Kubernetes Service &Ingress

1. Prerequisites
OS : CentOS v7.6
Arch : x86

2. k8s클러스터는 1마스터 2노드로 구성
Master : 4cpu, ram16G
Node : 2cpu, ram8G

3. Service
pod은 Controller에 의해 관리되기 때문에 한군데에 고정되어있지 않다.
서비스를 사용하게 되면 pod가 클러스터 내 어디에 있던지 상관없이 고정된 주소를 사용해서 접근이 가능하게 된다.
ClusterIP
- 가장 기본 타입
- 클러스터 **내부의 노드**에서 접근 가능
- 클러스터 외부에서는 접근 불가

NodePort
- 각 노드의 지정된 포트를 할당하는 방식
- 노드의 포트를 사용하기 때문에 클러스터 **외부나 내부 모두 접근가능**
- 특이점 : pod이 1번노드에 떠있다고 하더라도 2번노드의 ip로 접근가능

LoadBalancer
- 클라우드 서비스를 사용할 때 사용 가능한 옵션
- pod을 클라우드에서 제공하는 로드밸런서와 연결해 해당 **로드밸런서의 ip를 이용**해 **외부에서 접근**할 수 있게 해줌
- **EXTERNAL-IP**에 IP가 표시됨

ExternalName
- 클러스터 **내부에서 외부로 접근**할 때 주로 사용
- 이 서비스로 접근하면 설정해둔 CNAME 값으로 연결되어 클러스터 외부로 접근할 수 있음

kube-proxy란?
쿠버네티스 클러스터의 각 노드마다 실행되고 있으면서
클러스터 **내부 IP로 연결되기 바라는 요청을 적절한 곳으로 전달**해주는 역할을 한다.

4. Ingress
클러스터 외부에서 내부로 접근하는 요청을 어떻게 처리할지 정리해둔 규칙들의 모음이다.
- 외부에서 접근 가능한 URL
- 트래픽 로드밸런싱
- SSL/TLS termination
- ip가 아닌 이름 기반 주소처리

이러한 설정들을 **ingress**라고 하고, 실제 행위자는 **ingress controller이다.**

쿠버네티스에서 공식적으로 제공하는 ingress controller는 2개이다.
1. ingress-gce
2. ingress-nginx

gce 같은 경우는 구글 클라우드를 사용하면 자동으로 사용할 수 있고 직접 클러스터에 설치해서 사용할 수 있는 것은 ingress-nginx이다.

실습
ingress 규칙 정의
$ vim ingress-controller.yaml
apiVersion :networking.k8s.io/v1

```
kind :Ingress
metadata :
  name :test-ingress
  annotations :
    nginx.ingress.kubernetes.io/rewrite-target :/
spec :
  rules :
  #- host: foo.bar.com       # host를 명시하지 않으면 ip로 연결
  -http :
       paths :                  # 각 path는 백엔드와 연결됨
       -path :/testpath
         backend :              # 연결될 서비스이름과 port
           serviceName :test
           servicePort :80
```

Tip) Host를 지정해야 할 경우

```
spec:
 rules:
 - host: foo.bar.com
    http:
      paths:
      - path: /testpath
        backend:
```

위와 같이 host를 지정하였으면 밑의 http는 "-"을 빼주어야 하며 http도 "-"를 붙이게 되면 위의 host와는 별도의 블럭으로 취급되어 적용이 안된다.

ingress 배포

```
$ kubectl apply -f ingress-controller.yaml
$ kubectl get ingress
```

```
[root@kube-m yaml]# kubectl apply -f ingress-controller.yaml
ingress.networking.k8s.io/test-ingress created
[root@kube-m yaml]# kubectl get ingress
NAME           HOSTS   ADDRESS   PORTS   AGE
test-ingress   *                 80      7s
```

