee3	cinderv3	volumev3
f82d23e5e70e4777abdc01b36858da55	octavia	load-balancer
f859d5ccde37411c9769d523ff8d2425	glance	image

```
(venv) root@openstack:~#
```

```
(venv) root@openstack:~# cat /etc/kolla/admin-openrc.sh
```

```
# Ansible managed
```

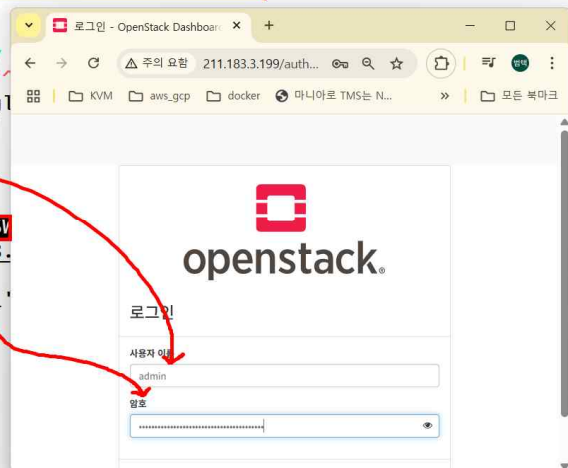
```
# Clear any old environment that may
for key in $( set | awk '{FS="="}' | cut -d= -f1 ); do
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
export OS_PROJECT_NAME='admin'
export OS_TENANT_NAME='admin'
export OS_USERNAME='admin'
export OS_PASSWORD='egglUrmzHHpj98eSv'
export OS_AUTH_URL='http://211.183.3.1:35357/v3/OS-F4FA1bf3e1154696981b1698c0dc1f06'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
export OS_REGION_NAME='RegionOne'
export OS_AUTH_PLUGIN='password'
```

```
(venv) root@openstack:~#
```

```
(venv) root@openstack:~#
```

```
(venv) root@openstack:~#
```

```
(venv) root@openstack:~#
```



10. amphora 이미지 생성 및 등록

선택1)

```
(venv) root@openstack:~# git clone https://opendev.org/openstack/octavia
(venv) root@openstack:~# apt -y install debootstrap qemu-utils git kpartx
(venv) root@openstack:~# pip install diskimage-builder
(venv) root@openstack:~# cd octavia/diskimage-create && ./diskimage-create.sh
```

선택2)(비추천)

```
(venv) root@openstack:~# wget \
https://tarballs.opendev.org/openstack/octavia/test-images/test-only-amphora-x64-
haproxy-ubuntu-noble.qcow2
```

이미지 등록

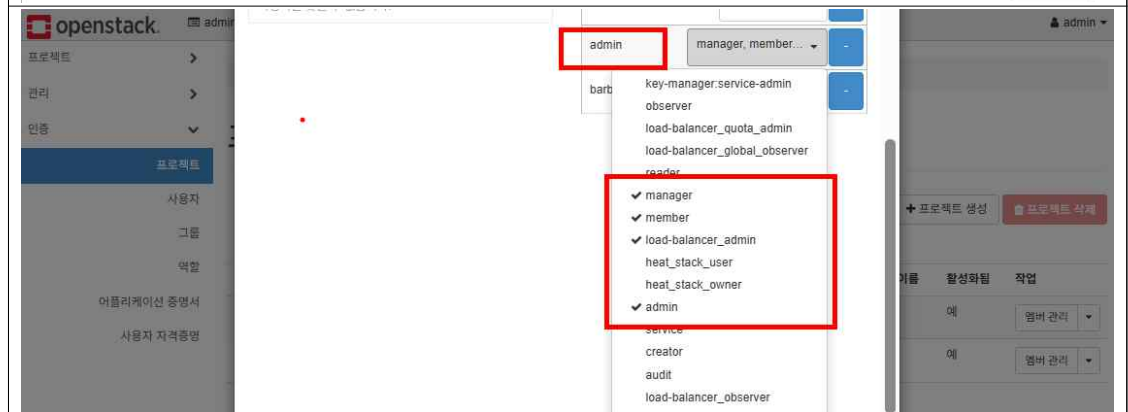
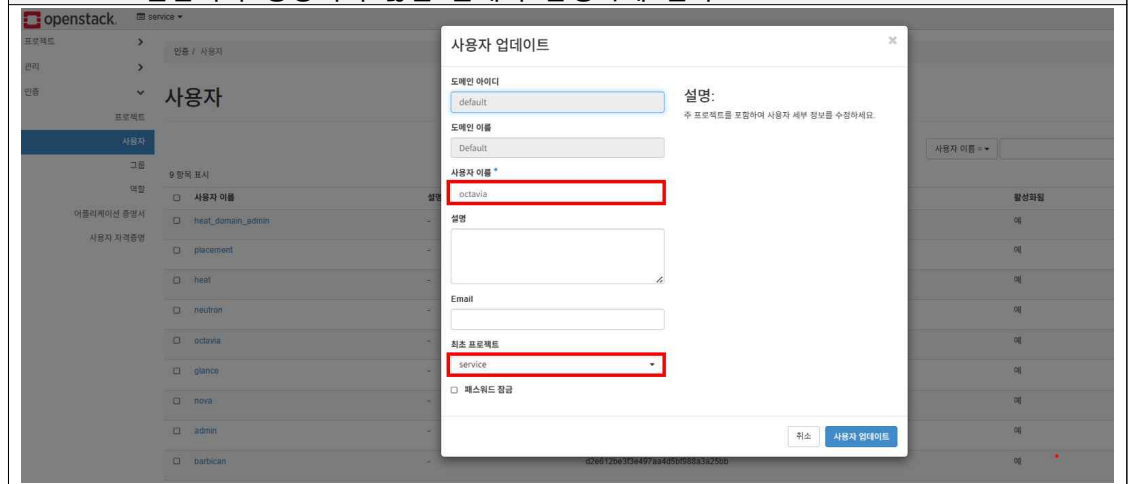
```
(venv) root@openstack:~# openstack image create amphora \
--container-format bare \
--disk-format qcow2 \
--private \
--tag amphora \
--file test-only-amphora-x64-haproxy-ubuntu-noble.qcow2 \
--property hw_architecture='x86_64' \
--property hw_rng_model=virtio
```

‘admin’ 사용자에게 ‘load-balancer_admin’ 권한 부여

```
(venv) root@openstack:~# openstack role add --project "admin" --user "admin"
load-balancer_admin
(venv) root@openstack:~#
(venv) root@openstack:~# pip install python-octaviaclient
```

11. 권한 부여 관련

octavia 의 경우 사용자 octavia를 통해 작업이 이루어진다. 또한 최초의 프로젝트는 service에서 이루어진다. 따라서 octavia 가 필요한 프로젝트를 진행하는 사용자 admin 의 경우 초기 프로젝트인 admin에서 진행할 경우 octavia 에 대한 권한이 없어 정상적으로 로드밸런서가 생성되지 않는 문제가 발생하게 된다.



이제 대시보드에서 admin 은 service 프로젝트에서 작업을 진행해야 한다.	cli 에서도 기존 /etc/kolla/admin-openrc.sh 파일을 복사하여 service 용 파일로 로그인 한다
---	---



```
(venv) root@openstack:~# pwd
/root
(venv) root@openstack:~# cat admin-openrc-service.sh
# Ansible managed

# Clear any old environment that may conflict.
for key in $(set | awk '{FS="="} /OS_/ {print $1}'); do unset $key; done
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
export OS_PROJECT_NAME='service'
export OS_TENANT_NAME='service'
export OS_USERNAME='admin'
export OS_PASSWORD='qgngHsuvkboiJf9fPj2k3E0MajCfhX0ccneVfn7M'
export OS_AUTH_URL='http://211.183.3.44:5000'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
export OS_REGION_NAME='RegionOne'
export OS_AUTH_PLUGIN='password'
(venv) root@openstack:~# source ~/admin-openrc-service.sh
```

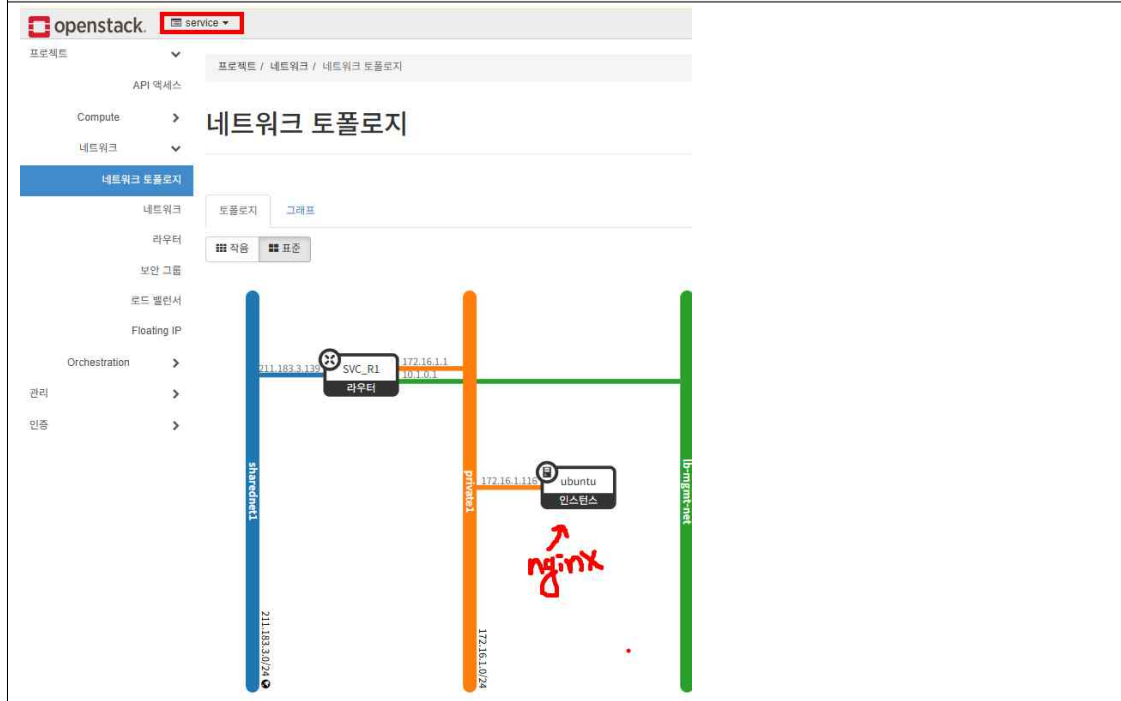
11. 외부 네트워크 구성

```
(venv) root@openstack:~# source /etc/kolla/admin-openrc.sh
(venv) root@openstack:~# ovs-vsctl add-port br-ex ens33 # 이미 설정되어 있음
(venv) root@openstack:~# ovs-vsctl set open .
external-ids:ovn-bridge-mappings=physnet1:br-ex
(venv) root@openstack:~#
(venv) root@openstack:~# openstack project list | grep service | awk '{print $2}'
938573308d9942ea872fdffd5208f8a5
(venv) root@openstack:~# projectID=$(openstack project list | grep service | awk
'{print $2}')
(venv) root@openstack:~# openstack network create --project $projectID --share
--provider-network-type flat --provider-physical-network physnet1 sharednet1
--external
(venv) root@openstack:~# openstack subnet create subnet1 --network sharednet1
--project $projectID --subnet-range 211.183.3.0/24 --allocation-pool
start=211.183.3.111,end=211.183.3.199 --gateway 211.183.3.2 --dns-nameserver
8.8.8.8
```



일반 인스턴스를 위한 사설 네트워크 구성 및 라우터 연결
 octavia 와의 연결을 위해 간단히 ubuntu 인스턴스 하나를 생성하고 nginx를 설치하여
 웹 서비스가 동작하도록 준비해 둔다.
 또한 octavia 네트워크인 lb-mgmt-net 도 라우터에 연결해 둔다. 이때 라우터의 외부

연결용 IP 주소를 미리 확인해 둔다



12. octavia 설정

octavia 관리를 api 컨테이너에는 아래와 같이 cert 관련 설정이 있지만 certs 디렉토리는 존재하지 않는다.

```
(venv) root@openstack:~# docker container exec -it octavia_api cat
/etc/octavia/octavia.conf | grep pem
ca_private_key = /etc/octavia/certs/server_ca.key.pem
ca_certificate = /etc/octavia/certs/server_ca.cert.pem
server_ca = /etc/octavia/certs/server_ca.cert.pem
client_cert = /etc/octavia/certs/client.cert-and-key.pem
client_ca = /etc/octavia/certs/client_ca.cert.pem
(venv) root@openstack:~#
```

컨테이너 내부에 certs 디렉토리를 생성하고 로컬 컨트롤러의 /etc/kolla/config/octavia 에 생성된 4개의 파일을 'docker container cp' 를 이용하여 api 에 넣어준다. 이후 컨테이너가 해당 파일을 다시 읽어 들일 수 있도록 'docker container restart octavia_api' 한다.

