4th Week

네번째 뵙겠습니다 ?!

▷ 출석 체크도 한 번 해보시면 어떠세요?!

- https://modulabs.co.kr/
- 모두연 홈페이지 → 로그인 → 마이페이지 → 참여한 랩·풀잎 → 자세히 보기 → 내 풀잎스쿨 출석 확인하기

▷ 지난 주 갑작스럽게 휴강(?)하게 되어 죄송합니다.

- **4 week : 02월 12일**
- **5 week : 02월 19일**
- **6 week : 02월 26일**
- **7 week : 03월 05일**
- **8 week : 03월 12일**

잡담 &
지난 수업 관련 이야기

///

Agenda

- ▷ [10m] make-friendship
- ▷ [20m] DNS / FQDN / Endpoints / Headless
- ▷ [20m] Flip - Ingress
- ▷ [20m] Wrap-Up & Hands-On
- ▷ [10m] Break-Time

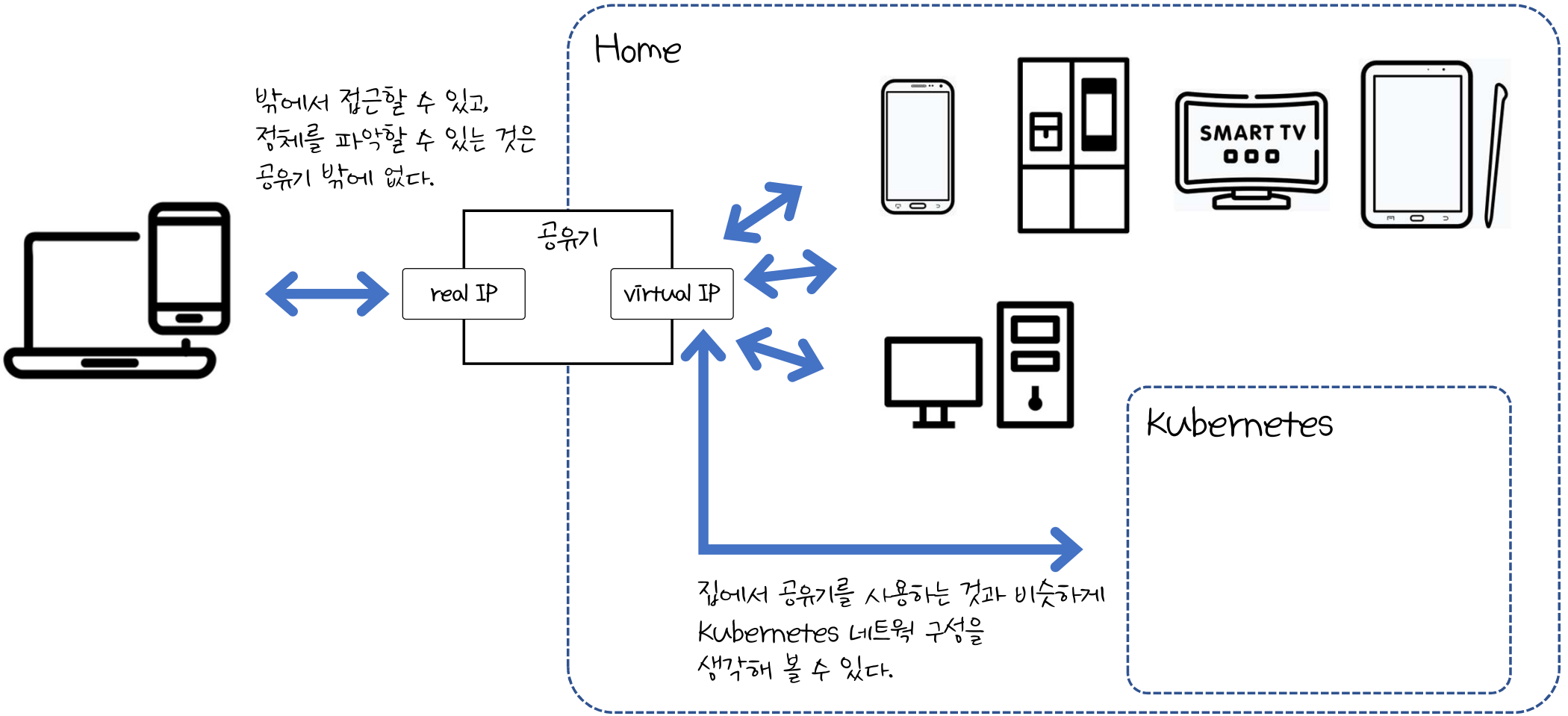
- ▷ [30m] MetalLB
- ▷ [40m] Advanced Network
- ▷ [20m] Quiz

///

잠깐만~!

Network overview

Real World (Public Internet)



///

Kubernetes

Network - DNS / FQDN

DNS (Domain Name System)

- 특정 컴퓨터(또는 네트워크로 연결된 임의의 장치)의 주소를 찾기 위해, 사람이 이해하기 쉬운 도메인 이름을 숫자로 된 식별 번호(IP 주소)로 변환

※ 참고 : https://www.youtube.com/watch?v=e2xLV7pCOLI

※ 참고 : https://ko.wikipedia.org/wiki/도메인_네임_시스템

FQDN (Fully Qualified Domain Name)

- 명확한 도메인 표기법

  . 소프트웨어 설치 중 도메인명을 요구하면, YAHOO.COM. 을 입력할지, WWW.YAHOO.COM. 을 입력할지 모호

  → Namespace 계층상에서 최종 호스트명을 포함하는 도메인명을 의미하는 FQDN을 요청하면 명확

- 원칙적으로 도메인의 표기는 네임스페이스상의 경로를 명확히 하기 위해 끝에 도트('.' 루트 도메인)를 포함

  . 하지만, 보통 도트를 생략하고 사용

www	호스트명
google.com.	도메인명
www.google.com.	FQDN

※ 참고 : http://doc.kldp.org/KoreanDoc/html/PoweredByDNS-KLDP/fqdn.html

CoreDNS

- CoreDNS는 쿠버네티스 클러스터의 DNS 역할을 수행할 수 있는, 유연하고 확장 가능한 DNS 서버

- CoreDNS 프로젝트도 CNCF가 관리

- 사용자는 기존 deployment인 kube-dns를 교체 가능

```
remote ❯ kubectl get pods --namespace kube-system -o wide

NAME                                    READY   STATUS    RESTARTS      AGE   IP               NODE      NOMINATED NODE   READINESS GATES
calico-kube-controllers-5788f6558-g5fl2 1/1     Running   3 (46m ago)   23h   192.168.100.201  worker1   <none>           <none>
calico-node-67cdg                       1/1     Running   2 (46m ago)   23h   192.168.100.202  worker2   <none>           <none>
calico-node-722h6                       1/1     Running   2 (47m ago)   23h   192.168.100.201  worker1   <none>           <none>
calico-node-jtspn                       1/1     Running   2 (47m ago)   23h   192.168.100.200  master    <none>           <none>
coredns-8474476ff8-8mxwm                1/1     Running   2 (46m ago)   23h   10.233.103.3     worker2   <none>           <none>
coredns-8474476ff8-z96xn                1/1     Running   2 (47m ago)   23h   10.233.70.7      master    <none>           <none>
dns-autoscaler-5ffdc7f89d-hhx9z         1/1     Running   2 (47m ago)   23h   10.233.70.9      master    <none>           <none>
kube-apiserver-master                   1/1     Running   3 (47m ago)   23h   192.168.100.200  master    <none>           <none>
kube-controller-manager-master          1/1     Running   3 (47m ago)   23h   192.168.100.200  master    <none>           <none>
kube-proxy-gfts9                        1/1     Running   2 (47m ago)   23h   192.168.100.201  worker1   <none>           <none>
kube-proxy-q8w2g                        1/1     Running   2 (46m ago)   23h   192.168.100.202  worker2   <none>           <none>
kube-proxy-r6l2t                        1/1     Running   2 (47m ago)   23h   192.168.100.200  master    <none>           <none>
kube-scheduler-master                   1/1     Running   3 (47m ago)   23h   192.168.100.200  master    <none>           <none>
kubernetes-dashboard-548847967d-ntgzz   1/1     Running   3 (47m ago)   23h   10.233.110.6     worker1   <none>           <none>
kubernetes-metrics-scraper-6d49f96c97-56q6s 1/1 Running   2 (47m ago)   23h   10.233.110.5     worker1   <none>           <none>
metrics-server-6978dd689f-dgst9         1/1     Running   2 (47m ago)   23h   10.233.70.8      master    <none>           <none>
...
```

※ 참고 : https://kubernetes.io/ko/docs/tasks/administer-cluster/coredns/

Hands-On - FQDN 1/3

FQDN 실습을 위해 ReplicaSet 생성해서 Pod들을 준비해 놓자

rs-node-web.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-labels

spec:
  replicas: 3

  selector:
    matchLabels:
      app: node-web

  template:
    metadata:
      labels:
        app: node-web
    spec:
      containers:
      - name: node-web
        image: whatwant/node-web:1.0
        ports:
        - containerPort: 8080
```

