# 5.1. 서비스 소개

## 서비스 등장 배경

**POD는 일시적이다.**

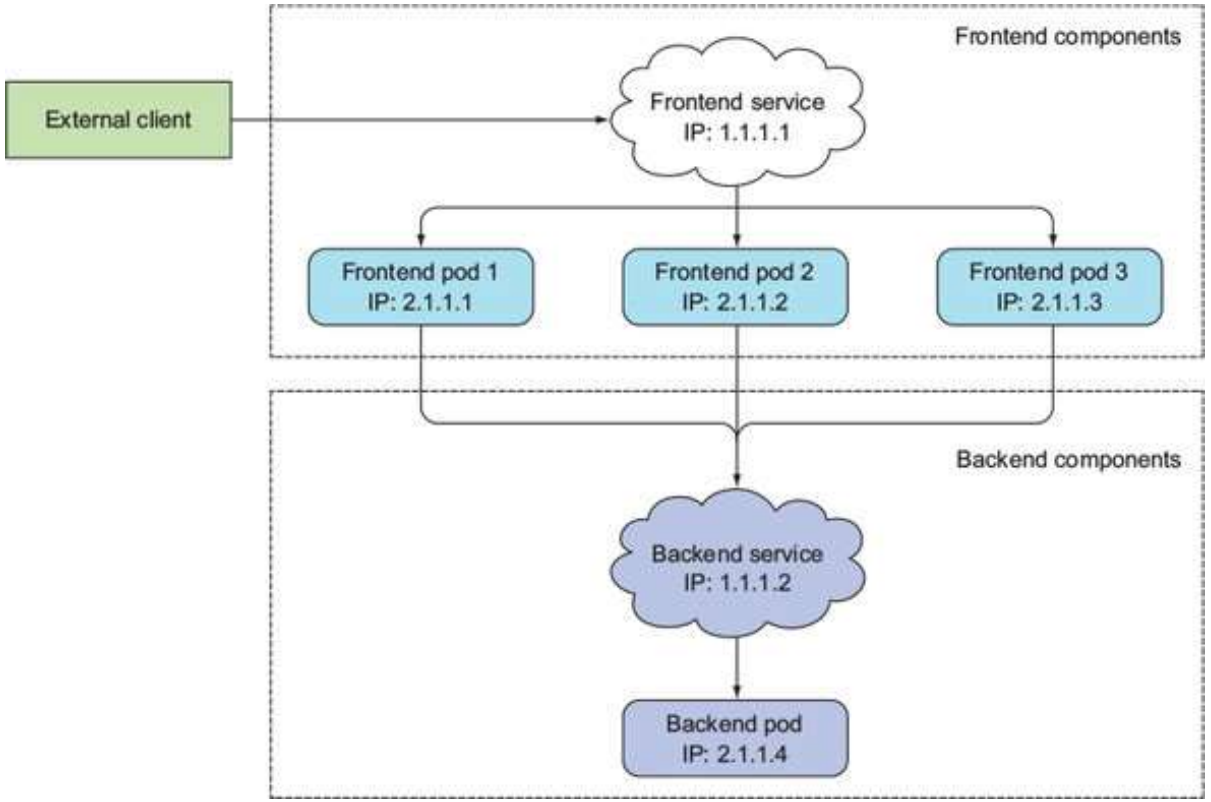
**POD의 IP를 예측할 수 없다..**

**동일한 서비스를 제공하는 여러 POD에 접근하기 위한 단일 IP가 필요하다.**

# 5.1. 서비스 소개

## 시나리오

- ◆ 웹서버 숫자와 상관없이 외부 클라이언트는 프론트엔드 파드에 연결 되어야 한다.
- ◆ 프론트엔드는 백엔드 데이터 베이스와 연결 해야 한다.
- ◆ 파드의 IP 변경이 되더라도 재설정 하지 않아야 한다.



# 5.1. 서비스 소개

## 서비스 생성

◆ kubectl expose 로 만들기
EX) kubectl expose deploy nginx –name nginx-svc

◆ Kubernetes API 서버에 YAML을 게시하여 생성하기
EX) kubectl create –f yaml파일명

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

App= nginx 레이블이 있는 파드가 이 서비스에 포함

서비스가 사용할 포트

서비스가 포워드 할 컨테이너 포트

# 5.1. 서비스 소개

## 서비스 생성

◆ 서비스생성 확인

kubectl get svc

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP   14m
my-service   ClusterIP   10.8.5.189    <none>        80/TCP    8m10s
```

# 5.1. 서비스 소개

## 서비스 생성 실습

◆ 파드와 서비스 생성

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: java
  labels:
    run: java
spec:
  containers:
  - name: java
    image: hifrodo/modu:v1
    ports:
      - containerPort: 80
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-svc
spec:
  selector:
    run: java
  ports:
  - port: 80
    targetPort: 80
```

## 서비스 생성 실습

◆ 파드와 서비스 확인

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get pod
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          19s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP   26s
nginx        ClusterIP   10.8.13.252   <none>        80/TCP    16s
```

# 5.1. 서비스 소개

## 서비스 생성 실습

◆ 테스트를 위한 파드 실행하기
kubectl run --generator=run-pod/v1 -it --rm centosshell --image=centos /bin/bash

◆ 접속 후 ClusterIP 로 curl 테스트
curl *ClusterIP*

```
PS C:\k8s\modu13\5.service> kubectl.1.14.exe get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
java-svc     ClusterIP   10.8.13.223   <none>        80/TCP    105s
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP   3m55s
PS C:\k8s\modu13\5.service> kubectl run --generator=run-pod/v1 -it --rm centosshell --image=centos /bin/bash
Flag --generator has been deprecated, has no effect and will be removed in the future.
If you don't see a command prompt, try pressing enter.
[root@centosshell /]# curl 10.8.13.223
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Frodo word</title>
</head>
<body>
    <h1><a href="/frodo"> 접속 정보 확인 하기 </a></h1>
```

서비스 기본 Type인 ClusterIP는 클러스터 내 IP라 외부로 노출이 되지 않는다.
따라서 내부 IP를 가지는 파드에서 테스트가 가능하다.

## 서비스 생성 실습2

◆ RC 와 서비스 생성

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: java-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      run: java
  template:
    metadata:
      labels:
        run: java
    spec:
      containers:
      - name: java
        image: hifrodo/modu:v1
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-svc
spec:
  selector:
    run: java
  ports:
  - port: 80
    targetPort: 80
```

## 서비스 생성 실습2

◆ RS, 파드, 서비스 확인

```
PS C:\k8s\modu13\5.service> kubectl.1.14.exe get rs
NAME      DESIRED   CURRENT   READY   AGE
java-rs   3         3         3       21s
PS C:\k8s\modu13\5.service> kubectl.1.14.exe get pod
NAME            READY   STATUS    RESTARTS   AGE
java-rs-4pld4   1/1     Running   0          27s
java-rs-rfvgj   1/1     Running   0          27s
java-rs-sm4jc   1/1     Running   0          27s
PS C:\k8s\modu13\5.service> kubectl.1.14.exe get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
java-svc     ClusterIP   10.8.11.125   <none>        80/TCP    25s
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP   87s
PS C:\k8s\modu13\5.service>
```

# 5.1. 서비스 소개

## 서비스 생성 실습

◆ 테스트를 위한 파드 실행하기

kubectl run --generator=run-pod/v1 -it --rm shell --image=busybox /bin/sh

◆ 접속 후 ClusterIP 로 wget 테스트

wget -q -O- http://*Cluster IP*

```
PS C:\k8s\modu13\5.service> kubectl.1.14.exe get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
java-svc     ClusterIP   10.8.11.125   <none>        80/TCP    25s
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP   87s
PS C:\k8s\modu13\5.service> kubectl run --generator=run-pod/v1 -it --rm shell --image=busybox /bin/sh
Flag --generator has been deprecated, has no effect and will be removed in the future.
If you don't see a command prompt, try pressing enter.
/ # wget -q -O- http://10.8.11.125
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Frodo word</title>
</head>
<body>
    <h1><a href="/frodo"> 접속 정보 확인 하기 </a></h1>


</body>
```

## 동일한 서비스에서 여러 개의 포트 호출 하기

◆ 멀티 컨테이너 파드를 포함 한 RS 및 서비스 생성

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      run: web-was
  template:
    metadata:
      labels:
        run: web-was
    spec:
      containers:
      - name: nginx
        image: hifrodo/moud:nginxv1
        ports:
          - containerPort: 80
      - name: tomcat
        image: hifrodo/modu:tomcatv1
        ports:
          - containerPort: 8080
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: tomcat-nginx
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  - name: was
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    run: web-was
```

여러 포트가 있는 서비스를 만들 때는 포트 이름을 지정해야 한다.

## 동일한 서비스에서 여러 개의 포트 호출 하기

◆ 멀티 컨테이너 파드를 포함 한 RS 및 서비스 생성 확인 하기

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get rs
NAME       DESIRED   CURRENT   READY   AGE
nginx-rs   3         3         3       4m15s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-9tlbt   2/2     Running   0          4m21s
nginx-rs-tq4fl   2/2     Running   0          4m21s
nginx-rs-vspd9   2/2     Running   0          4m21s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get svc
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)             AGE
kubernetes     ClusterIP   10.8.0.1       <none>        443/TCP             5m6s
tomcat-nginx   ClusterIP   10.8.12.183    <none>        80/TCP,8080/TCP     4m27s
```

# 5.1. 서비스 소개

## 동일한 서비스에서 여러 개의 포트 호출 하기

◆ 테스트를 위한 파드 생성 후 서비스 확인
kubectl run --generator=run-pod/v1 -it --rm centosshell --image=centos /bin/bash

