

Managing Kubernetes

2021-05-01
written by whatwant

Agenda

Chapter1. Kubernetes Overview

1주차: **Docker and Kubernetes**

Chapter2. Kubernetes Core

2주차: **Environment & POD**

3주차: **Replication and other controllers**

4주차: **Services & Volumes**

5주차: **ConfigMaps and Secrets & Kubernetes REST API**

6주차: **Deployment & StatefulSet**

7주차: **Summary & HandsOn**

Chapter3. Kubernetes Managing

8주차: **Authentication and User Management & Authorization & Admission Control**

9주차: **Networking**

10주차: **Monitoring**

11주차: **Disaster Recovery**

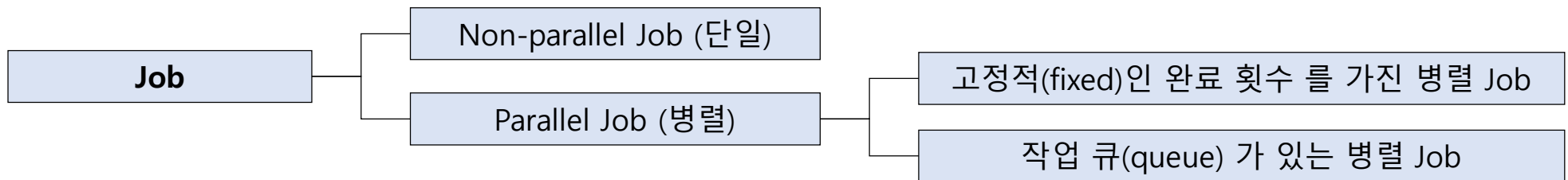
※ 참고 : <https://home.modulabs.co.kr/product/managing-kubernetes/>

4 week
Services & Volumes

Supplementary Lessons

Job - 1/2

- 고정적(fixed)인 완료 횟수 를 가진 병렬 Job
 - .spec.completions 에 0이 아닌 양수 값을 지정
 - . Job은 전체 작업을 나타내며 1에서 .spec.completions 까지의 범위의 각 값에 대해 한 개씩 성공한 Pod가 있으면 완료
- 작업 큐(queue) 가 있는 병렬 Job
 - .spec.completions 를 지정하지 않고, .spec.parallelism 에 양수 값 설정
 - . Job의 모든 Pod가 성공적으로 종료되면, 새로운 Pod는 생성되지 않음
 - . 하나 이상의 Pod가 성공적으로 종료되고, 모든 Pod가 종료되면 Job은 성공적으로 완료



※ 참고 : <https://kubernetes.io/ko/docs/concepts/workloads/controllers/job/>

Job - 2/2

- **completions** : 몇 번의 Completed가 나올 때까지
- **parallelism** : 한 번에 몇 개씩
- **backoffLimit** : 최대 실행 횟수

job-parallel.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: throw-dice-job
spec:
  completions: 3
  parallelism: 3
  backoffLimit: 25
  template:
    spec:
      containers:
        - name: math-add
          image: kodekloud/throw-dice
      restartPolicy: Never
```

```
> kubectl create -f job-parallel.yaml
```

```
job.batch/throw-dice-job created
```

```
> kubectl get jobs
```

NAME	COMPLETIONS	DURATION	AGE
throw-dice-job	3/3	52s	57s