13. 로드밸런서 생성

대시보드에서 구현하는 로드밸런서는 아직 완벽한 상태는 아니다
먼저 프로젝트 > 네트워크 > 로드밸런서로 진입하여 로드 밸런서 생성을 클릭한다
자동으로 선택되는 값들이 있으므로 이름과 서브넷 정도만 선택해도 문제 없다

로드 밸런서 생성

로드 밸런서에 대한 자세한 정보를 제공하십시오.

이름: mylb

IP 주소:

설명:

가용 구역:

Flavor:

서브넷:

네트워크	네트워크 ID	서브넷	서브넷 ID	CIDR
sharednet1	0ba76541-4ace-4001-939b-90310d87aaa9	subnet1	0d37a2c2-d674-4db1-b11e-9527c51b05b9	211.183.3.0/24
lb-mgmt-net	870bb592-821a-4476-9e16-51fe4e42e78f	lb-mgmt-subnet	3c1ce6f1-2d3c-4373-9445-f306970de668	10.1.0.0/24
private1	f3e7513c-9150-4f0f-a5a4-78a8511522c9	private1_subnet	8f815f2b-f0a8-4032-958d-6654305057fb	172.16.1.0/24

리스너는 로드밸런서로의 유입트래픽에 대한 정의이다. 로드밸런서의 80번 포트로 유입되는 트래픽을 ubuntu 인스턴스의 nginx 로 포워딩 하기 위해 아래와 같이 구성한다.
별도의 80 번 포트를 허용하는 보안 그룹을 작성할 필요는 없다. 동적으로 생성된다.

로드 밸런서 생성

리스너에 대한 자세한 정보를 제공하십시오.

리스너 생성: 예

이름:

설명:

프로토콜: HTTP

포트: 80

클라이언트 데이터 타임아웃: 50000

TCP 검사 타임아웃: 0

멤버 연결 타임아웃: 5000

멤버 데이터 타임아웃: 50000

연결 제한: -1

Allowed Cidrs: 0.0.0.0/0

헤더 삽입:

X-Forwarded-For: ☐

X-Forwarded-Proto: ☐

X-Forwarded-Port: ☐

관리자 업 상태: 예

< 뒤로 Next > 로드 밸런서 생성

least connection, round robin, source ip 중 선택할 수 있다

로드 밸런서 생성

로드 밸런서 세부정보

리스너 세부정보

풀 세부정보

모니터 세부정보 *

풀 생성

예 아니오

이름

설명

알고리즘 *

ROUND_ROBIN

LEAST_CONNECTIONS

ROUND_ROBIN

SOURCE_IP

예 아니오

관리자 업 상태

예 아니오

취소

< 뒤로

Next >

로드 밸런서 생성

ubuntu 인스턴스의 nginx 서비스 포트를 지정한다

로드 밸런서 생성

로드 밸런서 세부정보

리스너 세부정보

풀 세부정보

모니터 세부정보 *

로드 밸런서 풀에 멤버를 추가합니다.

활당된 멤버

IP 주소 *

서브넷 *

포트 *

가중치

172.16.1.116

private1_subnet

80

1

제거

외부 멤버 추가

사용 가능한 인스턴스

필터

헬스체크 항목을 지정한 뒤, '로드 밸런서 생성'을 클릭한다

로드 밸런서 생성

로드 밸런서 세부정보

리스너 세부정보

풀 세부정보

모니터 세부정보 *

상태 모니터의 세부 사항을 제공하십시오.

상태 모니터 생성

예 아니오

이름

유형 *

HTTP

최대 재시도 다음 *

3

지연 (초) *

5

최대 재시도 *

3

타임아웃 (초) *

5

HTTP 메서드

GET

예상 코드

200

URL 경로

/index.html

관리자 업 상태

예 아니오

취소

< 뒤로

Next >

로드 밸런서 생성

아래와 같이 '운영상태'가 '온라인', '프로비저닝 상태'가 '활성'이면 정상이지만 '프로비저닝 상태'는 계속해서 '비활성(PENDING_CREATE)' 상태가 된다. 이는 컨트롤 노드와 로드밸런서가 9443 번 포트를 통해 상태 확인이 불가능하기 때문에 발생하는 문

제이다.

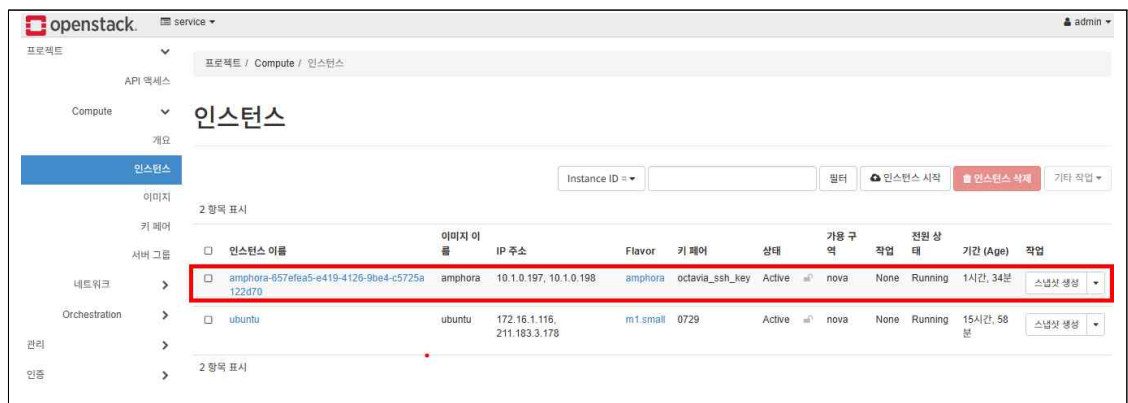


위의 문제점을 해결하기 위해서는 컨트롤노드에서 내부로 라우팅이 가능하도록 라우팅 테이블 작성이 필요하다. 라우터의 외부 IP 주소를 다시한번 확인한 뒤, 아래와 같이 yaml 파일을 작성한다.

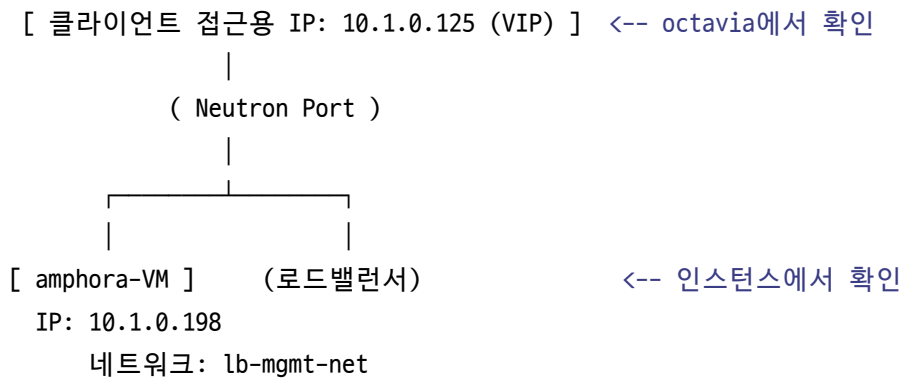
```
(venv) root@openstack:~# ls /etc/netplan/
01-network-manager-all.yaml octavia.yaml
(venv) root@openstack:~#
(venv) root@openstack:~# cat /etc/netplan/octavia.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    br-ex:
      routes:
        - to: 10.1.0.0/24      # 로드밸런서 네트워크
          via: 211.183.3.139   # 라우터의 외부 IP 주소
(venv) root@openstack:~#
(venv) root@openstack:~# systemctl restart systemd-networkd && netplan apply

(venv) root@openstack:~# ip route
default via 211.183.3.2 dev ens32 proto static
10.1.0.0/24 via 211.183.3.139 dev br-ex proto static onlink
211.183.3.0/24 dev ens32 proto kernel scope link src 211.183.3.50
(venv) root@openstack:~#
```

여기에서 이상한 부분이 있다. 로드밸런서를 생성하면 인스턴스에도 하나의 인스턴스가 자동으로 생성된다.

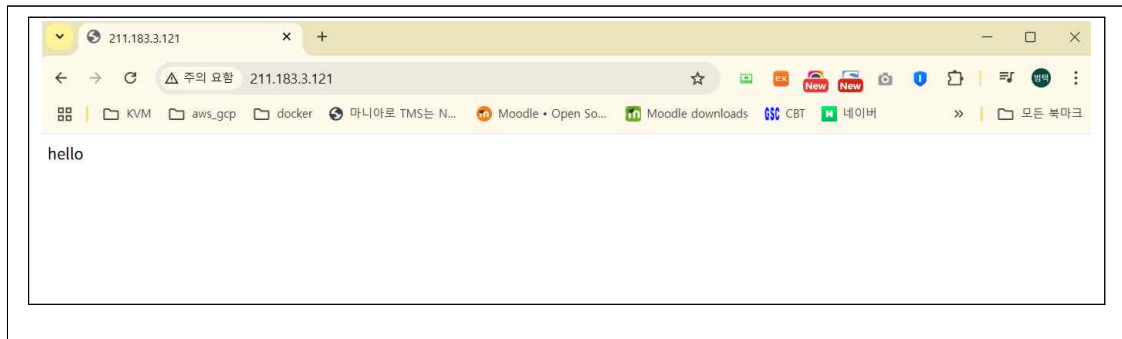


하나의 로드밸런서를 생성하면 아래와 같은 구조가 만들어 진다



- * 10.1.0.125 (Virtual IP) 는 사용자가 로드밸런서를 접근할 때 사용하는 주소이며 loadbalancer create 시 지정된 네트워크/서브넷에서 생성되며 외부에 노출되어야 하므로 floating ip를 부착해서 사용한다.
- * 10.1.0.198 (Amphora 관리 IP) 는 octavia 가 내부적으로 생성한 로드밸런서 VM 의 lb-mgm-net 상의 주소이며 외부 접근용이 아니다. 인스턴스에서 확인 가능하다 floating ip를 추가할 필요 없다





14. 오픈스택내에 ubuntu 24.04 (cpu 2, memory 2 GB, disk 20 GB) 이미지를 이용하여 2 대의 인스턴스(master, node1)를 생성하였고 kubeadm 기반의 kubernetes 1.29를 구축한 뒤, 위의 octavia 와의 연동을 통해 매니페스트 파일에 있는 service - type: LoadBalancer 와 연동이 되도록 구축해 본다.