◆ 테스트를 위한 파드 생성 후 서비스 확인
curl ClusterIP:80
curl ClusterIP:8080

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK₩yaml>kubectl run --generator=run-pod/v1 -it --rm shell --image=centos /bin/bash
If you don't see a command prompt, try pressing enter.
[root@shell /]# curl 10.8.12.183
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Frodo word</title>
</head>
<body>
    <h1>Hello, Frodo world!!!! I love frodo_V1 </h1>
</body>
</html>
[root@shell /]# curl 10.8.12.183:8080
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
<body>
  <h1>
    Hello World
  </h1>
</body>
```

# 5.1. 서비스 소개

## 서비스 개념 정리

서비스를 만들면 파드에 접속할 수 있는 안정적인 IP가 생긴다.

이 주소는 서비스가 유지 되는 동안 변경 되지 않는다.

파드의 변화와 상관없이 해당 IP주소로 접근 할 수 있다.

## POD 의 환경 변수로 서비스 IP 검색 하기

◆ 파드를 생성하기 전에 서비스를 먼저 생성하면 해당 파드의 환경 변수를 통해서도 서비스 IP 를 확인 할 수 있다

kubectl exec *파드명* env

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl exec nginx-rs-9tlbt env
Defaulting container name to nginx.
Use 'kubectl describe pod/nginx-rs-9tlbt -n default' to see all of the containers in this pod.
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx-rs-9tlbt
KUBERNETES_SERVICE_PORT=443
TOMCAT_NGINX_SERVICE_PORT_HTTP=80
TOMCAT_NGINX_PORT=tcp://10.8.12.183:80
KUBERNETES_SERVICE_PORT_HTTPS=443
TOMCAT_NGINX_SERVICE_HOST=10.8.12.183
TOMCAT_NGINX_SERVICE_PORT=80
TOMCAT_NGINX_PORT_80_TCP_PORT=80
TOMCAT_NGINX_PORT_80_TCP_ADDR=10.8.12.183
TOMCAT_NGINX_PORT_8080_TCP=tcp://10.8.12.183:8080
KUBERNETES_PORT_443_TCP_ADDR=10.8.0.1
TOMCAT_NGINX_PORT_80_TCP_PROTO=tcp
TOMCAT_NGINX_PORT_8080_TCP_PROTO=tcp
KUBERNETES_SERVICE_HOST=10.8.0.1
KUBERNETES_PORT=tcp://10.8.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.8.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
TOMCAT_NGINX_SERVICE_PORT_WAS=8080
TOMCAT_NGINX_PORT_80_TCP=tcp://10.8.12.183:80
TOMCAT_NGINX_PORT_8080_TCP_PORT=8080
TOMCAT_NGINX_PORT_8080_TCP_ADDR=10.8.12.183
HOME=/root
```

## FQDN을 이용한 서비스 연결

- Kube-dns에 의해서 서비스 FQDN의 형식으로 연결이 가능하다.
- 서비스 명이  web 인 서비스를 만들고 테스트를 해본다.

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)           AGE
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP           14m
web          ClusterIP   10.8.11.162   <none>        80/TCP,8080/TCP   13m
```

- 앞 에서와 같은 방식으로 shell용 pod를 만들고 서비스 FQDN이름으로 연결 해본다.
- FQDN형식은 서비스명.네임스페이스명.svc.cluster.local 이다
- 네임스페이스.svc.cluster.local 접미사는 생략이 가능하다.

```
[root@shell /]# curl http://web.default.svc.cluster.local
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Frodo word</title>
</head>
<body>
    <h1>Hello, Frodo world!!!! I love frodo_V1 </h1>
</body>
</html>
```

# 5.2 클러스트 외부에 있는 서비스 연결

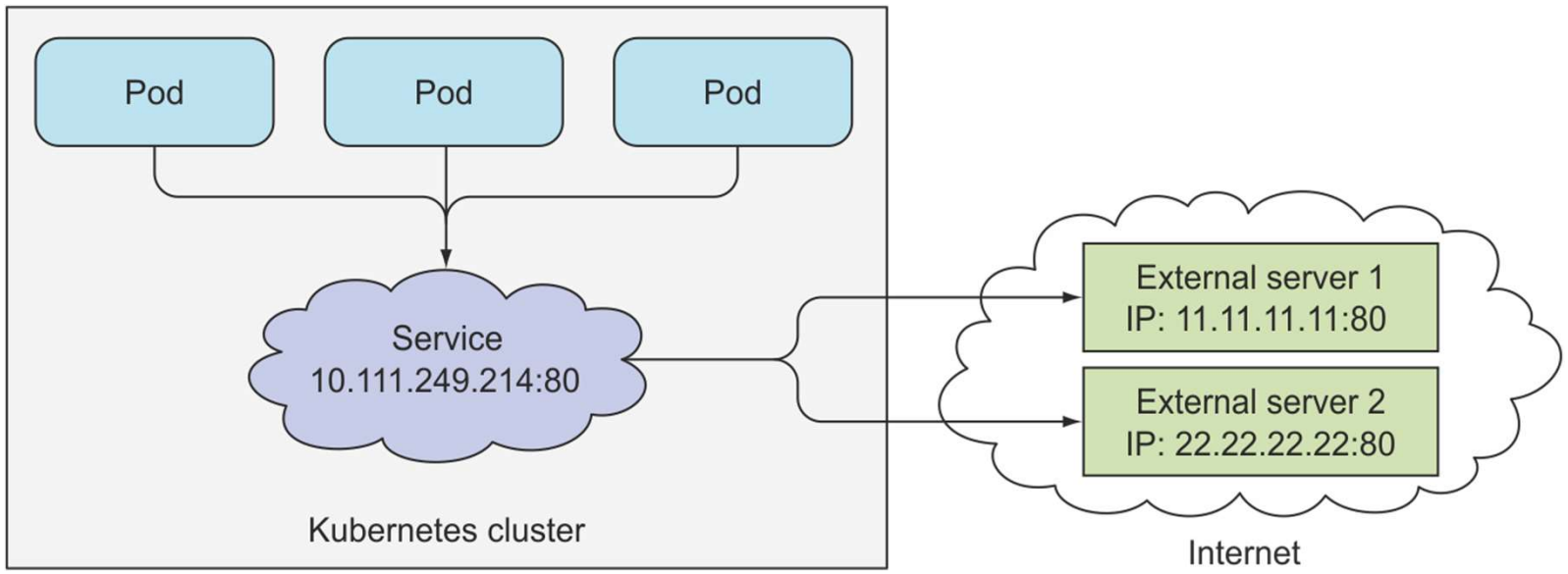
## 서비스 엔드포인란

◆ 서비스는 파드에 직접 연결 되지 않고 엔드포인트를 통해서 연결 된다.

```
PS C:\k8s\modu13\5.service> kubectl get pod -o wide
NAME            READY   STATUS    RESTARTS   AGE    IP
 GATES
java-rs-2xdg5   1/1     Running   0          3m6s   10.4.0.18
java-rs-5d7j9   1/1     Running   0          3m6s   10.4.1.11
java-rs-986f8   1/1     Running   0          3m6s   10.4.0.17
PS C:\k8s\modu13\5.service> kubectl describe svc java-svc
Name:              java-svc
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=java
Type:              ClusterIP
IP:                10.8.12.167
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.4.0.17:80,10.4.0.18:80,10.4.1.11:80
Session Affinity:  None
Events:            <none>
```

## 서비스 엔드포인트 수동 구성

◆ 엔드포인트 수동 연결의 예



## 서비스 엔드포인트 수동 구성

◆ 서비스를 생성 시 레이블 셀렉터를 만들면 해당 레이블이 있는 파드와 연결하는 엔드포인트를 자동으로 생성하고 것이고 그렇지 않으면 별도의 엔드포인트를 구성해야 한다.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

엔드포인트 이름과 동일해야 한다.

셀렉터가 정의 되어 있지 않다.

# 5.2 클러스트 외부에 있는 서비스 연결

## 서비스 엔드포인트 연결 실습

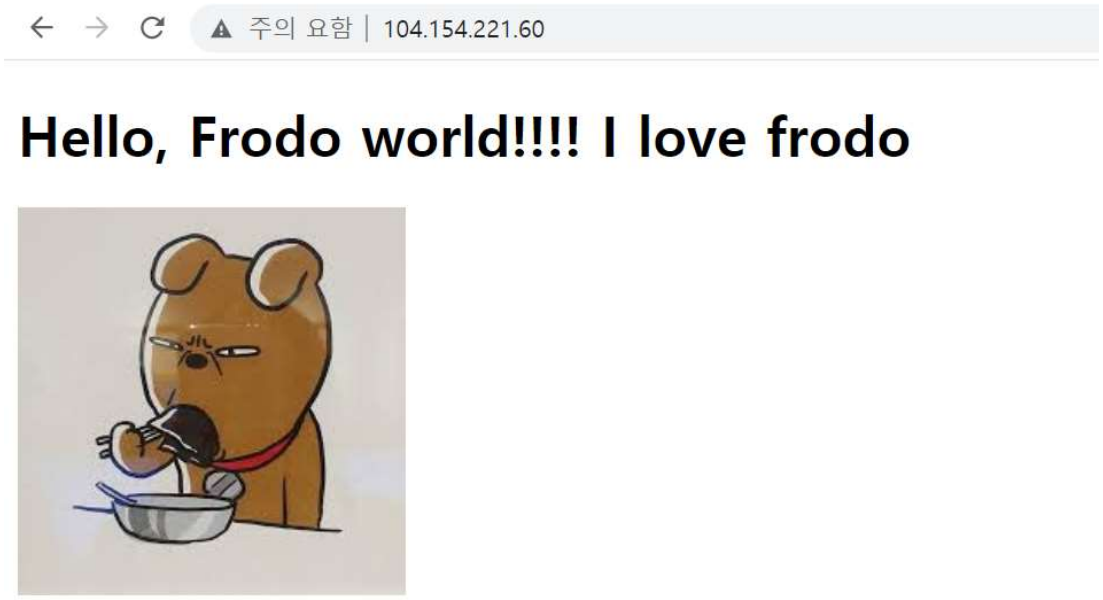
◆ 셀렉터가 없는 서비스 생성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

★ 서비스 외부 노출을 위해 Type을 LoadBalacer로 변경 (뒤에 해당 내용은 설명 되어짐)

## 서비스 엔드포인트 연결 실습

◆ 엔드포인트 생성 – 이 실습에서는 외부 웹사이트를 사용 함



# 5.2 클러스트 외부에 있는 서비스 연결

## 서비스 엔드포인트 연결 실습

◆ 엔드포인트 오브젝트 생성

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: my-service
subsets:
  - addresses:
      - ip: 104.154.221.60
    ports:
      - port: 80
```

