5th Week

다섯 번째 뵙겠습니다 ?!

▷ 잠시만 기다렸다가 30분 되면 시작하겠습니다~^^

▷ 지난 주에는 조금 일찍 끝났으니, 이번 주에는 … ?!

- 신나게 달려봅시다~~~!!!

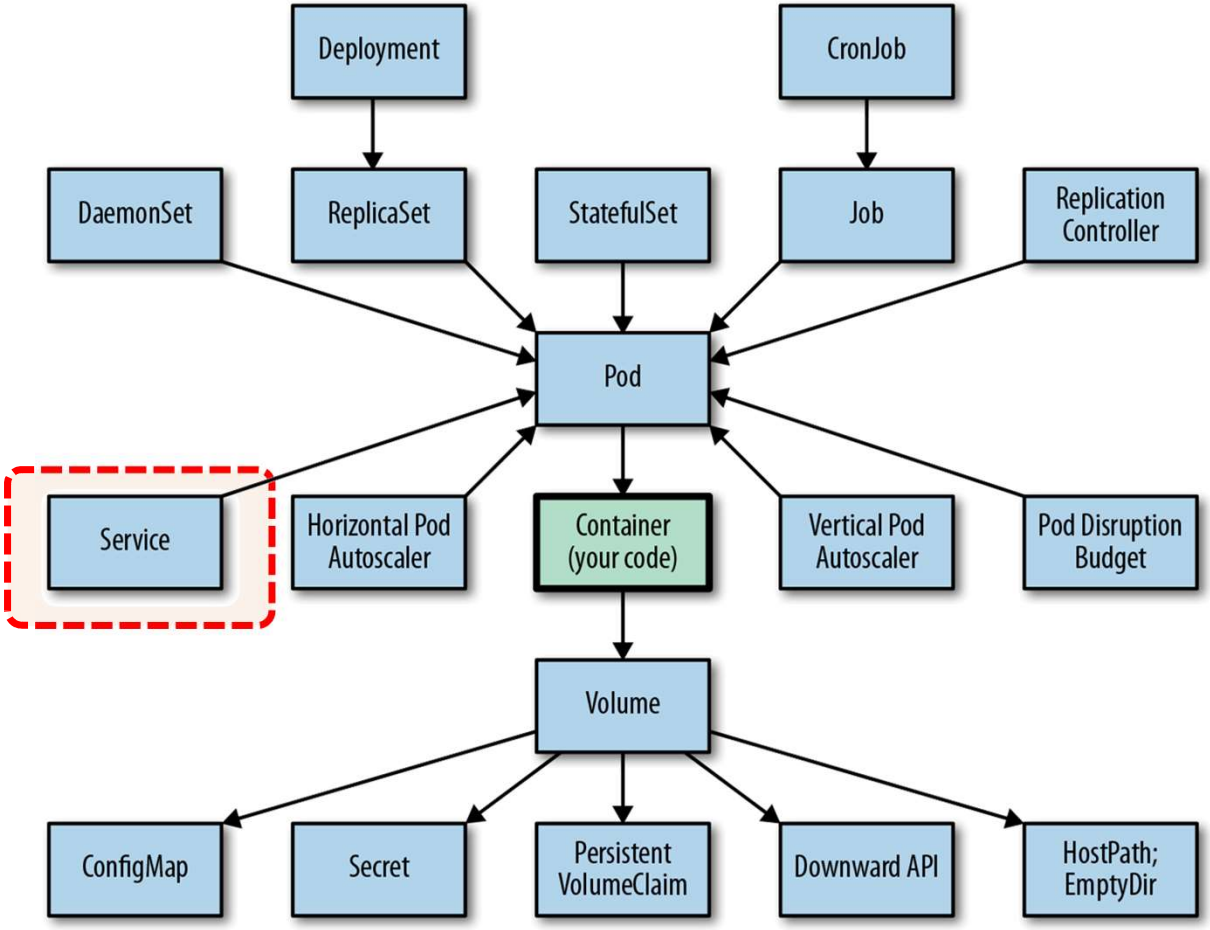
▷ Camera는 가급적 켜 주시면 대단히 감사하겠습니다 !!!

- 너무 부끄러우면 Snap Camera를 사용하시는 것 까지는~ ^^

▷ 오늘 수업 자료는 아래 링크에서 다운로드 받으실 수 있어요.

- https://github.com/whatwant-school/kubernetes

Service



※ 참고 : https://www.oreilly.com/library/view/kubernetes-patterns/9781492050278/ch01.html

///

지난 수업
복습

https://kahoot.it/

///

Service

Ingress / LoadBalancer

///

Flip Learning

(Service - Ingress / LoadBalancer)