모든 노드에 /etc/kubernetes/cloud.conf 파일 생성

```
(venv) root@openstack:~# openstack network list
```

ID	Name	Subnets
0ba76541-4ace-4001-939b-90310d87aea9	sharednet1	0d37a2c2-d674-4db1-b11e-9527c51b05b9
870bb592-821a-4476-9e16-51fede42e78f	lb-mgmt-net	3c1cef6f-2d3c-4373-9445-f306970de668
f3e7513c-9150-4f0f-a5a4-78a8511522c9	private1	8f815f2b-f0e8-4032-958d-6654305057fb

```
(venv) root@openstack:~# openstack subnet list
```

ID	Name	Network	Subnet
0d37a2c2-d674-4db1-b11e-9527c51b05b9	subnet1	0ba76541-4ace-4001-939b-90310d87aea9	211.183.3.0/24
3c1cef6f-2d3c-4373-9445-f306970de668	lb-mgmt-subnet	870bb592-821a-4476-9e16-51fede42e78f	10.1.0.0/24
8f815f2b-f0e8-4032-958d-6654305057fb	private1_subnet	f3e7513c-9150-4f0f-a5a4-78a8511522c9	172.16.1.0/24

```
root@master:~# cat /etc/kubernetes/cloud.conf
```

```
[Global]
auth-url = http://211.183.3.44:5000
username = admin
password = OS_PASSWORD에 기록된 패스워드작성
tenant-name = service
domain-name = Default
region = RegionOne

[LoadBalancer]
use-octavia = true
subnet-id = 3c1cef6f-2d3c-4373-9445-f306970de668 # lb-mgmt-net 서브넷 UUID
floating-network-id = 0ba76541-4ace-4001-939b-90310d87aea9 # sharednet1 같은 외부
```

네트워크 UUID

root@master:~#

kube-controller-manager 내용 수정(master) > 변경된 내용 반영 하여 자동 재생성

root@master:~# cat /etc/kubernetes/manifests/kube-controller-manager.yaml

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-controller-manager
    tier: control-plane
  name: kube-controller-manager
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-controller-manager
    - --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf
    - --authorization-kubeconfig=/etc/kubernetes/controller-manager.conf
    - --bind-address=127.0.0.1
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --cluster-name=kubernetes
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
    - --controllers=*,bootstrapsigner,tokencleaner
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
    - --leader-elect=true
    - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
    - --root-ca-file=/etc/kubernetes/pki/ca.crt
    - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
    - --use-service-account-credentials=true
    - --cloud-provider=external # 추가 됨
    image: registry.k8s.io/kube-controller-manager:v1.29.15
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 127.0.0.1
        path: /healthz
        port: 10257
        scheme: HTTPS
      initialDelaySeconds: 10
```

```
    periodSeconds: 10
    timeoutSeconds: 15
name: kube-controller-manager
resources:
  requests:
    cpu: 200m
startupProbe:
  failureThreshold: 24
  httpGet:
    host: 127.0.0.1
    path: /healthz
    port: 10257
    scheme: HTTPS
  initialDelaySeconds: 10
  periodSeconds: 10
  timeoutSeconds: 15
volumeMounts:
- mountPath: /etc/ssl/certs
  name: ca-certs
  readOnly: true
- mountPath: /etc/ca-certificates
  name: etc-ca-certificates
  readOnly: true
- mountPath: /usr/libexec/kubernetes/kubelet-plugins/volume/exec
  name: flexvolume-dir
- mountPath: /etc/kubernetes/pki
  name: k8s-certs
  readOnly: true
- mountPath: /etc/kubernetes/controller-manager.conf
  name: kubeconfig
  readOnly: true
- mountPath: /usr/local/share/ca-certificates
  name: usr-local-share-ca-certificates
  readOnly: true
- mountPath: /usr/share/ca-certificates
  name: usr-share-ca-certificates
  readOnly: true
- mountPath: /etc/kubernetes/cloud.conf
  name: cloud-config
  readOnly: true
hostNetwork: true
priority: 2000001000
priorityClassName: system-node-critical
securityContext:
  seccompProfile:
```

```

    type: RuntimeDefault
volumes:
- hostPath:
    path: /etc/ssl/certs
    type: DirectoryOrCreate
    name: ca-certs
- hostPath:
    path: /etc/ca-certificates
    type: DirectoryOrCreate
    name: etc-ca-certificates
- hostPath:
    path: /usr/libexec/kubernetes/kubelet-plugins/volume/exec
    type: DirectoryOrCreate
    name: flexvolume-dir
- hostPath:
    path: /etc/kubernetes/pki
    type: DirectoryOrCreate
    name: k8s-certs
- hostPath:
    path: /etc/kubernetes/controller-manager.conf
    type: FileOrCreate
    name: kubeconfig
- hostPath:
    path: /usr/local/share/ca-certificates
    type: DirectoryOrCreate
    name: usr-local-share-ca-certificates
- hostPath:
    path: /usr/share/ca-certificates
    type: DirectoryOrCreate
    name: usr-share-ca-certificates
- hostPath:
    path: /etc/kubernetes/cloud.conf
    type: File
    name: cloud-config
status: {}
root@master:~#

```

--cloud-provider=external

- 예전에는 --cloud-provider=openstack 같은 식으로 kube-controller-manager 자체가 클라우드 API(OpenStack, AWS 등)에 직접 접근해서 노드, 볼륨, 로드밸런서를 관리 했음 > 이 방식은 k8s 내부 코드에 클라우드별 로직이 모두 들어가 복잡하다. 또한 클라우드 벤더별 업데이트 속도에 따라 k8s 릴리스와 맞추기 어려움이 있었다
- --cloud-provider=external
 - 더 이상 클라우드 API(오픈스택 등)에 직접 접근하지 않는다
 - 클라우드 연계 기능은 외부 컴포넌트(Cloud Controller Manager, CCM)에게 위임

- 이러한 역할을 'openstack-cloud-controller-manager' 가 그 역할 대신 수행

OpenStack Cloud Controller Manager(CCM) 배포

```
root@master:~# cat openstack-ccm.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cloud-controller-manager
  namespace: kube-system
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: openstack-cloud-controller-manager
  namespace: kube-system
spec:
  selector:
    matchLabels:
      k8s-app: openstack-cloud-controller-manager
  template:
    metadata:
      labels:
        k8s-app: openstack-cloud-controller-manager
    spec:
      serviceAccountName: cloud-controller-manager
      tolerations:
        - key: "node-role.kubernetes.io/control-plane"
          effect: "NoSchedule"
        - key: "node.kubernetes.io/unreachable"
          operator: "Exists"
          effect: "NoSchedule"
      containers:
        - name: openstack-cloud-controller-manager
          image: docker.io/k8scloudprovider/openstack-cloud-controller-manager:latest
          command:
            - /bin/openstack-cloud-controller-manager
            - --cloud-provider=openstack
            - --cloud-config=/etc/kubernetes/cloud.conf
            - --use-service-account-credentials=true
            - --v=4
          volumeMounts:
            - mountPath: /etc/kubernetes/cloud.conf
              name: cloud-config
              readOnly: true
```



```

    volumes:
      - name: cloud-config
        hostPath:
          path: /etc/kubernetes/cloud.conf
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:cloud-controller-manager
rules:
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "patch", "update"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["*"]
  - apiGroups: [""]
    resources: ["services", "endpoints"]
    verbs: ["get", "list", "watch", "update", "patch"]
  - apiGroups: [""]
    resources: ["services/status"]
    verbs: ["update", "patch"]
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "update", "patch"]
  - apiGroups: ["coordination.k8s.io"]
    resources: ["leases"]
    verbs: ["get", "watch", "list", "update", "create"]
  - apiGroups: [""]
    resources: ["serviceaccounts"]
    verbs: ["get", "list", "watch", "create", "update"]
  - apiGroups: [""]
    resources: ["serviceaccounts/token"]
    verbs: ["create"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: system:cloud-controller-manager

```

```

roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:cloud-controller-manager
subjects:
  - kind: ServiceAccount
    name: cloud-controller-manager
    namespace: kube-system

```

```
root@master:~# kubectl apply -f openstack-ccm.yaml
```

openstack-cloud-controller-manager (OCCM) > Kubernetes가 “외부 클라우드 (OpenStack)” 자원을 사용할 수 있도록 연결해 주는 역할

Kubernetes와 OpenStack을 연결하는 Cloud Controller Manager(CCM)

- 클라우드 벤더(OpenStack)의 리소스를 Kubernetes 오브젝트와 연결해주는 역할
- 주요 기능
 - 노드 관리(Node Controller) : OpenStack VM 인스턴스를 Kubernetes 노드로 인식하고 상태를 동기화
 - 라우트 관리(Route Controller) : (지원하는 경우) 노드 간 네트워크 라우팅 설정
 - 서비스 관리(Service Controller) : Service type=LoadBalancer 를 선언하면, OpenStack Octavia를 통해 자동으로 로드밸런서를 생성하고, Floating IP를 붙여 외부 접근 가능하게 함
 - Persistent Volume 관리(PV Controller) : OpenStack Cinder 볼륨을 동적으로 프로비저닝하여 PVC 요청을 처리

RoleBinding 설정 > CCM Pod 의 기본 serviceaccount 가 kube-system namespace 의 configMap 보기 권한을 갖을 수 있도록 해 준다

```
root@master:~# k get pod -n kube-system | grep manager
```

```

kube-controller-manager-master      1/1      Running   0          16m
openstack-cloud-controller-manager-5f994  1/1      Running   0          2m
openstack-cloud-controller-manager-cdzp8  1/1      Running   0          2m
root@master:~#

```

발생할 수 있는 오류들

1. /etc/kubernetes/cloud.conf 는 각 호스트에서 pod 로 hostPath를 통해 연결되므로 반드시 모든 노드에 파일을 배치 시켜야 한다

2. 모든 노드에 파일을 배치 시키는 것보다 아래와 같이 secret를 이용할 수도 있다

```

kubectl create secret generic cloud-config \
  --from-file=cloud.conf=/etc/kubernetes/cloud.conf \
  -n kube-system

```

데몬셋 설정 파일에서 아래처럼 일부 수정

```
volumes:
  - name: cloud-config
    secret:
      secretName: cloud-config
volumeMounts:
  - name: cloud-config
    mountPath: /etc/kubernetes
    readOnly: true
```

3. default 계정을 그대로 사용할 경우 권한이 너무 많아 보안적으로 문제가 될 수 있다. 별도의 sa를 생성하고 이를 적용 시키는 것이 보안적으로 효과적일 수 있다.

openstack-cloud-controller-manager-all-in-one.yaml (all-in-one 파일)

```
---
apiVersion: v1
kind: Secret
metadata:
  name: cloud-config
  namespace: kube-system
type: Opaque
stringData:
  cloud.conf: |
    [Global]
    auth-url = http://211.183.3.99:5000/v3
    username = admin
    password = 9EmjbTMNe0Tm7ZgZ9LjdhlA4ADv59ck8L16o3632
    region = RegionOne
    project-name = service
    user-domain-name = Default
    project-domain-name = Default

    [LoadBalancer]
    use-octavia = true
    subnet-id = 64e2d187-a617-4789-a20d-190ba78ead8d # lb-mgmt-subnet
    floating-network-id = b571ae12-0f1a-43b0-94bf-c539cc782e88 # sharednet1
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cloud-controller-manager
  namespace: kube-system
---
# ClusterRole: system:cloud-controller-manager
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:cloud-controller-manager
```