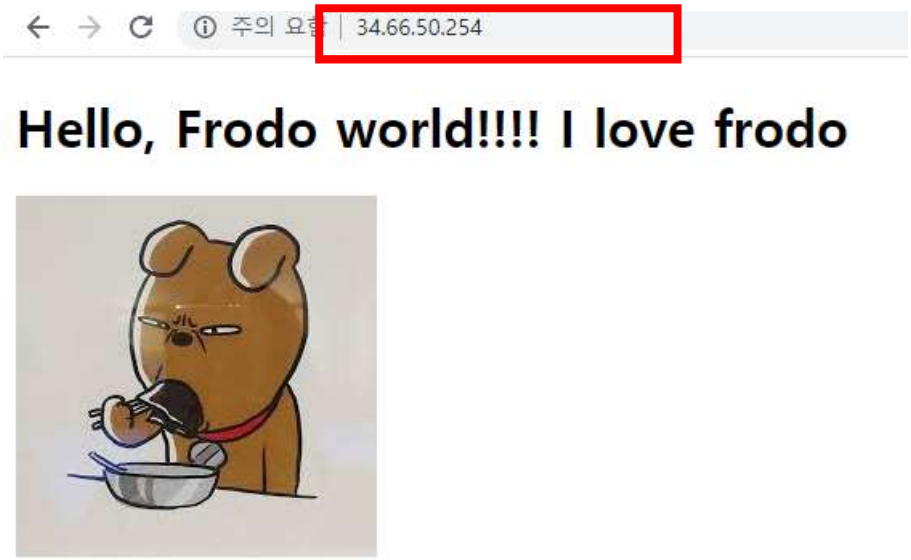
→ 서비스 명과 일치 해야 함

→ 앞에 만든 웹서버 IP

## 서비스 엔드포인트 연결 실습

◆ 접속 확인

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK₩yaml>kubectl get svc
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes   ClusterIP      10.8.0.1      <none>         443/TCP          56m
my-service   LoadBalancer   10.8.6.66     34.66.50.254   80:31034/TCP     2m31s
web          ClusterIP      10.8.11.162   <none>         80/TCP,8080/TCP  55m
```

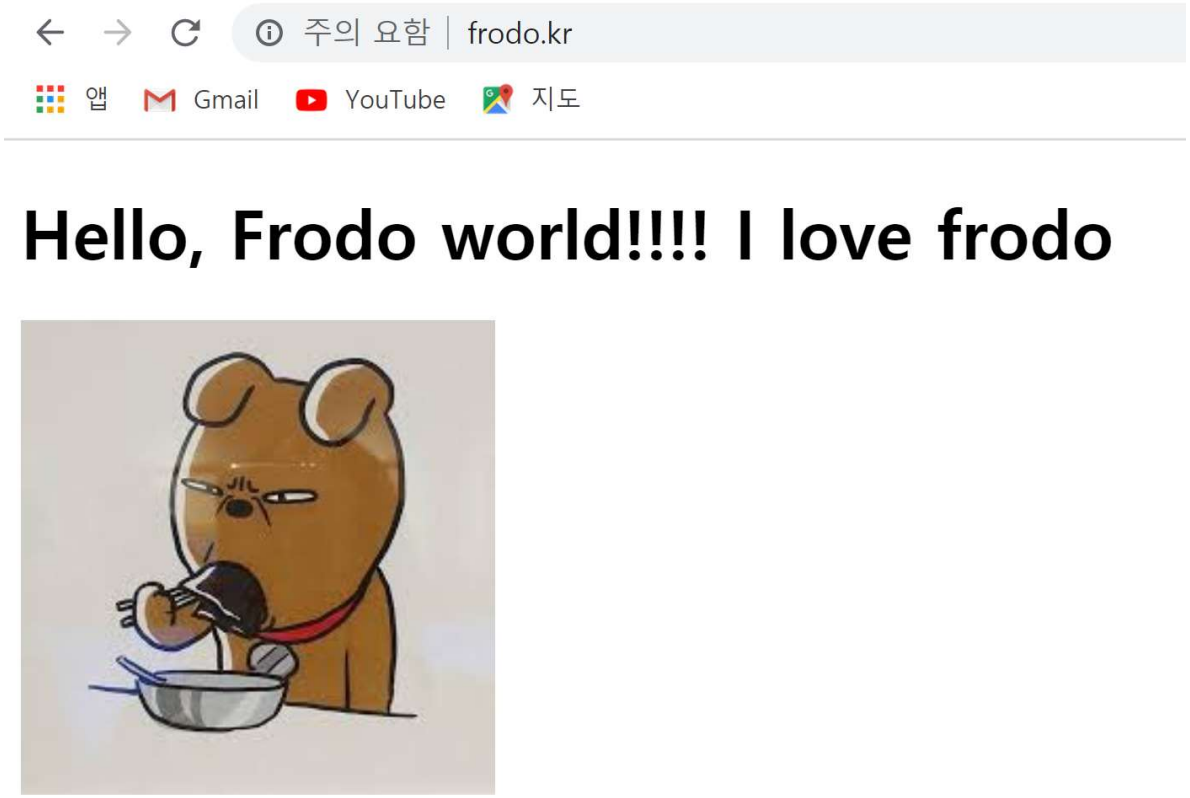


**서비스 엔드포인트 연결 실습**

데모

## ExternalName 서비스 생성 실습

◆ FQDN이 있는 외부 서비스 생성



## ExternalName 서비스 생성 실습

◆ 서비스 오브젝트 생성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: external-service
spec:
  type: ExternalName
  externalName: www.frodo.shop
  ports:
  - port: 80
```

**ExternalName 서비스 생성 실습**

데모

## ExternalName 서비스 생성 실습

◆ 연결 테스트
연결을 위한 파드를 생성 후 서비스 명으로 접속

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK₩yaml>kubectl get svc
NAME               TYPE           CLUSTER-IP   EXTERNAL-IP    PORT(S)   AGE
external-service   ExternalName   <none>       www.frodo.kr   80/TCP    34s
kubernetes         ClusterIP      10.8.0.1     <none>         443/TCP   4m18s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK₩yaml>kubectl run --generator=run-pod/v1 -it --rm shell --image=centos /bin/bash
If you don't see a command prompt, try pressing enter.
[root@shell /]# curl external-service
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Frodo word</title>
</head>
<body>
    <h1>Hello, Frodo world!!!! I love frodo </h1>
    <img src = "image/frodo.jpg">

</body>
</html>
[root@shell /]# exit
```

# 5.3 외부 클라이언트에 서비스 노출

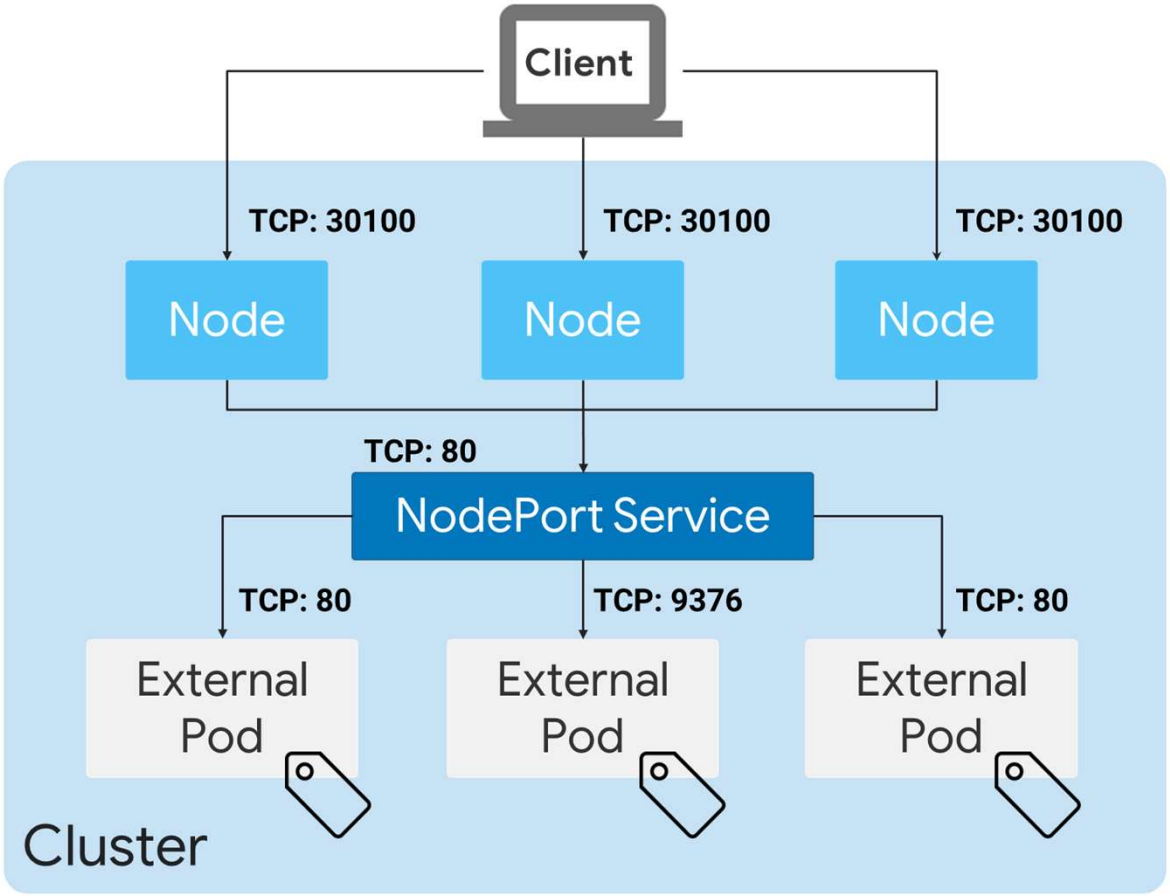
## 외부 노출 서비스 유형

◆ 노드포트
노드 자체에서 서비스를 오픈하고 해당 포트로 수신된 서비스 전달

◆ 로드발란스
전용로드발란서가 서비스를 수신 한 후 노드 포트로 전달

◆ 인스레스
HTTP레벨에서 작동

## 노드포트 서비스 사용




default NodePort Range (30000-32767)

## 노드포트 서비스 사용

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-svc
spec:
  selector:
    run: java
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30100
```

**노드포트를 이용한 실습**

◆ 노드포트로 서비스할 RS만들기
포트 80으로 서비스 하는 nginx pod로 구성된 RS를 만든다.

◆ 노드포트 포트 만들기
포트 30001로 노틑 포트를 구성한다.

