

2024.07.18

오전

```
kubectl create -f ./nginx-pod.yaml
```

```
encore@myserver01:~$ k cluster-info
```

```
encore@myserver01:~$ k get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
myserver01	Ready	control-plane	20h	v1.26.15
myserver02	Ready	<none>	20h	v1.26.15
myserver03	Ready	<none>	20h	v1.26.15

control-plane이 마스터다

nodes실질적인 컴퓨터 현재에서는 가상머신이다.

```
encore@myserver01:~$ kubectl delete pod nginx-pod
```

정의파일 manifest파일에서 pods만들고 파일올려서 일괄삭제 등등한다.

pod생성

```
k run hello-world --image=hello-world --restart=Never
```

쿠버네티스는 죽어도 다시부활이 default인데 never은 죽어도 그냥 놔둬라..

```
k get all
```

pod 정보와 service 정보

쿠버플로우

```
k get pod -namespace
```

```
k get pod -n calico-system
```

```
k get pod --namespace calico-system
```

calico-system은 네트워크

쿠버플로우는 클러스터에서 움직이는데, 하나의 공간에 소속되어있는것처럼 되어있어야하므로 소통이되어야한다.

cni computer network interface

calico

calico가 설치되는 순간 k get nodes가 missing에서 ready로 바뀐다.

calico-system의 coredns는 calico가 설치전 running이 아니다.

csi-node-driver는 안에 컨테이너가 2개씩 들어가있다.

vim nginx-test01.yaml

```
apiVersion:v1
kind:Pod
metadata:
  name:nginx01
spec:
  containers:
    - name:nginx-test01
      image:nginx:latest
```

pod로 해주고 metadata에 object로 선언하기

encore@myserver01:~\$ k apply -f nginx-test01.yaml

k delete apply -f nginx-test01.yam

디플리먼트

k8s를 구성하는 핵심

레플리카셋 파드개수관리

파드를 관리하기 위한 Deployment

ReplicaSet 파드의 개수만큼 복제??

yaml manifest파일에 replicaset이 명시되어있어야하고 유지해야하는 파드개수 가용성 보장

레플리카셋을 관리하는 컨트롤러가 디플리먼트이다..

각 포드들 위에 레플리카셋이 있고 그 위에 디플로이먼트가 있다

k create deployment deploy-hello --image=hello-world

k get all

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deploy-hello	0/1	1	0	13s
NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/deploy-hello-54df7ff57c	1	1	0	13s

k get pod

k get deployment

k delete deployment deploy-hello

replicaset 조정

사용자가 요청한 파드수만큼 파드 유지

k9s에서 죽여도 이름만바뀌고 다시살아난다.

```
k create deployment deploy-nginx --image=nginx --replicas=3
```

k get deploy,rs,po

```
k get deploy,rs,po -o wide
```

삭제

```
encore@myserver01:~$ k delete pod deploy-nginx-7d74c85c6f-c4ph8
```

지워도 다시살아난다..

```
watch -n 1 k get dplo,rs,po -o wide
```

apiVersion: apps/v1

kind: Deployment

metadata:

name: deploy-test01

spec:

replicas: 3

selector:

matchLabels:

app.kubernetes.io/name: web-deploy

template:

metadata:

labels:

app.kubernetes.io/name: web-deploy

spec:

containers:

- name: nginx

image: nginx:latest

```
k apply -f deploy-test01.yaml
```

selector - 파드에 라벨붙이기

deployment가 관리할 파드를 연결

matchLabels:

app.kubernetes.io/name: web-deploy

selector로 적용하느 ㄴ 이름이 됨

파드를 생성했을때 이름과 동일해야함....

template 생성한 파드의 정보 표현

metadata:

labels:

app.kubernetes.io/name: web-deploy

디플리먼트가 관리할 파드..

encore@myserver01:~\$ k delete deployment deploy-nginx

k delete -f deploy-test01.yaml

롤아웃

nginx 버전을 1.24로 바꾸고

k apply -f deploy-test02.yaml

기본으로 docker hub바라보고있는데

gcr.io/nginx:1.24를하면 구글을 바라본다. 레드햇에있다

image: quay.io/nginx/nginx-unprivileged:1.24 바꾸기

리전은 진짜 국가에 있는거

롤링업데이트

k describe deployment deploy-test01

버전 1.25로 바꾸고

rollback

\$ k rollout undo deployment deploy-test01

서비스

clusterip nodeport loadbalancer externalname

외부하고 통신되는 것

k get pod -o wide 에서 나오는 ip는 클러스터의 ip이다

각각의 pod안의 nginx에는 ip가 있는데 각 pod안의 ip들끼리 소통하기위하여 service가 필요하다

clusterip

k8s의 기본설정값

클러스터 내에서만 파드에 접근할수있도록 하는 유형

클러스터 내부에서만 접근 가능한 ip생성 외부 접근 x

yaml 생성.

service-test01.yml

```
apiVersion: v1
kind: Service
metadata:
name: web-service
spec:
selector:
app.kubernetes.io/name: web-deploy
type: ClusterIP
ports:
- protocol: TCP
port: 80
```

service 올리기전에 pod먼저 올리고

```
k apply -f yaml
```

service 띄우니 5개 pod 켜지고 replicaset이랑 deployment켜진다.

도커안에 컨테이너 들어간것처럼