```

rules:
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch", "create", "update", "delete"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "patch", "update"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["*"]
  - apiGroups: [""]
    resources: ["services", "services/status"]
    verbs: ["*"]
  - apiGroups: [""]
    resources: ["endpoints"]
    verbs: ["create", "get", "list", "watch", "update", "delete"]
  - apiGroups: ["coordination.k8s.io"]
    resources: ["leases"]
    verbs: ["get", "create", "update"]

```

```
---
```

```

# ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: system:cloud-controller-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:cloud-controller-manager
subjects:
  - kind: ServiceAccount
    name: cloud-controller-manager
    namespace: kube-system

```

```
---
```

```

apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: openstack-cloud-controller-manager
  namespace: kube-system
spec:
  selector:
    matchLabels:
      k8s-app: openstack-cloud-controller-manager
  template:
    metadata:
      labels:
        k8s-app: openstack-cloud-controller-manager

```

```

spec:
  serviceAccountName: cloud-controller-manager
  tolerations:
    - key: "node-role.kubernetes.io/control-plane"
      effect: "NoSchedule"
    - key: "node-role.kubernetes.io/master"
      effect: "NoSchedule"
    - key: "node.kubernetes.io/unreachable"
      operator: "Exists"
      effect: "NoSchedule"
  containers:
    - name: openstack-cloud-controller-manager
      image: docker.io/k8scloudprovider/openstack-cloud-controller-manager:latest
      command:
        - /bin/openstack-cloud-controller-manager
        - --cloud-provider=openstack
        - --cloud-config=/etc/kubernetes/cloud.conf
        - --use-service-account-credentials=true
        - --v=4
      volumeMounts:
        - mountPath: /etc/kubernetes/cloud.conf
          subPath: cloud.conf
          name: cloud-config
          readOnly: true
  volumes:
    - name: cloud-config
      secret:
        secretName: cloud-config

```

nginx pod 와 lb 배포하기

```

root@master:~# cat nginx.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:

```

```
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:latest
      ports:
        - containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80

root@master:~# k apply -f nginx.yaml
```

확인

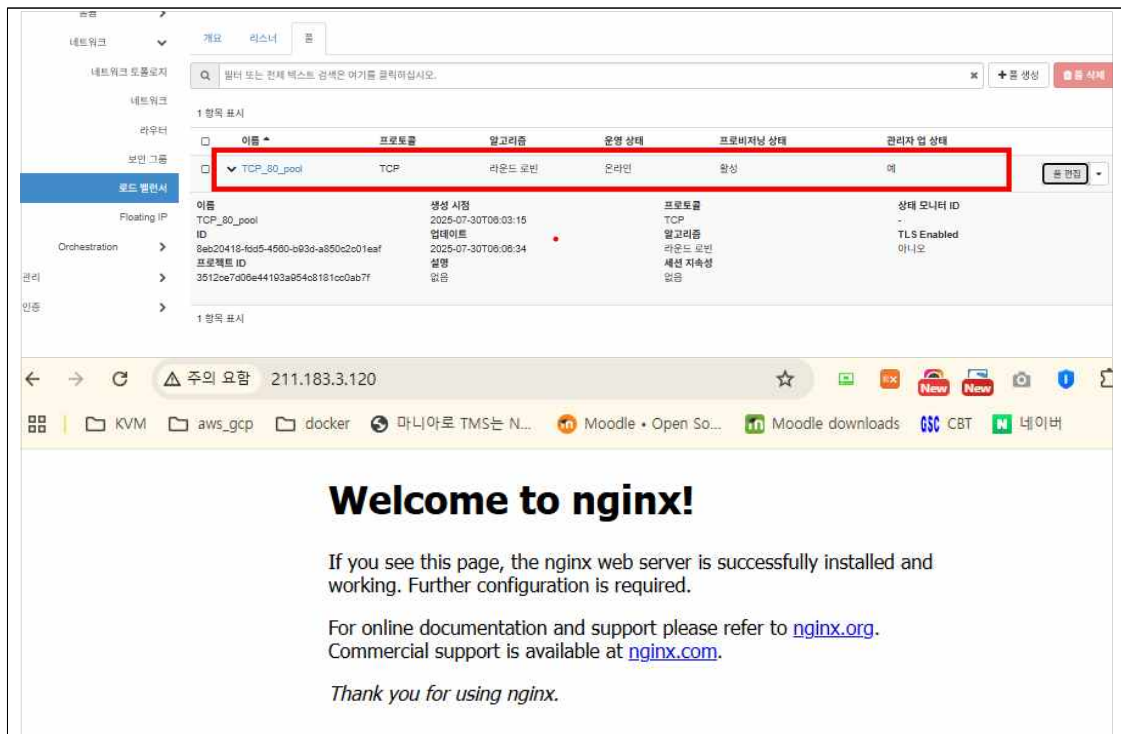
자동으로 floatingIP 까지 할당된다



이름	IP 주소	가용 구역	운영 상태	프로비저닝 상태	관리자 업 상태
kube_service_kubernetes_default_nginx-service	10.1.0.135	-	온라인	활성	예
nginx	10.1.0.156	-	온라인	활성	예

리스트/풀 까지 자동으로 등록된다

floatingIP를 통해 웹서버(nginx pod)로 연결 가능하다



k8s master 에서도 확인 가능하다

```
root@master:~# k get pod,svc
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-deployment-7c79c4bf97-dqq2h	1/1	Running	0	74m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
service/nginx-service	LoadBalancer	10.99.26.106	211.183.3.120	80:32214/TCP 52m

root@master:~#

cinder 볼륨을 활용한 동적 프로비전

앞선 /etc/kubernetes/cloud.conf 파일을 활용하여 secret을 생성하고 이를 이용한 볼륨을 동적 프로비전 해 볼 수 있다

```
root@master:~# cp /etc/kubernetes/cloud.conf .
```

```
root@master:~# cat cloud.conf # [BlockStorage] 섹션 추가
```

```
[Global]
```

```
auth-url = http://211.183.3.99:5000
```

```
username = admin
```



```
password = 9EmjbTMNe0Tm7ZgZ9LjdhlA4ADv59ck8L16o3632
```

```
tenant-name = service
```

```
domain-name = Default
```

```
region = RegionOne
```

[LoadBalancer]

```
use-octavia = true
```

```
subnet-id = 64e2d187-a617-4789-a20d-190ba78ead8d # lb-mgmt-net
```

```
floating-network-id = b571ae12-0f1a-43b0-94bf-c539cc782e88 # sharednet1
```

[BlockStorage]

```
bs-version=v3
```

```
root@master:~#
```

```
root@master:~# kubectl create secret generic cloud-config  
--from-file=cloud.conf=/root/cloud.conf -n kube-system
```

```
root@master:~# k get secret -n kube-system
```

NAME	TYPE	DATA	AGE
bootstrap-token-b1n3sp	bootstrap.kubernetes.io/token	7	4h35m
cloud-config	Opaque	1	168m

```
root@master:~#
```

cinder-csi 관련

1. cinder-csi-controllerplugin

역할: CSI (Container Storage Interface) 컨트롤러 플러그인

주요 기능:

볼륨 프로비저닝: 사용자가 PVC를 생성하면 Cinder 볼륨을 생성하는 요청을 OpenStack Cinder API로 전달하여 볼륨을 만듭니다.

볼륨 삭제: PVC 삭제 시 연결된 Cinder 볼륨을 삭제합니다.

볼륨 관리: 볼륨 확장, 스냅샷 생성/삭제, 볼륨 조회 등 컨트롤러 단에서 필요한 모든 API 호출을 처리합니다.

멀티노드 접근 관리: 여러 노드에서 볼륨 접근이 가능하도록 필요한 컨트롤러 서비스를 제공합니다.

구동 위치: 보통 Kubernetes 클러스터 내 컨트롤러 노드(마스터 또는 별도의 노드)에 Deployment 형태로 구동

2. cinder-csi-nodeplugin

역할: CSI 노드 플러그인

주요 기능:

볼륨 마운트/언마운트: 실제 Kubernetes 워커 노드에서 Cinder 볼륨을 pod에 연결 (attach)하고, mount 시키거나 detach, unmount하는 역할

볼륨 상태 보고: 노드 상태, 볼륨 마운트 상태 등을 컨트롤러에 리포트

노드별 볼륨 액세스 지원: 해당 노드에서 볼륨 사용 가능하도록 하는 역할 (ex. iSCSI, FC 등의 attach/detach 수행)

구동 위치: Kubernetes 각 워커 노드마다 DaemonSet 형태로 구동되어, 그 노드에 연결된 볼륨을 관리

master, node에서 미리 이미지를 pull 해 둔다

```
docker pull registry.k8s.io/sig-storage/csi-attacher:v4.7.0
docker pull registry.k8s.io/sig-storage/csi-provisioner:v5.1.0
docker pull registry.k8s.io/sig-storage/csi-snapshotter:v8.1.0
docker pull registry.k8s.io/sig-storage/csi-resizer:v1.12.0
docker pull registry.k8s.io/sig-storage/livenessprobe:v2.14.0
docker pull registry.k8s.io/provider-os/cinder-csi-plugin:v1.33.0
docker pull registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.12.0
docker pull registry.k8s.io/sig-storage/livenessprobe:v2.14.0
docker pull registry.k8s.io/provider-os/cinder-csi-plugin:v1.33.0
```

controller plugin/node plugin 설치

```
# Controller Plugin 설치
kubectl apply -f \
https://raw.githubusercontent.com/kubernetes/cloud-provider-openstack/master/manifests/cinder-csi-plugin/cinder-csi-controllerplugin.yaml

# Node Plugin 설치
kubectl apply -f \
https://raw.githubusercontent.com/kubernetes/cloud-provider-openstack/master/manifests/cinder-csi-plugin/cinder-csi-nodeplugin.yaml
```

또는 아래와 같이 파일을 직접 apply 한다

```
cinder-csi-controllerplugin.yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: csi-cinder-controllerplugin
  namespace: kube-system
```

```
spec:
  replicas: 1
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
      maxSurge: 1
  selector:
    matchLabels:
      app: csi-cinder-controllerplugin
  template:
    metadata:
      labels:
        app: csi-cinder-controllerplugin
    spec:
      serviceAccount: csi-cinder-controller-sa
      containers:
        - name: csi-attacher
          image: registry.k8s.io/sig-storage/csi-attacher:v4.7.0
          args:
            - "--csi-address=$(ADDRESS)"
            - "--timeout=3m"
            - "--leader-election=true"
            - "--default-fstype=ext4"
          env:
            - name: ADDRESS
              value: /var/lib/csi/sockets/pluginproxy/csi.sock
          imagePullPolicy: "IfNotPresent"
          volumeMounts:
            - name: socket-dir
              mountPath: /var/lib/csi/sockets/pluginproxy/
        - name: csi-provisioner
          image: registry.k8s.io/sig-storage/csi-provisioner:v5.1.0
          args:
            - "--csi-address=$(ADDRESS)"
            - "--timeout=3m"
            - "--default-fstype=ext4"
            - "--feature-gates=Topology=true"
            - "--extra-create-metadata"
            - "--leader-election=true"
```

```

env:
  - name: ADDRESS
    value: /var/lib/csi/sockets/pluginproxy/csi.sock
imagePullPolicy: "IfNotPresent"
volumeMounts:
  - name: socket-dir
    mountPath: /var/lib/csi/sockets/pluginproxy/
- name: csi-snapshotter
  image: registry.k8s.io/sig-storage/csi-snapshotter:v8.1.0
  args:
    - "--csi-address=$(ADDRESS)"
    - "--timeout=3m"
    - "--extra-create-metadata"
    - "--leader-election=true"
  env:
    - name: ADDRESS
      value: /var/lib/csi/sockets/pluginproxy/csi.sock
  imagePullPolicy: Always
  volumeMounts:
    - mountPath: /var/lib/csi/sockets/pluginproxy/
      name: socket-dir
- name: csi-resizer
  image: registry.k8s.io/sig-storage/csi-resizer:v1.12.0
  args:
    - "--csi-address=$(ADDRESS)"
    - "--timeout=3m"
    - "--handle-volume-inuse-error=false"
    - "--leader-election=true"
  env:
    - name: ADDRESS
      value: /var/lib/csi/sockets/pluginproxy/csi.sock
  imagePullPolicy: "IfNotPresent"
  volumeMounts:
    - name: socket-dir
      mountPath: /var/lib/csi/sockets/pluginproxy/
- name: liveness-probe
  image: registry.k8s.io/sig-storage/livenessprobe:v2.14.0
  args:
    - "--csi-address=$(ADDRESS)"
  env:

```

```
- name: ADDRESS
  value: /var/lib/csi/sockets/pluginproxy/csi.sock
volumeMounts:
  - mountPath: /var/lib/csi/sockets/pluginproxy/
    name: socket-dir
- name: cinder-csi-plugin
  image: registry.k8s.io/provider-os/cinder-csi-plugin:v1.33.0
  args:
    - /bin/cinder-csi-plugin
    - "--endpoint=$(CSI_ENDPOINT)"
    - "--cloud-config=$(CLOUD_CONFIG)"
    - "--cluster=$(CLUSTER_NAME)"
    - "--pvc-annotations"
    - "--v=1"
  env:
    - name: CSI_ENDPOINT
      value: unix://csi/csi.sock
    - name: CLOUD_CONFIG
      value: /etc/config/cloud.conf
    - name: CLUSTER_NAME
      value: kubernetes
  imagePullPolicy: "IfNotPresent"
  ports:
    - containerPort: 9808
      name: healthz
      protocol: TCP
  # The probe
  livenessProbe:
    failureThreshold: 5
    httpGet:
      path: /healthz
      port: healthz
    initialDelaySeconds: 10
    timeoutSeconds: 10
    periodSeconds: 60
  volumeMounts:
    - name: socket-dir
      mountPath: /csi
    - name: secret-cinderplugin
      mountPath: /etc/config
```

```

        readOnly: true
        # - name: cacert
        #   mountPath: /etc/cacert
        #   readOnly: true
volumes:
  - name: socket-dir
    emptyDir: {}
  - name: secret-cinderplugin
    secret:
      secretName: cloud-config
  # - name: cacert
  #   hostPath:
  #     path: /etc/cacert

```

cinder-csi-nodeplugin.yaml

```

kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: csi-cinder-nodeplugin
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: csi-cinder-nodeplugin
  template:
    metadata:
      labels:
        app: csi-cinder-nodeplugin
    spec:
      tolerations:
        - operator: Exists
      serviceAccount: csi-cinder-node-sa
      hostNetwork: true
      containers:
        - name: node-driver-registrar
          image: registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.12.0
          args:
            - "--csi-address=$(ADDRESS)"
            - "--kubelet-registration-path=$(DRIVER_REG_SOCK_PATH)"
          env:

```

```

      - name: ADDRESS
        value: /csi/csi.sock
      - name: DRIVER_REG_SOCK_PATH
        value: /var/lib/kubelet/plugins/cinder.csi.openstack.org/csi.sock
      - name: KUBE_NODE_NAME
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
        imagePullPolicy: "IfNotPresent"
    volumeMounts:
      - name: socket-dir
        mountPath: /csi
      - name: registration-dir
        mountPath: /registration
  - name: liveness-probe
    image: registry.k8s.io/sig-storage/livenessprobe:v2.14.0
    args:
      - --csi-address=/csi/csi.sock
    volumeMounts:
      - name: socket-dir
        mountPath: /csi
  - name: cinder-csi-plugin
    securityContext:
      privileged: true
      capabilities:
        add: ["SYS_ADMIN"]
      allowPrivilegeEscalation: true
    image: registry.k8s.io/provider-os/cinder-csi-plugin:v1.33.0
    args:
      - /bin/cinder-csi-plugin
      - "--endpoint=$(CSI_ENDPOINT)"
      - "--provide-controller-service=false"
      - "--cloud-config=$(CLOUD_CONFIG)"
      - "--v=1"
    env:
      - name: CSI_ENDPOINT
        value: unix:///csi/csi.sock
      - name: CLOUD_CONFIG
        value: /etc/config/cloud.conf

```



```
imagePullPolicy: "IfNotPresent"
ports:
  - containerPort: 9808
    name: healthz
    protocol: TCP
# The probe
livenessProbe:
  failureThreshold: 5
  httpGet:
    path: /healthz
    port: healthz
  initialDelaySeconds: 10
  timeoutSeconds: 3
  periodSeconds: 10
volumeMounts:
  - name: socket-dir
    mountPath: /csi
  - name: kubelet-dir
    mountPath: /var/lib/kubelet
    mountPropagation: "Bidirectional"
  - name: pods-probe-dir
    mountPath: /dev
    mountPropagation: "HostToContainer"
  - name: secret-cinderplugin
    mountPath: /etc/config
    readOnly: true
  # - name: cacert
  #   mountPath: /etc/cacert
  #   readOnly: true
volumes:
  - name: socket-dir
    hostPath:
      path: /var/lib/kubelet/plugins/cinder.csi.openstack.org
      type: DirectoryOrCreate
  - name: registration-dir
    hostPath:
      path: /var/lib/kubelet/plugins_registry/
      type: Directory
  - name: kubelet-dir
    hostPath:
```

```

        path: /var/lib/kubelet
        type: Directory
      - name: pods-probe-dir
        hostPath:
          path: /dev
          type: Directory
      - name: secret-cinderplugin
        secret:
          secretName: cloud-config
# - name: cacert
#   hostPath:
#     path: /etc/cacert

```

csi-cinder-controller, csi-cinder-node 가 볼륨을 만들고 연결하기 위해서는 serviceaccount, clusterrole, clusterrolebinding 이 필요하다

```

serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: csi-cinder-controller-sa
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: csi-cinder-node-sa
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-cinder-controller-rolebinding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-controller-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: cluster-admin # (권한 조정 가능)

```

```

  apiGroup: rbac.authorization.k8s.io
  ---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-cinder-node-rolebinding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-node-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: cluster-admin # (권한 조정 가능)
  apiGroup: rbac.authorization.k8s.io

```

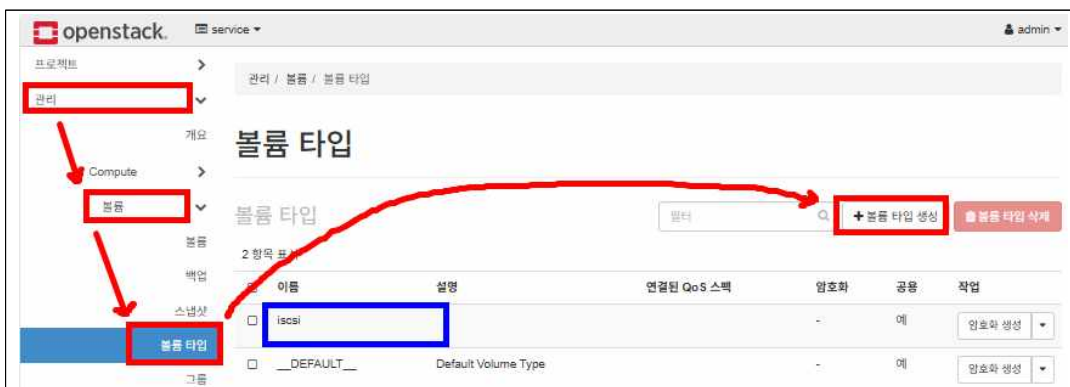
controllerplugin, nodeplugin을 확인한다

```

root@master:~/openstack# k get pod -n kube-system -o wide | grep csi-cinder
csi-cinder-controllerplugin-75968df487-2l696    6/6    Running    0    30m
192.168.166.130    node1    <none>    <none>
csi-cinder-nodeplugin-2b98s                    3/3    Running    0    30m
172.16.1.177      node1    <none>    <none>
csi-cinder-nodeplugin-qkhqf                    3/3    Running    0    30m
172.16.1.102      master  <none>    <none>
root@master:~/openstack#

```

storageClass 작성하기 > 'openstack volume type list' 이름을 그대로 활용한다. 만약 아래의 yaml 파일에서의 타입처럼 'iscsi' 가 없다면 미리 만들어 두어야 한다



```

cinder-storageClass.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cinder-sc

```

```
provisioner: cinder.csi.openstack.org
parameters:
  type: iscsi      # OpenStack에서 정의된 볼륨 타입
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

pvc 작성하기

```
pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cinder-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: cinder-sc
```

테스트용 pod 작성 및 배포

```
test_pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-with-cinder
spec:
  containers:
    - image: nginx
      name: nginx
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: cinder-storage
  volumes:
    - name: cinder-storage
      persistentVolumeClaim:
        claimName: cinder-pvc
```

k8s/openstack에서 확인하기

쿠버네티스에서 확인하기

```
root@master:~/openstack# k get pod,pv,pvc
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx-with-cinder              1/1     Running   0           4m5s

NAME                                CAPACITY  ACCESS MO
DES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIB
UTESCLASS REASON AGE
persistentvolume/pvc-b2563f31-7551-43e5-b6de-3756ad13bda7 1Gi RWO
Delete Bound default/cinder-pvc cinder-sc <unset>
4m8s

NAME                                STATUS  VOLUME
CAPACITY  ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE
persistentvolumeclaim/cinder-pvc Bound pvc-b2563f31-7551-43e5-b6de-3756ad13
bda7 1Gi RWO cinder-sc <unset> 4m9s
root@master:~/openstack#
```

오픈스택에서 확인

```
[root@localhost ~(keystone_admin)]# openstack volume list
+-----+-----+-----+-----+-----+
| ID | Status | Size | Attached to | Name |
+-----+-----+-----+-----+-----+
| 0e10609b-3914-413c-9b74-2236ec5a94dd | in-use | 1 | Attached to node1 on /dev/vdc | pvc-b2563f31-7551-43e5-b6de-3756ad13bda7 |
+-----+-----+-----+-----+-----+