◆ 서비스 확인
클라이언트 브라우저에서 노드포트 IP와 30100번 포트로 접속

## 노드포트를 이용한 실습

◆ 노드포트로 서비스할 RS만들기

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: java-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: java
  template:
    metadata:
      labels:
        app: java
    spec:
      containers:
      - name: java
        image: hifrodo/modu:v1
```

## 노드포트를 이용한 실습

◆ 노드포트로 서비스 생성하기

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-svc
spec:
  selector:
    app: java
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30100
```

# 5.3 외부 클라이언트에 서비스 노출

## 노드포트를 이용한 실습

◆ 접속확인하기
GCP에서 노드의 external IP를 확인 한 후 노드의 external IP:30100 으로 접속

| Name | Zone | Recommendation | In use by | Internal IP | External IP | Network | Network tags | Connect |
|---|---|---|---|---|---|---|---|---|
| gke-cluster-1-default-pool-913c65bc-66fk | us-central1-c | | gke-cluster-1-default-pool-913c65bc-grp | 10.128.0.64 (nic0) | 34.72.46.192 | default | gke-cluster-1-11412396-node | SSH |
| gke-cluster-1-default-pool-913c65bc-cjj8 | us-central1-c | | gke-cluster-1-default-pool-913c65bc-grp | 10.128.0.67 (nic0) | 35.192.139.77 | default | gke-cluster-1-11412396-node | SSH |
| gke-cluster-1-default-pool-913c65bc-dcvc | us-central1-c | | gke-cluster-1-default-pool-913c65bc-grp | 10.128.0.66 (nic0) | 34.66.50.254 | default | gke-cluster-1-11412396-node | SSH |

◆ 접속이 불가한 이유
외부에서 30100으로 접속하고자 하는 포트에 대한 방화벽 오픈 필요

gcloud compute firewall-rules create tcp-30100 --allow=tcp:30100

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>gcloud compute firewall-rules create tcp-30100 --allow=tcp:30100
Creating firewall...-Created [https://www.googleapis.com/compute/v1/projects/frodo-sandbox-20191112/global/firewalls/tcp-30100].
Creating firewall...done.
NAME        NETWORK  DIRECTION  PRIORITY  ALLOW      DENY  DISABLED
tcp-30100   default  INGRESS    1000      tcp:30100        False
```

**노드포트를 이용한 실습**

◆ 접속확인하기

**- 서버 접속 정보 -**

**- made by Frodo -**

*** HostName :<java-rs-qs2ql>**

*** RemoteAddr : 10.128.0.30**

*** ServerName : 35.184.169.150**

*** RequestURL : http://35.184.169.150:30100/frodo**

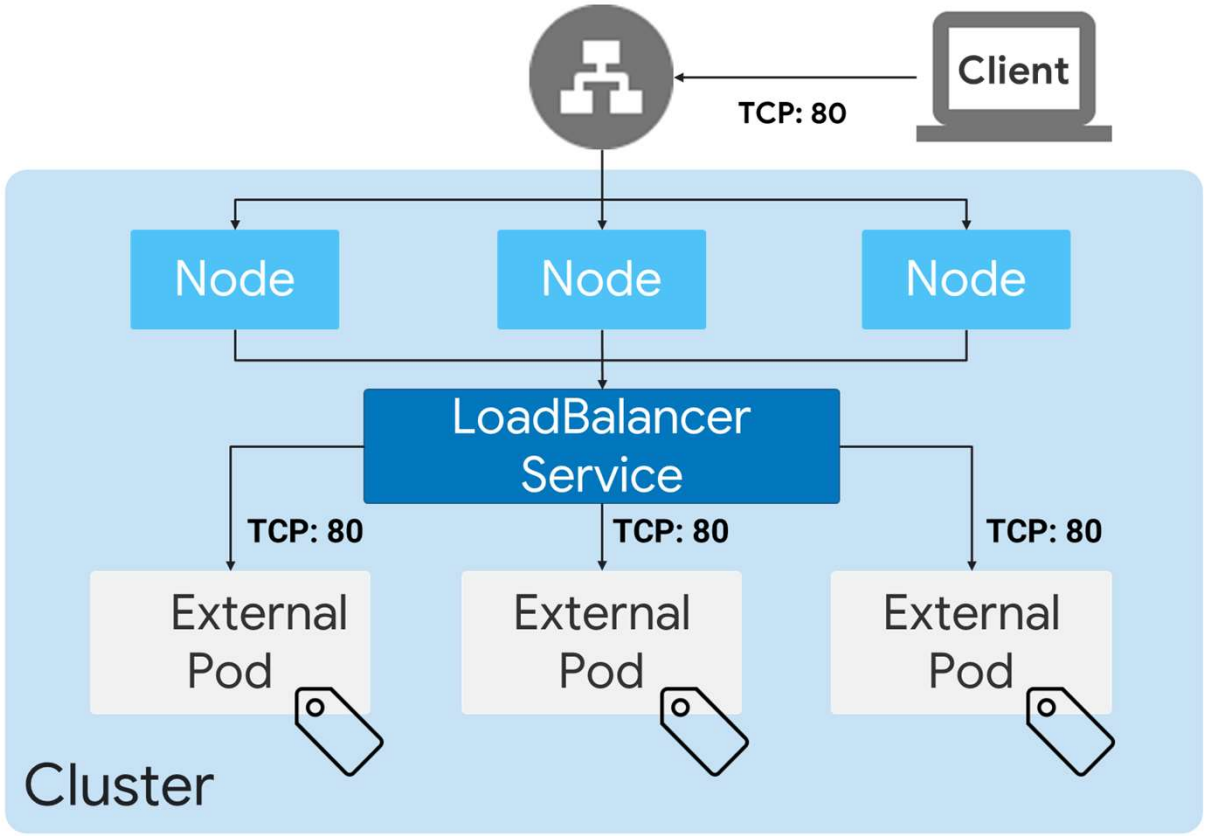
# 5.3 외부 클라이언트에 서비스 노출

## 외부 로드발란서로 서비스 노출

- 서비스 Type을 LoadBalacer로 선언 하면 클라우드 사업자에 의해서 로드발란서가 생성이 된다.
- 생성된 로드발란서는 외부 공개 IP를 가지게 되며 해당 IP로 접근이 가능하다.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-svc
Spec:
  selector:
    app: java
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
```

## 외부 로드발란서로 서비스 노출



**로드발란서 생성 실습**

◆ 로드발란서로 서비스할 RS만들기
포트 80으로 서비스 하는 nginx pod로 구성된 RS를 만든다.

◆ 로드발란서 서비스 만들기

◆ 서비스 확인
로드발란서 IP로 접속 테스트 하기

## 로드발란서 생성 실습

◆ 로드발란서로 서비스할 RS만들기

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: java-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: java
  template:
    metadata:
      labels:
        app: java
    spec:
      containers:
      - name: java
        image: hifrodo/modu:v1
```

## 로드발란서 생성 실습

◆ 로드발란서 서비스 생성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-svc
spec:
  selector:
    app: java
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 80
```

## 로드발란서 생성 실습

◆ 접속 테스트를 위한 서비스 상태 확인
서비스 외부 IP가 생성 된 것을 확인 한 후 해당 IP로 접속 테스트를 한다.

```
PS C:\k8s\modu13\5.service> kubectl get pod
NAME            READY   STATUS    RESTARTS   AGE
java-rs-hrlpl   1/1     Running   0          35s
java-rs-jrvwc   1/1     Running   0          35s
java-rs-lzgwk   1/1     Running   0          35s
PS C:\k8s\modu13\5.service> kubectl get svc
NAME         TYPE           CLUSTER-IP   EXTERNAL-IP      PORT(S)        AGE
java-svc     LoadBalancer   10.8.12.11   35.232.174.143   80:31777/TCP   52s
kubernetes   ClusterIP      10.8.0.1     <none>           443/TCP        56s
```

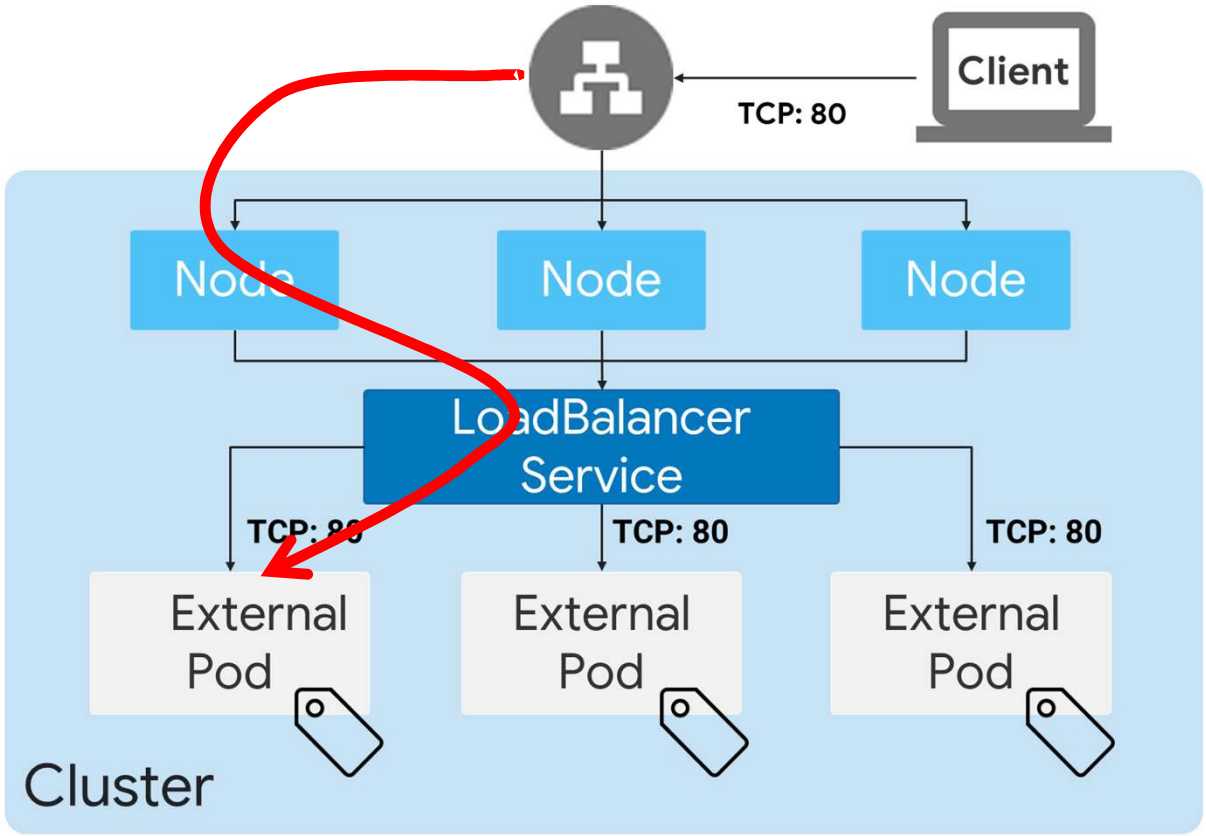
클라우드 사업자로부터 LB가 생성 되는 과정동안 External IP가 pending 상태로 보인다

## 로드발란서 생성 실습

◆ 접속 테스트를 위한 서비스 상태 확인
서비스 외부 IP가 생성 된 것을 확인 한 후 해당 IP로 접속 테스트를 한다.

