4th Week

네번째 뵙겠습니다 ?!

▷ 잠시만 기다렸다가 30분 되면 시작하겠습니다~^^

▷ 2주 만에 만나게 되어 대단히 반갑습니다 !!!

- 다시 신나게 달려봅시다~~~!!!

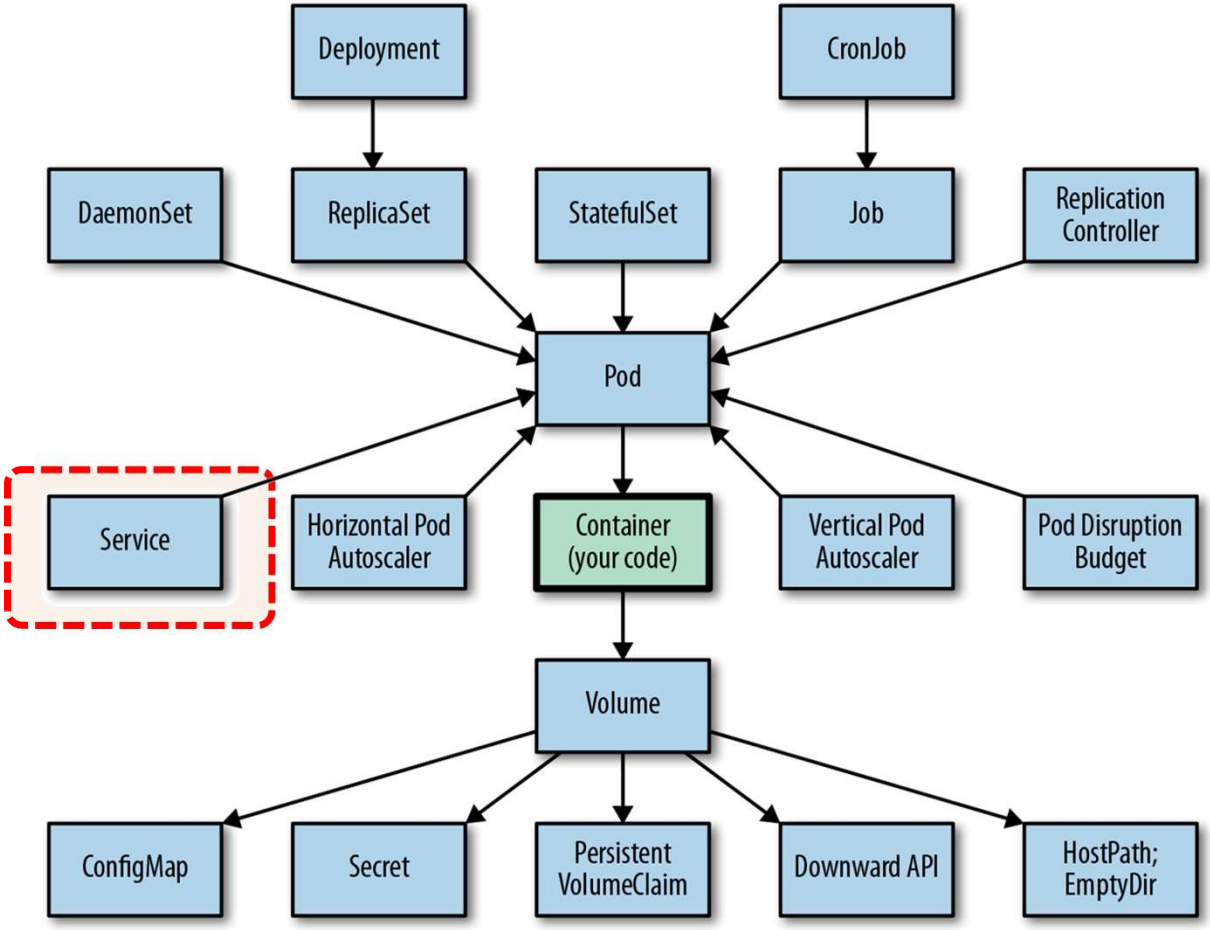
▷ Camera는 가급적 켜 주시면 대단히 감사하겠습니다 !!!