```
remote ❯ cd advanced-kubernetes/03-week/Service

remote ❯ kubectl create -f ./rs-node-web.yaml

replicaset.apps/rs-labels created


remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE   IP             NODE      NOMINATED NODE   READINESS GATES
rs-labels-8vzrh   1/1     Running   0          58s   10.233.103.5   worker2   <none>           <none>
rs-labels-tbshs   1/1     Running   0          57s   10.233.103.4   worker2   <none>           <none>
rs-labels-vhb7v   1/1     Running   0          57s   10.233.110.7   worker1   <none>           <none>
```

Hands-On - FQDN 2/3

ClusterIP 타입으로 Service를 생성하자.

svc-node-web.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-node

spec:
  ports:
  - port: 80
    targetPort: 8080

  selector:
    app: node-web
```

```
remote ❯ kubectl create -f ./svc-node-web.yaml

service/svc-node created


remote ❯ kubectl get services -o wide

NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE   SELECTOR
kubernetes   ClusterIP   10.233.0.1      <none>        443/TCP   23h   <none>
svc-node     ClusterIP   10.233.53.207   <none>        80/TCP    31s   app=node-web
```

당연히 Remote 환경에서는 접근을 할 수 없지만, Pod에서는 Service를 통해 접근할 수 있다.

```
remote ❯ curl -s http://10.233.53.207

^C


remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://10.233.53.207

You've hit rs-labels-tbshs
```

Hands-On - FQDN 3/3

- Kubernetes는 각 container의 /etc/resolv.conf 파일을 직접 관리

- FQDN (Fully Qualified Domain Name) : service name . namespace name . svc.cluster.local

```
remote ❯ kubectl exec -it rs-labels-8vzrh -- cat /etc/resolv.conf

search default.svc.cluster.local svc.cluster.local cluster.local skbroadband
nameserver 169.254.25.10
options ndots:5

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node

You've hit rs-labels-vhb7v

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default

You've hit rs-labels-vhb7v

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default.svc

You've hit rs-labels-8vzrh

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default.svc.cluster

^Ccommand terminated with exit code 130

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default.svc.cluster.local

^Ccommand terminated with exit code 130
```

Trouble-Shooting

- 앞에서 진행한 결과를 보면 원하는 내용과 다르다. 원인은 잘못된 DNS 정보 삽입. 이유는 Network 설정 이슈!

- 아래 내용은 각 상황에 따라 다를 수 있음을 염두에 두어야 한다.

```
remote ❯ ssh vagrant@192.168.100.200

master ❯ sudo nano /etc/netplan/00-installer-config.yaml



master ❯ sudo nano /etc/netplan/50-vagrant.yaml


master ❯ sudo reboot
```

Network 설정 파일 2개를 옆의 내용 참고해서 수정한 뒤, 재부팅 하자

master 뿐만 아니라, worker1 / worker2 도 마찬가지로 수정해주어야 한다.

공유기 환경에서 DHCP 설정으로 인해 공유기의 DNS 설정이 반영되어 발생하는 문제이다.

00-installer-config.yaml

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      dhcp4: no
  version: 2
```

50-vagrant.yaml

```
---
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses:
      - 192.168.100.200/24
      dhcp4: no
      dhcp6: no
      gateway4: 192.168.100.1
      nameservers:
        addresses: [8.8.8.8,8.8.4.4]
```

Hands-On - FQDN 3/3 - Fix!

- Kubernetes는 각 container의 /etc/resolv.conf 파일을 직접 관리

- FQDN (Fully Qualified Domain Name) : service name . namespace name . svc.cluster.local

```
remote ❯ kubectl exec -it rs-labels-8vzrh -- cat /etc/resolv.conf

search default.svc.cluster.local svc.cluster.local cluster.local
nameserver 169.254.25.10
options ndots:5

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node

You've hit rs-labels-vhb7v

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default

You've hit rs-labels-vhb7v

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default.svc

You've hit rs-labels-8vzrh

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default.svc.cluster

command terminated with exit code 6

remote ❯ kubectl exec -it rs-labels-8vzrh -- curl -s http://svc-node.default.svc.cluster.local

You've hit rs-labels-tbshs
```

///

Kubernetes

Network - Endpoints/Headless

Service

- Service로 노출되는 Pod의 IP 주소와 Port 목록

- Client → kube-proxy → Endpoints 중 하나의 IP/Port 선택 → 들어온 연결을 대상 Pod의 수신 대기 서버로 전달

```
remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS       AGE    IP              NODE      NOMINATED NODE   READINESS GATES
rs-labels-8vzrh   1/1     Running   3 (3h1m ago)   4h2m   10.233.103.13   worker2   <none>           <none>
rs-labels-tbshs   1/1     Running   3 (3h1m ago)   4h2m   10.233.103.12   worker2   <none>           <none>
rs-labels-vhb7v   1/1     Running   3 (3h1m ago)   4h2m   10.233.110.16   worker1   <none>           <none>


remote ❯ kubectl describe service svc-node

Name:              svc-node
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=node-web
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.233.53.207
IPs:               10.233.53.207
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.233.103.12:8080,10.233.103.13:8080,10.233.110.16:8080
Session Affinity:  None
Events:            <none>
```

svc-node-web.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-node

spec:
  ports:
  - port: 80
    targetPort: 8080

  selector:
    app: node-web
```

Endpoints

- Endpoint 리소스도 만들 수 있을까?

  . Selectors를 지정하지 않고, 직접 생성한 Endpoints 활용하기

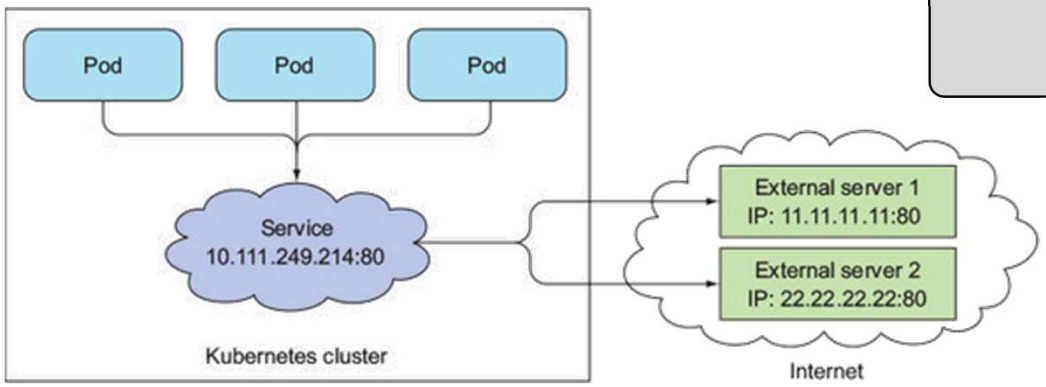
  . Service와 Endpoints 명칭 일치 필요

svc-no-selector.yaml

```yaml
apiVersion: v1
kind: Service

metadata:
  name: external-service

spec:
  ports:
  - port: 80
```

ep-addresses.yaml

```yaml
apiVersion: v1
kind: Endpoints

metadata:
  name: external-service

subsets:
  - addresses:
    - ip: 11.11.11.11
    - ip: 22.22.22.22
    ports:
    - port: 80
```



※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-5/144

Headless Service

- Pod 하나에 연결하는 것이 아니라, 생성된 모든 Pod에 연결하고 싶은 경우. 또는, 같은 Service에 묶인 다른 Pod와의 통신이 필요한 경우 사용

- `clusterIP: None` 설정을 하고, FQDN을 통해 접근

svc-node-web-headless.yaml

```yaml
apiVersion: v1
kind: Service

metadata:
  name: svc-node-web-headless

spec:
  clusterIP: None

  selector:
    app: node-web

  ports:
  - port: 80
    targetPort: 8080
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/04-week/headless

remote ❯ kubectl create -f ./svc-node-web-headless.yaml

service/svc-node-web-headless created


remote ❯ kubectl get services -o wide

NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE     SELECTOR
kubernetes             ClusterIP   10.233.0.1      <none>        443/TCP   28h     <none>
svc-node               ClusterIP   10.233.53.207   <none>        80/TCP    4h47m   app=node-web
svc-node-web-headless  ClusterIP   None            <none>        80/TCP    26s     app=node-web
```

※ 참고 : https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/

Headless Service Check

Headless Service는 DNS를 통해서 활용하는 것이기에 이를 확인하기 위해 `nslookup` 명령어를 이용해보고자 했다.
`nslookup` 명령어가 설치되어 있는 Pod를 하나 생성하고 이를 사용했다.

```
remote ❯ kubectl apply -f https://k8s.io/examples/admin/dns/dnsutils.yaml

pod/dnsutils created


remote ❯ kubectl exec -it dnsutils -- nslookup svc-node-web-headless

Server:     169.254.25.10
Address:    169.254.25.10#53

Name:       svc-node-web-headless.default.svc.cluster.local
Address:    10.233.103.13
Name:       svc-node-web-headless.default.svc.cluster.local
Address:    10.233.103.12
Name:       svc-node-web-headless.default.svc.cluster.local
Address:    10.233.110.16


remote ❯ kubectl exec -it dnsutils -- nslookup svc-node

Server:     169.254.25.10
Address:    169.254.25.10#53

Name:       svc-node.default.svc.cluster.local
Address:    10.233.53.207
```

