k8s - 쿠버네티스 아키텍처 (2)

k8s는 어떤 사상으로 움직이는지 알아보자

클러스터 인스턴스 만들기

Kubespray로 클러스터 구성을 위한 VM instance 5를 만들어줍시다.

OS 는 Ubuntu 18 LTS로 설정 후 실행 시

```
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1096-gcp x86_64)
```

grep 으로 google cloud에서 설정한 cpu와 일치하는지 확인 후

```
lscpu | grep -i -E "^CPU\(s\):|core|socket"
```

```
deed1515@instance-1:~$ lscpu | grep -i -E "^CPU\(s\):|core|socket"
CPU(s):                2
Thread(s) per core:    2
Core(s) per socket:    1
Socket(s):             1
```

이후 앤서블에서 원격 접속을 위한 ssh key를 생성 후 공개키를 각 노드를 담당할 인스턴스에 배포 한 후

(구글 클라우드에서 메타 데이터 이용해 배표 했습니다.)

```
d1515@instance-1:~$ ls -al .ssh/
al 24
------ 2 deed1515 deed1515 4096 Jan 14 12:48 .
xr-xr-x 7 deed1515 deed1515 4096 Jan  7 11:49 ..
------- 1 deed1515 deed1515  419 Jan 14 12:48 authorized_keys
------- 1 deed1515 deed1515 1679 Jan  7 09:48 id_rsa
-r--r-- 1 deed1515 deed1515  401 Jan  7 09:48 id_rsa.pub
-r--r-- 1 deed1515 deed1515  444 Jan  7 10:00 known_hosts
```

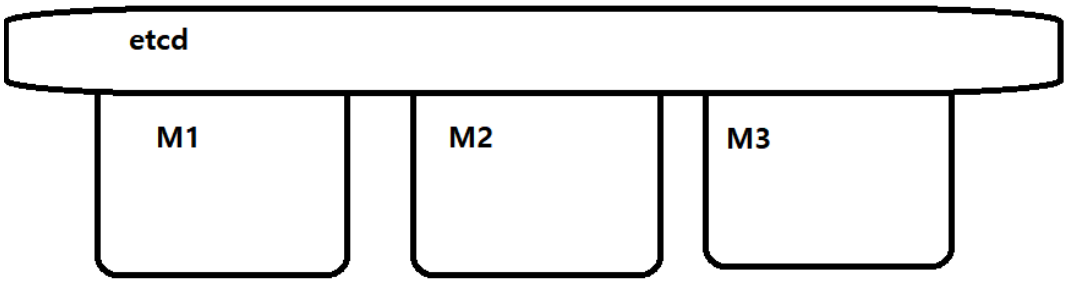
```
# key gen 이후 ssh 설정 확인
ssh-keygen -t rsa
ls -al .ssh/

## !!! Google 메타데이터 설정 이후

# ssh 연결 체크
ssh {instance name} hostname
```

git에서 Kubespray v2.11.0 설치 이후 필요한 팩키지를 설치해 줍니다. 해줍니다.

```
deed1515@instance-1:~/kubespray$ git branch
* (HEAD detached at origin/release-2.11)
  master
  release-2.11
  v2.11.0
```

```
## !!! kubespray v2.11.0버전 및 pip 설치 이후 requirement.txt.로 pip install
pip install -r requirements.txt
vi inventory/mycluster/inventory.ini

#root권한으로 ansible -playbook init
ansible-playbook -i inventory/mycluster/inventory.init -v --become --become-user=root
 cluster.yml
```

```
[all]
intance-1 ansible_ssh_host=10.142.0.11 ip=10.142.0.11 etcd_member_name=etcd1
intance-2 ansible_ssh_host=10.142.0.12 ip=10.142.0.12 etcd_member_name=etcd2
intance-3 ansible_ssh_host=10.142.0.13 ip=10.142.0.13 etcd_member_name=etcd3
intance-4 ansible_ssh_host=10.142.0.14 ip=10.142.0.14
intance-5 ansible_ssh_host=10.142.0.15 ip=10.142.0.15

[kube-master]
instance-1
instance-2
instance-3

[etcd]
instance-1
instance-2
instance-3

[kube-node]
instance-4
instance-5

[calico-rr]

[k8s-cluster:children]
kube-master
kube-node
calico-rr
```

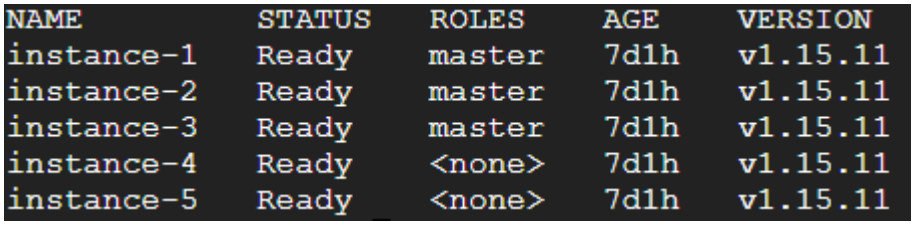


이런 모양으로 설정되는듯?