```
> kubectl get pods
```

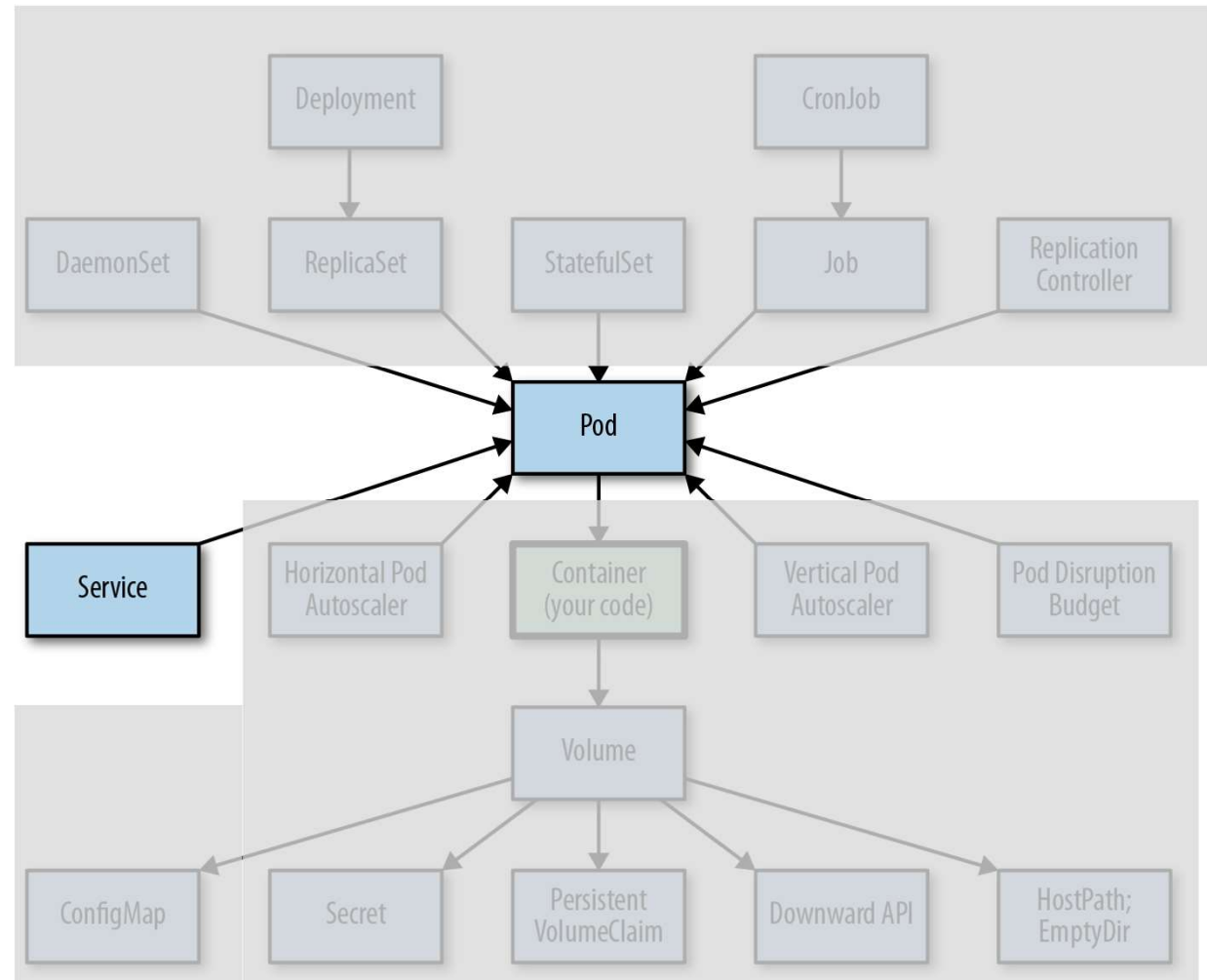
NAME	READY	STATUS	RESTARTS	AGE
throw-dice-job-5mclr	0/1	Error	0	60s
throw-dice-job-9pfp8	0/1	Error	0	60s
throw-dice-job-p56gc	0/1	Error	0	60s
throw-dice-job-ll8c6	0/1	Error	0	56s
throw-dice-job-7bjsk	0/1	Error	0	46s
throw-dice-job-9rq5g	0/1	Error	0	46s
throw-dice-job-m7qtf	0/1	Error	0	46s
throw-dice-job-mg29n	0/1	Completed	0	26s
throw-dice-job-nb8xv	0/1	Error	0	26s
throw-dice-job-q5cgg	0/1	Error	0	26s
throw-dice-job-hl7lz	0/1	Error	0	22s
throw-dice-job-85nl5	0/1	Completed	0	12s
throw-dice-job-zppnx	0/1	Completed	0	12s

※ 참고 : <https://github.com/chandramgc/kubernetes-snippets/blob/f0eca649f7e41e2c9143b4c6d5e92b211ea50d94/Certification/Code/01.%20Imperative.yaml>

Today

Today ...

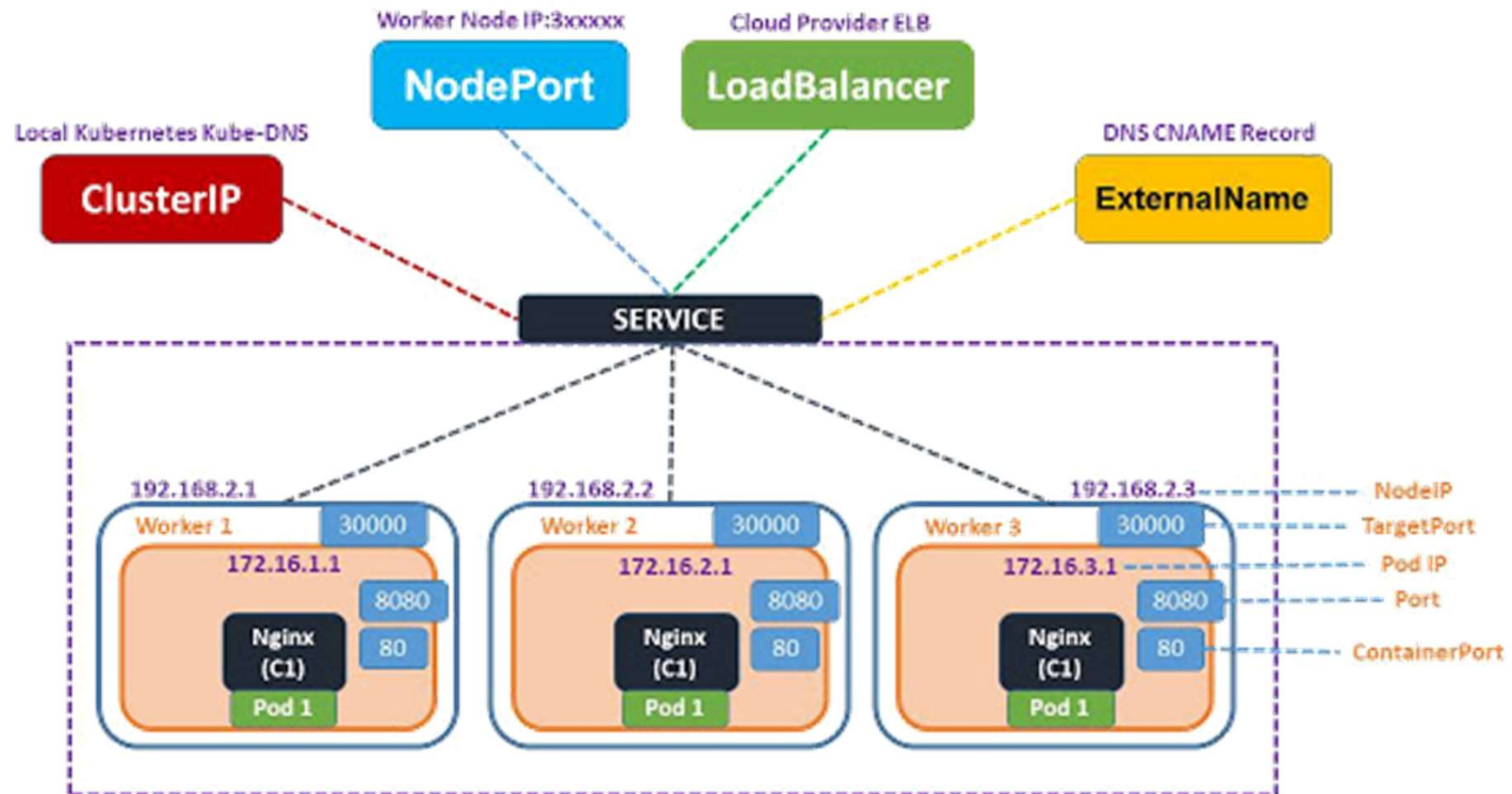
Service



Service

Service is

- 동일한 서비스를 제공하는 Pod 그룹에 지속적인 단일 접점을 만들려고 할 때 생성하는 리소스



※ 참고 : <https://www.learnitguide.net/2020/05/kubernetes-services-explained-examples.html>

Pod Network 실습 - 1/3

① 실습에 사용할 Container Image 설명 

② 실습에 사용할 ReplicaSet YAML

③ 실행

```
> kubectl create -f rs-node-web.yaml
```

```
replicaset.apps/rs-node-web created
```

```
> kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
rs-node-web	3	3	3	9s

```
> kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS	GATES
rs-node-web-dk4hd	1/1	Running	0	15m	10.244.2.21	worker2	<none>		<none>	
rs-node-web-gjs8m	1/1	Running	0	15m	10.244.1.87	worker1	<none>		<none>	
rs-node-web-j2q8s	1/1	Running	0	17m	10.244.1.82	worker1	<none>		<none>	

rs-node-web.yaml

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

```
metadata:
  name: rs-node-web
spec:
  replicas: 3
```

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

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

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

Pod Network 실습 - 2/3