- 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

///

Service

ClusterIP/NodePort/ExternalName/ Endpoints/Headless

///

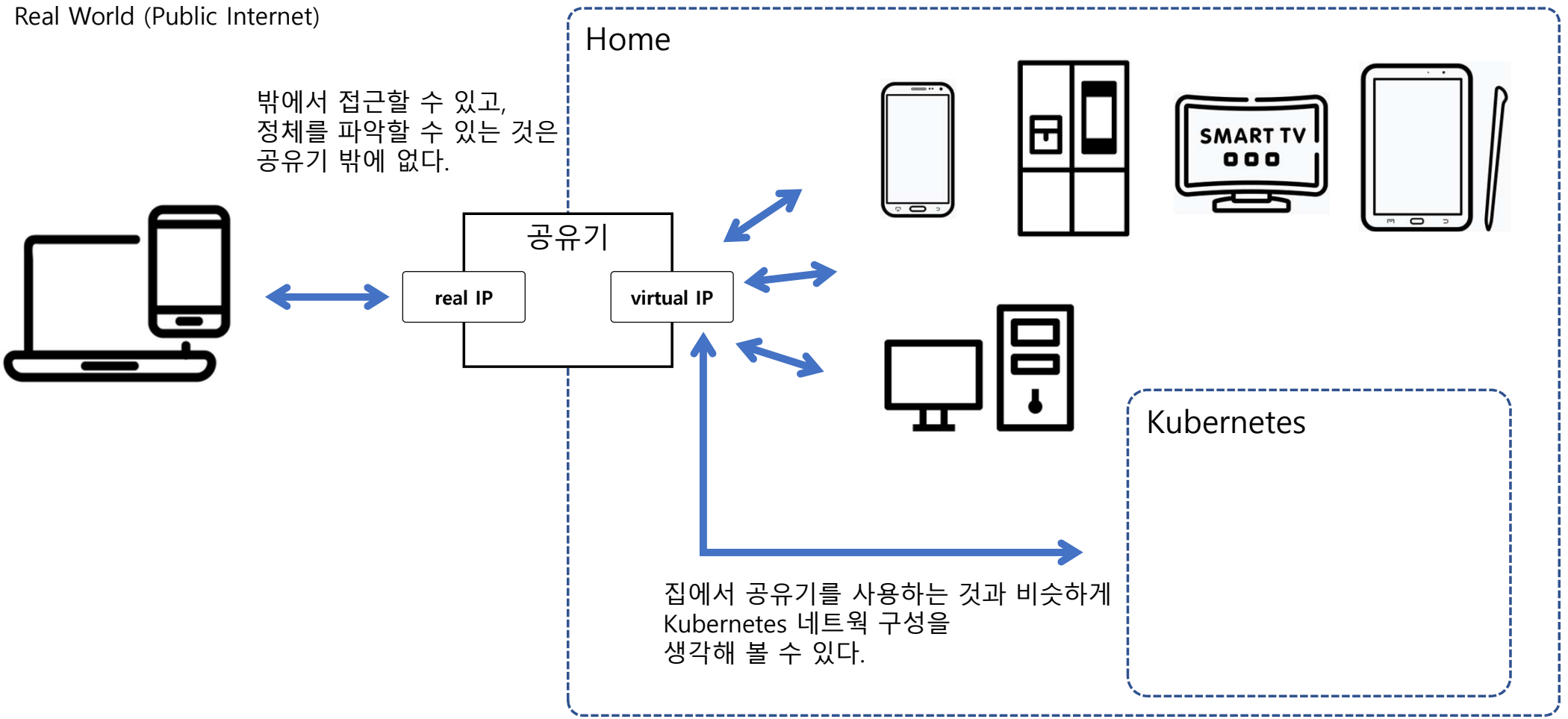
Flip Learning

(Service - ClusterIP/NodePort/ExternalName)