상세 내용 확인

```
$ kubectl describe ingress test-ingress
```

```
[root@kube-m yaml]# kubectl describe ingress test-ingress
Name:             test-ingress
Namespace:        default
Address:
Default backend:  default-http-backend:80 (<none>)
Rules:
  Host  Path  Backends
  ----  ----  --------
  *
        /testpath   test:80 (<none>)
Annotations:
  kubectl.kubernetes.io/last-applied-configuration:  {"apiVersion":"networking.k8s.io/v1beta1","kind":"Ingress"
,"metadata":{"annotations":{"nginx.ingress.kubernetes.io/rewrite-target":"/"},"name":"test-ingress","namespace"
:"default"},"spec":{"rules":[{"http":{"paths":[{"backend":{"serviceName":"test","servicePort":80},"path":"/test
path"}]}}]}}
```

/testpath로 접속하면 test서비스의 80포트로 연결된다는 규칙을 확인

ingress controller(ingress-nginx) 생성

가이드 : Nginx Ingress Controller/Deployment Guide

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.45.0/deploy/static/provider/baremetal/deploy.yaml

$ kubectl get deployment -A

```
[root@kube-m yaml]# kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/master/deploy/static/mandatory.yaml
namespace/ingress-nginx created
configmap/nginx-configuration created
configmap/tcp-services created
configmap/udp-services created
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created
deployment.apps/nginx-ingress-controller created
```

```
[root@kube-m yaml]# kubectl get deployment -A
NAMESPACE       NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
ingress-nginx   nginx-ingress-controller   1/1     1            1           94s
kube-system     calico-kube-controllers    1/1     1            1           10m
kube-system     coredns                    2/2     2            2           10m
```

ingress contoller에 접근하기 위한 서비스 생성
$ kubectl expose deploy nginx-ingress-controller --type =NodePort -n ingress-nginx

```
[root@kube-m yaml]# kubectl get svc -n ingress-nginx
NAME                       TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
nginx-ingress-controller   NodePort   10.108.148.116   <none>        80:30116/TCP,443:30491/TCP   8m38s
```

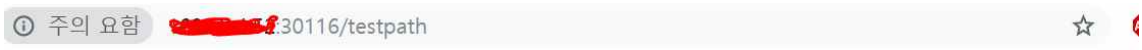
ip:port로 접근

404 Not Found

openresty/1.15.8.2

아무 path를 설정하지 않았을 경우 default 페이지가 뜨게되며 위에서 default로 설정을 하지 않았기 때문에 404에러가 뜬다.

위의 ingress규칙대로 /testpath로 이동



503 Service Temporarily Unavailable

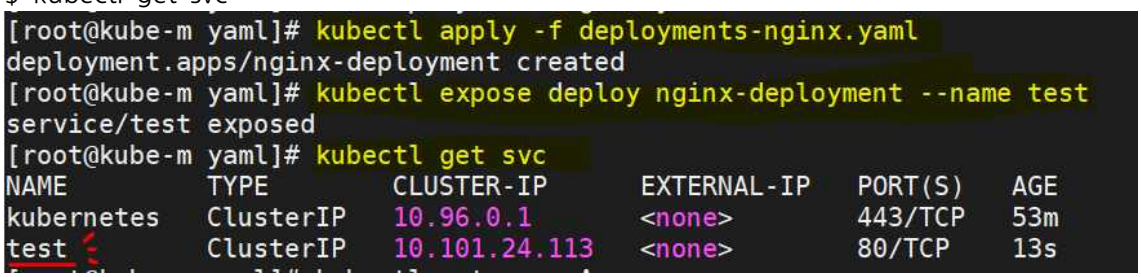
openresty/1.15.8.2

설정은 있지만, 서비스가 뜨지 않았으므로 503 에러가 발생
서비스 생성
/testpath와 이어질 서비스를 올려준다.

올릴 pod에 대한 정보는 아래와 같다.
$ vim deployments-nginx.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

$ kubectl apply -f deployments-nginx.yaml
$ kubectl expose deploy nginx-deployment --name test
$ kubectl get svc



새로고침
ingress를 사용하지 않았다면 서비스타입이 ClusterIP니까 외부에서 접근이 안되는게 정상이며
하지만 다시 페이지로 돌아가서 새로고침을 하면 :

요함 | [redacted]:30116/testpath

Welcome to nginx!

