열한번째 주 둘째 날 oracle live 환경에서 openstack 구축, core compoent, 각 node 구성

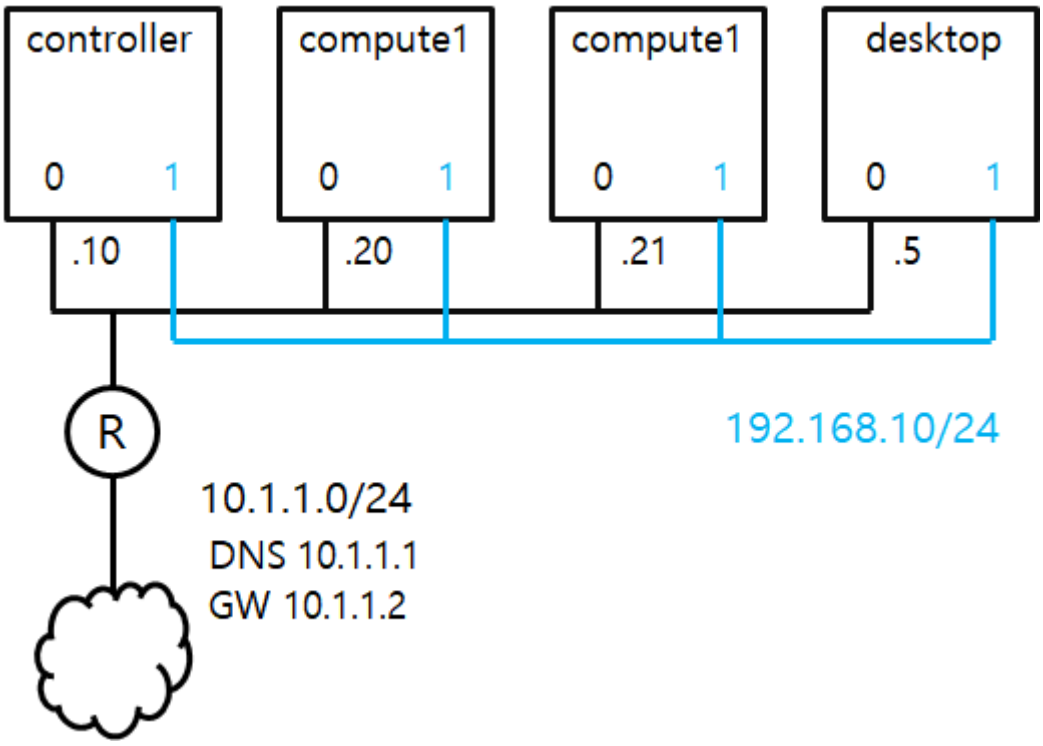
노트북:	필기노트		
만든 날짜:	2019-08-06 오후 9:19	**업데이트:**	2019-08-07 오전 12:13
작성자:	이종민		
태그:	오픈스택		
URL:	https://github.com/torlgit/cccr/blob/master/11th/48day		

48day

오라클 라이브(중첩 가상화) : 가상화 안에서 가상화

ip setting
home directory 밑에 ip.cfg를 /etc~network-scr.. 파일에 넣어두고 network.service 재시작
rocky ver 설치.

controller, compute1, compute2, desktop 네트워크 구성



Domain : nblab.com

cp /root/ifcfg-* /etc/sysconfig/network-scripts/
systemctl restart network
sh audit.sh

yum install centos-release-openstack-rocky
yum install openstack-packstack
packstack --gen-answer-file answers.txt *

vi answers.txt

```
CONFIG_DEFAULT_PASSWORD=P@ssw0rd
CONFIG_HEAT_INSTALL=y
CONFIG_NTP_SERVERS=kr.pool.ntp.org
CONFIG_COMPUTE_HOSTS=10.1.1.20,10.1.1.21
CONFIG_KEYSTONE_ADMIN_PW=P@ssw0rd
CONFIG_LBAAS_INSTALL=y
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=extnet:br-ex
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-ex:eth0
CONFIG_NEUTRON_OVS_TUNNEL_IF=eth1
CONFIG_NEUTRON_OVS_TUNNEL_SUBNETS=192.168.10.0/24
CONFIG_PROVISION_DEMO=n
```

packstack -d --answer-file answers.txt *

맨 처음 프로젝트
nova
swift

core compoent 라고한다.

core compoent = answer file 구성
openstack의 최초 버전은 nova(nasa), swift

nova
nasa는 슈퍼컴퓨팅 형태로 많은 시스템을 병렬처리 하거나 관리할 필요성이 있어서 네뷸라 시스템을 구축해서 운영했는데, 그 내용을 오픈소스로 만든게 nova이다.

swift(오픈소스)는 서버를 빌려주는 호스팅 업체에서 만든 클라우드 스토리지이다.
클라우드 스토리지는 drophox, google driver 와 유사하다. 통신사 중 kt는 cloud stoage를 제공한다.

nova와 swift의 상관관계는 없었다. 밀접한 관계를 갖고 있지 않았는데, 이 두 가지로 오픈스택이라는 프로젝트를 만들었다.
그 이후 필요한 요소 neutron, keystone, glance, cellometerm, cinder 등을 만들었다.