권재혁 님

///

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

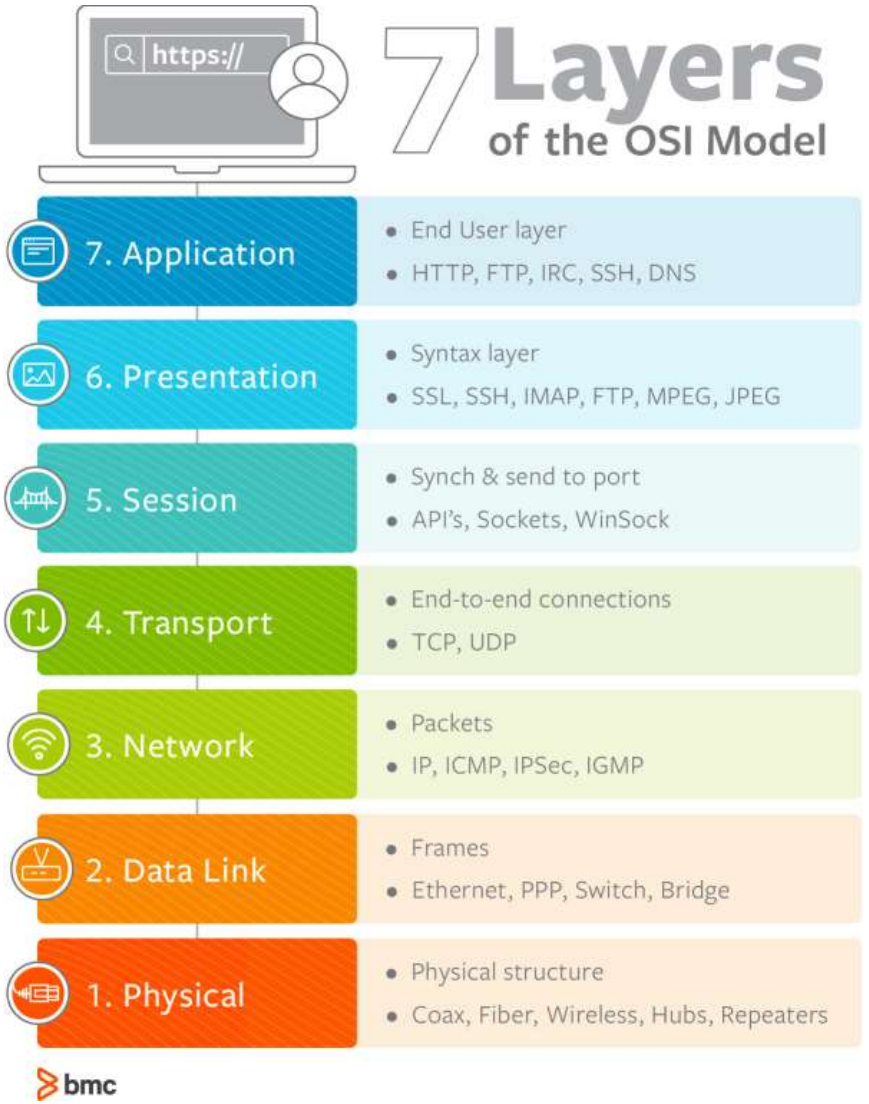
  . 오픈소스 (Ingress NGINX Controller) : https://github.com/kubernetes/ingress-nginx

  . 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/

OSI 7 Layers



※ 참고 : https://www.bmc.com/blogs/osi-model-7-layers/

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

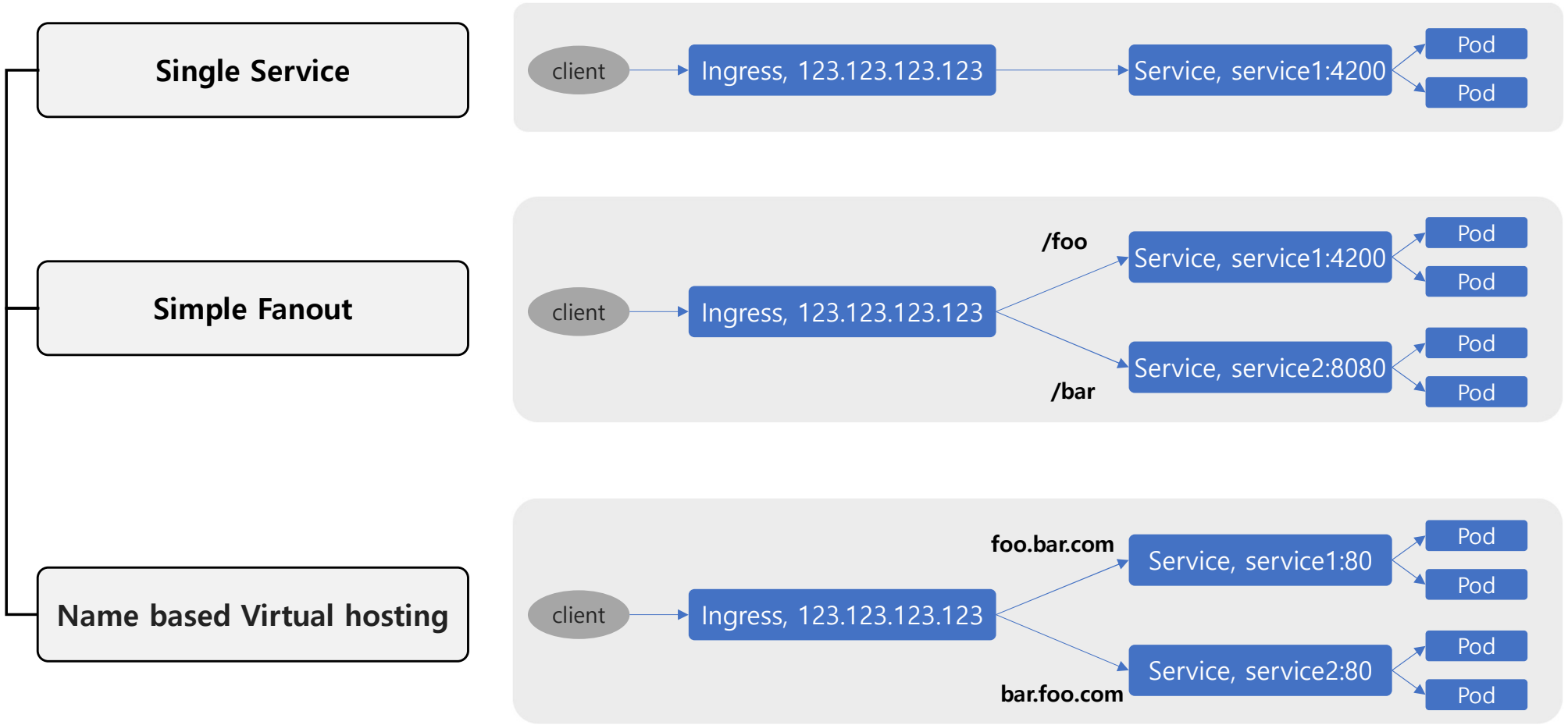
```
remote ❯ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.3.0/deploy/static/provider/baremetal/deploy.yaml

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

Ingress Controller Check

```
remote ❯ kubectl get namespaces

NAME              STATUS   AGE
default           Active   23d
ingress-nginx     Active   2m19s
...