3개가 다 확인된다

1개가 확인되는 것이 일반적이다.

dnsutils.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: dnsutils
  namespace: default

spec:
  containers:
  - name: dnsutils
    image: k8s.gcr.io/e2e-test-images/jessie-dnsutils:1.3

    command:
      - sleep
      - "3600"

    imagePullPolicy: IfNotPresent

  restartPolicy: Always
```

///

Flip Learning

(Network - Ingress)

박남준님

///

Kubernetes

Network - Ingress

Ingress Controller

- Ingress는 cluster 외부에서 내부 Service로 HTTP와 HTTPS 경로 노출

- 클라이언트가 요청한 호스트와 경로에 따라 요청을 전달할 서비스가 결정

  . HTTP(S)기반의 L7 LoadBalancing 기능을 제공하는 컴포넌트

- Ingress 구현체 : 각 구현체마다 설정 방법 차이 존재

  . 구글 클라우드 : https://github.com/kubernetes/ingress-gce

  . 오픈소스 (Ingress NGINX Controller) : https://github.com/kubernetes/ingress-nginx

  . 오픈소스 (NGINX Ingress Controller) : https://github.com/nginxinc/kubernetes-ingress

  . 상용 (F5 BIG IP Controller) : http://clouddocs.f5.com/products/connectors/k8s-bigip-ctlr

※ 참고 : https://kubernetes.io/ko/docs/concepts/services-networking/ingress/

Ingress Controller Install

Kubernetes에서 배포하고 있는 패키지로 설치해보는데, 아래 링크에서 bare-metal 설치 방법을 찾아보면 된다

- https://github.com/kubernetes/ingress-nginx/blob/main/docs/deploy/index.md#bare-metal-clusters

```
remote ❯ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.1/deploy/static/provider/baremetal/deploy.yaml

namespace/ingress-nginx created
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
```

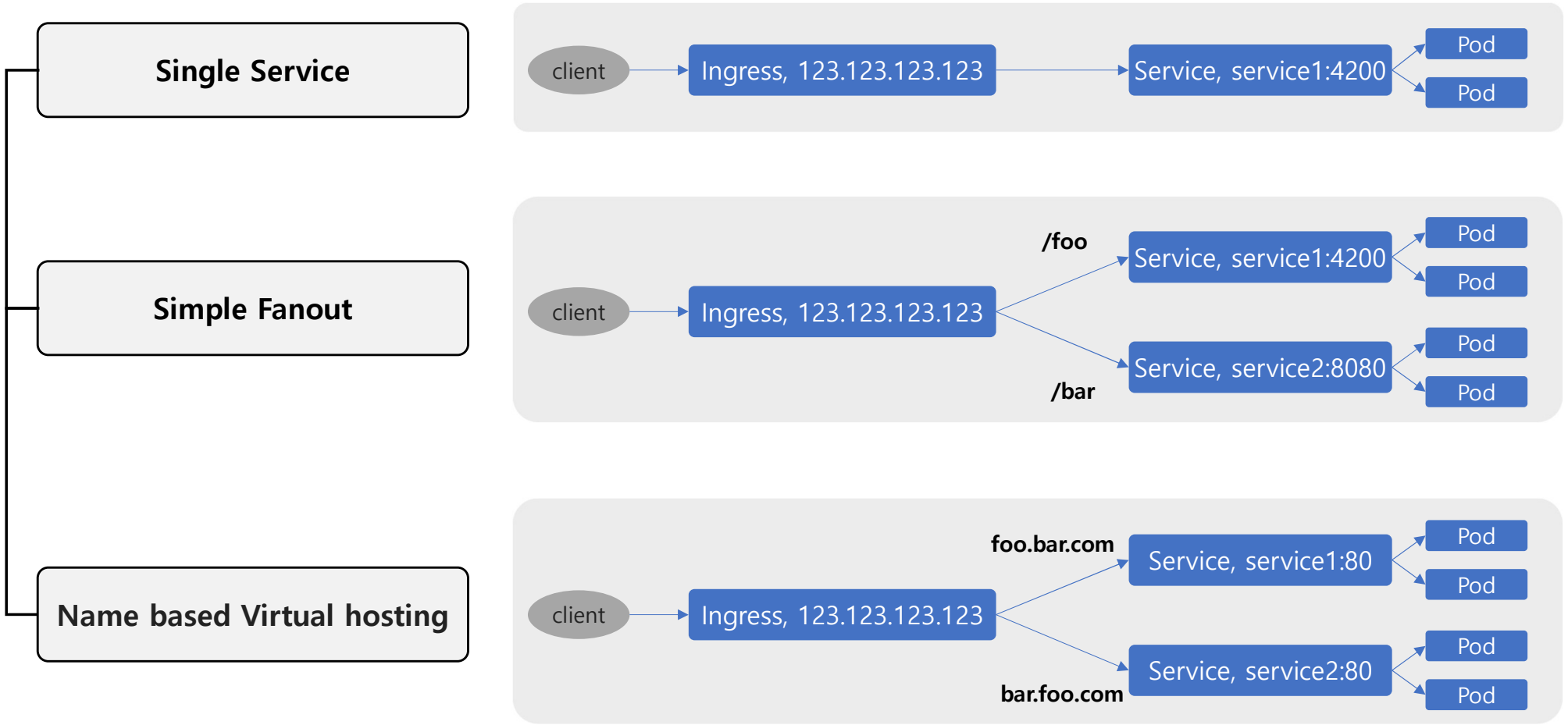
무언가 몽창 설치하는 것을 볼 수 있다.

Ingress Controller Check

```
remote ❯ kubectl get namespaces

NAME              STATUS   AGE
default           Active   2d7h
ingress-nginx     Active   5m47s
...


remote ❯ kubectl get pods -o wide --namespace kube-system

NAME                                    READY   STATUS    RESTARTS       AGE    IP               NODE      NOMINATED NODE   READINESS GATES
...
nginx-proxy-worker1                     1/1     Running   7 (44m ago)    2d7h   192.168.100.201   worker1   <none>           <none>
nginx-proxy-worker2                     1/1     Running   7 (44m ago)    2d7h   192.168.100.202   worker2   <none>           <none>
...


remote ❯ kubectl get pods -o wide --namespace ingress-nginx

NAME                                   READY   STATUS      RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
ingress-nginx-admission-create--1-gr5zf     0/1     Completed   0          10m   10.233.110.20   worker1   <none>           <none>
ingress-nginx-admission-patch--1-87plx      0/1     Completed   0          10m   10.233.103.16   worker2   <none>           <none>
ingress-nginx-controller-778574f59b-7rnzn   1/1     Running     0          10m   10.233.103.17   worker2   <none>           <none>


remote ❯ kubectl get serviceaccounts -o wide --namespace ingress-nginx

NAME                    SECRETS   AGE
default                 1         15m
ingress-nginx           1         15m
ingress-nginx-admission   1         15m
```

///

Types of Ingress



※ 참고 : https://kubernetes.io/docs/concepts/services-networking/ingress/

///

Hands-On: Container Image 준비

`Simple Fanout` 타입의 Ingress 실습을 위해서 Pod를 하나 더 필요해서, docker image를 하나 더 만들었다.

app.js

```
const http = require('http');
const os = require('os');

console.log("node-web server starting...");

var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "(Ver2.0)\n");
};

var www = http.createServer(handler);
www.listen(8080);
```

Dockerfile

```
FROM node:latest
ADD app.js /app.js
ENTRYPOINT ["node", "app.js"]
```

기존 `node-web:1.0`과 구분이 될 정도만 살짝 변경한 후

`node-web:2.0` 버전으로 push 했다.

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/04-week/container


remote ❯ docker build -t node-web:2.0 .


remote ❯ docker tag node-web:2.0 <user-id>/node-web:2.0


remote ❯ docker push <user-id>/node-web:2.0
```

Hands-On: ReplicaSet YAML

'ReplicaSet'을 이용해서 2종의 어플리케이션 그룹을 만들어 놓으려고 한다.

rs-web-v1.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-web-v1

spec:
  replicas: 3
  selector:
    matchLabels:
      app: node-web-v1

  template:
    metadata:
      labels:
        app: node-web-v1

    spec:
      containers:
      - name: node-web
        image: whatwant/node-web:1.0
        ports:
        - containerPort: 8080
        imagePullPolicy: Always
```

rs-web-v2.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-web-v2

spec:
  replicas: 3
  selector:
    matchLabels:
      app: node-web-v2

  template:
    metadata:
      labels:
        app: node-web-v2

    spec:
      containers:
      - name: node-web
        image: whatwant/node-web:2.0
        ports:
        - containerPort: 8080
        imagePullPolicy: Always
```