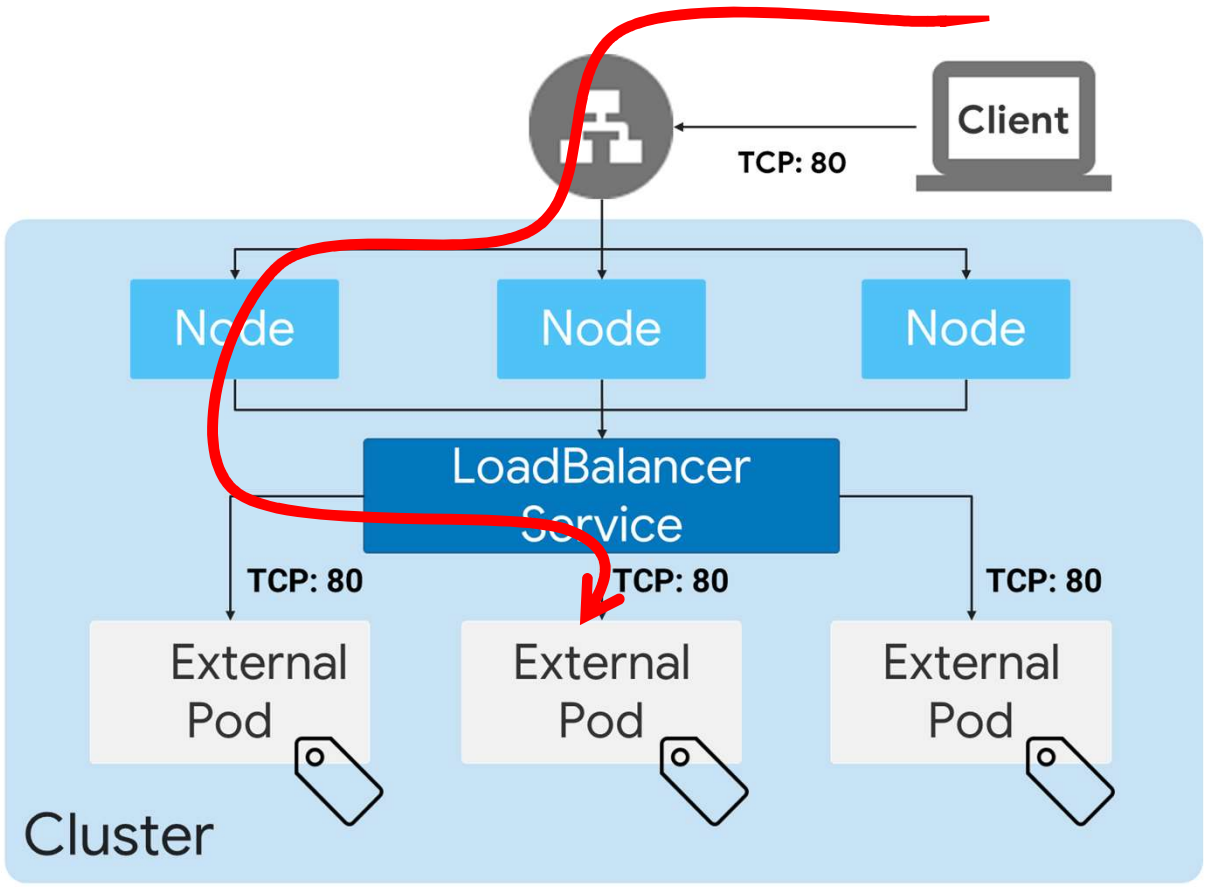
## 로드발란서 생성 실습

◆ 외부 로드발란서에서 응답한 노드의 파드로 전달



## 로드발란서 생성 실습

◆ 외부 로드발란서에서 응답한 노드에서 다른 노드의 파드로 전달



## 불필요 홉을 예방하기

◆ 응답 받은 노드 내의 파드로 전달
externalTrafficPolicy: Local 옵션 사용

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
spec:
  selector:
    run: nginx
  type: LoadBalancer
  externalTrafficPolicy: Local
  ports:
    - port: 80
      targetPort: 80
```

## 불필요 홉을 예방하기

◆ externalTrafficPolicy: Loca 특징
최초 응답 받은 노드에 있는 파드에서 응답 한다.
클라이언트 IP가 보존 된다.

**- 서버 접속 정보 -**

**- made by Frodo -**

*** HostName :<java-rs-jrvwc>**

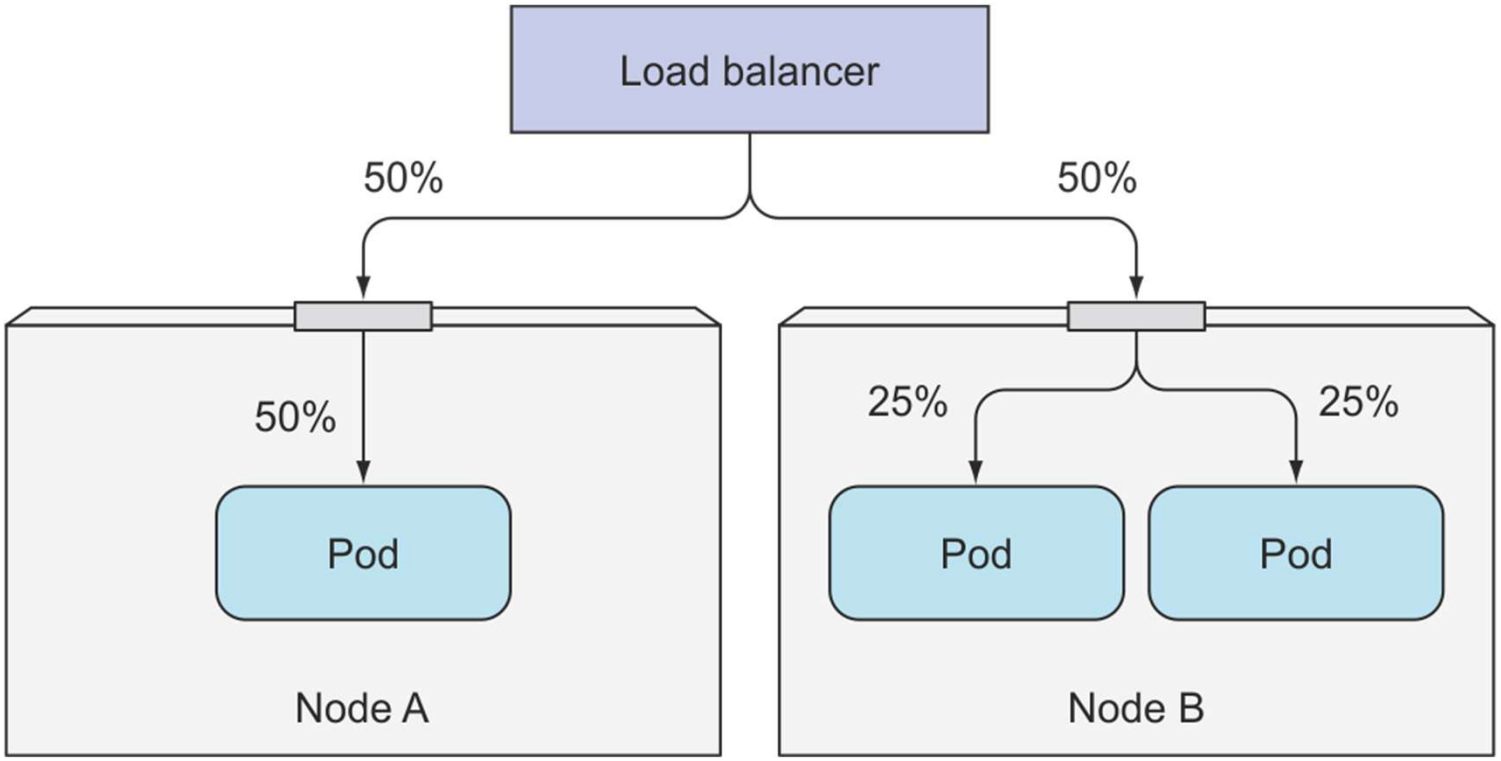
*** RemoteAddr : 110.10.173.191**

*** ServerName : 35.232.174.143**

*** RequestURL : http://35.232.174.143/frodo**

## 불필요 홉을 예방하기

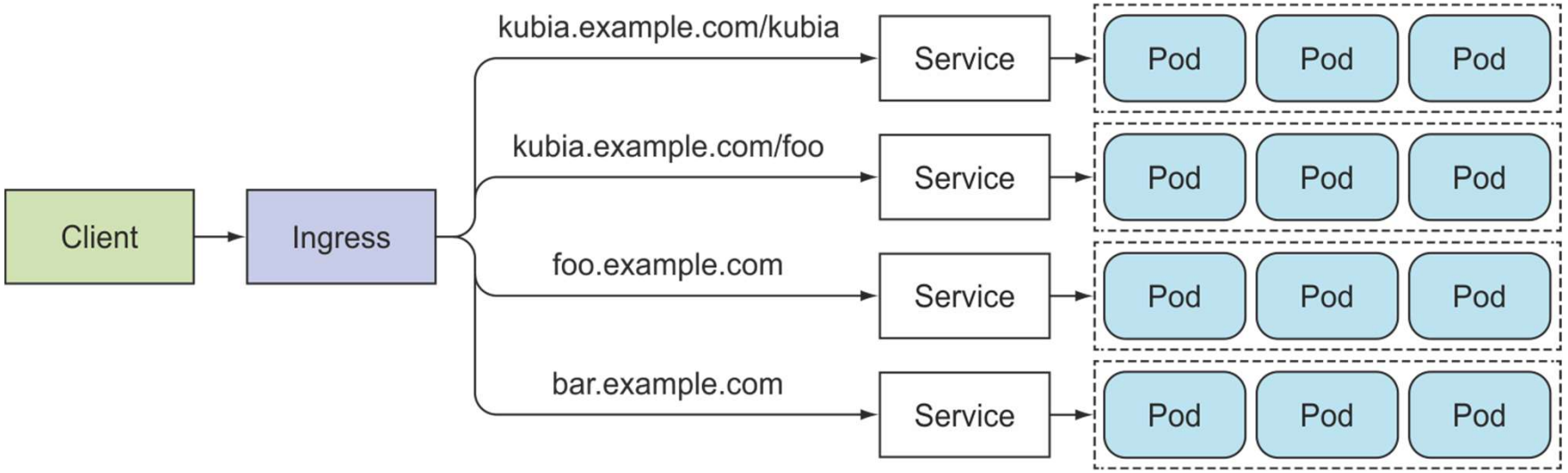
◆ externalTrafficPolicy: Local 단점은 파드 간 부하가 고르지 않을 수 있다



# 5.4 인그레스 리소스로 서비스 외부 노출

## 인그레스 서비스가 필요한 이유

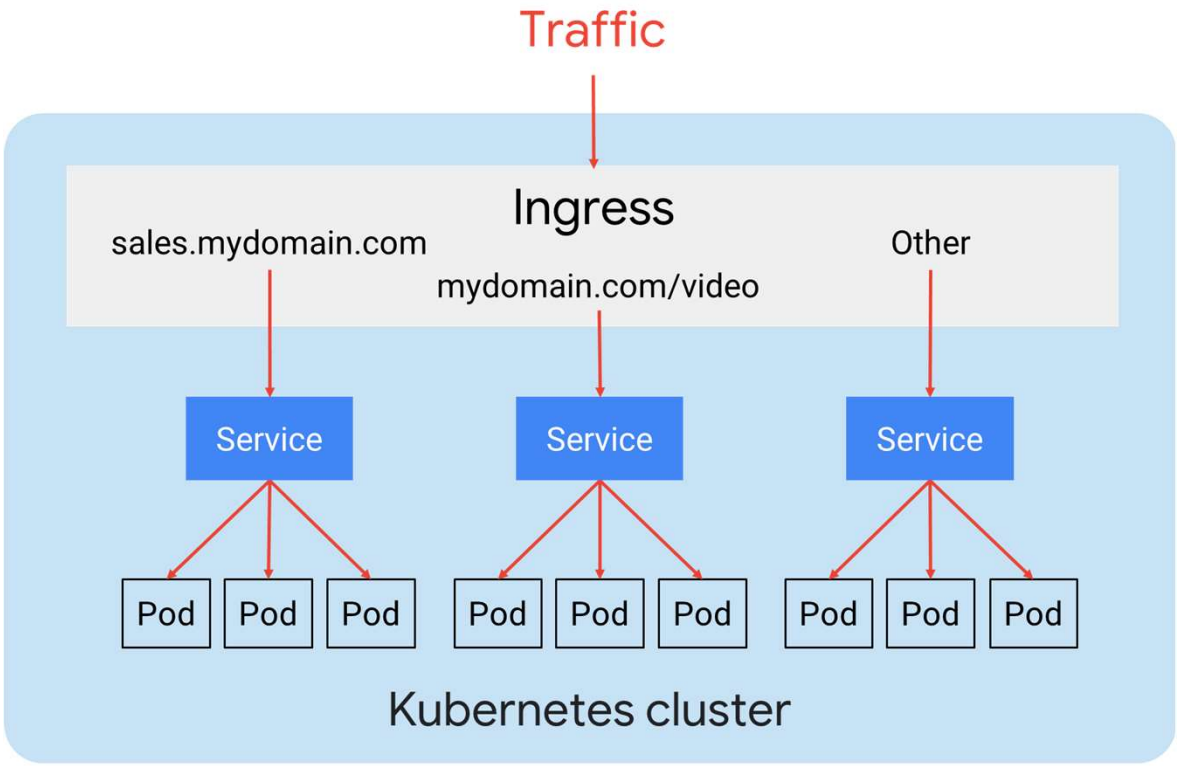
- 하나의 IP로 여러 개의 서비스가 접근이 가능하다.
- 클라이언트가 요청한 호스트와 경로에 따라 요청을 전달할 서비스가 결정 된다.



# 5.4 인그레스 리소스로 서비스 외부 노출

## 인그레스 컨트롤러가 필요한 경우

- 인그레스 리소스 작동을 위해서는 클러스터에 인스레스 컨트럴라가 실행 되어야 한다.
- 쿠버네티스 환경 마다 다른 컨트롤러를 제공 하며 GKE는 GCP에 고유한 HTTP 로드발란서 기능을을 사용해 인그레스 기능을 제공한다



# 5.4 인그레스 리소스로 서비스 외부 노출

## 인그레스 리소스 생성

- 서비스할 RS 작성
- 노드포트 타입의 서비스 작성
- 인그레스 서비스 작성
- DNS 등록 하기

## 인그레스 리소스 생성

◆ 서비스할 RS와 노드포트 서비스 작성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30100
```