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   11 (5m42s ago)    23d   192.168.100.201  worker1   <none>           <none>
nginx-proxy-worker2                 1/1     Running   9 (5m34s ago)     23d   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-6j9cv 0/1     Completed   0          3m49s   10.233.103.75  worker2   <none>           <none>
ingress-nginx-admission-patch-qmwvf  0/1     Completed   1          3m49s   10.233.110.24  worker1   <none>           <none>
ingress-nginx-controller-77cb5dbf4d-k5qmr   1/1     Running     0          3m49s   10.233.110.25  worker1   <none>           <none>


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

NAME                    SECRETS   AGE
default                 1         4m28s
ingress-nginx           1         4m28s
ingress-nginx-admission 1         4m28s
```

///

Types of Ingress



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

///

Ready: container images

별개의 사이트 구분을 할 수 있도록 웹서비스를 2종 준비했다.

기존 `node-web:1.0`과 구분이 될 정도만 살짝 변경한 후 `node-web:2.0` 버전으로 DockerHub에 push 했다.

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"]
```

DockerHub에 이미 있는 만들어 놓은

이미지를 사용해도 좋다.

```
remote > git clone https://github.com/whatwant-school/kubernetes.git
remote > cd kubernetes/05-Ingress-LoadBalancer/hands-on


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
```

Ready: ReplicaSet YAML

웹사이트 2종을 각각 띄울 `ReplicaSet`을 준비하자.

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
```

Ready: Create ReplicaSet

앞에서 작성해 놓은 ReplicaSet을 이용해서 Pods를 생성하자.

```
remote ❯ git clone https://github.com/whatwant-school/kubernetes.git
remote ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

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-ghbqs  1/1     Running   0          2m25s   10.233.110.29   worker1   <none>           <none>
rs-web-v1-m6gtz  1/1     Running   0          2m25s   10.233.103.79   worker2   <none>           <none>
rs-web-v1-rxpxz  1/1     Running   0          2m25s   10.233.103.80   worker2   <none>           <none>
rs-web-v2-cmvl2  1/1     Running   0          2m22s   10.233.103.82   worker2   <none>           <none>
rs-web-v2-k6src  1/1     Running   0          2m22s   10.233.110.30   worker1   <none>           <none>
rs-web-v2-qsh68  1/1     Running   0          2m22s   10.233.103.81   worker2   <none>           <none>
```

Ready: Create Service

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

svc-node-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-node-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 ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

remote ❯ kubectl create -f svc-node-web-nodeport-v1.yaml
remote ❯ kubectl create -f svc-node-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        24d   <none>
svc-node-web-v1  NodePort    10.233.24.241   <none>        80:30001/TCP   11s   app=node-web-v1
svc-node-web-v2  NodePort    10.233.26.196   <none>        80:30002/TCP   7s    app=node-web-v2
```

///

Status

```
remote ❯ kubectl get replicasets -o wide

NAME        DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES                   SELECTOR
rs-web-v1   3         3         3       8m55s   node-web     whatwant/node-web:1.0    app=node-web-v1
rs-web-v2   3         3         3       8m52s   node-web     whatwant/node-web:2.0    app=node-web-v2

remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-v1-ghbqs   1/1     Running   0          9m24s   10.233.110.29   worker1   <none>           <none>
rs-web-v1-m6gtz   1/1     Running   0          9m24s   10.233.103.79   worker2   <none>           <none>
rs-web-v1-rxpxz   1/1     Running   0          9m24s   10.233.103.80   worker2   <none>           <none>
rs-web-v2-cmvl2   1/1     Running   0          9m21s   10.233.103.82   worker2   <none>           <none>
rs-web-v2-k6src   1/1     Running   0          9m21s   10.233.110.30   worker1   <none>           <none>
rs-web-v2-qsh68   1/1     Running   0          9m21s   10.233.103.81   worker2   <none>           <none>

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        24d   <none>
svc-node-web-v1   NodePort    10.233.24.241   <none>        80:30001/TCP   11s   app=node-web-v1
svc-node-web-v2   NodePort    10.233.26.196   <none>        80:30002/TCP   7s    app=node-web-v2

remote ❯ curl -s http://192.168.100.200:30001

You've hit rs-web-v1-m6gtz

remote ❯ curl -s http://192.168.100.200:30002

You've hit rs-web-v2-k6src (Ver2.0)
```

Create Ingress (Simple Fanout)