無名님

///

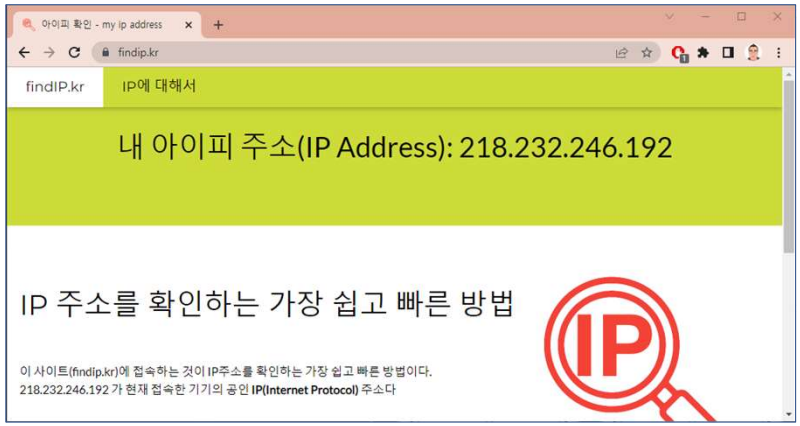
Network


Real World (Public Internet)
Home
밖에서 접근할 수 있고,
정체를 파악할 수 있는 것은
공유기 밖에 없다.
공유기
real IP
virtual IP
SMART TV
Kubernetes
집에서 공유기를 사용하는 것과 비슷하게
Kubernetes 네트웍 구성을
생각해 볼 수 있다.

My Public(Real) IP

Windows

(Host)




```
MINGW64:/c/Users/whatw

whatw@SUPER-PC MINGW64 ~
$ ipconfig

Windows IP Configuration


Ethernet adapter ▒▒▒▒:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5c4e:e7f8:28a5:7c1f%11
   IPv4 Address. . . . . . . . . . . : 192.168.100.10
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.100.1
```

Linux

(Guest - VM)


```
❯ ifconfig
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.200  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:fed1:f3a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:d1:0f:3a  txqueuelen 1000  (Ethernet)
        RX packets 29393  bytes 3218179 (3.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59709  bytes 50024345 (50.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Node IPs

192.168.100.200

Node

(master)

192.168.100.201

Node

(worker1)

192.168.100.202

Node

(worker2)

Work PC

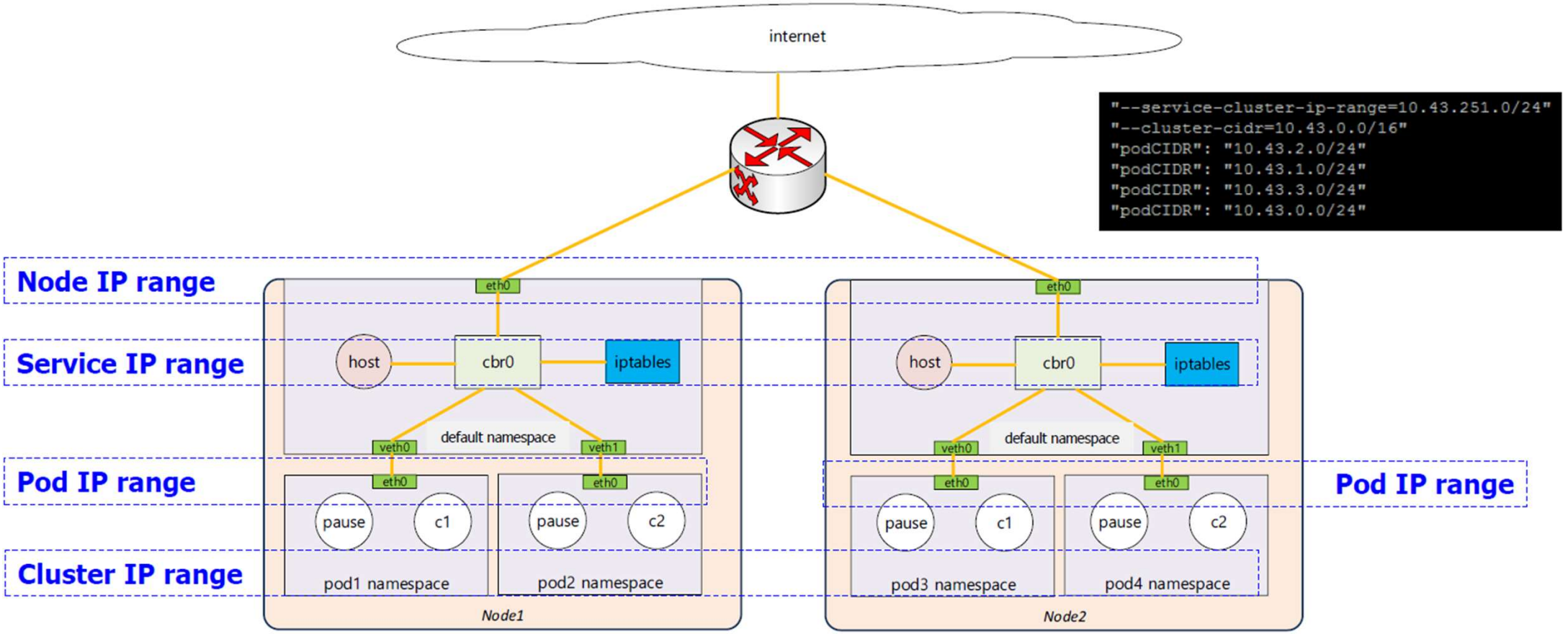
///

Kubernetes

Service

K8S Network IP Ranges

Kubernetes 안에서는 여러 계층의 Network Layer 존재



※ 참고 : https://www.slideshare.net/InfraEngineer/ss-186475759

Pod IP range

worker node에 따라 C-Class 대역이 다름을 볼 수 있다.