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: java
        image: hifrodo/modu:nginxv1
```

## 인그레스 리소스 생성

◆ 인그레스 작성

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  rules:
  - host: www.frodo.kr
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx-svc
          servicePort: 80
```

→ 노드포트 서비스명

# 5.4 인그레스 리소스로 서비스 외부 노출

## 인그레스 리소스 생성

◆ 서비스 IP 확인 후 DNS 등록하기

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get pod
NAME              READY   STATUS    RESTARTS   AGE
nginx-rs-nkzts    1/1     Running   0          8m6s
nginx-rs-s4pw7    1/1     Running   0          8m6s
nginx-rs-wbvpz    1/1     Running   0          8m6s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get svc
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.8.0.1       <none>        443/TCP        9m36s
nginx-svc     NodePort    10.8.15.157    <none>        80:30100/TCP   8m10s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get ingress
NAME            HOSTS           ADDRESS          PORTS   AGE
nginx-ingress   www.frodo.kr    34.107.241.97    80      2m17s
```

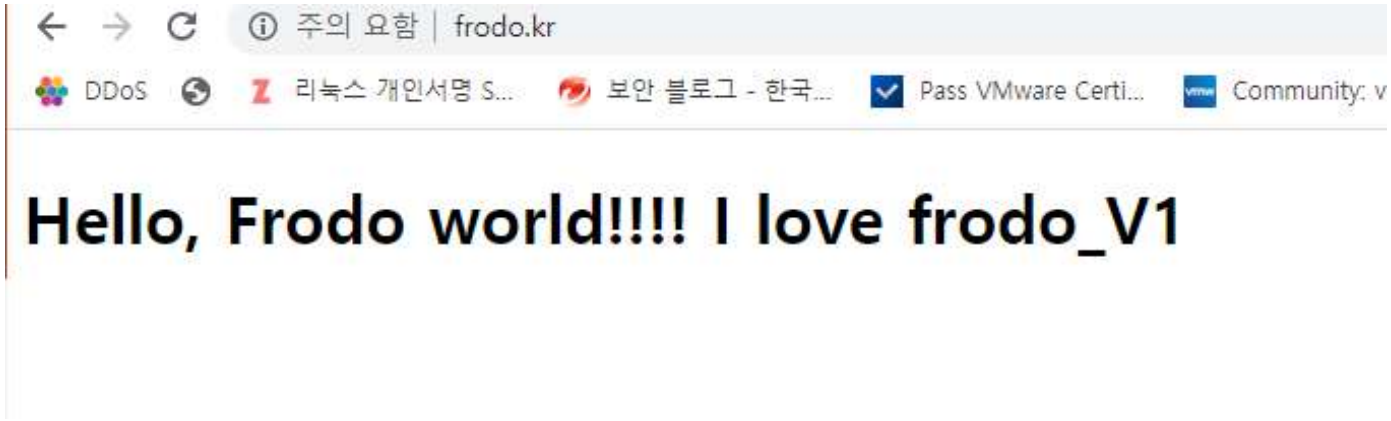
DNS 호스트

| | |
|---|---|
| www.frodo.kr | 34.107.241.97 |

## 인그레스 리소스 생성

◆ DNS 확인 후 접속 하기

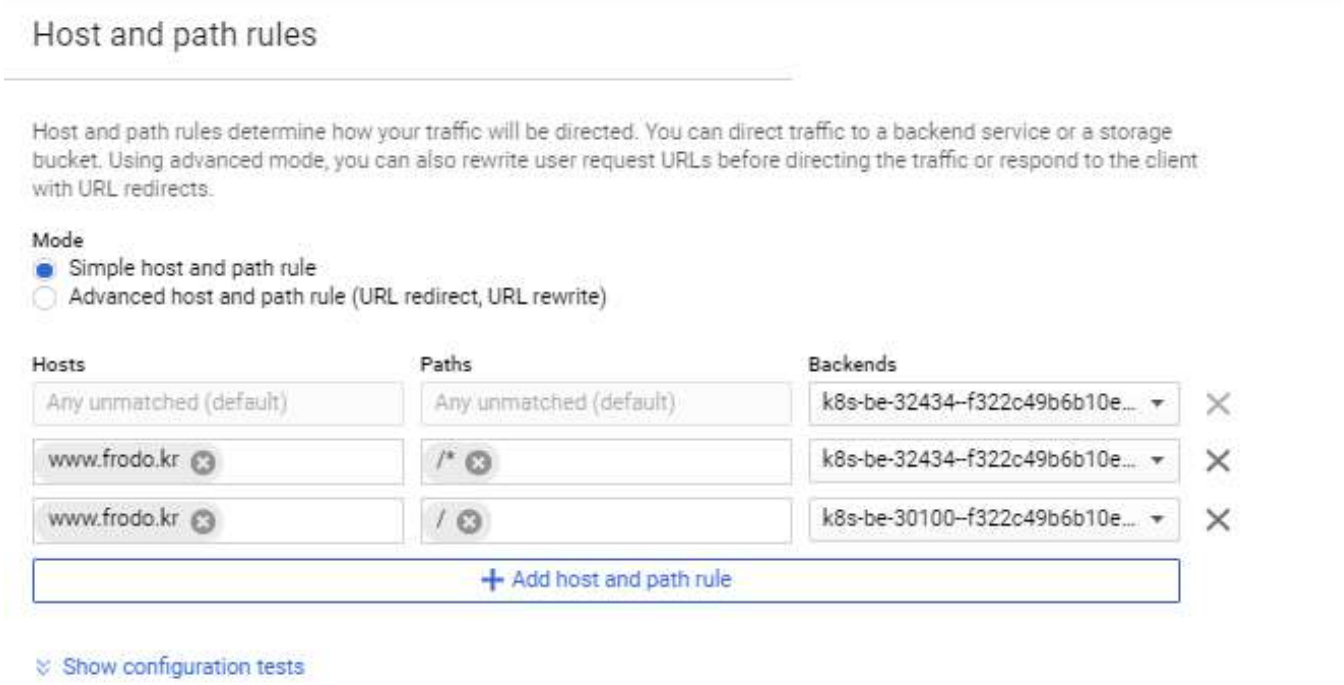
```
Ping was.frodo.kr [34.107.241.97] 32바이트 데이터 사용:
34.107.241.97의 응답: 바이트=32 시간=55ms TTL=52
34.107.241.97의 응답: 바이트=32 시간=59ms TTL=52
34.107.241.97의 응답: 바이트=32 시간=40ms TTL=52
34.107.241.97의 응답: 바이트=32 시간=43ms TTL=52
```



# 5.4 인그레스 리소스로 서비스 외부 노출

## 인그레스 리소스 생성

◆ GCP HTTP LB의 호스트패스 룰 확인 하기



## 멀티 인그레스 리소스 생성

◆ 추가 RS와 노드포트 서비스 만들기

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx2-svc
spec:
  selector:
    app: nginx2
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30200
```

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx2-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx2
  template:
    metadata:
      labels:
        app: nginx2
    spec:
      containers:
      - name: nginx2
        image: hifrodo/modu:nginxv3
```

## 멀티 인그레스 리소스 생성

◆ 인그레스 다시 만들기

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  rules:
  - host: www.frodo.kr
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx-svc
          servicePort: 80
  - host: frodo.frodo.kr
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx2-svc
          servicePort: 80
```