ingress-node-web.yaml

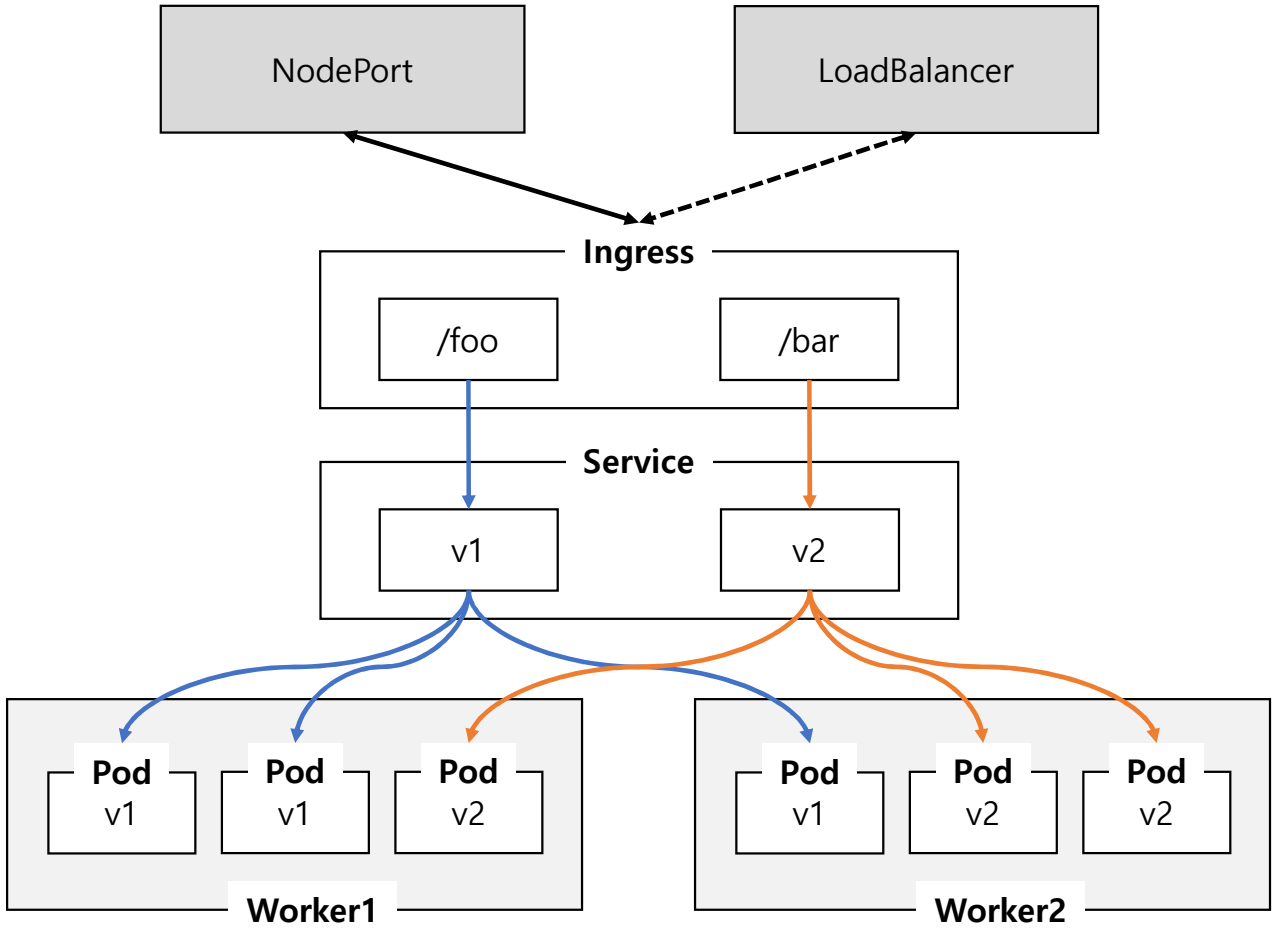
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress

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
```

```
remote ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

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

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


remote ❯ kubectl get ingresses -o wide

NAME               CLASS    HOSTS   ADDRESS           PORTS   AGE
ingress-node-web   <none>   *       192.168.100.201   80      11s


remote ❯ kubectl get services --namespace ingress-nginx

NAME                                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.233.63.114  <none>        80:30977/TCP,443:30168/TCP   13h
ingress-nginx-controller-admission   ClusterIP   10.233.59.2    <none>        443/TCP                      13h


remote ❯ curl -s http://192.168.100.201:30977/foo

You've hit rs-web-v1-m6gtz


remote ❯ curl -s http://192.168.100.201:30977/bar

You've hit rs-web-v2-cmvl2 (Ver2.0)
```

Describe Ingress

```
remote ❯ kubectl describe ingress ingress-node-web

Name:             ingress-node-web
Labels:           <none>
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 (10.233.103.79:8080,10.233.103.80:8080,10.233.110.29:8080)
              /bar   svc-node-web-v2:80 (10.233.103.81:8080,10.233.103.82:8080,10.233.110.30:8080)
Annotations:  kubernetes.io/ingress.class: nginx
              nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason  Age                 From                      Message
  ----    ------  ----                ----                      -------
  Normal  Sync    12m (x2 over 13m)  nginx-ingress-controller  Scheduled for sync
```

Design Thinking



///

externalTrafficPolicy

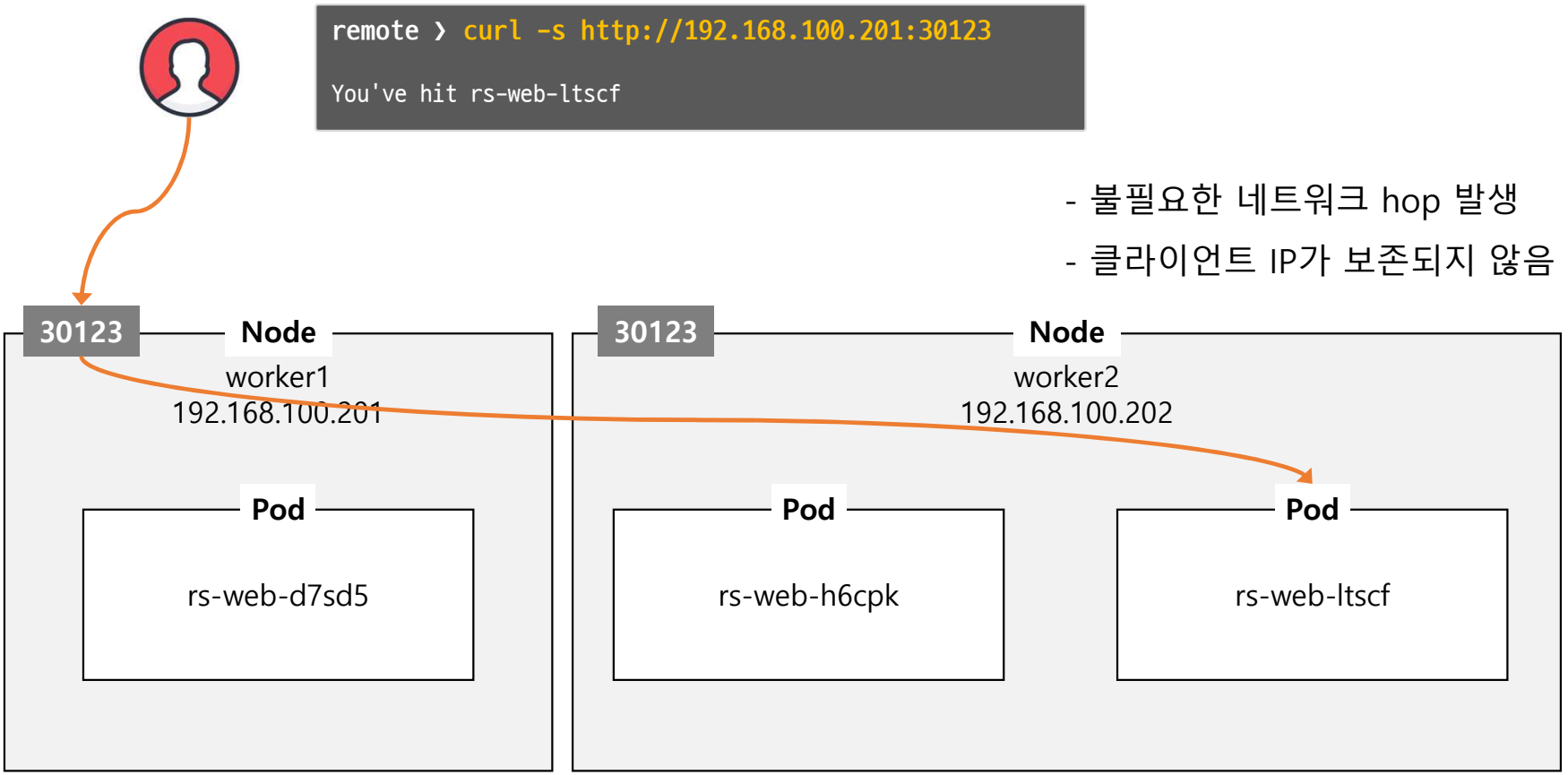
NodePort
공부하고 이어서 바로 했으면
더 좋았을 것 같다.

Ready

- 앞에서 NodePort 실습했던 내용을 활용해서 살펴보도록 하겠다.