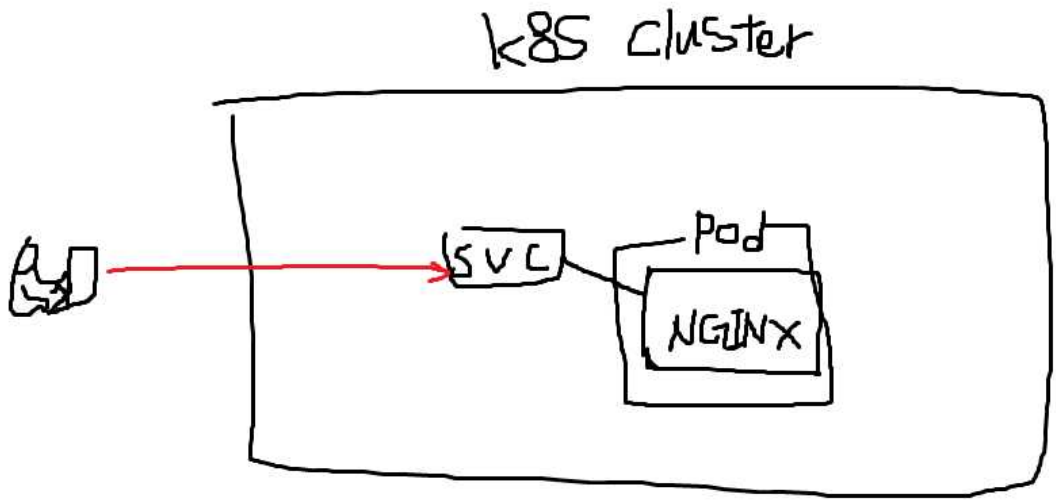
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

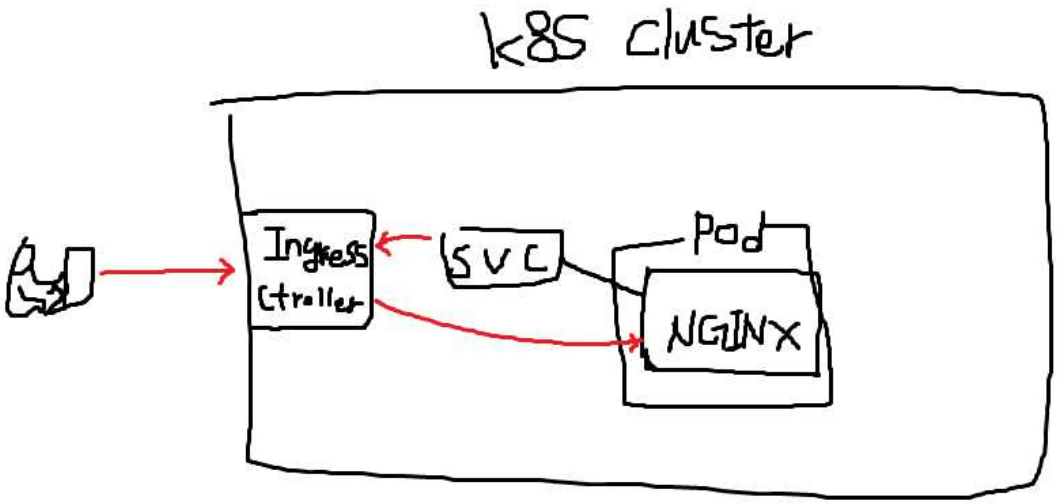
Thank you for using nginx.

ingress controller에서 서비스로 연결을 시켜주는 것을 확인

정리
일반적으로 서비스를 사용할 때, 서비스의 ip정보를 바라보게 됨 :



ingress를 사용하게 되면 서비스의 pod정보를 ingress controller로 가져오고, controller에서 서비스를 통하지 않고 pod으로 직접 연결


k8s cluster
Ingress
Ctroller
SVC
Pod
NGINX