# 5.4 인그레스 리소스로 서비스 외부 노출

## 멀티 인그레스 리소스 생성

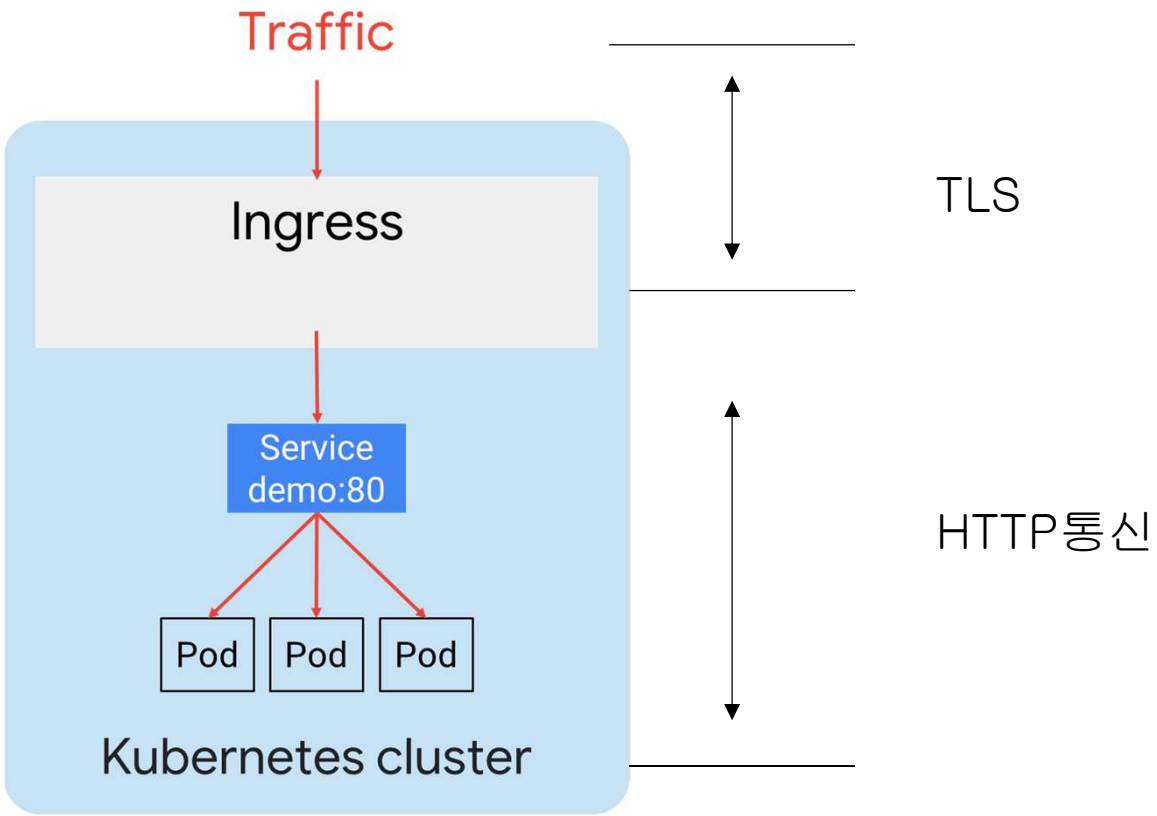
◆ 멀티 인그레스 IP확인 후 DNS등록 하기

```
C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get pod
NAME              READY   STATUS    RESTARTS   AGE
nginx-rs-bhm2s    1/1     Running   0          2m56s
nginx-rs-jf8sc    1/1     Running   0          2m56s
nginx-rs-twzdj    1/1     Running   0          2m56s
web2-rs-np5xs     1/1     Running   0          2m52s
web2-rs-qpgrd     1/1     Running   0          2m52s
web2-rs-zq5jz     1/1     Running   0          2m52s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.8.0.1      <none>        443/TCP        3m4s
nginx-svc    NodePort    10.8.2.169    <none>        80:30100/TCP   3m4s
web2-svc     NodePort    10.8.12.110   <none>        80:30200/TCP   3m1s

C:₩Users₩hifro₩AppData₩Local₩Google₩Cloud SDK>kubectl get ingress
NAME            HOSTS                          ADDRESS         PORTS   AGE
nginx-ingress   www.frodo.kr,frodo.frodo.kr    34.107.241.97   80      119s
```

| | | |
|---|---|---|
| A | www | 34.107.241.97 |
| A | frodo | 34.107.241.97 |

## TLS를 처리하도록 인그레스 구성

- ◆ 인그레스가 HTTPS를 지원
- ◆ Client 에서 인그레스 컨트롤러까지는는 TLS통신으로 처리 하고 인그레스 컨트롤러에서 파드간은 암호화 되지 않은 통신을 한다. (TLS termination)



**TLS를 처리하도록 인그레스 구성 실습**

- ◆ TLS 암호화를 위한 인증서를 준비한다.
- ◆ RS와 노드포트 서비스 생성
- ◆ TLS 인그레스 생성
- ◆ 접속 테스트

## TLS를 처리하도록 인그레스 구성 실습

◆ TLS 암호화를 위한 인증서를 준비한다.
개인키와 인증서를 .kubectl을 실행할 경로에 복사 한 후
$ kubectl create secret tls *secret명* --cert=*서버키 명* --key=*개인키명* 실행

```
-a----     2020-05-20  오전 12:25          2201 www.frodo.kr.crt
-a----     2020-05-20  오전 12:29          1706 www.frodo.kr.key


PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl create secret tls tls-secret --cert=www.frodo.kr.crt --key=www.frodo.kr.key
secret/tls-secret created
```

## TLS를 처리하도록 인그레스 구성 실습

◆ RS와 노트포트 서비스 생성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
spec:
  selector:
    run: nginx
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30100
```

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      run: nginx
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - name: nginx
        image: hifrodo/modu:nginxv1
```

## TLS를 처리하도록 인그레스 구성 실습

◆ TLS 인그레스 생성

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  tls:
  - hosts:
    - www.frodo.kr
    secretName: tls-secret
  rules:
  - host: www.frodo.kr
    http:
      paths:
      - path: /
        backend:
          serviceName: nginx-svc
          servicePort: 80
```

앞서 작성한 tls-secret를 참조

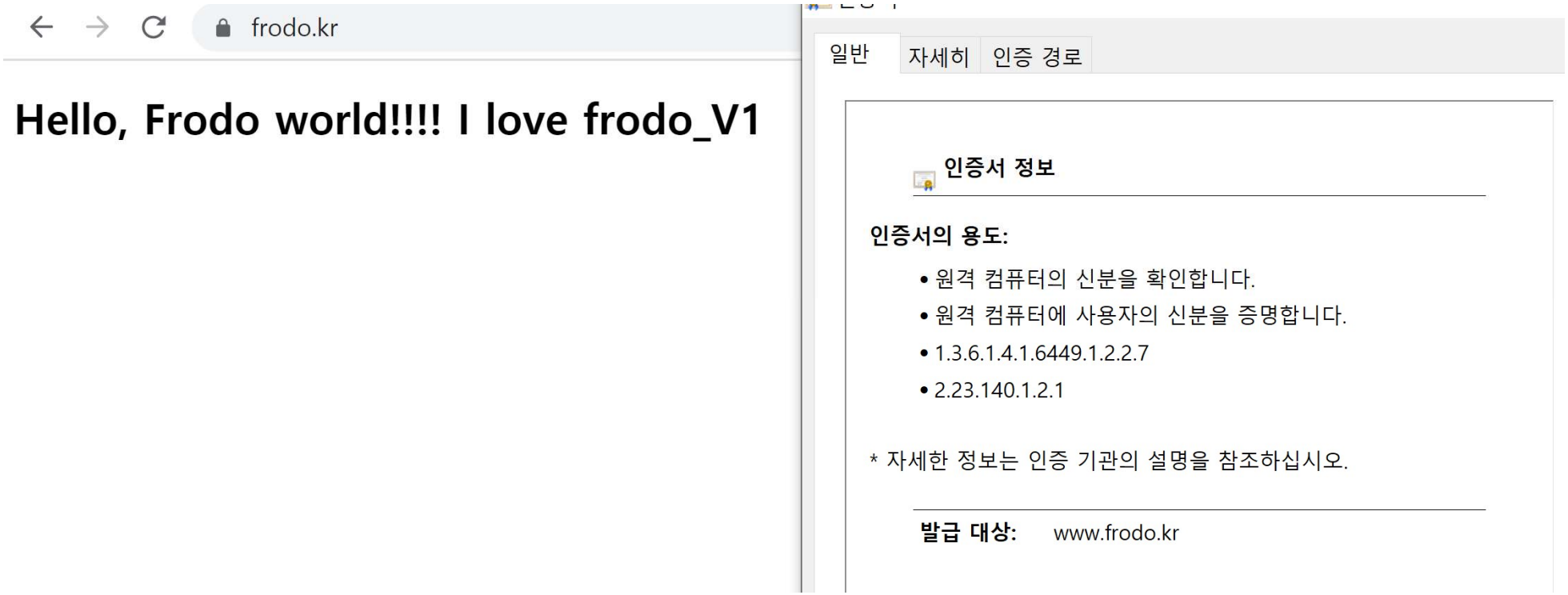
## TLS를 처리하도록 인그레스 구성 실습

◆ Ingress 확인 – 인그레스에 443이 추가 된 것을 확인

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-chmcr   1/1     Running   0          4m31s
nginx-rs-lst9p   1/1     Running   0          4m31s
nginx-rs-q4vkq   1/1     Running   0          4m31s
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.8.0.1       <none>        443/TCP        40m
nginx-svc    NodePort    10.8.14.226    <none>        80:30100/TCP   39m
web2-svc     NodePort    10.8.3.11      <none>        80:30200/TCP   38m
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get ingress
NAME            HOSTS           ADDRESS          PORTS     AGE
nginx-ingress   www.frodo.kr    34.107.241.97    80, 443   2m51s
```

## TLS를 처리하도록 인그레스 구성 실습

◆ HTTPS로 접속 확인



# 5.5 파드가 연결을 수락할 준비가 됐을 때 신호 보내기

## 파드 실행

- 적절한 레이블을 가진 파드는 만들어지자 마자 서비스의 일부가 되어 요청을 전달 받는다
- 이때 파드가 처리할 준비가 되지 않았다면 응답이 느려 질 수 있다
- 이런 경우 파드가 요청을 즉시 받지 않도록 가동중인 파드에게 요청을 전달 하지 않을 수 있다.

## 레디니스 프로브 소개

- livenessProbe: 컨테이너가 동작 중인지 여부를 나타낸다. 만약 활성 프로브(liveness probe)에 실패한다면, kubelet은 컨테이너를 죽이고, 해당 컨테이너는 재시작 정책의 대상이 된다. 만약 컨테이너가 활성 프로브를 제공하지 않는 경우, 기본 상태는 Success이다.