```
remote ❯ cd kubernetes/03-RS-DS-JOB-CJ/hands-on
remote ❯ kubectl create -f rs-web.yaml

remote ❯ cd kubernetes/04-ClusterIP-NodePort-ExternalName/hands-on
remote ❯ kubectl create -f svc-web-node.yaml

remote ❯ kubectl get pods -o wide

NAME          READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-d7sd5  1/1     Running   0          2m33s   10.233.110.31   worker1   <none>           <none>
rs-web-h6cpk  1/1     Running   0          2m33s   10.233.103.85   worker2   <none>           <none>
rs-web-ltscf  1/1     Running   0          2m33s   10.233.103.84   worker2   <none>           <none>


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        24d   <none>
svc-web-node  NodePort    10.233.7.149   <none>        80:30123/TCP   77s   app=node-web


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

You've hit rs-web-ltscf
```

Problem


remote ❯ curl -s http://192.168.100.201:30123
You've hit rs-web-ltscf
- 불필요한 네트워크 hop 발생
- 클라이언트 IP가 보존되지 않음
30123
Node
worker1
192.168.100.201
Pod
rs-web-d7sd5
30123
Node
worker2
192.168.100.202
Pod
rs-web-h6cpk
Pod
rs-web-ltscf

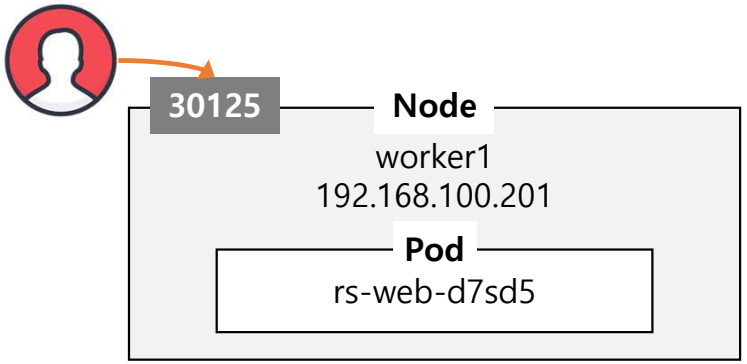
externalTrafficPolicy

svc-web-node-et.yaml

```yaml
apiVersion: v1
kind: Service

metadata:
  name: svc-web-node-et

spec:
  type: NodePort

  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30125

  selector:
    app: node-web

  externalTrafficPolicy: Local
```

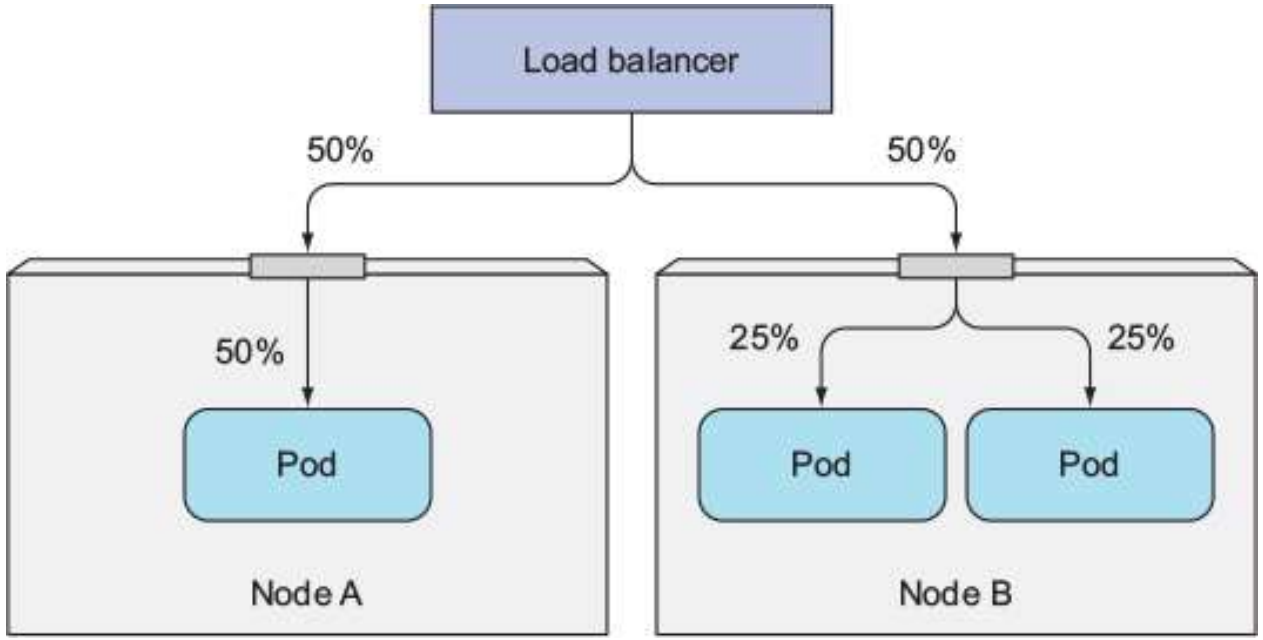
```
remote ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