```
remote ❯ cd kubernetes/03-RS-DS-JOB-CJ/hands-on

remote ❯ kubectl create -f rs-web.yaml

remote ❯ kubectl get pods -o wide

NAME          READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-22d9l   1/1     Running   0          8s    10.233.103.66   worker2   <none>           <none>
rs-web-9gp6x   1/1     Running   0          9s    10.233.110.17   worker1   <none>           <none>
rs-web-fd7br   1/1     Running   0          9s    10.233.103.65   worker2   <none>           <none>
```

Pod끼리 서로 통신할 수 있음

```
remote ❯ kubectl exec -it rs-web-fd7br -- /bin/bash

root@rs-web-fd7br:/# curl http://10.233.103.66:8080

You've hit rs-web-22d9l

root@rs-web-fd7br:/# curl http://10.233.110.17:8080

You've hit rs-web-9gp6x

root@rs-web-fd7br:/# exit
exit
```

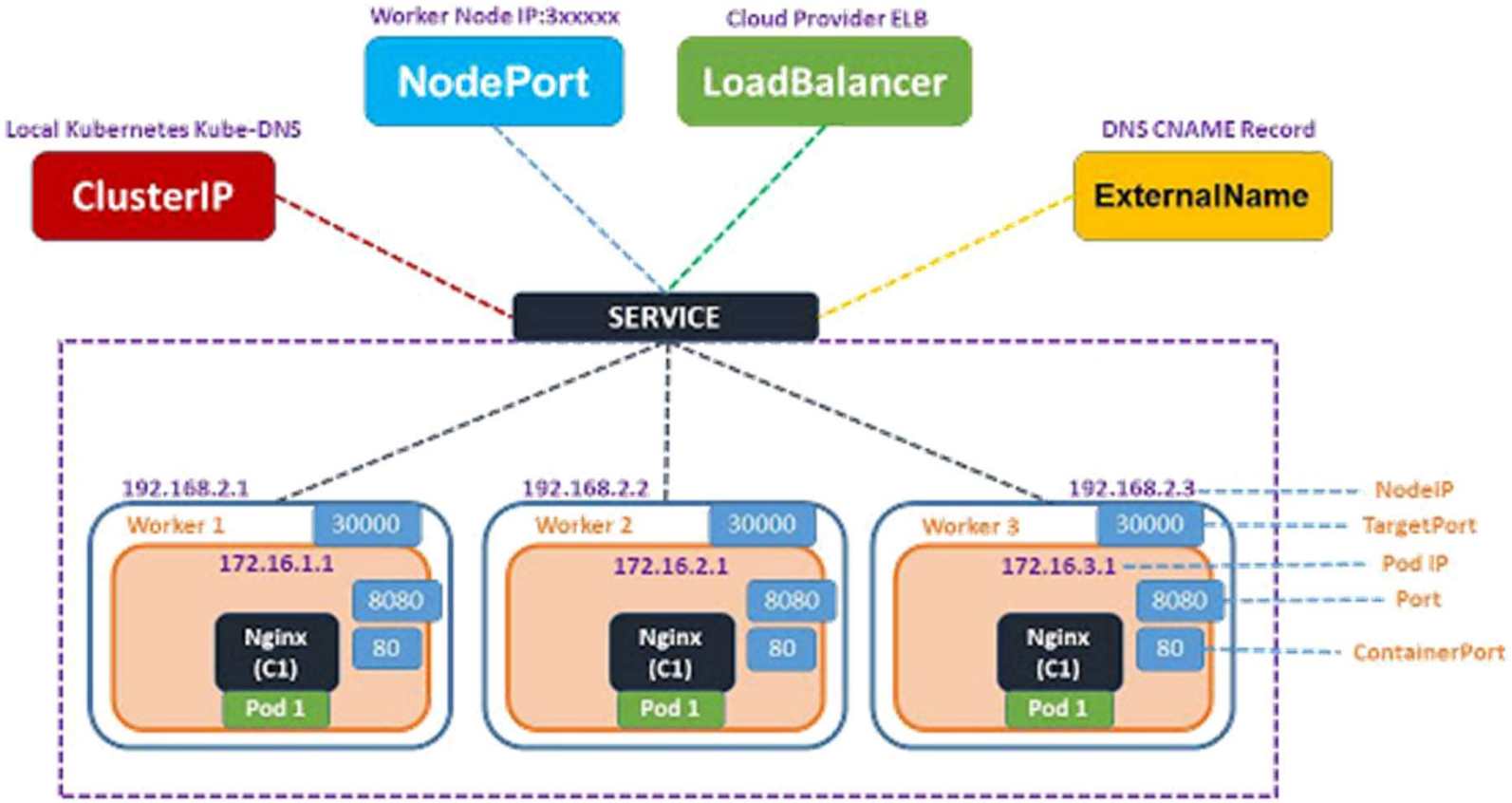
Node IP range

node IP를 확인해볼 수 있다