```
> kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
master-stg	Ready	control-plane,master	121d	v1.20.1	192.168.100.119	<none>	Ubuntu 20.04.2 LTS	5.4.0-72-generic	docker://20.10.1
worker1	Ready	<none>	121d	v1.20.1	192.168.100.112	<none>	Ubuntu 20.04.2 LTS	5.4.0-72-generic	docker://20.10.1
worker2	Ready	<none>	4d1h	v1.20.1	192.168.100.111	<none>	Ubuntu 20.04.2 LTS	5.4.0-72-generic	docker://20.10.1

```
> kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
rs-node-web-dk4hd	1/1	Running	0	17m	10.244.2.21	worker2	<none>	<none>
rs-node-web-gjs8m	1/1	Running	0	17m	10.244.1.87	worker1	<none>	<none>
rs-node-web-j2q8s	1/1	Running	0	18m	10.244.1.82	worker1	<none>	<none>

```
> kubectl exec -it rs-node-web-dk4hd -- curl -s http://10.244.2.21:8080
```

You've hit rs-node-web-dk4hd

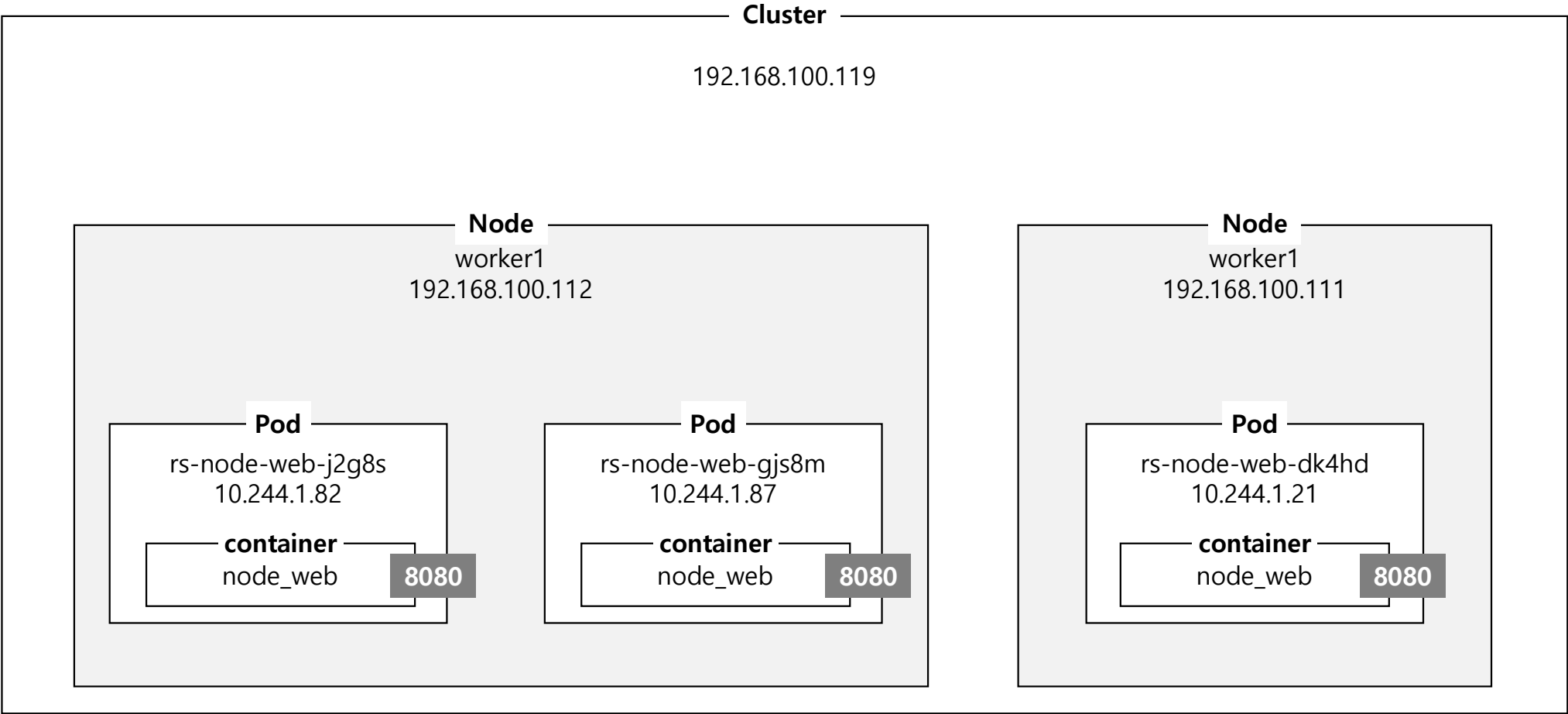
```
> kubectl exec -it rs-node-web-dk4hd -- curl -s http://10.244.1.87:8080
```

You've hit rs-node-web-gjs8m

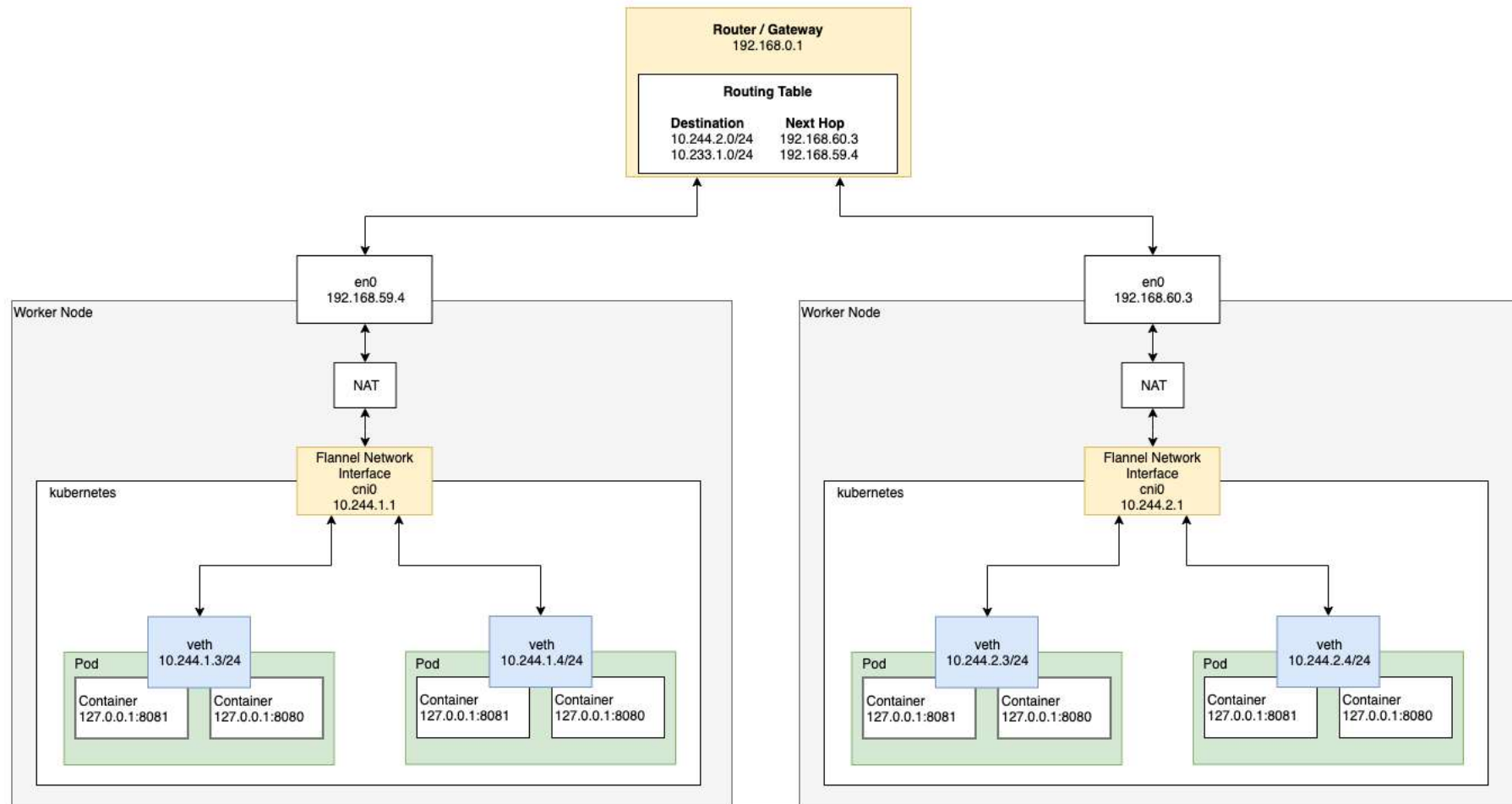
```
> kubectl exec -it rs-node-web-dk4hd -- curl -s http://10.244.1.82:8080
```

You've hit rs-node-web-j2q8s

Pod Network 실습 - 3/3



Pod Network



※ 참고 : <https://medium.com/finda-tech/kubernetes-네트워크-정리-fccd4fd0ae6>

Create Service

svc-none-node-web.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: svc-node
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: node-web
```

```
> kubectl create -f svc-none-node-web.yaml
```