etcd → 클러스터 데이터 저장소 (워커노드와 같은 노드)

이후 root에서 kubectl로 node를 확인하면

```
#root 계정이동
$ sudo -i
kubectl get node
```



NAME	STATUS	ROLES	AGE	VERSION
instance-1	Ready	master	7d1h	v1.15.11
instance-2	Ready	master	7d1h	v1.15.11
instance-3	Ready	master	7d1h	v1.15.11
instance-4	Ready	<none>	7d1h	v1.15.11
instance-5	Ready	<none>	7d1h	v1.15.11

쿠버네티스 기본 사용법 배우기

clude VM 에서 실행 해 본 뒤 로컬에서 kubectl으로 echoserver pod를 사용해보겠습니다..

```
# kubectl [command] [Type] [name] [flags]
kubectl get pods

# echo server create -> service
kubectl run echoserver --image="k8s.gcr.io/echoserver:1.10" --port=8080
kubectl expose po echoserver --type=NodePort

# 확인해보기
kubectl get pods
kubectl get service

# port foward
kubectl port-forward svc/echoserver 8080:8080

# 신규 쉘에서 통신 테스트
kubectl logs -f echoserver
curl 127.0.0.1:8080

#pod 삭제
kubectl delete pod echoserver
```

```
#클러스터 자원 확인
kubectl api-resources
```

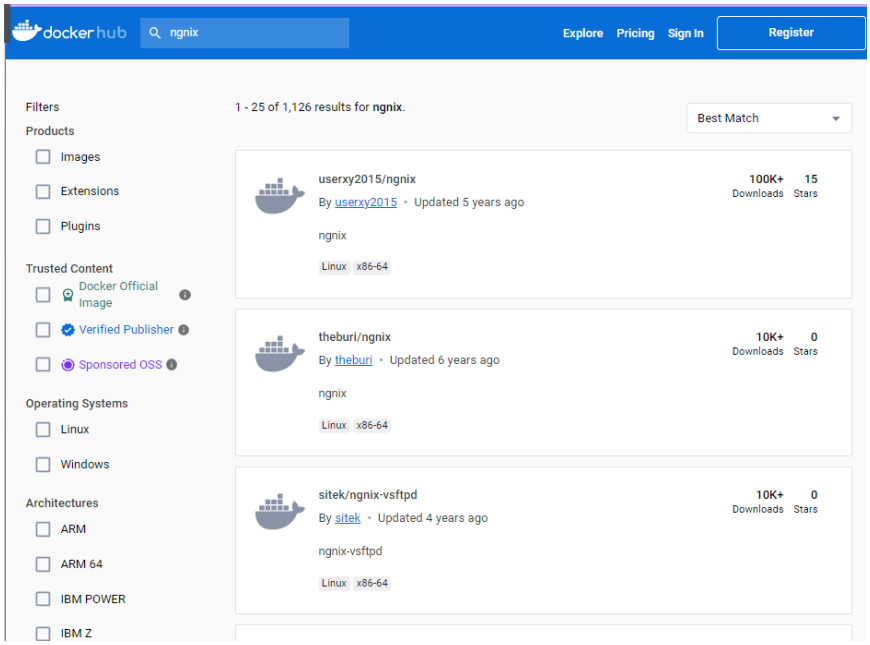
```
PS C:¥Users¥user> kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
```

api 이름 | 단축명 | 버전 | 하위자원여부 | 객체 자원항목

```
#docker-desktop으로 context를 이동해 다중클러스커 접근 시 사용(?)
kubectl config use-context docker-desktop
```

```
# kubectl run {디플로이} --image {컨테이너 이미지} --port={포트번호}
kubectl run ngnixapp --image niinx --port=80
kubectl create deployment ngnix-app --image nginx

#pod 및 deployments 확인
kubectl get pods
kubectl get deployments

kubectl scale deploy ngnix-app --replicas=2
```

이미지 파일을 도커허브에서 찾아 설치됩니다.



https://hub.docker.com/search?q=ngnix

AVvaileiable → 헬스체크 이후 사용가능한 pod개수

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx-app
spec:
  selector:
    matchLabels:
      app: nginx-app
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

```
# kubectl apply 명령어로 선언적 관리 가능(권장!)
kubectl apply -f ngnix-app.yaml
```