[root@localhost ~(keystone_admin)]#
```

15. VPNaaS 구성

단순히 VPNaaS를 활성화 시키기만 해도 VPN 사용이 가능하다.

설정값은 미리 생성되어 있는 Router 에 적용되므로 라우터의 external ip 주소를 endpoint 로 구성하고 interesting traffic 은 라우터에 연결된 사설 서버넷과 상대방 사설 서버넷을 연결한다. 상대방 endpoint 역시 상대 라우터나 VPN 기기의 public ip

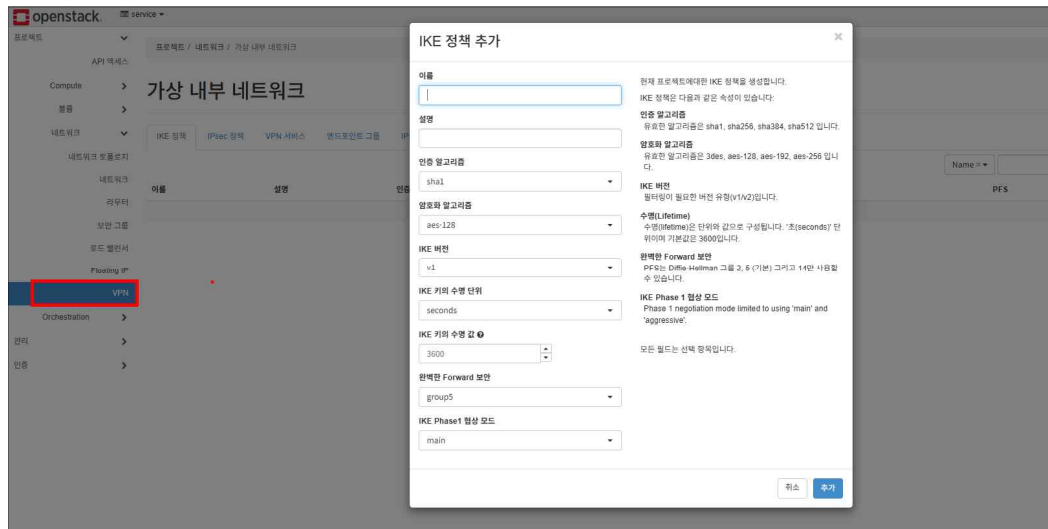
로 구성한다

```
(venv) root@openstack:~# cat /etc/kolla/globals.yml | grep vpn
```

```
enable_horizon_neutron_vpnaas: "{ enable_neutron_vpnaas | bool }{"
```

```
enable_neutron_vpnaas: "yes"
```

```
(venv) root@openstack:~#
```



16. DBaaS : Trove를 이용한 관리형 RDBMS 서비스

1. Trove CLI 설치

```
root@openstack:~# source venv/bin/activate
(venv) root@openstack:~# source admin-openrc-service.sh # service 프로젝트용
(venv) root@openstack:~# pip install python-troveclient
```

2. globals.yml 파일 설정

```
vi /etc/kolla/globals.yml

enable_horizon_trove: "{{ enable_trove | bool }}"
enable_trove: "yes"
trove_guest_image_name: "trove-guest-ubuntu"
```

3. Trove guest 이미지 다운로드 및 등록

```
(venv) root@openstack:~# wget \

https://tarballs.opendev.org/openstack/trove/images/trove-master-guest-ubuntu.qcow2

(venv) root@openstack:~# openstack image create trove-guest-ubuntu \
  --file trove-guest-ubuntu.qcow2 \
  --disk-format qcow2 \
  --container-format bare \
  --public \
  --tag trove \
  --tag mariadb

(venv) root@openstack:~# kolla-ansible reconfigure -i ./all-in-one --tags
trove,horizon
```

4. datastore

```
(venv) root@openstack:~# IMAGE_ID=$(openstack image show trove-guest-ubuntu -f
value -c id)
(venv) root@openstack:~# openstack datastore version create \
  mariadb-10.4 \
  mariadb \
  mariadb \
  "$IMAGE_ID" \
  --active \
```

```
--default \  
--version-number 10.4
```

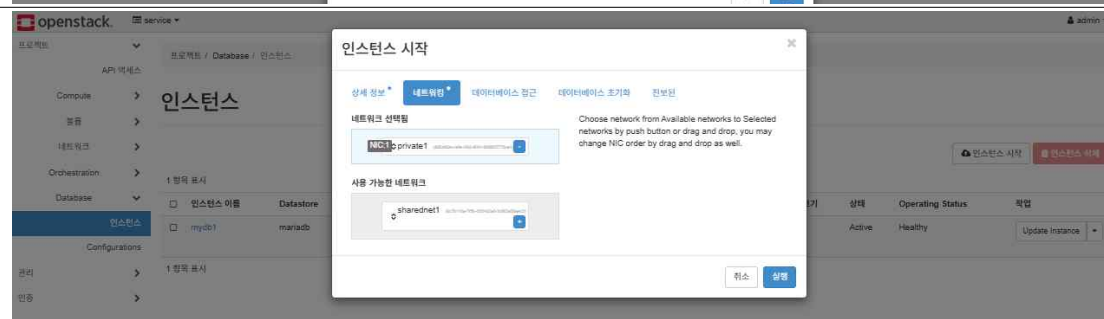
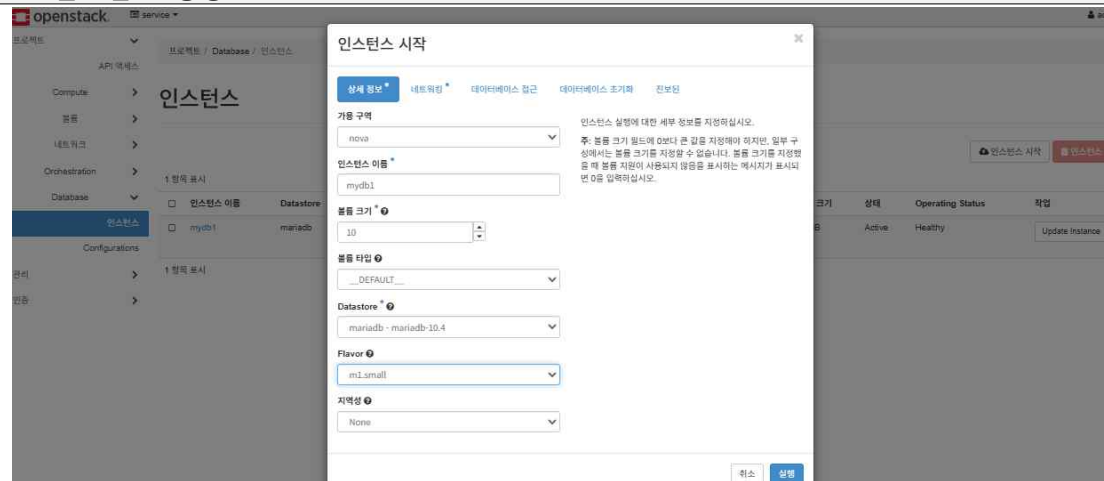
#형식: datastore version create <version-name> <datastore-name> <manager>
<image-id>

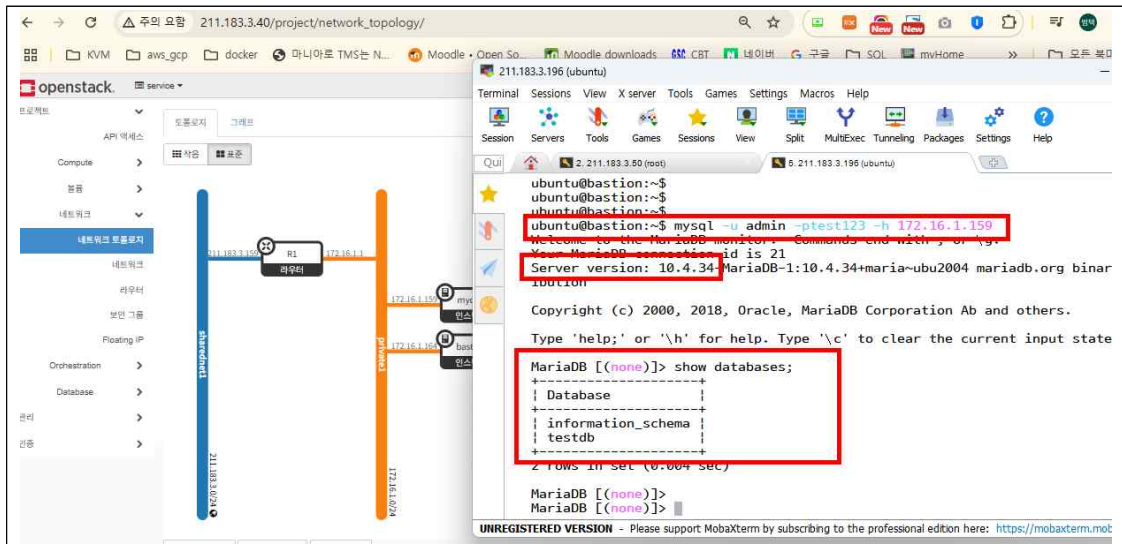
```
(venv) root@openstack:~# openstack datastore version list mariadb
```

ID	Name	Version
2eba86d9-6d03-4750-a974-b3be1c6c4574	mariadb-10.4	10.4

```
(venv) root@openstack:~#
```

5. 인스턴스 생성





17. CaaS : 컨테이너 서비스

globals.yml 수정하기

```
enable_zun: "yes"
enable_etcd: "yes"
enable_kuryr: "yes" # 컨테이너 네트워킹
neutron_plugin_agent: "openvswitch"
docker_configure_for_zun: "yes"
containerd_configure_for_zun: "yes"
containerd_grpc_gid: 42463
```

도커 동작 상태 확인

```
(venv) root@openstack:~# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: e
   Active: active (running) since Fri 2025-08-08 07:37:24 KST; 14min ago
   TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 1581 (dockerd)
```

배포 및 클라이언트 설치

```
(venv) root@openstack:~# kolla-ansible deploy -i ./all-in-one
(venv) root@openstack:~# kolla-ansible reconfigure -i ./all-in-one --tags horizon
(venv) root@openstack:~# pip install python-zunclient
```

Zun API 테스트

```
(venv) root@openstack:~# cat service-openrc.sh

# Ansible managed

# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="} /^OS_/ {print $1}' ); do unset $key ; done
export OS_PROJECT_DOMAIN_NAME='Default'
export OS_USER_DOMAIN_NAME='Default'
```

```
export OS_PROJECT_NAME='service'
export OS_TENANT_NAME='service'
export OS_USERNAME='admin'
export OS_PASSWORD='UhT0JcdVNvwKMZ7XDp1dxikkbpijnOzPThcYyiHh'
export OS_AUTH_URL='http://211.183.3.40:5000'
export OS_INTERFACE='internal'
export OS_ENDPOINT_TYPE='internalURL'
export OS_IDENTITY_API_VERSION='3'
export OS_REGION_NAME='RegionOne'
export OS_AUTH_PLUGIN='password'
(venv) root@openstack:~# source service-openrc.sh
```

```
(venv) root@openstack:~# openstack appcontainer run --name my-nginx \
--net network="private network 의 ID 또는 이름" \
--cpu 1 \
--memory 512 \
nginx
```

885.. 는 private 네트워크의 ID
가장 마지막의 nginx 는 이미지 명

[결과 확인]

```
(venv) root@openstack:~# docker ps | grep zun_compute
f3fb6b6f0156          quay.io/openstack.kolla/zun-compute:master-ubuntu-noble
    "dumb-init --single-..."   13 minutes ago      Up Less than a second (health:
starting)                  zun_compute
(venv) root@openstack:~#
```

와 같은 오류 메시지를 출력하고 zun compute 는 계속해서 배포/중지를 반복하게 된다.
이는 (venv) root@openstack:~# docker exec -it zun_compute tail -n 100 /var/log/kolla/zun/zun-compute.log를 확인해 보면 아래와 같은 오류 메시지를 확인할 수 있다.