service/svc-node created

```
> kubectl get services -o wide
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	123d	<none>
svc-node	ClusterIP	10.104.191.116	<none>	80/TCP	18s	app=node-web

```
> curl http://10.104.191.116
```

You've hit rs-node-web-dk4hd

```
> curl http://10.104.191.116
```

You've hit rs-node-web-j2q8s

```
> curl http://10.104.191.116
```

You've hit rs-node-web-dk4hd

```
> curl http://10.104.191.116
```

You've hit rs-node-web-gjs8m

sessionAffinity

※ affinity : 친밀감, 관련성

- 특정 클라이언트의 연결이 매번 동일한 Pod로 전달 : `service.spec.sessionAffinity = "ClientIP"`
- 기본값은 "None"

svc-clientip-node-web.yaml

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

spec:
  ports:
    - port: 80
      targetPort: 8080

  selector:
    app: node-web

  sessionAffinity: ClientIP
```

```
> kubectl apply -f svc-client-node-web.yaml
```

```
Warning: resource services/svc-node is missing the kubect.kubernetes.io/last-applied-configuration annotation which is required
by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or
kubectl apply. The missing annotation will be patched automatically.
service/svc-node configured
```

```
> kubectl describe service svc-node
```

```
Name:          svc-node
Namespace:     default
Labels:        <none>
Annotations:   <none>
Selector:      app=node-web
Type:          ClusterIP
IP Families:   <none>
IP:            10.104.191.116
IPs:           10.104.191.116
Port:          <unset> 80/TCP
TargetPort:    8080/TCP
Endpoints:     10.244.1.88:8080,10.244.1.89:8080,10.244.2.22:8080
Session Affinity: ClientIP
Events:        <none>
```

```
> curl http://10.104.191.116
```

You've hit rs-node-web-j2q8s

```
> curl http://10.104.191.116
```

You've hit rs-node-web-j2q8s

```
> curl http://10.104.191.116
```

You've hit rs-node-web-j2q8s

ports name

rs-node-web.yaml

```
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-node-web
spec:
  replicas: 3
  selector:
    matchExpressions:
      - key: app
        operator: In
        values:
          - node-web
  template:
    metadata:
      labels:
        app: node-web
    spec:
      containers:
        - name: node-web
          image: whatwant/node-web:1.0
          ports:
            - containerPort: 8080
```

rs-node-web-ports-name.yaml

```
apiVersion: apps/v1
kind: ReplicaSet

metadata:
  name: rs-node-web
spec:
  replicas: 3
  selector:
    matchExpressions:
      - key: app
        operator: In
        values:
          - node-web
  template:
    metadata:
      labels:
        app: node-web
    spec:
      containers:
        - name: node-web
          image: whatwant/node-web:1.0
          ports:
            - name: node-http
              containerPort: 8080
```

svc-clientip-node-web-ports-name.yaml

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

spec:
  ports:
    - port: 80
      targetPort: node-http

  selector:
    app: node-web

  sessionAffinity: ClientIP
```

- Pod에서 정의된 ports에 name을 붙이면
- Service에서 그 이름을 사용할 수 있다.
- Service spec을 변경하지 않고도 ports 변경 가능

Service check

- Pod가 시작되면 K8s는 해당 시점에 존재하는 Service를 가리키는 환경변수를 설정

```
> kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	123d
svc-node	ClusterIP	10.104.191.116	<none>	80/TCP	67m

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
rs-node-web-dk4hd	1/1	Running	1	45h
rs-node-web-gjs8m	1/1	Running	1	45h
rs-node-web-j2q8s	1/1	Running	1	45h

```
> kubectl exec -it rs-node-web-dk4hd -- env
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=rs-node-web-dk4hd
TERM=xterm
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
NODE_VERSION=15.14.0
YARN_VERSION=1.22.5
HOME=/root
```

```
> kubectl delete pods rs-node-web-dk4hd
```

```
pod "rs-node-web-dk4hd" deleted
```