쿠버네티스 연결을 담당하는 서비스

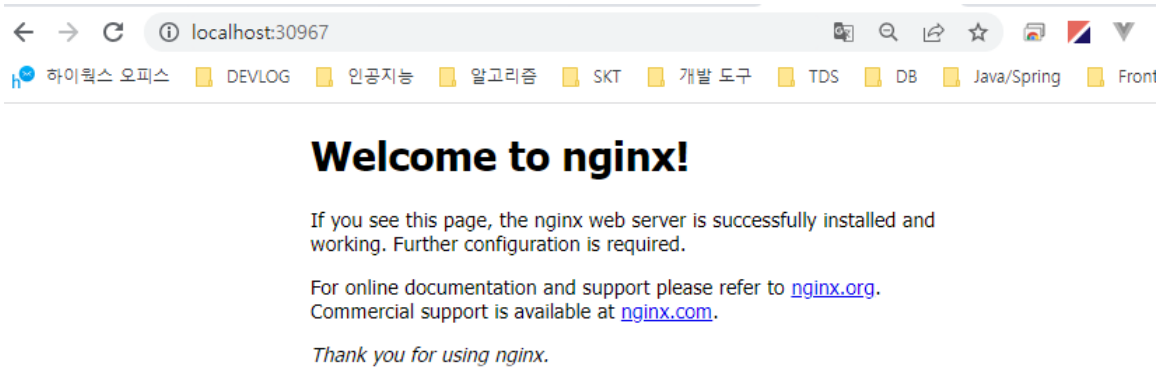
deployments 를 접근 가능하게 하는service를 생성해봅니다

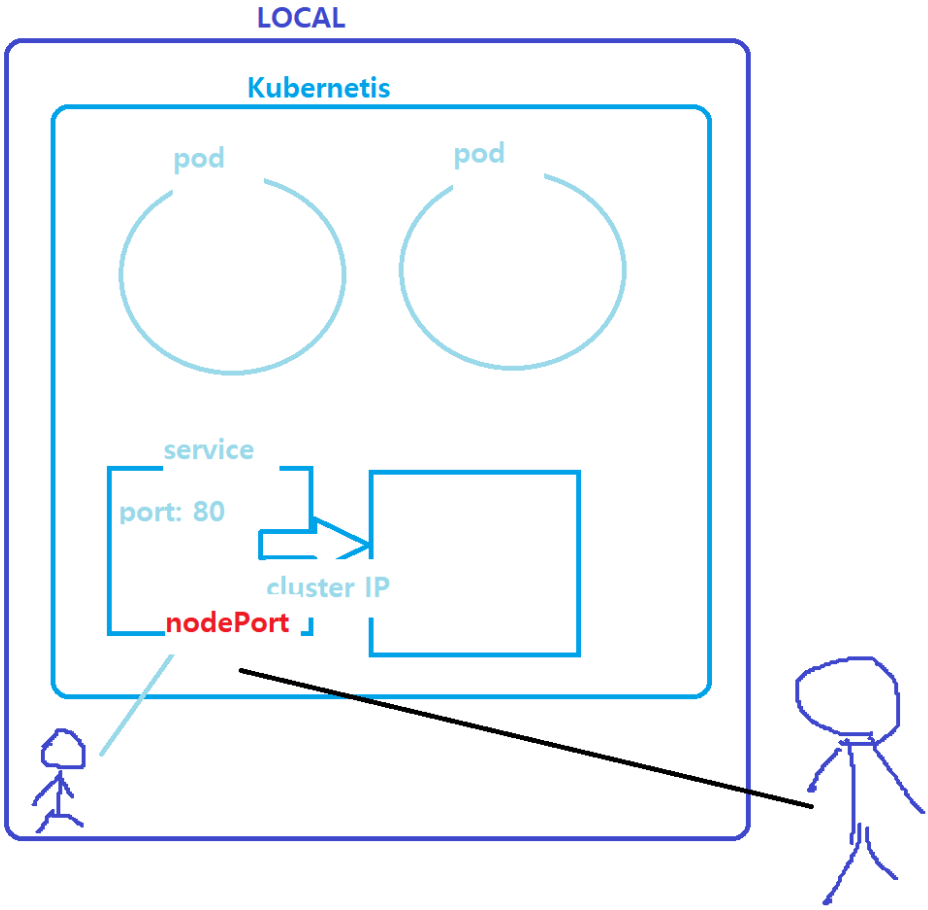
```
# NodePort Type으로 nginx-deployment생성
kubectl expose deployments nginx-deployment --type=NodePort

# 생성 확인
kubectl get service
kubectl describe service nginx-deployment
```

```
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
echoserver          NodePort    10.101.236.91   <none>        8080:31547/TCP   93m
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          7d7h
nginx-deployment    NodePort    10.108.159.1    <none>        80:30967/TCP     7m41s
```

이름 **타입** **클러스터 IP** **외부 IP** **포트**

port의 80:30967중 30967으로 접속 시 nginx서버를 접속 할 수 있다.





서비스의 유형과 통신 형태는 이후 챕터인, 서비스에서 설명 예정입니다.