```
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   19d   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>                 19d   v1.23.7   192.168.100.201   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
worker2   Ready    <none>                 19d   v1.23.7   192.168.100.202   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
```

Service is ...

동일한 서비스를 제공하는 Pod 그룹에 지속적인 **단일 접점**을 만들려고 할 때 생성하는 리소스



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

Keywords

▷ ClusterIP

- 디폴트 설정으로, 서비스에 클러스터 IP (내부 IP)를 할당한다.
- 쿠버네티스 클러스터 내에서는 이 서비스에 접근이 가능하지만, 클러스터 외부에서는 외부 IP 를 할당 받지 못했기 때문에 접근이 불가능하다.

▷ NodePort

- 클러스터 IP로만 접근이 가능한 것이 아니라, 모든 노드의 IP와 포트를 통해서도 접근이 가능하게 된다.
- 예를 들어 hello-node-svc 라는 서비스를 NodePort 타입으로 선언을 하고 nodePort를 30036으로 설정하면,
- 설정에 따라 클러스터 IP의 80포트로도 접근이 가능하지만, 모든 노드의 30036 포트로도 서비스를 접근할 수 있다.

▷ Load Balancer

- 보통 클라우드 벤더에서 제공하는 설정 방식으로, 외부 IP 를 가지고 있는 LoadBalancer를 할당한다.