```
2025-08-08      09:15:52.041      7      ERROR      zun      File
"/var/lib/kolla/venv/lib/python3.12/site-packages/requests/adapters.py",      line
700, in send
2025-08-08 09:15:52.041 7 ERROR zun      raise ConnectionError(e, request=request)
2025-08-08 09:15:52.041 7 ERROR zun      requests.exceptions.ConnectionError:
HTTPConnectionPool(host='211.183.3.50', port=2375): Max retries exceeded with
url: /v1.26/info (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f69e7be4bf0>:
Failed to establish a new connection: [Errno 111] ECONNREFUSED'))
```

2025-08-08 09:15:52.041 7 ERROR zun

이는 zun-compute 프로세스가 Docker 에 접속하려고 TCP 로 연결을 시도하지만 해당 포트에서 응답이 없다는 뜻이다. 결론적으로 Docker 와 통신이 되지 않는다는 뜻이다. 도커는 unix 소켓 통신을 하므로 잘못된 설정이 zun-compute 에 설정되어 있다는 것을 알 수 있다.

수정해야 한다.

```
(venv) root@openstack:~# docker cp zun_compute:/etc/zun/zun.conf ./zun.conf
```

```
(venv) root@openstack:~# vi zun.conf
```

```
87
88 [docker]
89 api_url = unix:///var/run/docker.sock
90 docker_remote_api_host = unix:///var/run/docker.sock
91
92 █
```

```
(venv) root@openstack:~# mkdir -p /etc/kolla/config/
```

```
(venv) root@openstack:~# cp zun.conf /etc/kolla/config
```

```
(venv) root@openstack:~# docker rm -f zun_compute
```

```
(venv) root@openstack:~# kolla-ansible reconfigure -i ./all-in-one --tags zun
```

컨테이너 생성

```
(venv) root@openstack:~# openstack appcontainer run --name nginx-container
--net network=svc_net1 nginx
```

생성된 포트는 별도의 포트 매핑이 없으므로 floating ip 와의 연계와 security group를 통해 외부 노출 및 외부로부터의 접속이 가능해 진다

외부 노출 서비스(horizon)

The screenshot shows the OpenStack Horizon interface. On the left, a sidebar contains navigation links for 'Compute', 'Network', 'Orchestration', and 'Containers'. The 'Containers' link is selected. The main panel displays the 'Containers' section with a search bar and a table of containers. The table has columns for '이름' (Name), '이미지' (Image), '상태' (Status), and '작업 상태' (Action). One container, 'nginx-container', is listed with the image 'nginx' and status 'Running'. This row is highlighted with a red rectangular box.

이름	이미지	상태	작업 상태
nginx-container	nginx	Running	-

The top screenshot shows the OpenStack dashboard's 'Security Groups' page. A red box highlights a rule for port 80 (HTTP) with protocol TCP and port range 80. The bottom screenshot shows the 'Floating IP' page, with a red box highlighting a floating IP address (211.183.3.199) and its associated fixed IP address (172.16.1.105). Below the screenshots, a browser window shows the 'Welcome to nginx!' page, indicating that the nginx web server is successfully installed and working.

18. Designate : DNS 서비스

globals.yml 파일 수정

enable_designate: "yes"

enable_redis: "yes" # Designate 가 내부에서 Redis를 캐시 및 동기화 백엔드로 사용

designate_backend: "bind9"

designate_ns_record:

- "ns1.beomtaek.pri"

```
dns_interface: "{{ network_interface }}"
```

```
# dns서버는 이제 ens32에 지정되어 있는 211.183.3.50을 사용할 수 있다
```

변경사항 적용 및 반영

```
(venv) root@openstack:~# pip install python-designateclient
```

```
(venv) root@openstack:~# kolla-ansible reconfigure -i ./all-in-one
```

실제 동작 시켜보면 openvswitch 로 인해 발생하는 문제, dial tcp: lookup quay.io on 127.0.0.53:53: read udp 127.0.0.1:36763->127.0.0.53:53: read: connection refused와 같이 Designate 부트스트랩 컨테이너 이미지를 quay.io에서 가져오지 못하는 문제등이 발생하여 정상적으로 배포되지 않는 경우가 자주 발생한다.

1. 호스트 OS DNS 문제 해결

선택1	<pre># DNS서버 직접 지정하기 (venv) root@openstack:~# systemctl disable systemd-resolved --now (venv) root@openstack:~# rm -rf /etc/resolv.conf (venv) root@openstack:~# echo "nameserver 127.0.0.1" >> /etc/resolv.conf (venv) root@openstack:~# echo "nameserver 8.8.8.8" >> /etc/resolv.conf (venv) root@openstack:~# echo "nameserver 1.1.1.1" >> /etc/resolv.conf</pre>
선택2	<pre># 도커 데몬이 자체적으로 127.0.0.53을 참조하는 경우 /etc/docker/daemon.json 에 DNS 설정 { "dns": ["8.8.8.8", "1.1.1.1"] } 이후, systemctl restart docker</pre>

2. 배포 > 간단히 스크립트를 작성하여 변경된 내용을 적용시켜준다

```
(venv) root@openstack:~# cat kolla-deploy.sh
#!/bin/bash

while [ 1 ]
do
    echo "##### openvswitch reconfiguring #####"
    sleep 3
    kolla-ansible reconfigure -i ./all-in-one --tags openvswitch

    if [ $? -eq 0 ]
    then
        kolla-ansible deploy -i ./all-in-one
    else
        echo "##### openvswitch reconfiguring error #####"
    fi
done
```

```

        continue
    fi

    if [ $? -eq 0 ]
    then
        echo "##### DEPLOYED #####"
        break
    else
        echo "##### DEPLOY ERROR #####"
    fi
done

echo "##### FINISHED #####"
(venv) root@openstack:~#
(venv) root@openstack:~# ./kolla-deploy.sh

```

현재 상황

Name	ID	Image	Flavor	OS Image	Status	Tags	Project	State	Age	Action
snc_ubuntu2	test	172.16.2.131, 211.183.3.131	m1.small	0813	Active	novi	None	Running	21시간, 42분	스태팅 변경
snc_ubuntu1	ubuntu	172.16.2.181, 211.183.3.151	m1.small	0813	Active	novi	None	Running	22시간, 36분	스태팅 변경

- 대시보드 상에 새로운 서비스인 "DNS" 가 추가됨
- 인스턴스 2대가 동작 중이며 테스트를 위해 각 인스턴스에 nginx를 이용하여 웹 서버를 실행 중
- DNS서비스는 도메인이름을 질의할 경우 IP 주소를 반환하는 정방향 DNS(Forward DNS) 서비스와 IP주소에 대한 도메인이름을 반환하는 역방향 DNS(reverse DNS) 를 구성할 수 있다
- 테스트를 위해 인스턴스의 /etc/resolv.conf 파일을 아래처럼 구성함

```

# option for /etc/resolv.conf.
nameserver 211.183.3.50
nameserver 127.0.0.53
options edns0 trust-ad
search beomtaek.pri
root@snc-ubuntu2:~#

```

zone 구성

Create Zone

이름 *
openstack.pri.
Zone name ending in .

설명
Details about the zone.

Email Address *
beomtaek@test.pri.
Email address to contact the zone owner.

TTL *
3600
Time To Live in seconds.

유형
Primary
Select the type of zone

× 취소 Submit

DNS Zones

필터 또는 전체 텍스트 검색은 여기를 클릭하십시오.

Create Zone

이름	유형	상태	
openstack.pri.	Primary	Active	Create Record Set

Create Record Set

유형 *
A - Address record
Select the type of record set

이름 *
ubuntu1.openstack.pri.
DNS name for this record set, ending in .

설명
Details about the zone.

TTL *
3600
Time To Live in seconds.

Records
Record
211.183.3.151
Records for the record set.

+ Add Record

× 취소 Submit

이름 *

ubuntu2.openstack.pri.

DNS name for the record set, ending in '.'

설명

Details about the zone.

TTL *

3600

Time To Live in seconds.

Records

Record

211.183.3.131

Records for the record set.

```

(venv) root@openstack:~# curl http://ubuntu1.openstack.pri
<center><h2>HELLO FROM UBUNTU1</h2></center>

(venv) root@openstack:~#
(venv) root@openstack:~# curl http://ubuntu2.openstack.pri
<center><h2>HELLO FROM UBUNTU2</h2></center>
(venv) root@openstack:~#
(venv) root@openstack:~#

```

19. OpenSearch

OpenSearch를 이용하여 이벤트/로그 통합하기

kolla-ansible 은 기본적으로 Fluentd > OpenSearch > OpenSearch Dashboards를 통해 구성 된다.

globals.yml 수정 및 재배포

```

video_drivers: {}
enable_opensearch: "{ enable_central_logging | bool or enable_osprofiler | bool
or (enable_cloudkitty | bool and cloudkitty_storage_backend == 'opensearch') }}"
#enable_opensearch_dashboards: "{ enable_opensearch | bool }"
enable_celbs: no
enable_central_logging: "yes"
#enable_ceph_row: "no"
# of CloudKitty Storage backend version 2.
cloudkitty_storage_backend: "opensearch"

enable_elcu: no
enable_fluentd: "yes"
#enable_fluentd_systemd: "{ (enable_fluentd | bool) and (enable_central_logging

```

```

(venv) root@openstack:~# ./kolla-deploy.sh

```

```

(venv) root@openstack:~# cat kolla-deploy.sh
#!/bin/bash
#

while [ 1 ]
do
    echo "##### openvswitch reconfiguring #####"
```

```

sleep 3
kolla-ansible reconfigure -i ./all-in-one --tags openvswitch

if [ $? -eq 0 ]
then
    kolla-ansible deploy -i ./all-in-one
else
    echo "##### openvswitch reconfiguring error #####"
    continue
fi

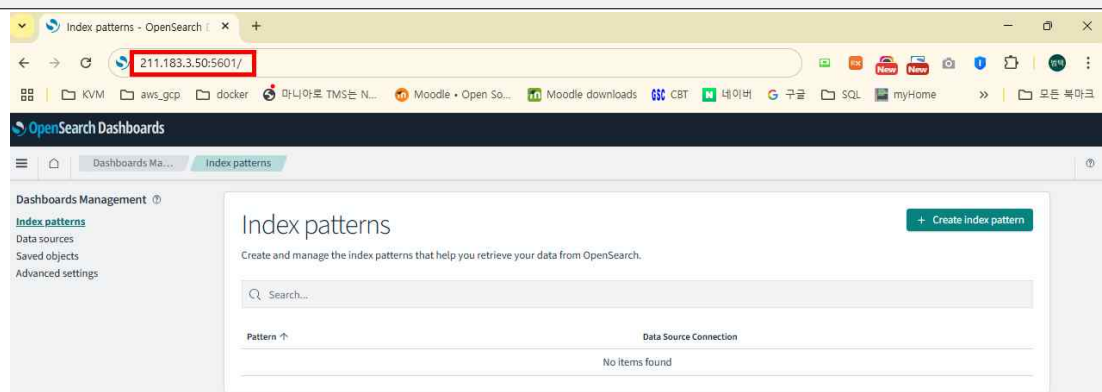
if [ $? -eq 0 ]
then
    echo "##### DEPLOYED #####"
    break
else
    echo "##### DEPLOY ERROR #####"
fi
done

echo "##### FINISHED #####"
(env) root@openstack:~#

```

접속하기

api 주소(vip) 인 49 가 아니라 ens32 의 주소인 .50 으로 접속한다



49 로 접속하면 haproxy를 통해 50으로 접속된다. 이때 haproxy를 통하면 haproxy 에서 는 인증을 통해 접속되므로 vip 인 49는 인증, 50 은 무인증으로 접속이 된다. 둘다 무인증 접속을 위해 /etc/kolla/haproxy/services.d/opensearch-dashboards.cfg 파일을 수정한다.