remote ❯ kubectl create -f svc-web-node-et.yaml

service/svc-web-node-et 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        24d   <none>
svc-web-node      NodePort    10.233.7.149    <none>        80:30123/TCP   89m   app=node-web
svc-web-node-et   NodePort    10.233.6.235    <none>        80:30125/TCP   27s   app=node-web


remote ❯ curl -s http://192.168.100.201:30125

You've hit rs-web-d7sd5
```

- externalTrafficPolicy 기본값은 Cluster



But,

- 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

///

Break

돌아오셨으면 채팅창에
복귀!
타이핑하기!

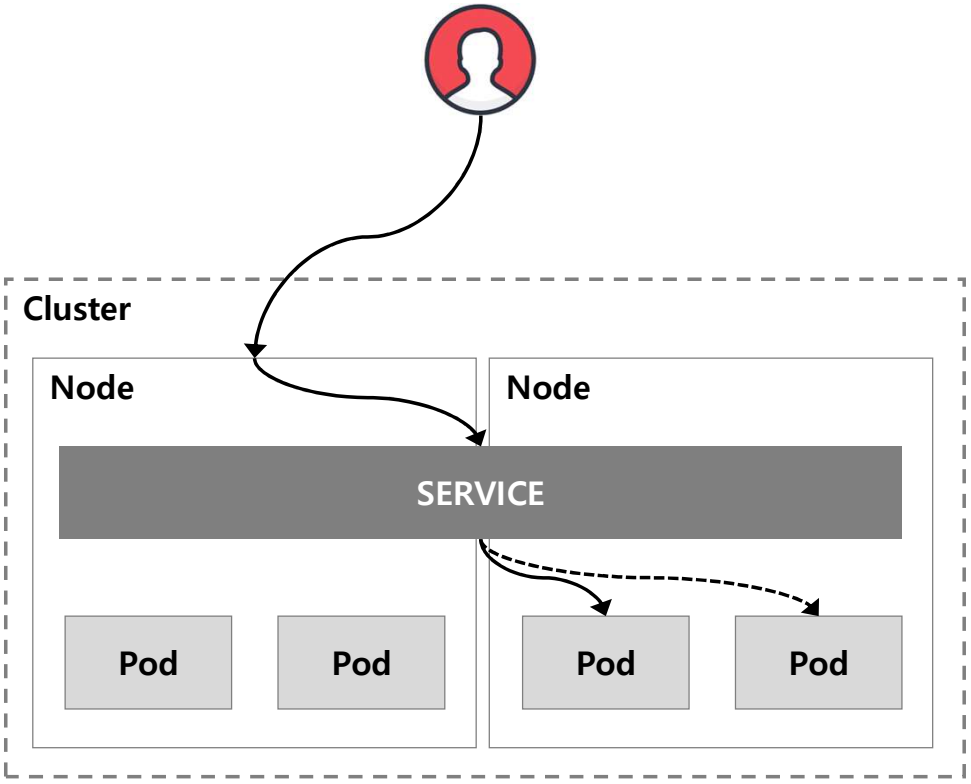
///

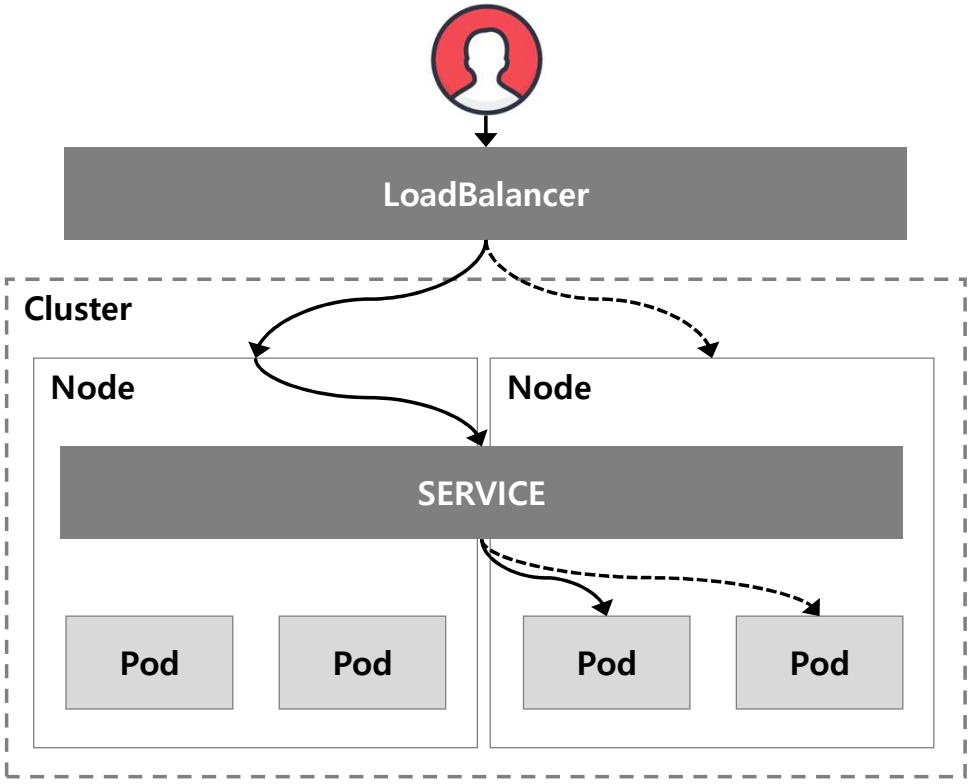
Kubernetes

Network - LoadBalancer

Why LoadBalancer ?

NodePort vs LoadBalancer





///

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   24d   v1.23.7   192.168.100.200   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
worker1   Ready    <none>                 24d   v1.23.7   192.168.100.201   <none>        Ubuntu 20.04.4 LTS   5.4.0-124-generic   containerd://1.6.4
worker2   Ready    <none>                 24d   v1.23.7   192.168.100.202   <none>        Ubuntu 20.04.4 LTS   5.4.0-124-generic   containerd://1.6.4
```