Hands-On: ReplicaSet 생성

3개 1묶음으로 해서 2개의 묶음으로 미리 만들어 놓자

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/04-week/replica

remote ❯ kubectl create -f ./rs-web-v1.yaml

replicaset.apps/rs-web-v1 created


remote ❯ kubectl create -f ./rs-web-v2.yaml

replicaset.apps/rs-web-v2 created


remote ❯ kubectl get pods -o wide

NAME             READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-v1-7zv4l  1/1     Running   0          3m27s   10.233.110.25   worker1   <none>           <none>
rs-web-v1-9lcxc  1/1     Running   0          3m27s   10.233.103.25   worker2   <none>           <none>
rs-web-v1-d59sr  1/1     Running   0          3m27s   10.233.103.26   worker2   <none>           <none>
rs-web-v2-cdv4s  1/1     Running   0          3m8s    10.233.103.27   worker2   <none>           <none>
rs-web-v2-gkhs6  1/1     Running   0          3m8s    10.233.110.26   worker1   <none>           <none>
rs-web-v2-zhccb  1/1     Running   0          3m8s    10.233.103.28   worker2   <none>           <none>
```

Hands-On: Service 생성

`ClusterIP` 유형으로 해도 되지만, 여기에서는 `NodePort` 유형으로 `Service`를 만들어보자.

svc-web-nodeport-v1.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-node-web-v1

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30001
  selector:
    app: node-web-v1
```

svc-web-nodeport-v2.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-node-web-v2

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30002
  selector:
    app: node-web-v2
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/04-week/service

remote ❯ kubectl create -f ./svc-web-nodeport-v1.yaml
remote ❯ kubectl create -f ./svc-web-nodeport-v2.yaml

remote ❯ kubectl get services -o wide

NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE     SELECTOR
kubernetes       ClusterIP   10.233.0.1      <none>        443/TCP        7d20h   <none>
svc-node-web-v1  NodePort    10.233.28.43    <none>        80:30001/TCP   8s      app=node-web-v1
svc-node-web-v2  NodePort    10.233.26.164   <none>        80:30002/TCP   5s      app=node-web-v2
```

///

Hands-On: Status Check

```
remote ❯ kubectl get pods -o wide

NAME             READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-v1-7zv4l  1/1     Running   0          4h40m   10.233.110.25   worker1   <none>           <none>
rs-web-v1-9lcxc  1/1     Running   0          4h40m   10.233.103.25   worker2   <none>           <none>
rs-web-v1-d59sr  1/1     Running   0          4h40m   10.233.103.26   worker2   <none>           <none>
rs-web-v2-gzbqb  1/1     Running   0          44s     10.233.110.27   worker1   <none>           <none>
rs-web-v2-p85wx  1/1     Running   0          44s     10.233.103.29   worker2   <none>           <none>
rs-web-v2-sd2ds  1/1     Running   0          44s     10.233.103.30   worker2   <none>           <none>

remote ❯ kubectl get replicasets -o wide

NAME       DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES                   SELECTOR
rs-web-v1  3         3         3       4h40m   node-web     whatwant/node-web:1.0    app=node-web-v1
rs-web-v2  3         3         3       50s     node-web     whatwant/node-web:2.0    app=node-web-v2

remote ❯ kubectl get services -o wide

NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE     SELECTOR
kubernetes       ClusterIP   10.233.0.1      <none>        443/TCP        7d20h   <none>
svc-node-web-v1  NodePort    10.233.28.43    <none>        80:30001/TCP   81s     app=node-web-v1
svc-node-web-v2  NodePort    10.233.26.164   <none>        80:30002/TCP   78s     app=node-web-v2

remote ❯ curl -s http://192.168.100.201:30001

You've hit rs-web-v1-wkkk9

remote ❯ curl -s http://192.168.100.201:30002

You've hit rs-web-v2-gzbqb (Ver2.0)
```

Hands-On: Ingress Create (Simple Fanout)

ingress-node-web.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress

metadata:
  name: ingress-node-web
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: /

spec:
  rules:
  - http:
      paths:
      - path: /foo
        pathType: Prefix
        backend:
          service:
            name: svc-node-web-v1
            port:
              number: 80
      - path: /bar
        pathType: Prefix
        backend:
          service:
            name: svc-node-web-v2
            port:
              number: 80
```

`host`를 별도로 지정하지 않으면, 들어오는 모든 입력에 적용된다.
즉, IP로 접근을 하더라도 해당 내용이 적용되는 것이다.

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/04-week/ingress

remote ❯ kubectl create -f ./ingress-node-web.yaml

ingress.networking.k8s.io/ingress-node-web created
```

Hands-On: Simple Fanout 1/2

앞에서 생성한 Ingress를 확인해 보자

```
remote ❯ kubectl get ingresses -o wide

NAME              CLASS    HOSTS   ADDRESS           PORTS   AGE
ingress-node-web  <none>   *       192.168.100.201   80      29s


remote ❯ kubectl describe ingress ingress-node-web

Name:             ingress-node-web
Namespace:        default
Address:          192.168.100.201
Default backend:  default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /foo   svc-node-web-v1:80 (10.233.110.33:8080,10.233.110.34:8080,10.233.110.35:8080)
              /bar   svc-node-web-v2:80 (10.233.110.32:8080,10.233.110.36:8080,10.233.110.37:8080)
Annotations:  kubernetes.io/ingress.class: nginx
              nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason  Age                From                      Message
  ----    ------  ----               ----                      -------
  Normal  Sync    43s (x2 over 49s)  nginx-ingress-controller  Scheduled for sync
```

Hands-On: Simple Fanout 2/2

우리가 설치한 Ingress Controller는 NodePort로 설정되어 있으며, 80/433 포트가 이미 등록되어 있다. 확인해 보자.

```
remote ❯ kubectl get services -o wide --namespace ingress-nginx

NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE     SELECTOR
ingress-nginx-controller             NodePort    10.233.45.186   <none>        80:31459/TCP,443:30693/TCP   5d13h   app.kubernetes.io/component=co..
ingress-nginx-controller-admission   ClusterIP   10.233.30.23    <none>        443/TCP                      5d13h   app.kubernetes.io/component=co..
```

그러면, 이제 우리가 생성한 Ingress를 이용해서 접속이 잘되는지 확인해 보자.