```

(env) root@openstack:~# vi ₩
/etc/kolla/haproxy/services.d/opensearch-dashboards.cfg

```

```
#userlist opensearch-dashboards-user
# user opensearch insecure-password uxVvhkJPo1p9K1hwhf1xSoMLxmKJdQ95KBJvFQ8AJ

frontend opensearch-dashboards_front
  mode http
  http-request del-header X-Forwarded-Proto
  option httplog
  option forwardfor
  http-request set-header X-Forwarded-Proto https if { ssl_fc }
  bind 211.183.3.49:5601
  default_backend opensearch-dashboards_back

backend opensearch-dashboards_back
  mode http
  # acl auth_acl http_auth(opensearch-dashboards-user)
  http-request auth realm basicauth unless auth_acl
  option httpchk GET /api/status
  server openstack 211.183.3.50:5601 check inter 2000 rise 2 fall 5
```

수정후

(venv) root@openstack:~# docker container restart haproxy

OpenStack 서비스 로그(Nova, Neutron, Keystone ...)는 자동으로 /var/log/kolla/... → Fluentd → OpenSearch에 적재된다.

```
(venv) root@openstack:~# ls /var/log/kolla
ansible.log  designate  haproxy    libvirt    opensearch    proxysql
aodh         fluentd    heat       mariadb    opensearch-dashboards  rabbitmq
ceilometer   glance     horizon    neutron    openvswitch   redis
cinder       gnocchi    keystone   nova       placement
```

OpenSearch Dashboards (ex-Kibana)에서 시각화/검색 가능.

이제 nova(computing 서비스) 관련 로그를 fluentd에서 수집하여 이를 시각화하도록 할 계획이다. /etc/kolla/fluentd/fluentd.conf 파일에 아래의 내용을 추가하되, 기존 <match **> 앞에 작성해야 한다.

```
(venv) root@openstack:~# vi /etc/kolla/fluentd/fluentd.conf

</match>

# 여기에서부터 위쪽은 기존 내용
# === Nova 전용 분기: openstack_python 태그 중 programname 이 nova- 로 시작하는 것만 ===
<match openstack_python>
  @type copy

  # 기존 흐름은 그대로 두고(뒤의 catch-all 로 감),
  # 아래 store 하나를 더해 nova 전용 라벨로 "복제"해서 보냄
  <store>
```

```
@type relabel
@label @NOVA_ONLY
</store>
</match>

<label @NOVA_ONLY>
# openstack_python 중 Nova 계열만 통과시키기
<filter **>
  @type grep
  <regexp>
    key programname
    pattern ^nova-
  </regexp>
</filter>

# Nova 전용 인덱스에 적재
<match **>
  @type opensearch
  host 211.183.3.49
  port 9200
  scheme http

# 둘 중 하나 방식 사용 (둘 다 쓰지 말고 하나만)
# ① logstash 스타일:
logstash_format true
logstash_prefix nova

# ② 직접 인덱스명 지정 원하시면 위 2줄 대신 아래 1줄:
# index_name nova-%Y.%m.%d

suppress_type_name true
reconnect_on_error true
request_timeout 30s

<buffer>
  @type file
  path /var/lib/fluentd/data/opensearch.buffer/nova.*
  flush_interval 15s
  chunk_limit_size 8M
  retry_forever true
</buffer>
```

```

</match>
</label>

# -----
# 여기에서부터 아래는 기존내용
<match ** >

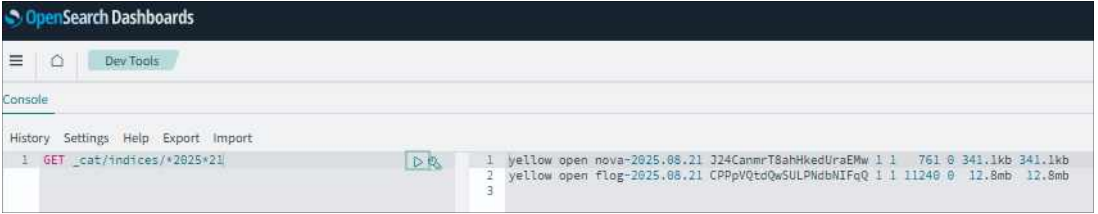
```

Nova 로그는 이미 rewrite_tag_filter에 의해 openstack_python 태그로 재태깅 되어 있음.
위 블록은 그 openstack_python 스트림을 한 번 “복제(copy)”하여 @NOVA_ONLY 라벨로 보낸 뒤, grep으로 programname이 nova-로 시작하는 것만 골라서 nova-* 인덱스로 저장.

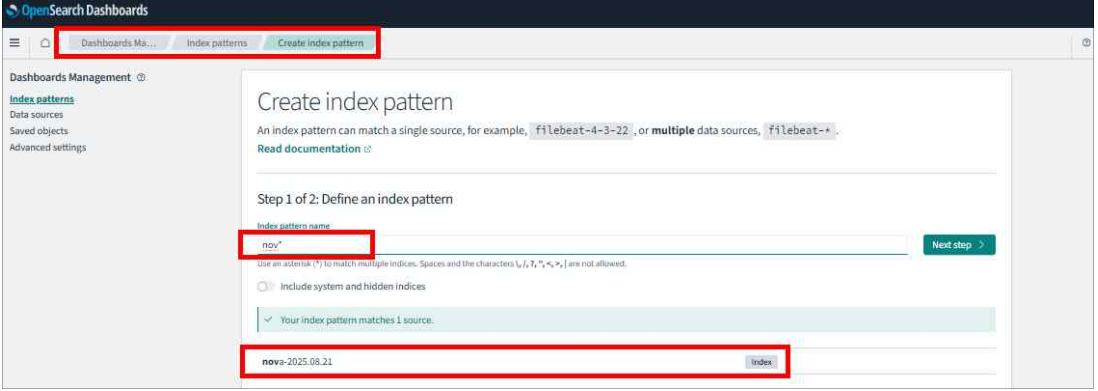
원래 흐름(전체 로그 → flog-*)은 그대로 유지되므로 결국 2개의 인덱스가 생성된다.

확인

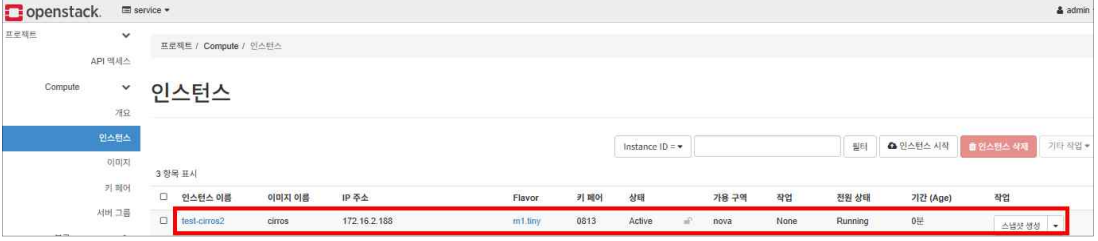
1. 생성된 인덱스 확인



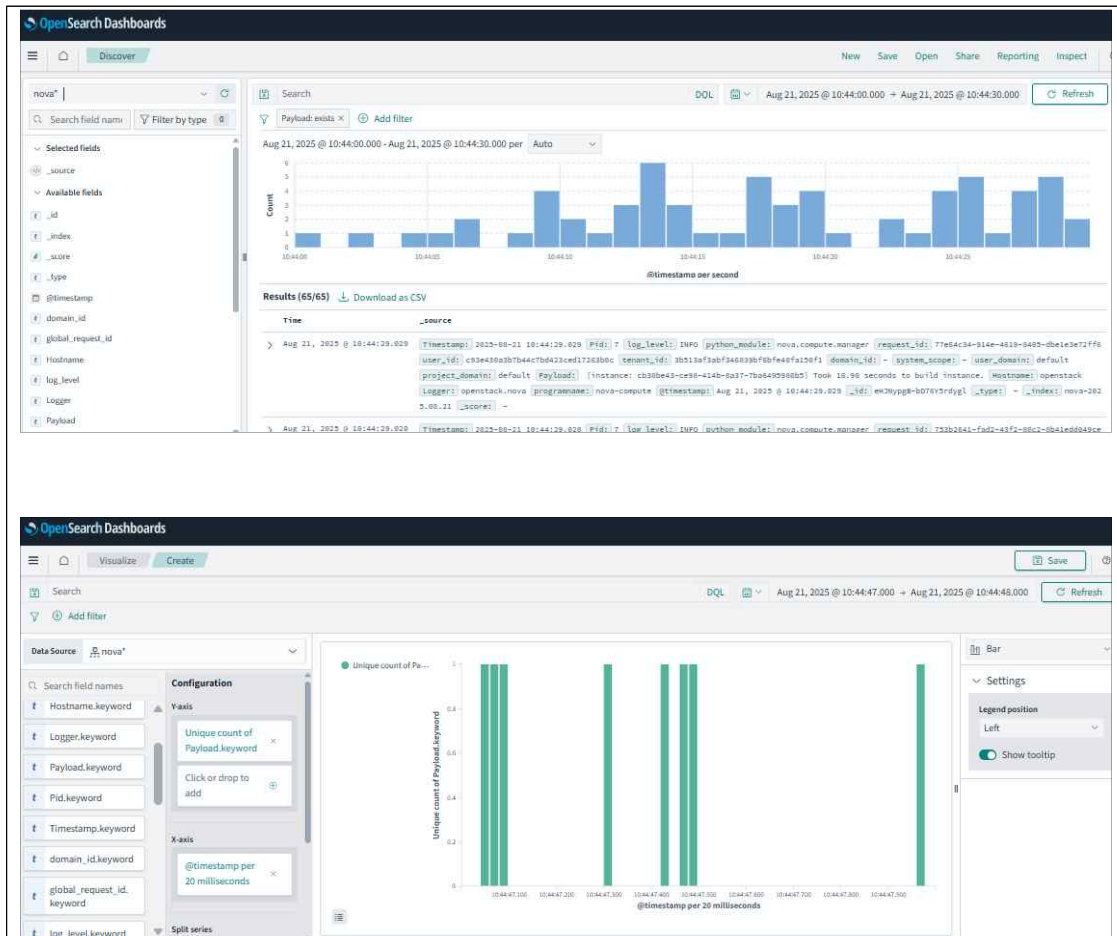
2. 시각화를 위해 인덱스 패턴 생성



3. 인스턴스 생성해 보고 로그 확인해 보기



인스턴스 이름	이미지 이름	IP 주소	Flavor	키 패어	상태	가용 구역	작업	현재 상태	기간 (Age)	작업
test-cirros2	cirros	172.16.2.185	m1.tiny	0013	Active	nova	None	Running	0분	스냅샷 생성



위와 같은 방법을 이용하면 nova 외에 다른 서비스에 대해서도 모니터링이 가능하다

만약 다른 오픈서치를 현재의 오픈서치 대시보드로 연결하고 싶다면

```
(venv) root@openstack:~# vi W
/etc/kolla/opensearch-dashboards/opensearch_dashboards.yml
(venv) root@openstack:~# cat /etc/kolla/opensearch-dashboards/opensearch_dashboards.yml
data.search.usageTelemetry.enabled: false
logging.dest: /var/log/kolla/opensearch-dashboards/opensearch-dashboards.log
opensearch.hosts: http://211.183.3.49:9200
opensearch.requestTimeout: 300000
opensearch.shardTimeout: 0
opensearch.ssl.verificationMode: full
opensearchDashboards.defaultAppId: discover
server.host: 211.183.3.50
server.port: 5601
data_source.enabled: true
(venv) root@openstack:~#
```

위와 같이 내용을 추가한 뒤, docker container restart opensearch_dashboards를 하면 아래처럼 기존의 데이터 소스 추가 메뉴외에 OpenSearch를 확인할 수 있다.