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

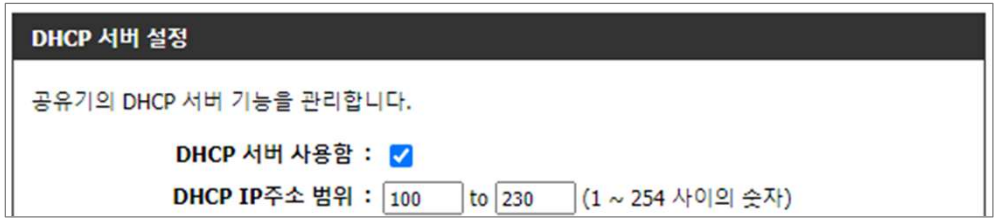
```
remote ❯ kubectl get pods --namespace kube-system

NAME                                      READY   STATUS    RESTARTS          AGE
calico-kube-controllers-6dd874f784-rjxtz  1/1     Running   14 (4h11m ago)    24d
calico-node-599ck                         1/1     Running   10 (4h12m ago)    24d
calico-node-qlhvf                         1/1     Running   10 (4h11m ago)    24d
calico-node-tpwvg                         1/1     Running   20 (4h11m ago)    24d
coredns-76b4fb4578-kc7vm                  1/1     Running   10 (4h12m ago)    24d
coredns-76b4fb4578-zbtvb                  1/1     Running   9 (4h11m ago)     24d
dns-autoscaler-7979fb6659-p5fpm           1/1     Running   10 (4h12m ago)    24d
kube-apiserver-master                     1/1     Running   11 (4h12m ago)    24d
kube-controller-manager-master            1/1     Running   11 (4h12m ago)    24d
...
```

Calico 환경에서는 알려진 문제가 있다지만,

BGP를 사용할 경우에만 해당하는 문제이니 일단 무시하고 진행~

Network addon	Compatible
Antrea	Yes (Tested on version 1.4 and 1.5)
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 수 있는 IP들을 확보해야 한다.

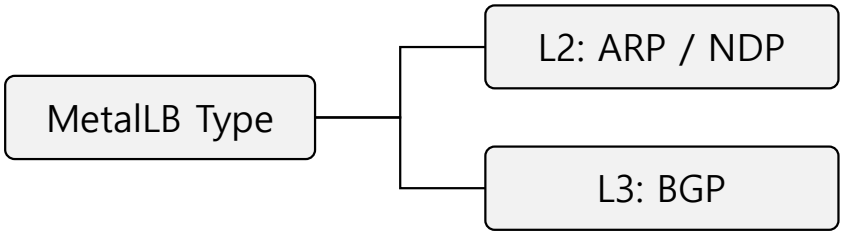
로컬 클라이언트 상태

MAC 주소	IP 주소	장치명
[illegible]	192.168.100.102	[illegible]
[illegible]	192.168.100.107	[illegible]
[illegible]	192.168.100.101	[illegible]
[illegible]	192.168.100.200	
[illegible]	192.168.100.201	
[illegible]	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"

- 실습환경 구축을 가이드한대로 진행했다고 하면, kube-proxy 기본 설정이 `iptables` 일 것이다.

- 만약, `IPVS` 설정으로 되어 있다면 strictARP 설정을 enable로 설정해줘야 한다.

- kube-proxy 설정 내역을 확인해 보려면 다음과 같이 해보면 된다.

```
remote ❯ kubectl describe configmap kube-proxy --namespace kube-system

...
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
...
```

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

///

Install MetalLB

```
remote ❯ kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.13.4/config/manifests/metallb-native.yaml

namespace/metallb-system created
customresourcedefinition.apiextensions.k8s.io/addresspools.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bfdprofiles.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bgpadvertisements.metallb.io created
customresourcedefinition.apiextensions.k8s.io/bgppeers.metallb.io created
customresourcedefinition.apiextensions.k8s.io/communities.metallb.io created
customresourcedefinition.apiextensions.k8s.io/ipaddresspools.metallb.io created
customresourcedefinition.apiextensions.k8s.io/l2advertisements.metallb.io created
serviceaccount/controller created
serviceaccount/speaker created
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
role.rbac.authorization.k8s.io/controller created
role.rbac.authorization.k8s.io/pod-lister created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
rolebinding.rbac.authorization.k8s.io/controller created
rolebinding.rbac.authorization.k8s.io/pod-lister created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker created
secret/webhook-server-cert created
service/webhook-service created
deployment.apps/controller created
daemonset.apps/speaker created
validatingwebhookconfiguration.admissionregistration.k8s.io/metallb-webhook-configuration created
```