- readinessProbe: 컨테이너가 요청을 처리할 준비가 되었는지 여부를 나타낸다. 만약 준비성 프로브(readiness probe)가 실패한다면, 엔드포인트 컨트롤러는 파드에 연관된 모든 서비스들의 엔드포인트에서 파드의 IP주소를 제거한다. 준비성 프로브의 초기 지연 이전의 기본 상태는 Failure이다. 만약 컨테이너가 준비성 프로브를 지원하지 않는다면, 기본 상태는 Success이다.

- startupProbe: 컨테이너 내의 애플리케이션이 시작되었는지를 나타낸다. 스타트업 프로브(startup probe)가 주어진 경우, 성공할 때 까지 다른 나머지 프로브는 활성화 되지 않는다. 만약 스타트업 프로브가 실패하면, kubelet이 컨테이너를 죽이고, 컨테이너는 재시작 정책에 따라 처리된다. 컨테이너에 스타트업 프로브가 없는 경우, 기본 상태는 Success이다.

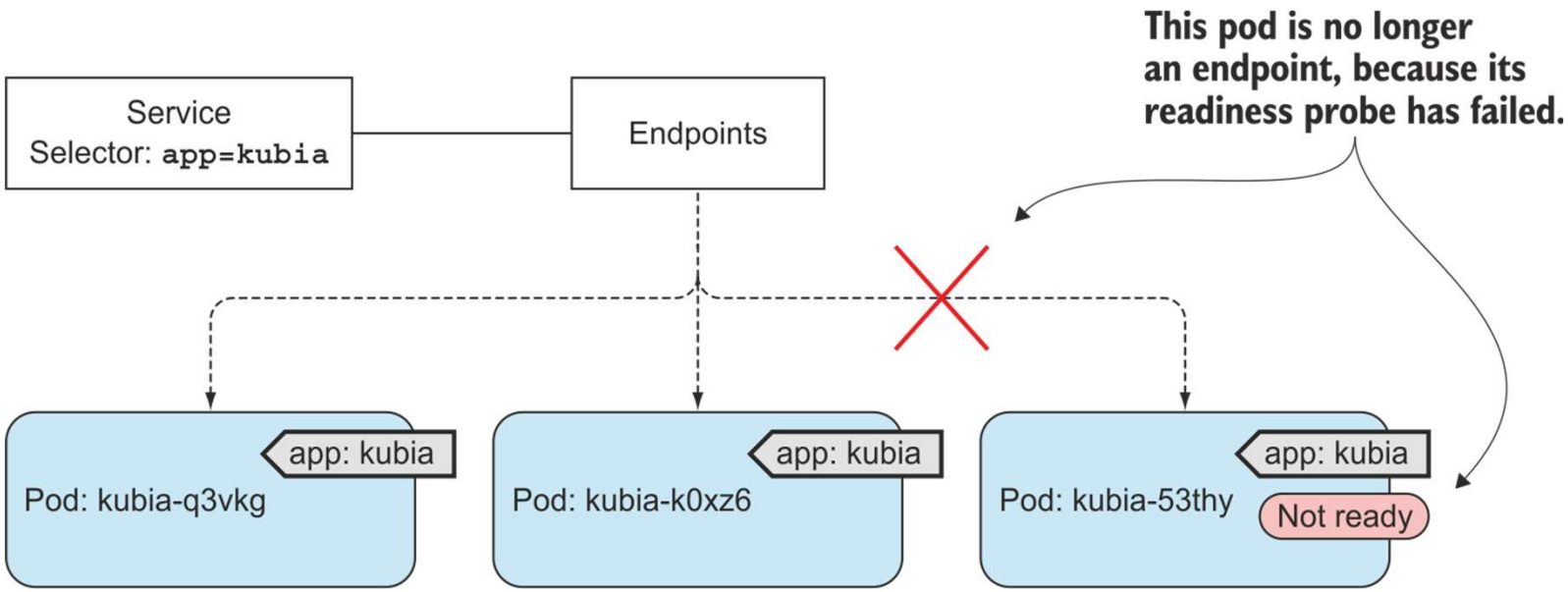
### 프르브 핸들러의 종류

- ExecAction 은 컨테이너 내에서 지정된 명령어를 실행한다. 명령어가 상태 코드 0으로 종료되면 진단이 성공한 것으로 간주한다.
- TCPSocketAction 은 지정된 포트에서 컨테이너의 IP주소에 대해 TCP 검사를 수행한다. 포트가 활성화되어 있다면 진단이 성공한 것으로 간주한다.
- HTTPGetAction 은 지정한 포트 및 경로에서 컨테이너의 IP주소에 대한 HTTP Get 요청을 수행한다. 응답의 상태 코드가 200보다 크고 400보다 작으면 진단이 성공한 것으로 간주한다.

# 5.5 레디니스 프로브 실습

## 프르브 핸들러의 종류

- ◆ 레디니스가 포함 된 RS생성
- ◆ 파드 상태 확인
- ◆ 레디니스가 응답 받을 수 있는 파일 추가 후 파드 동작



## 프르브 핸들러의 종류

◆ 레디니스가 포함 된 RS생성

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      run: nginx
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - name: nginx
        image: hifrodo/nginx:v1
        readinessProbe:
            exec:
              command:
                - ls
                - /var/test
```

## 프르브 핸들러의 종류

◆ 파드 상태 확인
파드가 READY인 것을 확인

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-6rvg6   0/1     Running   0          2m45s
nginx-rs-956ft   0/1     Running   0          2m45s
nginx-rs-mkrzn   0/1     Running   0          2m45s
```

# 5.5 레디니스 프로브 실습

## 프르브 핸들러의 종류

◆ 레디니스가 응답 받을 수 있는 파일 추가 후 파드 동작
파드 중 하나에 bash로 접속 후 /var/test 파일 생성

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl exec -it nginx-rs-6rvg6 bash
[root@nginx-rs-6rvg6 /]# touch /var/test
```

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get pod
NAME             READY   STATUS    RESTARTS   AGE
nginx-rs-6rvg6   1/1     Running   0          5m23s
nginx-rs-956ft   0/1     Running   0          5m23s
nginx-rs-mkrzn   0/1     Running   0          5m23s
```

# 5.6 헤드리스 서비스로 개별 파드 찾기



**헤드리스 서비스란**

- 단일 서비스 IP는 필요치 않다. 이 경우, "헤드리스" 서비스라는 것을 만들 수 있다
- 쿠바네티스는 클라이언트가 DNS 조회로 파드 IP 를 찾을 수 있다.
- 헤드리스 서비스 생성은  명시적으로 클러스터 IP (.spec.clusterIP)에 "None"을 지정한다.

**헤드리스 서비스 실습**

- ◆ 헤드리스 서비스의 RS와 서비스 생성
- ◆ 서비스 상태 확인
- ◆ FQDN으로 서비스 접속하기

# 5.6 헤드리스 서비스로 개별 파드 찾기

## 헤드리스 서비스 실습

◆ 헤드리스 서비스의 RS와 서비스 생성

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
     app: nginx
  template:
    metadata:
      labels:
       app: nginx
    spec:
      containers:
      - name: nginx
        image: hifrodo/nginx:v1
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
spec:
  selector:
    app: nginx
  clusterIP: None
  ports:
  - port: 80
    targetPort: 80
```

## 헤드리스 서비스 실습

◆ 서비스 상태 확인

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl run --generator=run-pod/v1 -it --rm centosshell --image=centos /bin/bash
If you don't see a command prompt, try pressing enter.
[root@centosshell /]# curl nginx-svc.default.svc.cluster.local
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Frodo word</title>
</head>
<body>
    <h1>Hello, Frodo world!!!! I love frodo_V1 </h1>
</body>
</html>
[root@centosshell /]#
```

# 5.6 헤드리스 서비스로 개별 파드 찾기

## 헤드리스 서비스 실습

◆ FQDN으로 서비스 접속하기
ping으로 FQDN 결과를 확인하면 여러 파드에서 응답을 주는 것을 확인 가능

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get pod -o wide
NAME            READY   STATUS        RESTARTS   AGE   IP          NODE                                          NOMINATED NODE   READINESS GATES
centosshell     0/1     Terminating   0          46s   <none>      gke-cluster-1-default-pool-61223293-56mp       <none>           <none>
nginx-rs-8zh9m  1/1     Running       0          14m   10.4.0.28   gke-cluster-1-default-pool-61223293-6xzr       <none>           <none>
nginx-rs-hj6ks  1/1     Running       0          14m   10.4.0.29   gke-cluster-1-default-pool-61223293-6xzr       <none>           <none>
nginx-rs-zfzh6  1/1     Running       0          14m   10.4.0.27   gke-cluster-1-default-pool-61223293-6xzr       <none>           <none>
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.8.0.1     <none>        443/TCP   16m
nginx-svc    ClusterIP   None         <none>        80/TCP    14m
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl run --generator=run-pod/v1 -it --rm centosshell --image=centos /bin/bash
If you don't see a command prompt, try pressing enter.
[root@centosshell /]# ping nginx-svc.default.svc.cluster.local
PING nginx-svc.default.svc.cluster.local (10.4.0.27) 56(84) bytes of data.
64 bytes from 10.4.0.27 (10.4.0.27): icmp_seq=1 ttl=62 time=1.16 ms
64 bytes from 10.4.0.27 (10.4.0.27): icmp_seq=2 ttl=62 time=0.271 ms
64 bytes from 10.4.0.27 (10.4.0.27): icmp_seq=3 ttl=62 time=0.284 ms
^C
--- nginx-svc.default.svc.cluster.local ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 34ms
rtt min/avg/max/mdev = 0.271/0.571/1.160/0.416 ms
[root@centosshell /]# ping nginx-svc.default.svc.cluster.local
PING nginx-svc.default.svc.cluster.local (10.4.0.29) 56(84) bytes of data.
64 bytes from 10.4.0.29 (10.4.0.29): icmp_seq=1 ttl=62 time=1.31 ms
64 bytes from 10.4.0.29 (10.4.0.29): icmp_seq=2 ttl=62 time=0.274 ms
64 bytes from 10.4.0.29 (10.4.0.29): icmp_seq=3 ttl=62 time=0.305 ms
```

# 5.6 헤드리스 서비스로 개별 파드 찾기

## 헤드리스 서비스 실습

◆ FQDN으로 서비스 접속하기
cluster 내부에 접속할 수 있는 centos bash 파드를 만들어 실습 한다.

```
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get pod
NAME              READY   STATUS    RESTARTS   AGE
nginx-rs-8zh9m    1/1     Running   0          25s
nginx-rs-hj6ks    1/1     Running   0          25s
nginx-rs-zfzh6    1/1     Running   0          25s
PS C:\Users\hifro\AppData\Local\Google\Cloud SDK\yaml> kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.8.0.1     <none>        443/TCP   2m37s
nginx-svc    ClusterIP   None         <none>        80/TCP    31s
```