- 외부 IP를 가지고 있기 때문에, 클러스터 외부에서 접근이 가능하다.

▷ ExternalName

- 외부 서비스를 쿠버네티스 내부에서 호출하고자 할 때 사용할 수 있다.
- Pod들은 클러스터 IP를 가지고 있기 때문에 클러스터 IP 대역 밖의 서비스를 호출하고자 하면, NAT 설정 等 복잡한 설정이 필요하다. 이 때 사용하면 된다.

※ 참고 : https://bcho.tistory.com/1262

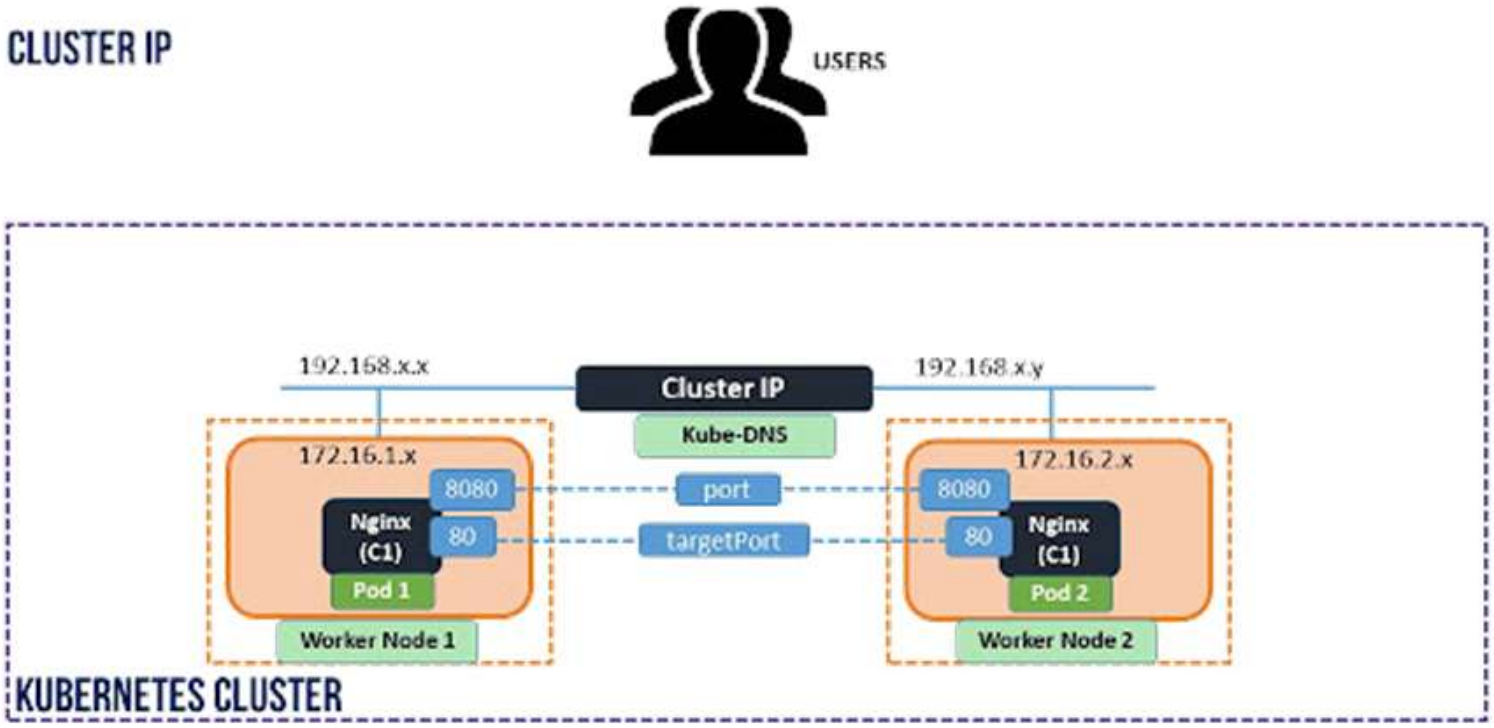
///

ClusterIP

ClusterIP

- 디폴트 설정으로, 서비스에 클러스터 IP (내부 IP)를 할당한다.

- 클러스터 내에서는 이 서비스에 접근이 가능하지만, 클러스터 외부에서는 외부 IP 를 할당 받지 못했기 때문에 접근이 불가능하다.



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

Ready ... ReplicaSet

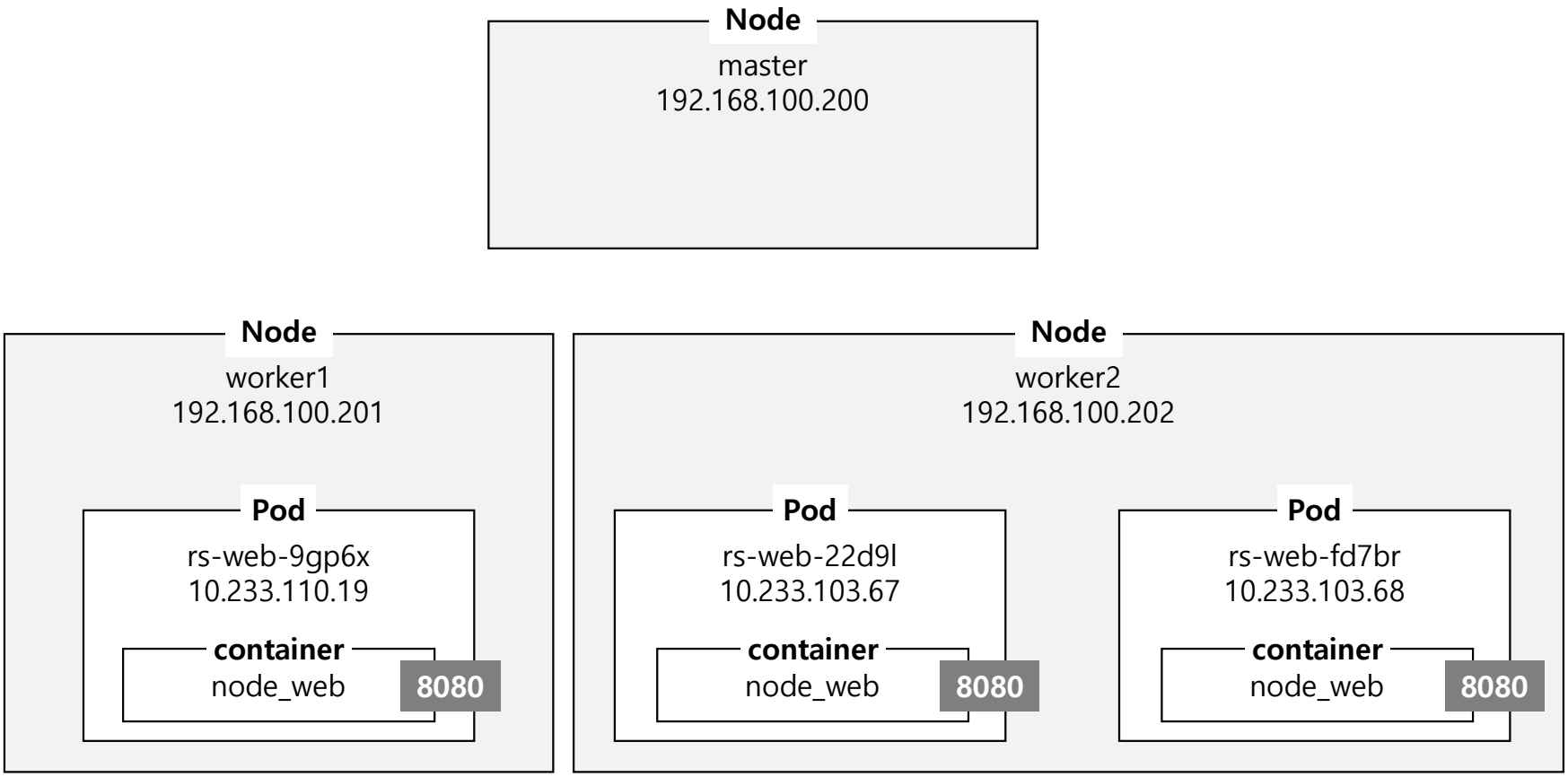
Service 가 필요한 Pod를 앞에서 공부했던 ReplicaSet을 통해 생성해 놓자

rs-web.yaml

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

metadata:
  name: rs-web

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 kubernetes/03-RS-DS-JOB-CJ/hands-on

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

replicaset.apps/rs-web created


remote ❯ kubectl get replicasets -o wide

NAME     DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                   SELECTOR
rs-web   3         3         3       2d    node-web     whatwant/node-web:1.0    app=node-web


remote ❯ kubectl get pods -o wide

NAME           READY   STATUS    RESTARTS       AGE   IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-22d9l   1/1     Running   1 (132m ago)   2d    10.233.103.67   worker2   <none>           <none>
rs-web-9gp6x   1/1     Running   1 (132m ago)   2d    10.233.110.19   worker1   <none>           <none>
rs-web-fd7br   1/1     Running   1 (132m ago)   2d    10.233.103.68   worker2   <none>           <none>


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   19d   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>                 19d   v1.23.7   192.168.100.201   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
worker2   Ready    <none>                 19d   v1.23.7   192.168.100.202   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
```