※ 참고 : https://metallb.universe.tf/installation/#installation-by-manifest

Check

```
remote ❯ kubectl get namespaces

NAME              STATUS   AGE
default           Active   24d
ingress-nginx     Active   16h
kube-node-lease   Active   24d
kube-public       Active   24d
kube-system       Active   24d
metallb-system    Active   3m12s


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

NAME                          READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
controller-64cc46b9f9-vxqb2   1/1     Running   0          4m40s   10.233.103.83   worker2   <none>           <none>
speaker-dwprs                 1/1     Running   0          4m40s   192.168.100.201   worker1   <none>           <none>
speaker-n8prv                 1/1     Running   0          4m40s   192.168.100.202   worker2   <none>           <none>
speaker-prhkx                 1/1     Running   0          4m40s   192.168.100.200   master    <none>           <none>
```

Configuration

- L2 / BGP 유형에 따라 설정이 다르지만, 여기에서는 L2 기준으로 진행하겠다.

metallb-ip.yaml

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool

metadata:
  name: lb-pool
  namespace: metallb-system

spec:
  addresses:
  - 192.168.100.240-192.168.100.250
```

metallb-l2.yaml

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement

metadata:
  name: lb
  namespace: metallb-system

spec:
  ipAddressPools:
  - lb-pool
```

```
remote ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

remote ❯ kubectl create -f metallb-ip.yaml

ipaddresspool.metallb.io/lb-pool created


remote ❯ kubectl create -f metallb-l2.yaml

l2advertisement.metallb.io/lb created
```

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

///

Ready

- 앞에서 진행했던 ReplicaSet (Pods) 및 Service를 그대로 사용해서 LoadBalancer를 실습해보도록 하겠다.

```
remote ❯ kubectl get replicasets -o wide

NAME        DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES                   SELECTOR
rs-web-v1   3         3         3       8m55s   node-web     whatwant/node-web:1.0    app=node-web-v1
rs-web-v2   3         3         3       8m52s   node-web     whatwant/node-web:2.0    app=node-web-v2

remote ❯ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-v1-ghbqs   1/1     Running   0          9m24s   10.233.110.29   worker1   <none>           <none>
rs-web-v1-m6gtz   1/1     Running   0          9m24s   10.233.103.79   worker2   <none>           <none>
rs-web-v1-rxpxz   1/1     Running   0          9m24s   10.233.103.80   worker2   <none>           <none>
rs-web-v2-cmvl2   1/1     Running   0          9m21s   10.233.103.82   worker2   <none>           <none>
rs-web-v2-k6src   1/1     Running   0          9m21s   10.233.110.30   worker1   <none>           <none>
rs-web-v2-qsh68   1/1     Running   0          9m21s   10.233.103.81   worker2   <none>           <none>

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        24d   <none>
svc-node-web-v1   NodePort    10.233.24.241   <none>        80:30001/TCP   11s   app=node-web-v1
svc-node-web-v2   NodePort    10.233.26.196   <none>        80:30002/TCP   7s    app=node-web-v2
```

Create LoadBalancer

- LoadBalancer 유형의 Service를 생성하는 것은 type 으로 명시만 해주면 된다.

svc-node-web-lb.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-v1
```

```
remote ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

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

service/svc-lb 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        24d     <none>
svc-lb           LoadBalancer   10.233.46.241   192.168.100.240   80:31336/TCP   4s      app=node-web-v1
svc-node-web-v1  NodePort       10.233.24.241   <none>            80:30001/TCP   7h54m   app=node-web-v1
svc-node-web-v2  NodePort       10.233.26.196   <none>            80:30002/TCP   7h54m   app=node-web-v2


remote ❯ curl -s http://192.168.100.240

You've hit rs-web-v1-m6gtz
```

Describe LoadBalancer

```
remote ❯ kubectl describe service svc-lb

Name:                     svc-lb
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=node-web-v1
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.233.46.241
IPs:                      10.233.46.241
LoadBalancer Ingress:     192.168.100.240
Port:                     http  80/TCP
TargetPort:               8080/TCP
NodePort:                 http  31336/TCP
Endpoints:                10.233.103.79:8080,10.233.103.80:8080,10.233.110.29:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason        Age                   From               Message
  ----    ------        ----                  ----               -------
  Normal  IPAllocated   3m19s                 metallb-controller  Assigned IP ["192.168.100.240"]
  Normal  nodeAssigned  2m2s (x25 over 3m19s)  metallb-speaker     announcing from node "master" with protocol "layer2"
```

///

sessionAffinity
(Sticky Session)

Problem

- 앞에서 실습한 LoadBalancer 를 조금 더 살펴보자.

```
remote ❯ curl -s http://192.168.100.240
You've hit rs-web-v1-ghbqs

remote ❯ curl -s http://192.168.100.240
You've hit rs-web-v1-rxpxz

remote ❯ curl -s http://192.168.100.240
You've hit rs-web-v1-m6gtz

remote ❯ curl -s http://192.168.100.240
You've hit rs-web-v1-rxpxz
```

- 접속할 때 마다 매번 바뀌는 Endpoints 를 볼 수 있다 → 만약 게시판과 같은 웹사이트라면 문제가 있을 수 있다!

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

sessionAffinity (Sticky Session)

- 세션이 유지되는 동안 접속되는 Pod가 동일하도록 하기 위한 설정을 해보자.

svc-node-web-lb-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-v1

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

```
remote ❯ cd kubernetes/05-Ingress-LoadBalancer/hands-on

remote ❯ kubectl create -f svc-node-web-lb-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          24d   <none>
svc-lb           LoadBalancer   10.233.46.241   192.168.100.240   80:31336/TCP     49m   app=node-web-v1
svc-lb-sa        LoadBalancer   10.233.20.177   192.168.100.241   80:30566/TCP     25s   app=node-web-v1


remote ❯ curl -s http://192.168.100.241
You've hit rs-web-v1-ghbqs

remote ❯ curl -s http://192.168.100.241
You've hit rs-web-v1-ghbqs

remote ❯ curl -s http://192.168.100.241
You've hit rs-web-v1-ghbqs
```

- `sessionAffinityConfig` 기본값은 10,800sec 이다. 즉, 지정하지 않아도 되는 설정이지만 공부를 위해서 추가해보았다.

///

https://kahoot.it/

///

자습 (복습)

- ▷ Ingress
- ▷ LoadBalancer