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
rs-node-web-ghqnq	1/1	Running	0	39s
rs-node-web-gjs8m	1/1	Running	1	45h
rs-node-web-j2q8s	1/1	Running	1	45h

```
> kubectl exec -it rs-node-web-ghqnq -- env
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=rs-node-web-ghqnq
TERM=xterm
SVC_NODE_SERVICE_HOST=10.104.191.116
SVC_NODE_PORT_80_TCP=tcp://10.104.191.116:80
KUBERNETES_PORT_443_TCP_PORT=443
SVC_NODE_PORT=tcp://10.104.191.116:80
SVC_NODE_PORT_80_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
SVC_NODE_PORT_80_TCP_PORT=80
SVC_NODE_PORT_80_TCP_ADDR=10.104.191.116
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_PORT_443_TCP_PROTO=tcp
SVC_NODE_SERVICE_PORT=80
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
NODE_VERSION=15.14.0
YARN_VERSION=1.22.5
HOME=/root
```

DNS - 1/2

- kube-system namespace 內 kube-dns Pod 존재

. DNS Server 실행하며 cluster에서 실행중인 다른 모든 Pod는 자동으로 이를 사용하도록 구성

```
> kubectl get pods --namespace kube-system -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
coredns-74ff55c5b-dz8kq	1/1	Running	10	123d	10.244.0.23	master-stg	<none>	<none>
coredns-74ff55c5b-h66ct	1/1	Running	10	123d	10.244.0.22	master-stg	<none>	<none>
etcd-master-stg	1/1	Running	29	123d	192.168.100.119	master-stg	<none>	<none>
kube-apiserver-master-stg	1/1	Running	26	123d	192.168.100.119	master-stg	<none>	<none>
kube-controller-manager-master-stg	1/1	Running	15	123d	192.168.100.119	master-stg	<none>	<none>
kube-flannel-ds-99psb	1/1	Running	14	123d	192.168.100.112	worker1	<none>	<none>
kube-flannel-ds-sdcpc	1/1	Running	4	5d23h	192.168.100.111	worker2	<none>	<none>
kube-flannel-ds-sdss8	1/1	Running	13	123d	192.168.100.119	master-stg	<none>	<none>
kube-proxy-5lvx5	1/1	Running	3	5d23h	192.168.100.111	worker2	<none>	<none>
kube-proxy-nkb8c	1/1	Running	10	123d	192.168.100.119	master-stg	<none>	<none>
kube-proxy-xvlvs	1/1	Running	10	123d	192.168.100.112	worker1	<none>	<none>
kube-scheduler-master-stg	1/1	Running	15	123d	192.168.100.119	master-stg	<none>	<none>

DNS - 2/2

- Kubernetes는 각 container의 /etc/resolv.conf 파일을 수정

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

```
> kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	123d
kube-node-lease	Active	123d
kube-public	Active	123d
kube-system	Active	123d

```
> kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	123d
svc-node	ClusterIP	10.104.191.116	<none>	80/TCP	153m

```
> kubectl exec -it rs-node-web-ghqng -- cat /etc/resolv.conf
```

```
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local skbroadband
options ndots:5
```

```
> kubectl exec -it rs-node-web-ghqng -- curl -s http://svc-node
```

You've hit rs-node-web-ghqng

```
> kubectl exec -it rs-node-web-ghqng -- curl -s http://svc-node.default
```

You've hit rs-node-web-ghqng

```
> kubectl exec -it rs-node-web-ghqng -- curl -s http://svc-node.default.svc
```

You've hit rs-node-web-ghqng

```
> kubectl exec -it rs-node-web-ghqng -- curl -s http://svc-node.default.svc.cluster
```

^Ccommand terminated with exit code 130

추가 확인 필요

```
> kubectl exec -it rs-node-web-ghqng -- curl -s http://svc-node.default.svc.cluster.local
```

^Ccommand terminated with exit code 130

Endpoints - 1/2

- Service로 노출되는 Pod의 IP 주소와 Port 목록
- client → kube-proxy → Endpoints 중 하나의 IP/Port 선택 → 들어온 연결을 대상 Pod의 수신 대기 서버로 전달

```
> kubectl describe service svc-node
```

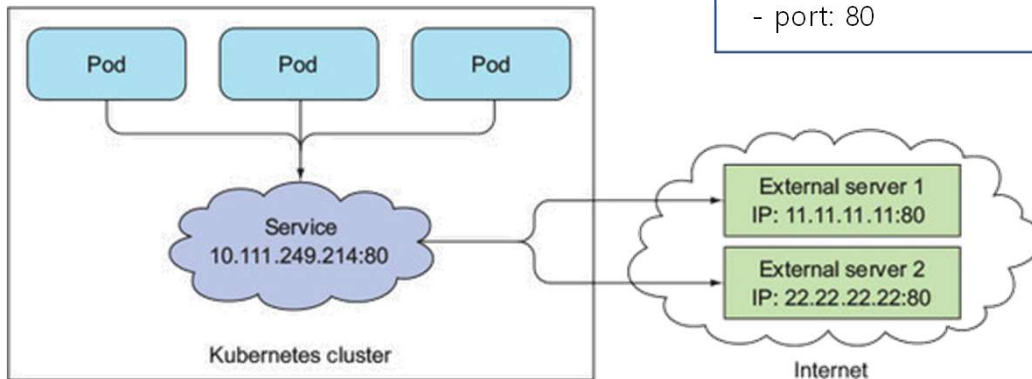
```
Name:          svc-node
Namespace:     default
Labels:        <none>
Annotations:   <none>
Selector:      app=node-web
Type:          ClusterIP
IP Families:   <none>
IP:            10.104.191.116
IPs:           10.104.191.116
Port:          <unset> 80/TCP
TargetPort:    8080/TCP
Endpoints:     10.244.1.88:8080,10.244.1.89:8080,10.244.2.23:8080
Session Affinity: ClientIP
Events:        <none>
```

```
> kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE
READINESS GATES								
rs-node-web-ghqnq	1/1	Running	0	99m	10.244.2.23	worker2	<none>	<none>
rs-node-web-gjs8m	1/1	Running	1	47h	10.244.1.88	worker1	<none>	<none>
rs-node-web-j2q8s	1/1	Running	1	47h	10.244.1.89	worker1	<none>	<none>