```
k exec -it nginx01 -- /bin/bash
```

```
curl ip:port
```

nginx 내용뜯다

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: deploy-test01
spec:
replicas: 3
selector:
matchLabels:
app.kubernetes.io/name: web-deploy
template:
metadata:
labels:
app.kubernetes.io/name: web-deploy
spec:
containers:
- name: nginx
image: nginx:latest
로 바꾸면 된다....
```

기존컨

apiVersion: v1

kind: Pod

metadata:

name: nginx01

spec:

containers:

- name: nginx-test01
image: nginx:latest

다

k delete -f deploy-test01.yaml

k delete -f service-test01.yaml

k delete -f nginx-test01.yaml

k apply -f nginx-test01.yaml

k apply -f deploy-test01.yaml

k apply -f service-test01.yaml

k exec -it nginx01 -- /bin/bash

curl clusterip:port

nginx pod에서 cluster ip로 보낸다

NodePort

각 노드의 특정포트를 통해 외부접근을 제공

nat를 사용하는 클러스터 내에서 각 노드가 지정된 포트를 외부로 노출

**

apiVersion: v1

kind: Service

metadata:

name: web-service-nodeport

spec:

selector:

app.kubernetes.io/name: web-deploy

type: NodePort

ports:

- protocol: TCP

nodePort: 31001

port: 80

targetPort: 80

**

nodeport, 서비스가 사용하는 port 파드가 받게 될 port

deplyoment 띄우고

service 띄우고

이제 ip:31001하면 nginx 접속이됩니다.

LoadBalancer

이걸 이용하며 ip 및 포트번호를 활용해 클러스터 외부에 서비스제공
nodeport의 상위개념

apiVersion: v1

kind: Service

metadata:

name: web-service-loadbalancer

spec:

selector:

app.kubernetes.io/name: web-deploy

type: LoadBalancer

ports:

- protocol: TCP
nodePort: 31002
port: 80
targetPort: 80
externalIPs:
 - 192.168.56.200

k create deployment web-deploy --image=sysnet4admin/echo-hname

이러고 chrom에 들어가면 200과 202는 안되고 201만된다

매핑오류

apiVersion: v1

kind: Service

metadata:

name: np-svc

spec:
selector:
app: np-pods
ports:
- name: http
protocol: TCP
port: 80
targetPort: 80
nodePort: 30000
type: NodePort

k create deployment np-pods --image=sysnet4admin/echo-hname

k apply -f nodeport.yml

파드개수 늘리기

k scale deployment np-pods --replicas=4

```
import requests
url = "http://ip:30000"
for x in range(200):
    print(requests.get(url).text)**
```

인그레스

nodeport 서비스는 port를 중복사용이 안된다.

결국 한개의 nodeport에는 한개의 디플로이먼트만 적용

여러개의 디플로이먼트면 그 수만큼 노드포트를 구성하긴 hard

인그레스는 고유한 주소 제공

목적에 따라 다른 응답

l4/l7 로드밸런서와 보안 인증서 처리

nginx 인그레스 컨트롤러 k8s에서 지원

사용자는 노드마다 설정된 노드포트를 통해 노드포트 서비스 접속 이때 노드포트 서비스를
nginx

ingress controller로 구성.

해당 controller는 사용자의 접속 경로에 따라 클러스터 ip서비스로 경로 제공

클러스터 ip 서비스는 사용자를 해당 pod로 연결

인그레스 컨트롤러는 파드와 직접통신x

노드포트 or 로드밸런서로 도움받기

k delete deployment np-pods

k delete -f nodeport.yml

****k create deployment in-hname-pod --image=sysnet4admin/echo-hname**

k create deployment in-ip-pod --image=sysnet4admin/echo-ip**

apiVersion: v1

kind: Service

metadata:

name: nginx-ingress-controller

spec:

ports:

- name: http

protocol: TCP

port: 80

targetPort: 80

nodePort: 30100

- name: https

protocol: TCP

port: 443

targetPort: 443

nodePort: 30101

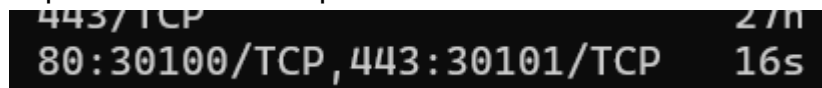
selector:

app.kubernetes.io/name: ingress-nginx

type: NodePort

port 범위는 30000~32767

https 443 보안들어간 port



```
443:30101/TCP 27m
80:30100/TCP, 443:30101/TCP 16s
```

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-nginx

annotations:

nginx.ingress.kubernetes.io/rewrite-target:

spec:

rules:

- http:

paths:

- path: /
pathType: Prefix
backend:
service:
name: hname-svc-default
port:
number: 80
- path: /ip
pathType: Prefix
backend:
service:
name: ip-svc
port:
number: 80
- path: /your-directory
pathType: Prefix
backend:
service:
name: your-svc
port:
number: 80

k apply로 적용한다음

k get ingress

k get ingress -o yaml

```
**wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-  
v1.8.0/deploy/static/provider/cloud/deploy.yaml**
```

k apply -f deploy.yaml

만든걸 외부로 노출

k expose deployment in-hname-pod --name=hname-svc-default --port=80,443

k expose deployment in-hname-pod --name=ip-svc --port=80,443

k get service를 하니 새로운 cluste-ip가 생성이됨