가장 기본적인 core service

compoent(오픈소스 구성요소)	Project	Role
Dashboard	Horizon	Web Interface(제공), Apache python(Django)
Compute	Nova	Instance(vm) Hyperviser(kvm, vm, 등 ..)
Network	Neutron	SW/routing, NAT, DHCP, LBaas FW ... Instance <- network
Image	Glance	Image(os pre-install) -> Instance
Block	Cinder	Block -> Instance LVM
Object	swift	Image 저장소, instance backup 저장소
Identification	keystone	SSO인증, 사용자/프로젝트(테넌트), Quota, 서비스 목록, Endpoint
Telemetry	Ceilometer	사용량, 알람
Orchestration	Heat	YAML(markup language) - xml /, IaC : 인프라를 코드로 구현
		-> Telemetry + Orchestration = Auto Scaling

Dashboard
오픈스택 환경을 운영 및 관리 할 수 있는 웹 기반 인터페이스

기업에서 오픈스택을 사용할 때 웹 기반 인터페이스를 직접 만드는 경우가 많다.
사용자가 사용하기에는 별로 좋은 ui, ux가 아니다. // ux : 사용자 경험, 인터페이스의 디자인 사용자 편의성이 좋지 않다.
명령어로 구조를 보는게 우리의 목적

Compute
Nova, 가상 컴퓨터를 제공, 가장 핵심적인 요소
라이프 사이클은 생명주기, 가상화에선 필요에 의해 생성하고 삭제하기 까지 과정
클라우드 instance는 몇 시간에서 며칠을 사용, 가상컴퓨터는 며칠에서 몇 년을 사용, 실제 피지컬 서버는 몇 년에서 수십년까지 사용
클라우드 instance는 라이프 사이클이 짧다. 클라우드는 필요한만큼 사용하고 비용을 지불하는 형태

가상컴퓨터를 생성하려면 nova가 직접하는게 아니라 hyperviser에 명령을 내린다. (nova가 libvirtd 에 요청)

vm -> QEMU -> KVM
vm이 늘어나면 QEMU가 늘어나는데, libvirtd가 관리한다.

Network
nova-network -> Quantum -> Netron // 오픈스택 네트워크의 발전형태

로드밸런서, 보안장비, 방화벽은 L4~L7 까지의 네트워크 기능을 사용하는데, Nova network로는 할 수 없어서 새로운 프로젝트를 만들었다.
Quantum = Neutron 둘은 같다고 보면 된다. nova는 퀀텀과 뉴트론 나오기 전에 네트워크 하이퍼바이저
Instance에 network를 제공하는게 기본적인 목적이다.

Image
가상컴퓨터는 전부 다 파일로 구성이 되어있다.
기본적으로 2가지로 구성이 되어있는데 metadata, disk 이 두 가지를 타르 아카이빙 한게 ova 이다.
하드웨어 관련된 정보를 metadata라고 하고 xda, xdb 등 block 장치를 disk 파일 이라고한다.

virtualbox = kvm을 가지고 만든 형태
vdi = virtualbox | vmdk = vmware | qcow2 = kvm | vhd = hyper-v
요즘은 소스가 오픈 되어 있어서 다른 프로그램에서도 다 사용이 가능하다.
가상화에서도 type2 가상화는 이미지 기능이 없다. ova를 import 시키는 형태
type1은 template 기능이 존재한다. 미리 설치된 운영체제를 복제해서 가상컴퓨터를 찍어낸다.
미리 설치된 운영체제를 가지고 배포, 찍어낸다. = deploy

대부분의 클라우드는 설치를 지원하지 않는다. Image service는 deploy, 만들어 질때만 관여한다.

Block
디스크 기반, 분산처리 기반, 슈도(Pseudo)
데이터에 입출력이 블록단위로 이루어지는걸 블록장치 라고한다. 파일시스템이 있는 instance가 사용한다.
vm에서 instance가 사용하기 위한 block deivce 이다. 논리볼륨은 instance에 제공, Nova(kvm), Cinder가 block device를 제공

기본적으로 인스턴스는 kvm이 제공하고 지우면 kvm 블록 장치를 지운다. = 썼다가 필요 없으면 버리는 형태
Cinder는 라이프 사이클과 nova의 라이프 사이클은 별개이다. 데이터를 영구적으로 저장하고 싶으면 cinder에 저장한다.
데이터를 영구적으로 저장할 필요가 없다면 instance만 만들어서 비용을 지불하면된다.

block 장치를 만들었다는건 block 장치에 대한 비용도 지불해야한다. -> 클라우드의 모든건 비용과 연관 되어있다.