Endpoints - 2/2

- Endpoints 명시하지 않으면? = Selectors를 지정하지 않기
- . Endpoints 직접 생성 필요 (Service와 Endpoints 이름이 같아야 한다)



svc-no-selector.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: external-service

spec:
  ports:
    - port: 80
```

svc-no-selector.yaml

```
apiVersion: v1
kind: Endpoints
metadata:
  name: external-service

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

ExternalName

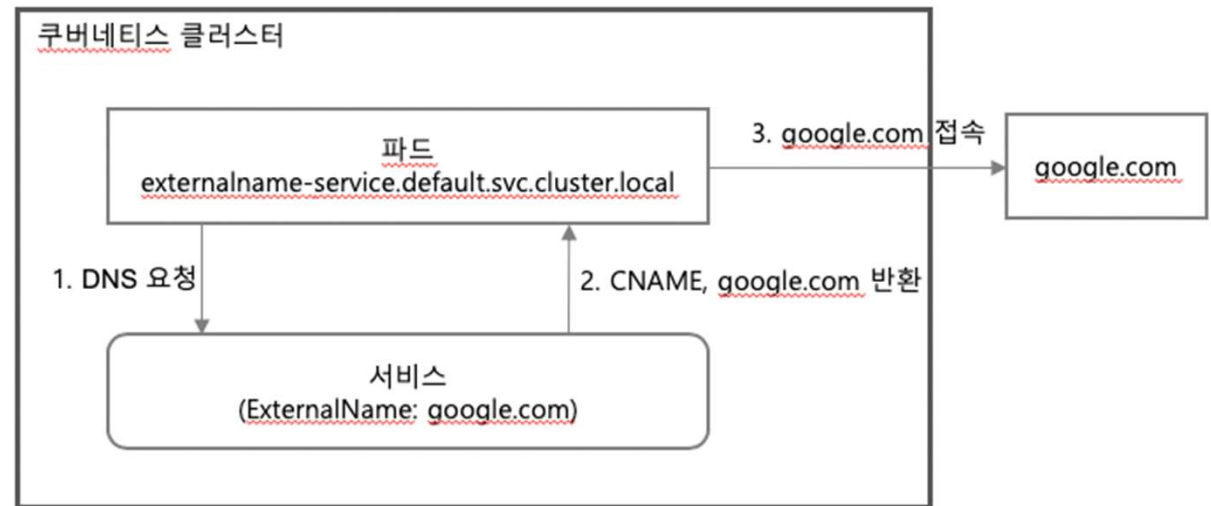
- 외부 서비스의 별칭으로 사용되는 서비스
- . Pod는 서비스의 FQDN을 사용하는 대신 external-service.default.svc.cluster.local 로 외부 서비스에 연결
- . Service를 사용하는 Pod에서 실제 서비스 이름과 위치가 숨겨짐
- . 나중에 Service spec 수정하여 다른 Service를 가리킬 수 있음

external-name.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: external-service

spec:
  type: ExternalName
  externalName: google.com
```

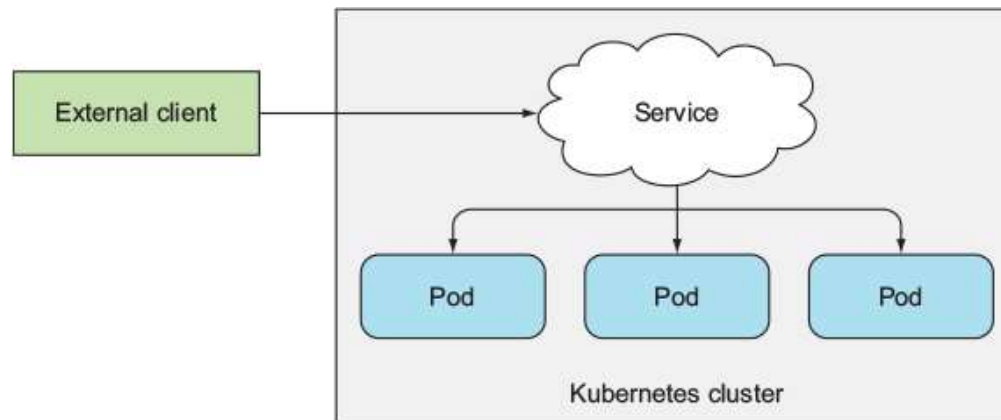
추가 확인 필요



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

Exposing a service to external clients

- NodePort
- LoadBalancer
- Ingress



※ 참고 : <https://livebook.manning.com/book/kubernetes-in-action/chapter-5/158>

NodePort

- 모든 Node에 특정 Port를 할당하고 Service를 구성하는 Pod로 들어오는 연결을 전달
- . 모든 Node에서 동일한 Port 번호 사용

svc-nodeport.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: svc-nodeport

spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30123

  selector:
    app: node-web
```

```
> kubectl create -f svc-nodeport.yaml
```

```
service/svc-nodeport created
```

```
> kubectl get services -o wide
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	124d	<none>
svc-nodeport	NodePort	10.110.15.244	<none>	80:30123/TCP	6s	app=node-web

```
> kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
master-stg	Ready	control-plane,master	124d	v1.20.1	192.168.100.119	<none>	Ubuntu 20.04.2 LTS	5.4.0-72-generic	docker://20.10.1
worker1	Ready	<none>	124d	v1.20.1	192.168.100.112	<none>	Ubuntu 20.04.2 LTS	5.4.0-72-generic	docker://20.10.1
worker2	Ready	<none>	6d3h	v1.20.1	192.168.100.111	<none>	Ubuntu 20.04.2 LTS	5.4.0-72-generic	docker://20.10.1

```
> curl http://192.168.100.119:30123
```

```
You've hit rs-node-web-6fz7s
```

```
> curl http://192.168.100.112:30123
```

```
You've hit rs-node-web-v6khr
```

```
> curl http://192.168.100.111:30123
```

```
You've hit rs-node-web-v6khr
```

LoadBalancer

- Service type을 LoadBalancer로 선언 하면 Provider에 의해서 LoadBalancer가 생성
- . 생성된 LoadBalancer는 외부 공개 IP를 가지게 되며 해당 IP로 접근 가능

svc-loadbalancer.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: svc-loadbalancer
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: node-web
```