```
remote ❯ curl -s http://192.168.100.201:31459/foo

You've hit rs-web-v1-wkkk9


remote ❯ curl -s http://192.168.100.201:31459/bar

You've hit rs-web-v2-gzbqb (Ver2.0)
```

///

Break

돌아오셨으면 채팅창에
복귀!
타이핑하기!

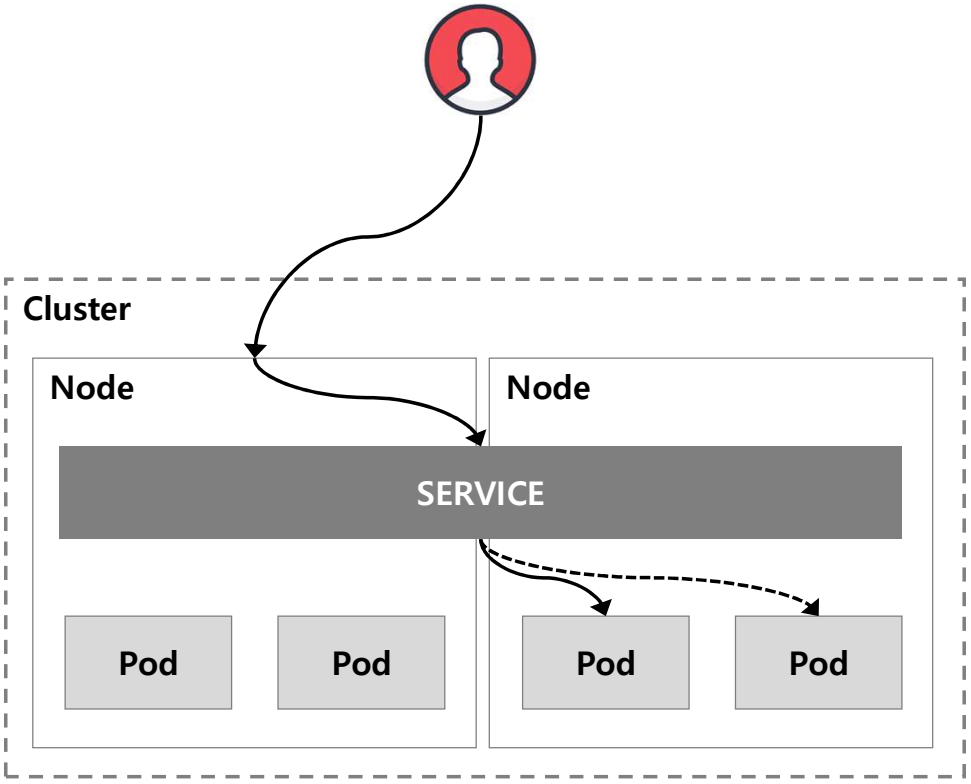
///

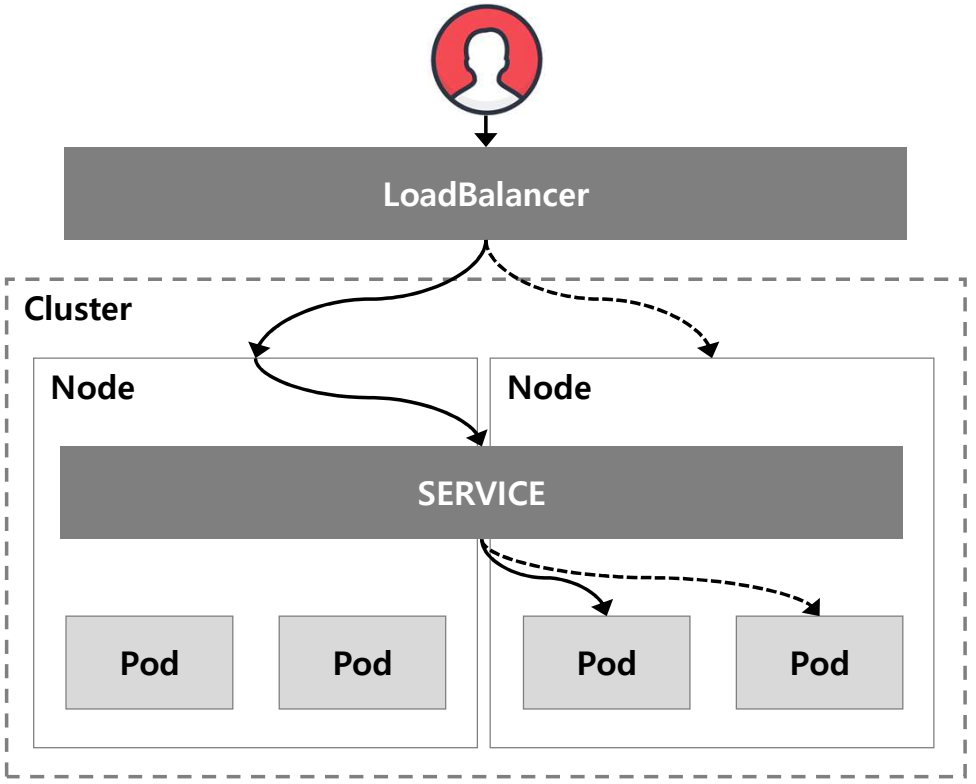
Kubernetes

Network - LoadBalancer

Why LoadBalancer ?

NodePort vs LoadBalancer





KCD 2020

Korea Community Day 2020

- https://kcd2020.festa.io/

시간	Track 1	Track 2
13:00 ~ 13:25	백 마디 말보다 중요한 것: Backend.AI의 오픈소스 문서 작성과 자동화 경험 신정규 (Backend.AI) 발표자료 / 발표영상	코드 주석으로 GitHub Pages를 만들 수 있다? 박동하 (C++ Korea) 발표자료 / 발표영상
13:25 ~ 13:50	쿠버네티스의 로드밸런서 서비스를 온프렘(On-Premises)에서 쓸순 없을까? 조훈 (IT인프라엔지니어그룹) 발표자료 / 발표영상	내가 태국(망해가는 회사)에서 일하는 방법 김태우 (비원이즈) 발표자료 / 발표영상
13:50 ~ 14:15	오픈소스 커뮤니티의 딜레마 김종민 (Elastic) 발표자료 / 발표영상	Win32 API로 다루는 WSL 남정현 (한국WSL사용자그룹) 발표자료 / 발표영상

[발표자료] : 링크

[강의영상] : https://www.youtube.com/watch?v=wlQHQbdiYQk

///

MetalLB

Installation

Requirement - 1/3

- MetalLB requires the following to function:

① A Kubernetes cluster, running Kubernetes 1.13.0 or later, that does not already have network load-balancing functionality.

② A cluster network configuration that can coexist with MetalLB.

③ Some IPv4 addresses for MetalLB to hand out.

④ When using the BGP operating mode, you will need one or more routers capable of speaking BGP.

⑤ Traffic on port 7946 (TCP & UDP) must be allowed between nodes, as required by memberlist.

① Kubernetes version

```
remote ❯ kubectl get nodes -o wide

NAME      STATUS   ROLES                  AGE   VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
master    Ready    control-plane,master   9d    v1.22.5   192.168.100.200   <none>        Ubuntu 20.04.3 LTS   5.4.0-96-generic   containerd://1.5.8
worker1   Ready    <none>                 9d    v1.22.5   192.168.100.201   <none>        Ubuntu 20.04.3 LTS   5.4.0-96-generic   containerd://1.5.8
worker2   Ready    <none>                 9d    v1.22.5   192.168.100.202   <none>        Ubuntu 20.04.3 LTS   5.4.0-96-generic   containerd://1.5.8
```

Kubernetes 1.13.0 버전 이상이고, 다른 load-balancing 기능을 갖고 있지 않으면 된다.

※ 참고 : https://metallb.universe.tf/#requirements

Requirement - 2/3

② Cluster Network

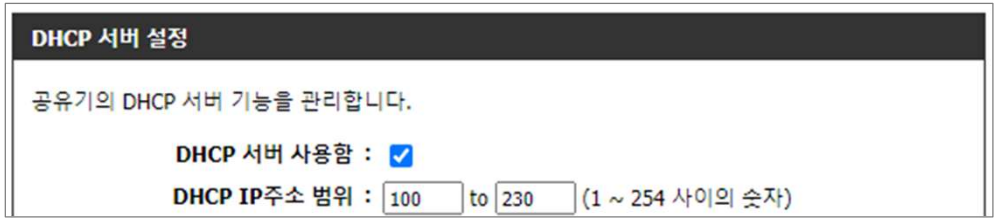
```
remote ❯ kubectl get pods --namespace kube-system

NAME                                       READY   STATUS    RESTARTS        AGE
calico-kube-controllers-5788f6558-g5fl2    1/1     Running   16 (34h ago)    9d
calico-node-67cdg                          1/1     Running   13 (34h ago)    9d
calico-node-722h6                          1/1     Running   13 (34h ago)    9d
calico-node-jtspn                          1/1     Running   14 (34h ago)    9d
coredns-8474476ff8-mk7wr                   1/1     Running   1 (34h ago)     2d
coredns-8474476ff8-z96xn                   1/1     Running   9 (34h ago)     9d
dns-autoscaler-5ffdc7f89d-hhx9z            1/1     Running   9 (34h ago)     9d
kube-apiserver-master                      1/1     Running   11 (34h ago)    9d
kube-controller-manager-master             1/1     Running   11 (34h ago)    9d
...
```

Calico 환경에서는 알려진 문제가 있다지만,

BGP를 사용할 경우에만 해당하는 문제이니 일단 무시하고 진행~

Network addon	Compatible
Calico	Mostly (see known issues)
Canal	Yes
Cilium	Yes
Flannel	Yes
Kube-ovn	Yes
Kube-router	Mostly (see known issues)
Weave Net	Mostly (see known issues)

③ IPv4 addresses



MetalLB에 할당할 IP 대역을 확보해야 한다.

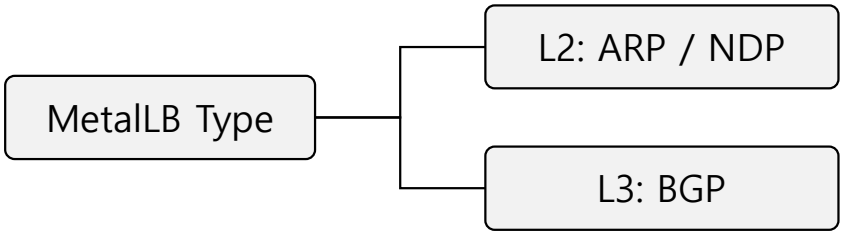
로컬 클라이언트 상태

MAC 주소	IP 주소	장치명
[redacted]	192.168.100.102	[redacted]
[redacted]	192.168.100.107	[redacted]
[redacted]	192.168.100.101	[redacted]
[redacted]	192.168.100.200	
[redacted]	192.168.100.201	
[redacted]	192.168.100.202	

※ 참고 : https://metallb.universe.tf/installation/network-addons/

Requirement - 3/3

④ When using the BGP(Boarder Gateway Protocol)...



MetalLB는 L2/L3 2가지 방식 중 하나를 선택해서 설치할 수 있다.

L3(BGP) 방식을 선택할 경우 추가적으로 고려할 사항들이 있다.

하지만, 여기에서는 일단 L2로 진행할 계획이기에 Skip ...

L2	Data Link	Mac 주소 기반	
L3	Network	IP 주소 기반	
L4	Transport	Port 기반	TCP, UDP
L7	Application	요청(URL) 기반	HTTP, HTTPS

⑤ Traffic on port 7946 (TCP & UDP)

MetalLB가 동작하기 위해서는 Node끼리 port 7946 (TCP/UDP) 통신이 되어야 한다.

보통의 상황에서는 문제가 없겠지만, 방화벽이 있는 환경에서는 해당 port가 열려 있도록 주의해야 한다.

Preparation

- If you're using kube-proxy in IPVS mode, since Kubernetes v1.14.2 you have to enable strict ARP mode.

```
remote ❯ export KUBE_EDITOR=nano

remote ❯ kubectl edit configmap -n kube-system kube-proxy
```

kube-proxy를 IPVS 모드에서 사용한다고 하면 strict ARP를 enable 시켜줘야 한다.

하지만, iptables 모드에서 사용하는 경우에는 Skip ...