블록장치는 정형화된 데이터를 저장 = 데이터베이스 들어가야 되는 내용, 형식이 정해져있다.

Object
cloud storage, google drive, dropbox 와 같은 형태
block storage가 instance를 저장할 저장소라면 swift는 client가 사용할 저장소 web 기반 스토리지 이다.

비정형화된 데이터를 저장, 형태가 없이 아무거나 막 집어넣는 형태라고 볼 수 있다.

보통 미디어 데이터를 저장한다. 대용량의 데이터들을 저장.
유튜브, 넷플릭스 등의 데이터를 블록장치에 저장할 수 없다. -> 페타바이트 저장장치

web storage라서 정적인 콘텐츠 저장도 가능하다. => 간단한 정적 콘텐츠의 웹 서버 구현이 가능하다.

Glance 실제 이미지를 저장하는 저장소로도 쓸 수 있다. swift는 mirror 구성과 비슷하다고 생각하면 된다. 기본적으로 복제를한다.
block 장치를 묶어서 object로 만드는 기술을 사용했다.

백업을 하면 swift에 저장해두고 swift 비용만 지불해서 다시 필요에 의해 끄집어내서 가동 시키는 형태
인스턴스를 띄워놓는것보다 가격이 저렴하다.

swift는 오픈스택이 처음 나왔을 때 kt가 storage를 swift로 구현해서 우리나라에서도 많이 알려졌다.
레퍼런스 (검증)을 kt가 storage를 구현함으로써 검증 됐다고 볼 수 있다.

3대 스토리지 image, block, object

Indentification
프로젝트는 horizon에서 쓰는 용어
keystone 에서는 리소스를 사용하는 기본적인 스콧 범위, 오브젝트를 분리 시킬 수 있는 그룹 리소스를 구별해주는 프로젝트라고 한다.
논리적으로 isolation (분리) 시킨다. Quota 프로젝트를 분리 시킨다.

catalog
서비스 목록과 endpoint를 가지고 있다. compoent를 서비스 라고 하는데 end point 서비스에 접근하기 위한 최종 포인트, 위치 keystone이 제공.

Telemetry
알람 기능, 모든 리소스에 대한 사용량을 측정, 알람

Orchestration
하드웨어를 구성하고 설치하고 소프트웨어를 구성하고 설치하는 코드로 관리한다.
IAC , 가상컴퓨터 그 자체, 그 안에 들아가는 소프트웨어 프로그램을 실행, 구현한다.
오픈스택 내에서 모든 compoment(서비스)를 yaml file을 이용해 조율한다.

cpu, ram 특정 기준을 넘게되면 자동으로 복제를 한다. 기준이 미달되면 다시 복제한걸 지운다.
database 같은건 자동 스케일링을 사용하지 않는다. (못한다고 볼 수 있다) 보통 web, was가 스케일링 대상

statefull - 저장된 데이터가 있다. (DB) // 부품을 추가하는 형태
stateless - 상태가 없다 (web, was) // 컴퓨터를 한 대 더 사는 형태

AIO(All-In-One) : 하나의 시스템에 모든 기능을 설치하는 형태
PoC(Proof-Of-Concept) : 필요한 기능이 제대로 작동하는지 확인하는 절차, AIO로 구성한다.
test용이므로 test용도로 많이 사용하지만, 기업에서 사용하는 형태의 자료는 없다.

packstack의 목적은 Poc에 있는데 kolla, ansible을 이용해서 오픈스택을 설치

packstack --allinone
packstack --> redhat 계열 openstack 자동화 설치 툴
All-In-One --> 하나의 시스템에 모든 기능을 설치하는 형태
둘을 합쳐서 생각하면 될 거 같다.

오픈스택은 4가지 용도가 있는데, 분리를 시켜서 설치를 한다.
기능을 시스템 별로 분리 시킨다 = node

controller node :
오픈스택 전체를 컨트롤 하는 용도
horizon, keystone, heat, ceilometer(데이터베이스로 이용), glance, mariadb, message queue(message brocker, AMQP=Advanced Message queue protocol), RabbitMQ
모든 정보는 database에 저장 되어있다. => 그 말은 db에 문제가 생기게 되면 아무것도 할 수 없게 된다.

디스크가 아닌 메모리에 있다. 먼저 들어가면 먼저 나온다.
큐는 중간에 뚫려있는 원통이라고 생각하면 된다.