```
> kubectl create -f svc-loadbalancer.yaml
```

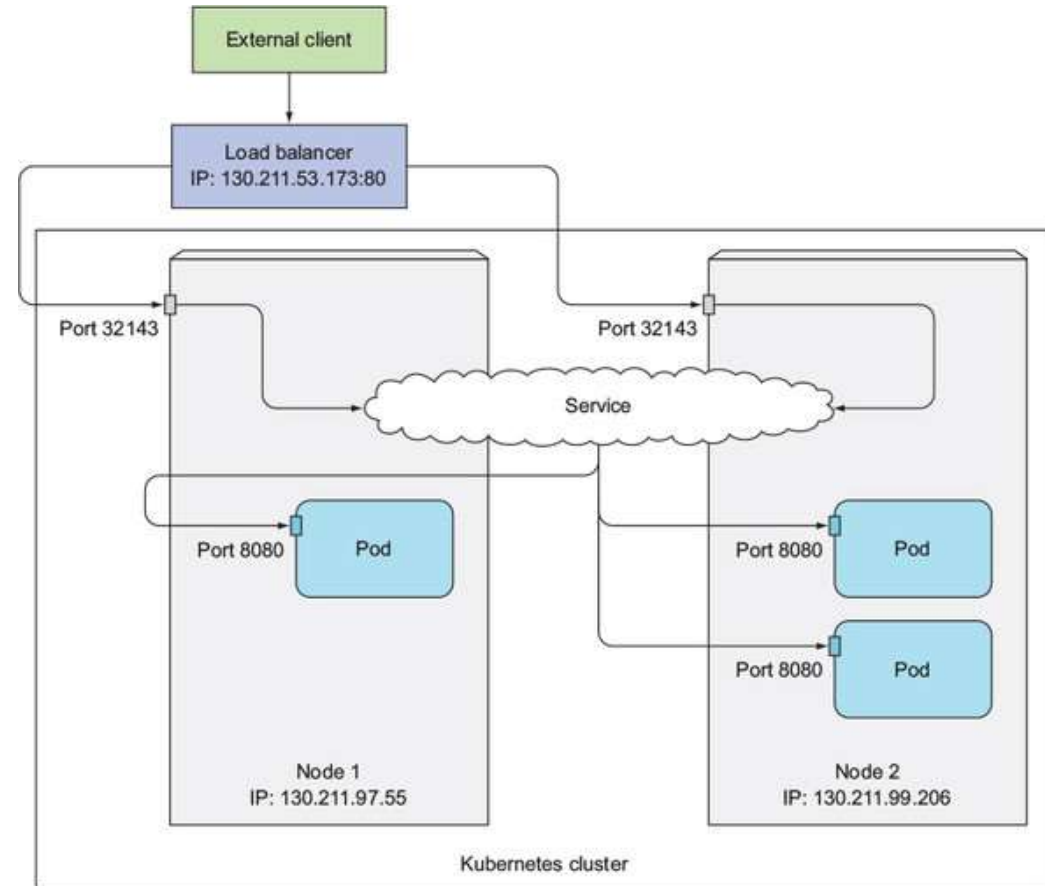
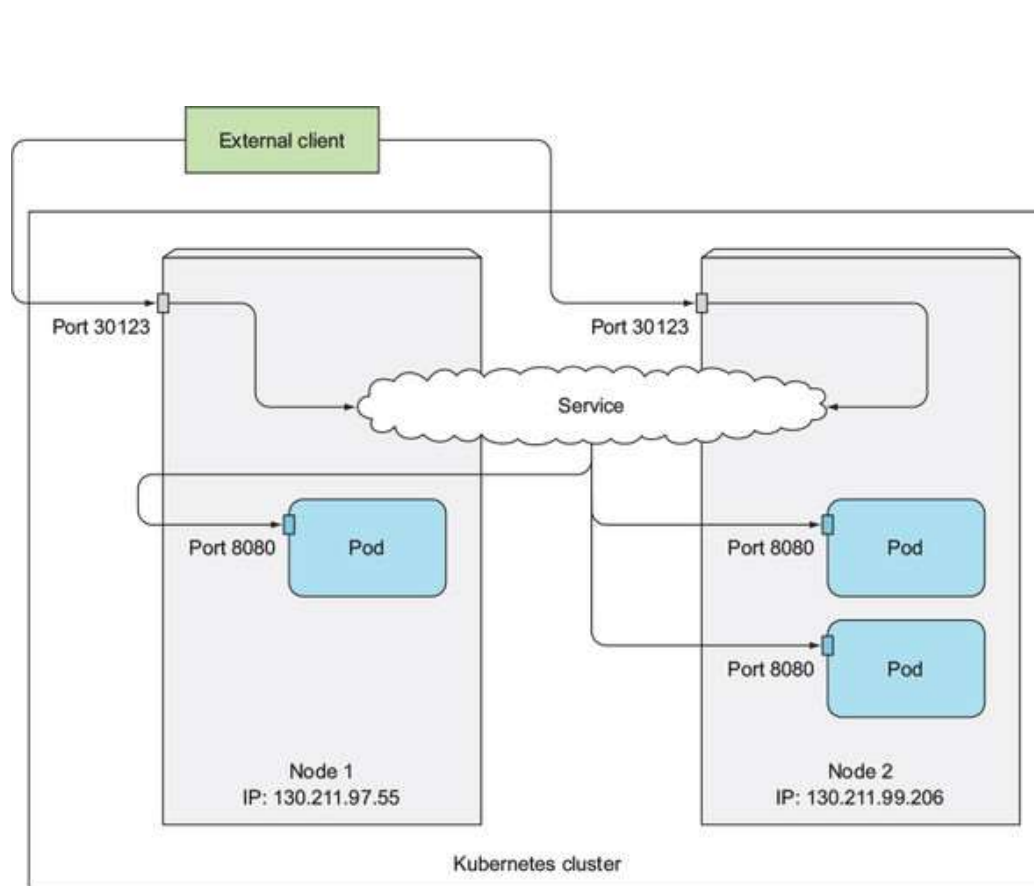
```
service/svc-loadbalancer created
```

```
> kubectl get services -o wide -w
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	124d	<none>
svc-loadbalancer	LoadBalancer	10.110.162.50	<pending>	80:32701/TCP	8m53s	app=node-web
svc-nodeport	NodePort	10.110.15.244	<none>	80:30123/TCP	24m	app=node-web