```
...
    iptables:
      masqueradeAll: false
      masqueradeBit: 14
      minSyncPeriod: 0s
      syncPeriod: 30s
    ipvs:
      excludeCIDRs: []
      minSyncPeriod: 0s
      scheduler: rr
      strictARP: false
      syncPeriod: 30s
      tcpFinTimeout: 0s
      tcpTimeout: 0s
      udpTimeout: 0s
    kind: KubeProxyConfiguration
    metricsBindAddress: 127.0.0.1:10249
    mode: iptables
    nodePortAddresses: []
...
```

※ 참고 : https://metallb.universe.tf/installation/

Check YAML (skippable)

- MetalLB를 설치하기에 앞서서 설치를 위한 YAML 파일을 다운로드 받은 후 한 번 살펴보기를 권장한다

```
remote ❯ wget https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/namespace.yaml

remote ❯ wget https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/metallb.yaml
```

namespace.yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: metallb-system
  labels:
    app: metallb
```

꼭 해야 하는 과정은 아니다.

그냥 막 설치하기에 앞서서 미리 한 번 살펴보라는 의미.

metallb.yaml

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  labels:
    app: metallb
  name: controller
  namespace: metallb-system
spec:
  allowPrivilegeEscalation: false
  allowedCapabilities: []
  allowedHostPaths: []
  defaultAddCapabilities: []
  defaultAllowPrivilegeEscalation: false
  fsGroup:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  hostIPC: false
  hostNetwork: false
...
```

Installation By Manifest

- 이제는 설치를 진행해 보자.

```
remote ❯ kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/namespace.yaml

namespace/metallb-system created


remote ❯ kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/metallb.yaml

Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

설치 후에 configmap 작업까지 해줘야 끝난다.

Check & Edit Config

- 설치 잘 되었는지 보고, 설정 마무리를 진행해 보자.

```
remote ❯ kubectl get pods -o wide --namespace metallb-system

NAME                          READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
controller-7dcc8764f4-8m4dv   1/1     Running   0          3m48s   10.233.103.31   worker2   <none>           <none>
speaker-nfjl8                 1/1     Running   0          3m48s   192.168.100.201   worker1   <none>           <none>
speaker-qjdtg                 1/1     Running   0          3m48s   192.168.100.202   worker2   <none>           <none>
speaker-zw7x7                 1/1     Running   0          3m48s   192.168.100.200   master    <none>           <none>
```

metallb-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 192.168.100.240-192.168.100.250
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/04-week/metallb

remote ❯ kubectl create -f ./metallb-config.yaml

configmap/config created
```

앞에서도 말했지만, 지금은 layer2 모드로 구성하고자 한다.

마찬가지로 앞에서 확인한 부여 가능한 IP 대역에 따라서 config 내역 잡아주고 생성하면 된다.

※ 참고 : https://metallb.universe.tf/configuration/#layer-2-configuration

///

Hands-On

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes/03-week/ReplicaSet

remote ❯ kubectl create -f ./rs-node-web.yaml

replicaset.apps/rs-labels created

remote ❯ cd ../../04-week/metallb

remote ❯ kubectl create -f ./svc-lb-web.yaml

service/svc-lb created
```

ReplicaSet으로 만들어진 여러 개의 Pod를 준비하고,

이를 LoadBalancer로 묶어주는 과정

→ 독립적인 IP를 할당 받아서 접근 가능

svc-lb-web.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-lb

spec:
  type: LoadBalancer

  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080

  selector:
    app: node-web
```

```
remote ❯ kubectl get services -o wide

NAME         TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)        AGE     SELECTOR
kubernetes   ClusterIP      10.233.0.1     <none>            443/TCP        9d      <none>
svc-lb       LoadBalancer   10.233.55.71   192.168.100.240   80:32380/TCP   2m38s   app=node-web


remote ❯ curl -s http://192.168.100.240

You've hit rs-labels-ksxlj


remote ❯ curl -s http://192.168.100.240

You've hit rs-labels-wdln2
```

///

Kubernetes

Advanced Network

sessionAffinity (Sticky Session) - 1/2

- Environments (ReplicaSet으로 3쌍의 Pod를 생성하고 이를 LoadBalancer를 이용하여 Service하고 있는 환경을 준비하자)

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./03-week/ReplicaSet/rs-node-web.yaml
remote ❯ kubectl create -f ./04-week/metallb/svc-lb-web.yaml
```

- 접속할 때마다 바뀌고 있는 Pod를 확인해 보자

```
remote ❯ curl -s http://192.168.100.240
You've hit rs-labels-ksxlj

remote ❯ curl -s http://192.168.100.240
You've hit rs-labels-wdln2
```

접속할 때마다 Pod가 바뀌는 것이 바람직할 때도 있겠지만,

원하지 않는 상황일 때도 있다.

※ 참고 : https://www.imperva.com/learn/availability/sticky-session-persistence-and-cookies/

sessionAffinity (Sticky Session) - 2/2

- 세션이 유지되는 동안 같은 Pod와 연결이 되도록 하려면 다음과 같이 하면 된다

svc-lb-web-sa.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-lb-sa

spec:
  type: LoadBalancer

  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080

  selector:
    app: node-web

  sessionAffinity: ClientIP
  sessionAffinityConfig:
    clientIP:
      timeoutSeconds: 3600
```

`sessionAffinityConfig`

기본값은 10,800sec 이다. 즉, 지정하지 않아도 되는 설정이다. (sessionAffinity 기본값은 None이다.)

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./04-week/advanced/svc-lb-web-sa.yaml

service/svc-lb-sa created


remote ❯ kubectl get services -o wide

NAME         TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)        AGE   SELECTOR
kubernetes   ClusterIP      10.233.0.1     <none>            443/TCP        9d    <none>
svc-lb-sa    LoadBalancer   10.233.39.34   192.168.100.240   80:31032/TCP   5s    app=node-web


remote ❯ curl -s http://192.168.100.240

You've hit rs-labels-98xnl

                                                처음 붙은 Pod로 계속 접속하게 된다.

remote ❯ curl -s http://192.168.100.240

You've hit rs-labels-98xnl
```

///

externalTrafficPolicy - 1/3

- Environments (ReplicaSet으로 3쌍의 Pod를 생성하고 이를 NodePort를 이용하여 Service하고 있는 환경을 준비하자)

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./03-week/ReplicaSet/rs-node-web.yaml
remote ❯ kubectl create -f ./03-week/Service/svc-node-web.yaml
```

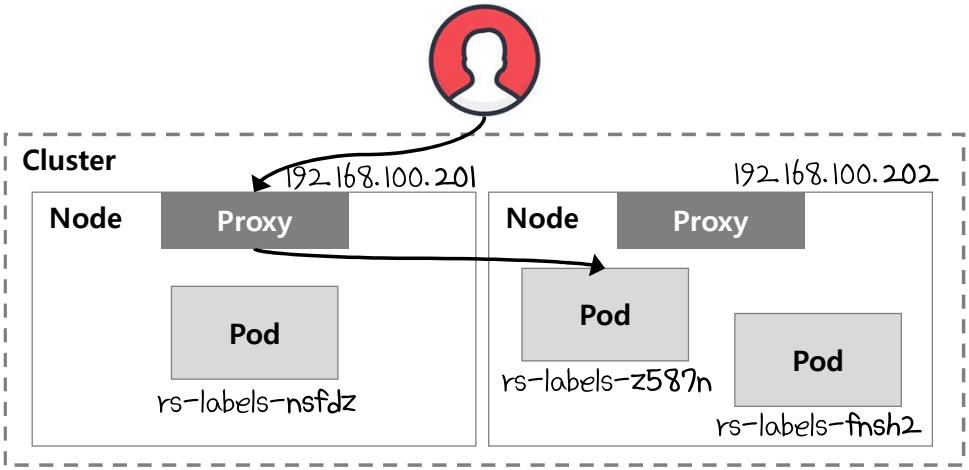
- NodePort 접속 위치와 Pod 위치를 확인해보자

```
remote ❯ kubectl get services -o wide

NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
kubernetes   ClusterIP   10.233.0.1      <none>        443/TCP        10d   <none>
svc-node     NodePort    10.233.28.186   <none>        80:30123/TCP   12s   app=node-web


remote ❯ curl -s http://192.168.100.201:30123

You've hit rs-labels-z587n


remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-labels-fnsh2   1/1     Running   0          64s   10.233.103.35   worker2   <none>           <none>
rs-labels-nsfdz   1/1     Running   0          64s   10.233.110.49   worker1   <none>           <none>
rs-labels-z587n   1/1     Running   0          64s   10.233.103.34   worker2   <none>           <none>
```

worker2 Node에 있는 Pod를 하나 확인하고,

worker1 Node의 IP로 접근하는 상황을 직접 확인해 보자.

externalTrafficPolicy - 2/3



- 불필요한 네트워크 hop 발생
- 클라이언트 IP가 보존되지 않음

svc-lb-web-sa.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-node-et

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30123
  selector:
    app: node-web

  externalTrafficPolicy: Local
```