스택은 양동이를 생각하면 되는데 처음에 들어가면 제일 마지막에 나온다.
비동기화 형식으로 메시지를 저장하기 위한 방식

요즘 웹은 비동기화 형태이다.
웹의 규모가 커지는데 그걸 다 데이터베이스에 저장 해야하는데, 내용들이 맨 처음에 보이고 was가 접근 제어를 해준다.
동기화 방식은 글을 올리고 동기화 될 때까지 아무 작동할 수 없다. 백그라운드에서 다른 작업을 하는 형태와 같다.

message queue가 없으면 동기화 방식이다. 동기화 방식은 좋지만 많은 작업을 할 때 시간이 오래 걸린다.
비동기화 하는게 message queue의 목적이다. 웹 어플리케이션에는 다 AMQP 표준으로 정해져있다.

controller를 여러 개 두는게 불가능하다. 상태를 가지고 있어서 하지만 한 대만 둬도 안되는게 DB 장애 발생 시 아무것도 못하게 된다.
이중화를 하는건 cluster 밖에 안된다. linux cluster 운영체제와 운영체제를 클러스트한다.
cluster를 해야 데이터베이스, controller 이중화가 가능하다. -> Pacemaker
Pacemaker : 클러스터에서 사용되는 오픈 소스 고가용성(high availability) 자원 관리자 소프트웨어

Network node :
현재는 없다. neutron이 설치되는 node -> 현재는 controller에 있다.
모든 외부와의 입출력을 neutron이 처리한다.

North-South Traffic = 내부와 인터넷망 사이에 트래픽을 의미한다.
West-East Traffic = 내부 트래픽을 의미한다.

요즘은 가상화, 클라우드 가상컴퓨터 간의 통신, 내부의 통신 규모가 많아서 동서방식을 사용

인스턴스가 인터넷을 하거나 인터넷에서 인스턴스를 거쳐서 들어올 때 뉴트론을 통해서 들어간다.
network service가 망가지면 north-south가 불가능하다.

compute node :
nova
kvm

storage node :
별도의 시스템으로 구축한다.
block - cinder
object - swift

프로젝트 *
사용자 *
Network - External 물리/인터넷 *
Internal Instance
-> Subnet
Image -> cloud용 이미지
Security Group = (FW)
Floating IP (DNAT)
ssh Keypair
Flavor (hw spec) *
Instance
Block
Object

*는 관리자가 할 수 있는 영역
Cloud는 Self Service 관리자가 계정을 생성해주면 필요한만큼 만들어쓴다.
가상화는 모든것들이 관리자 권한이 있어야한다. = Self Service 개념을 가지고 있지않다.

사용자는 프로젝트가 있어야한다.

Network
두 가지 종류의 네트워크가 있다.
external : 실제로 인터넷을 할 네트워크
internal : 인스턴스 네트워크 (대역대가 관계없다) 네트워크 대역이 다르다보니 라우터가 필요하다. -> Routing

network는 껍데기, external, internal을 지정
subnet을 실제 네트워크로 지정, 네트워크에 subnet이 없을 수 없다.
ip 대역, dhcp 사용유무 세부적인 사항, routing table

Image
image를 만들 때 ssh, 암호 등을 제거 해야한다. = 가상화나 탬플릿은 이미지에 sealing, 봉인(밀봉)을 해야한다.

Security Group
인스턴스에 부여할 그룹을 방화벽이라고 한다.

Floating IP
router는 SNAT (source nat)를 기본적으로 갖고있다. DNAT (destination nat)
- 내부에서 외부로 나갈 수 있ㅇ지만 외부에서 내부로 들어올 수 없다. / floating ip는 DNAT를 설정해준다.
- 외부에서 찾아와야 할 네트워크는 floating DNAT 설정이 있어야한다. (외부에서 알 수 있는 고정 IP)
- 포트포워딩과 매커니즘 차이는 없다. DNAT를 사용하기에 포트포워딩을 하지 않는다.

SSH Keypair
ssh key 개인키를 가지고 ssh 인증을 한다. 인스턴스에 공개키가 들어간다.
인터넷에서 받은 계정은 password가 없기 때문에 ssh 키 기반 인증을 해야한다.

Instance
위 내용을 가지고 Instance를 생성한다.

flavor

가상화에서는 일반적으로 cpu, memory, disk를 마음대로 설정이 가능한데, cloud는 정해져 있는 스펙에서 골라야한다.
이름에 따라서 사이즈가 다르게 설정된 걸 볼 수 있다. (아마존에서 따온 형태)

부수적인 내용
Block
Object

관리자는 openstack의 모든 관리자. 모든 걸 관리 할 수 있다.
관리자는 admin tab에서 모든 리소스를 관리 할 수 있다.
admin tab은 관리자만 나온다.