추가 확인 필요

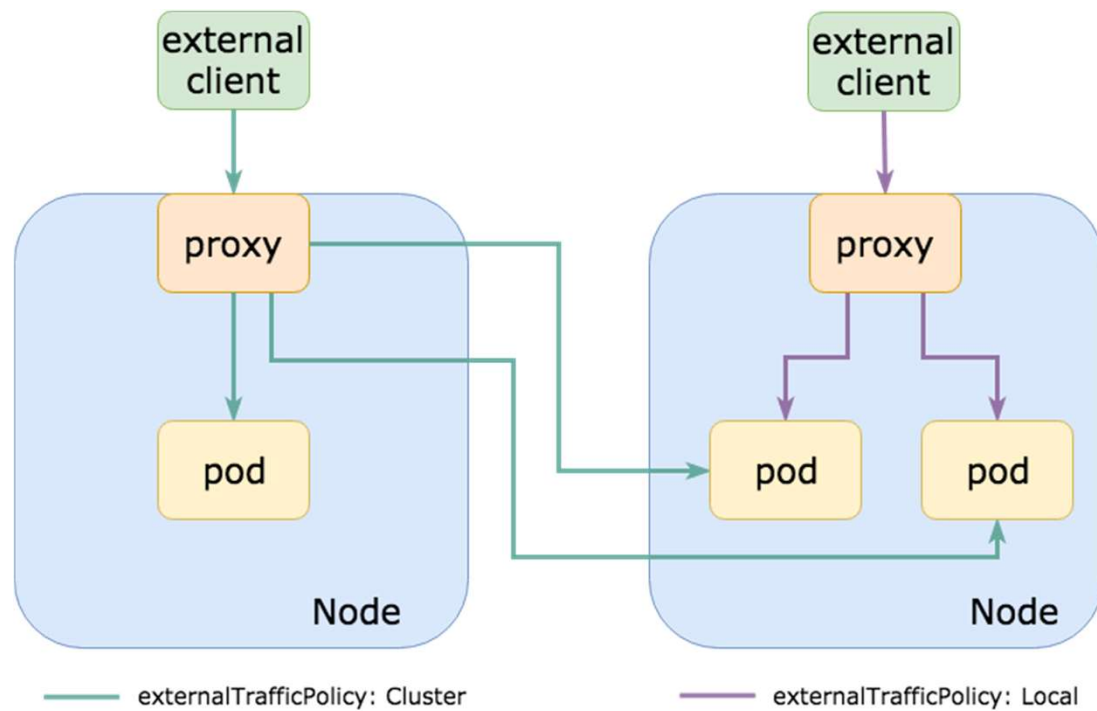
NodePort vs LoadBalancer



※ 참고 : <https://livebook.manning.com/book/kubernetes-in-action/chapter-5/219>

externalTrafficPolicy - 1/2

- 불필요한 네트워크 hop 발생
- 클라이언트 IP가 보존되지 않음



Proxy: iptables proxy rules for Service ExternalIPs or NodePort

※ 참고 : <https://coffeewhale.com/kubernetes/mistake/2020/11/29/mistake-10/>

externalTrafficPolicy - 2/2

- 오직 로컬 Endpoint로만 proxy 요청, 다른 node로 트래픽 전달하지 않음
- . 원본 소스 IP 주소를 보존

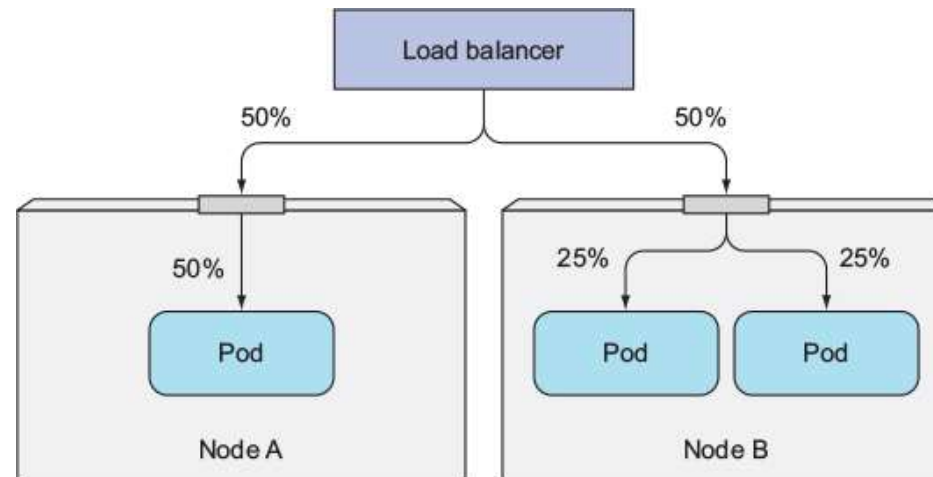
svc-externaltrafficpolicy.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: svc-loadbalancer
```

```
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
```

```
selector:
  app: node-web
```

externalTrafficPolicy: Local

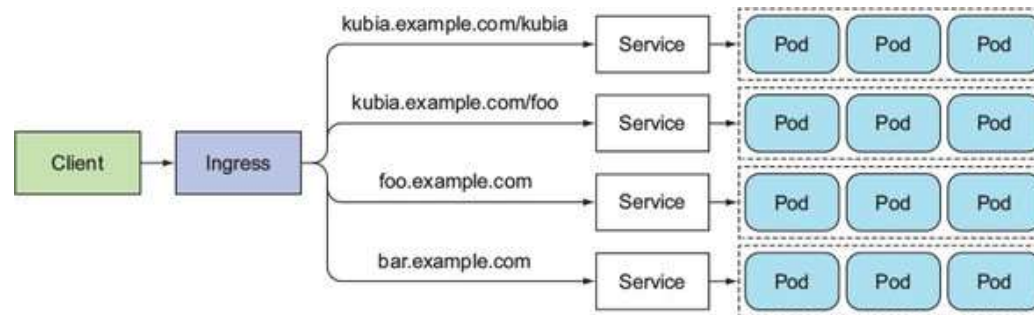


※ "externalTrafficPolicy: Local" 설정에 따른 문제점 : 균등히 배부되지 않을 수 있음

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

Ingress - 1/3

- 클라이언트가 요청한 호스트와 경로에 따라 요청을 전달할 서비스가 결정
 - . HTTP(S)기반의 L7 LoadBalancing 기능을 제공하는 컴포넌트
- Ingress 구현체
 - . 구글 클라우드 : <https://github.com/kubernetes/ingress-gce>
 - . 오픈소스 : <https://github.com/kubernetes/ingress-nginx>
 - . 상용 : F5 BIG IP Controller (<http://clouddocs.f5.com/products/connectors/k8s-bigip-ctrl>)
- 각 구현체마다 설정 방법 차이 존재



※ 참고 : <https://livebook.manning.com/book/kubernetes-in-action/chapter-5/243>

Ingress - 2/3

- 오픈소스 Ingress 구현체 설치

. <https://docs.nginx.com/nginx-ingress-controller/installation/installation-with-manifests/>

Ingress - 3/3

svc-externaltrafficpolicy.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: svc-ingress

spec:
  rules:
    - host: k8s.whatwant.com
      http:
        paths:
          - pathType: Prefix
            path: /
            backend:
              service:
                name: svc-nodeport
                port:
                  number: 80
```

```
> kubectl create -f svc-ingress.yaml
```

```
ingress.networking.k8s.io/svc-ingress created
```

```
> kubectl get services -o wide
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	124d	<none>
svc-loadbalancer	LoadBalancer	10.110.162.50	<pending>	80:32701/TCP	92m	app=node-web
svc-nodeport	NodePort	10.110.15.244	<none>	80:30123/TCP	108m	app=node-web

```
> kubectl get ingress -o wide
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
svc-ingress	<none>	k8s.whatwant.com		80	38s

추가 확인 필요

TLS - 1/

- ///

readinessProbe - 1/

- ///

Headless Service - 1/

- ///

<https://kahoot.it/>