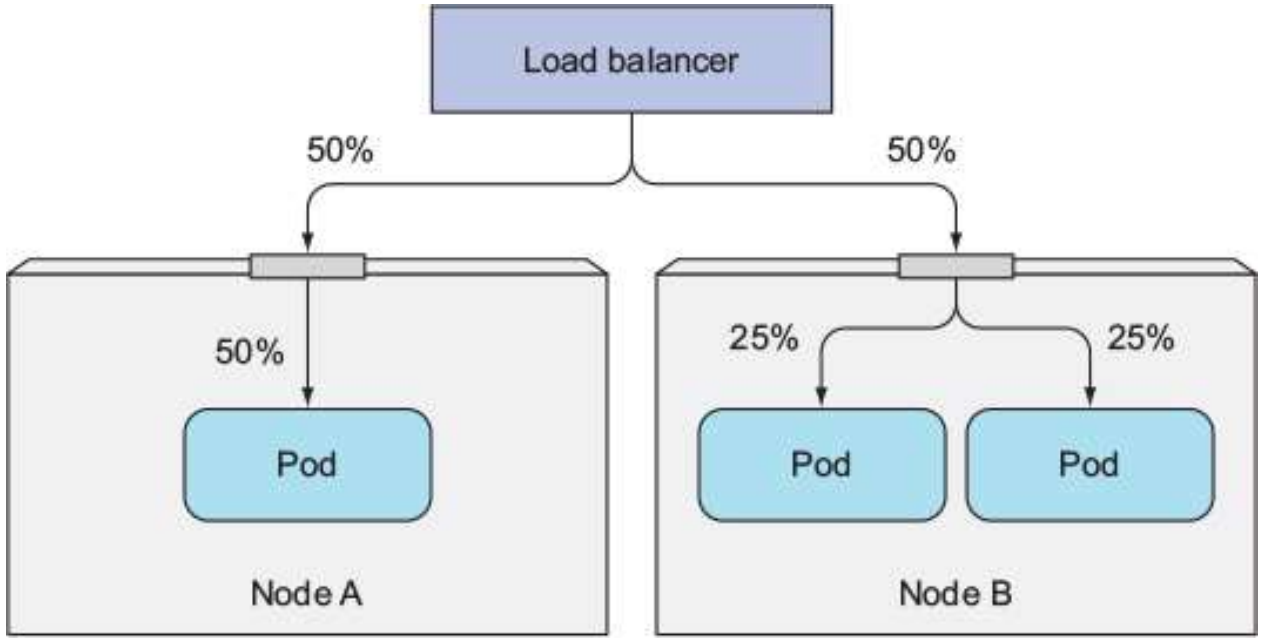
```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl delete services svc-node

service "svc-node" deleted

remote ❯ kubectl create -f ./04-week/advanced/svc-node-web-et.yaml

service/svc-node-et created

remote ❯ curl -s http://192.168.100.201:30123

You've hit rs-labels-nsfdz
```

특정 Node로 들어오게 되면,

해당 Node에 있는 Pod를 선택하도록 하기 위함이다.

externalTrafficPolicy - 3/3

- LoadBalancer를 사용하면서 externalTrafficPolicy를 적용하게 되면, 오히려 균등 배부가 되지 않을 수도 있다.



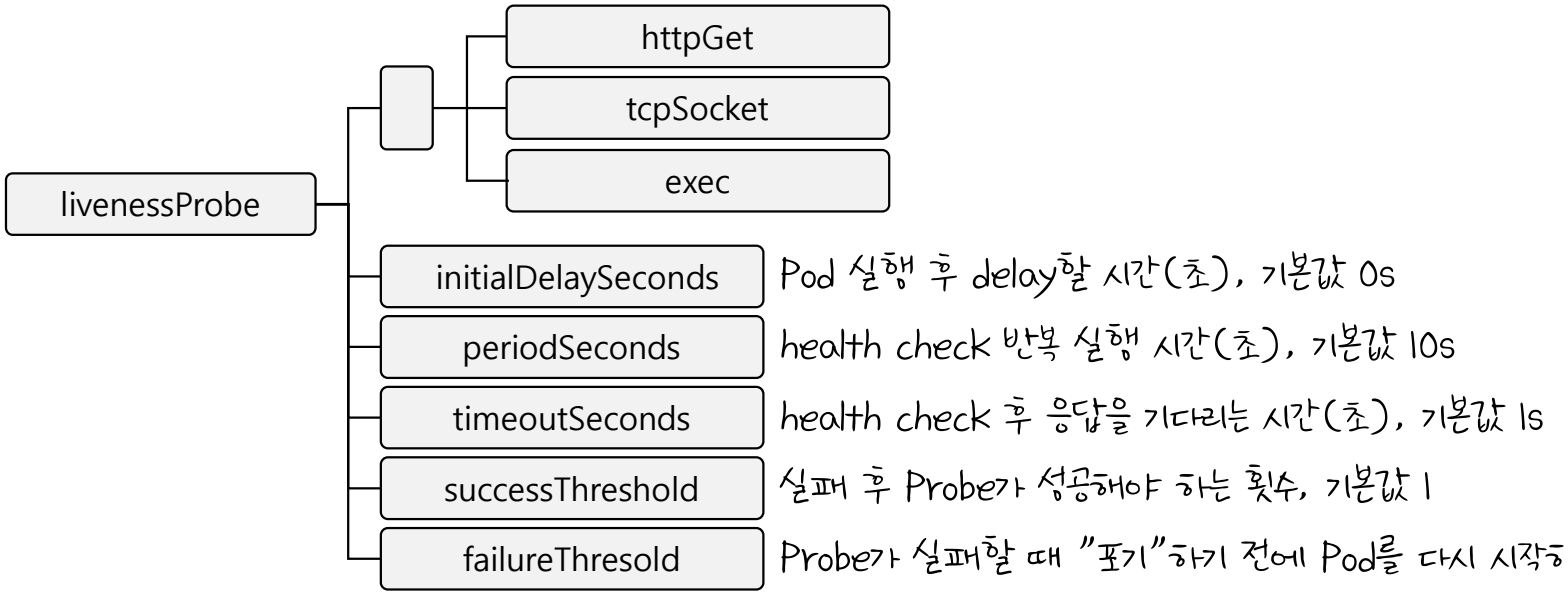
※ "externalTrafficPolicy: Local" 설정에 따른 문제점 : 균등히 배부되지 않을 수 있음

※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-5/230

///

livenessProbe - 1/4

- 외부로부터 Request를 요청 받을 수 있는 상태인지 판별
- **Pod의 spec**에 각 container의 livenessProbe를 지정할 수 있음
  . **HTTP GET Probe**: HTTP GET 요청의 응답 코드를 확인 (2xx, 3xx이면 성공)
  . **TCP Socket Probe**: TCP 연결 성공 여부
  . **Exec Probe**: 명령을 실행하고 exit code가 0이면 성공
- Probe가 실패하면 container를 재실행




```
apiVersion: v1
kind: Pod

metadata:
  name: kubia-liveness

spec:
  containers:
  - image: luksa/kubia-unhealthy
    name: kubia

    livenessProbe:
      httpGet:
        path: /
        port: 8080

      initialDelaySeconds: 15
      periodSeconds: 5
      timeoutSeconds: 1

      successThreshold: 1
      failureThrehsold: 3
```

livenessProbe - 2/4

- `luksa/kubia-unhealthy` 이미지에서 실행되는 node app의 내용을 좀 살펴보고 실습을 해보자.

app.js

```
const http = require('http');
const os = require('os');

console.log("Kubia server starting...");

var requestCount = 0;

var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  requestCount++;
  if (requestCount > 5) {
    response.writeHead(500);
    response.end("I'm not well. Please restart me!");
    return;
  }
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "\n");
};

var www = http.createServer(handler);
www.listen(8080);
```

pod-kubia-liveness.yaml

```
apiVersion: v1
kind: Pod

metadata:
  name: kubia-liveness

spec:
  containers:
  - image: luksa/kubia-unhealthy
    name: kubia

    livenessProbe:
      httpGet:
        path: /
        port: 8080

      initialDelaySeconds: 15
      periodSeconds: 5
      timeoutSeconds: 1
      successThreshold: 1
      failureThreshold: 3
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./04-week/advanced/pod-kubia-liveness.yaml
```

livenessProbe - 3/4

- Success가 될 때까지의 여정을 엿보자

```
remote ❯ kubectl get pods -o wide

NAME             READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
kubia-liveness   1/1     Running   0          70s   10.233.103.36   worker2   <none>           <none>


remote ❯ kubectl describe pod kubia-liveness

...
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  2m1s               default-scheduler  Successfully assigned default/kubia-liveness to worker2
  Normal   Pulled     92s                kubelet            Successfully pulled image "luksa/kubia-unhealthy" in 28.123275632s
  Warning  Unhealthy  41s (x3 over 51s)  kubelet            Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Killing    41s                kubelet            Container kubia failed liveness probe, will be restarted
  Normal   Pulling    11s (x2 over 2m)   kubelet            Pulling image "luksa/kubia-unhealthy"
  Normal   Created    9s (x2 over 91s)   kubelet            Created container kubia
  Normal   Started    9s (x2 over 91s)   kubelet            Started container kubia
  Normal   Pulled     9s                 kubelet            Successfully pulled image "luksa/kubia-unhealthy" in 1.984013259s
```

liveness 확인하고 fail로 판정되게 되면 restart 해버린다.

livenessProbe - 4/4

[효과적인 livenessProbe 구성]

- **특정 URL 경로**(e.g, /health)에 요청하도록 Probe를 구성

  . /health에서 모든 주요 구성 요소가 살아 있는지 확인하도록 로직 구성

- Probe는 application의 **내부만 체크**하고 외부 요인의 영향을 받지 않도록 해야 함

  . database 장애로 인해 front-end의 Probe가 실패해서는 안된다

- Probe를 가볍게 유지

  . **1초 내에 완료**되어야 한다.

- Probe에 재시도 루프를 구현하지 마라 (=중복)

- container 레벨의 livenessProbe는 Worker Node의 Kubelet이 수행

  . Master Node는 관여하지 않음

- Node crash로 인해 중단된 모든 Pod를 복구하는 것은 master의 몫

///

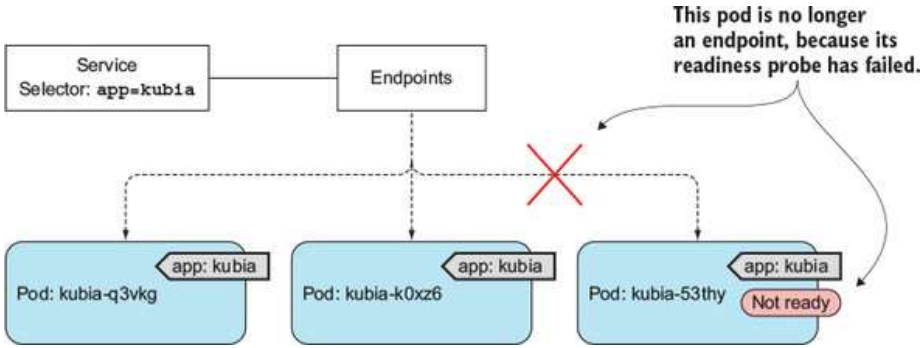
readinessProbe - 1/2

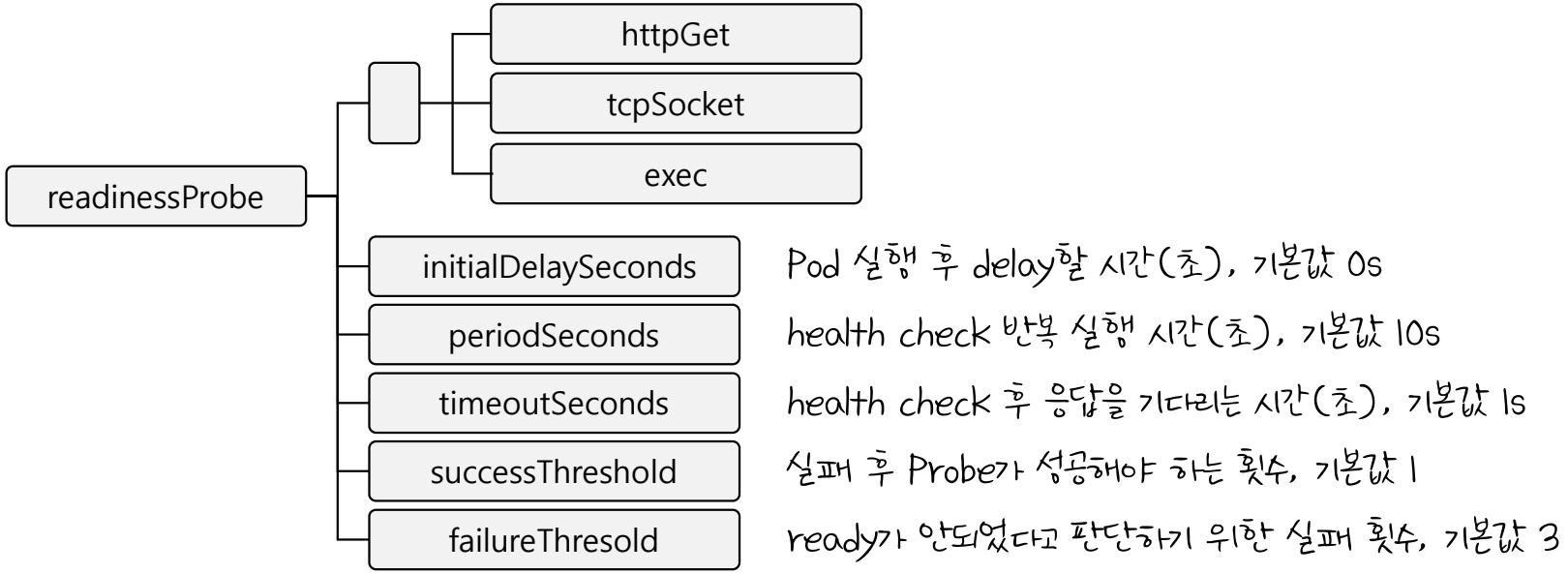
- Probe가 실패하면 Service에서 제거, 성공하면 다시 추가

  . Client 들이 healthy pod 에만 요청을 할 수 있도록 관리

  . 실패한다고 해서 **Container 종료/재시작 하지 않음**

- livenessProbe와 거의 동일한 사용법





※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-5/333

readinessProbe - 2/2

rs-node-web-readiness.yaml

```yaml
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-node-web

spec:
  replicas: 3

  selector:
    matchExpressions:
      - key: app
        operator: In
        values:
          - node-web
```

```yaml
  template:
    metadata:
      labels:
        app: node-web

    spec:
      containers:
      - name: node-web
        image: whatwant/node-web:1.0
        ports:
        - containerPort: 8080

        readinessProbe:
          exec:
            command:
            - ls
            - /var/ready
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f ./04-week/advanced/rs-node-web-readiness.yaml

replicaset.apps/rs-node-web created


remote ❯ kubectl get pods

NAME                READY   STATUS    RESTARTS   AGE
rs-node-web-cn8q4   0/1     Running   0          23s
rs-node-web-gj5b8   0/1     Running   0          23s
rs-node-web-nj8dm   0/1     Running   0          23s
```

```
remote ❯ kubectl exec -it rs-node-web-cn8q4 -- touch /var/ready

remote ❯ kubectl get pods -o wide

NAME                READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-node-web-cn8q4   1/1     Running   0          4m53s   10.233.110.56   worker1   <none>           <none>
rs-node-web-gj5b8   0/1     Running   0          4m53s   10.233.103.38   worker2   <none>           <none>
rs-node-web-nj8dm   0/1     Running   0          4m53s   10.233.103.37   worker2   <none>           <none>
```

///

Tips

k9s

- Kubernetes CLI To Manage Your Clusters In Style!

- LinuxBrew(HomeBrew) Install

```
remote ❯ sudo apt-get install build-essential curl file git

remote ❯ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/install.sh)"
```

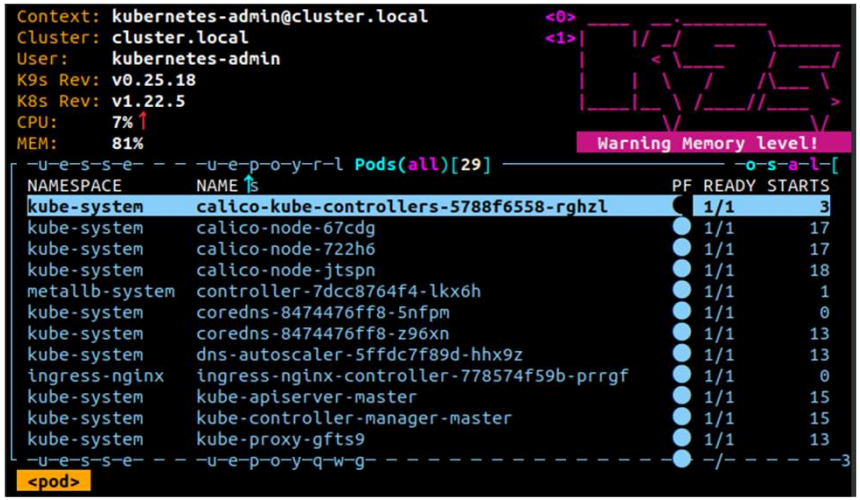
```
remote ❯ nano ~/.zshrc

...

remote ❯ source ~/.zshrc
```

~/.zshrc

```
export PATH="/home/linuxbrew/.linuxbrew/bin:$PATH"
export MANPATH="/home/linuxbrew/.linuxbrew/share/man:$MANPATH"
export INFOPATH="/home/linuxbrew/.linuxbrew/share/info:$INFOPATH"
```

- k9s Install

```
remote ❯ brew install derailed/k9s/k9s

remote ❯ k9s
```



※ 참고 : https://k9scli.io/

///

https://kahoot.it/

[Score]

이민준 (6)
박남준 (3)
이혜정 (3)
김상호 (2)
정현찬 (1)
김정은 (1)
김남형 (1)