Now Status


Node
master
192.168.100.200
Node
worker1
192.168.100.201
Pod
rs-web-9gp6x
10.233.110.19
container
node_web
8080
Node
worker2
192.168.100.202
Pod
rs-web-22d9l
10.233.103.67
container
node_web
8080
Pod
rs-web-fd7br
10.233.103.68
container
node_web
8080

Service Create - ClusterIP

우리가 원하는 것은 하나의 접점으로 Pod들 중 하나에 접근할 수 있도록 하는 것이다.

svc-web.yaml

```
apiVersion: v1
kind: Service

metadata:
  name: svc-web

spec:
  ports:
  - port: 80
    targetPort: 8080

  selector:
    app: node-web
```

```
remote ❯ cd kubernetes/04-ClusterIP-NodePort-ExternalName/hands-on

remote ❯ kubectl create -f svc-web.yaml

service/svc-web 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   20d   <none>
svc-web      ClusterIP   10.233.53.193   <none>        80/TCP    37s   app=node-web
```

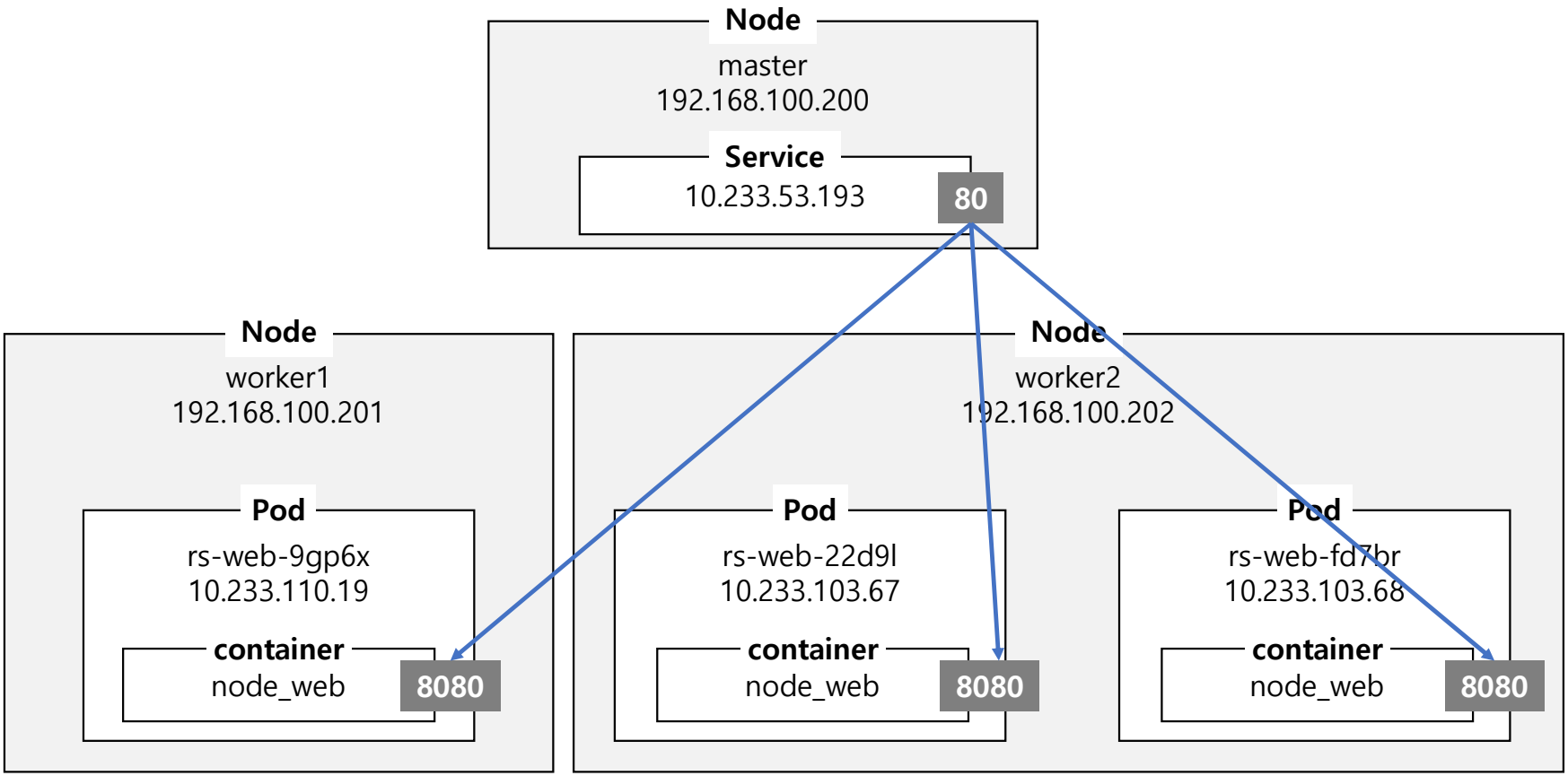
외부에서는 접근이 안된다.

```
remote ❯ curl http://10.233.53.193

^C
```

Cluster 내부에서는 접근이 된다 ! (worker node에서도 된다)

```
remote ❯ ssh vagrant@192.168.100.200


master ❯ curl http://10.233.53.193

You've hit rs-web-22d9l


master ❯ curl http://10.233.53.193

You've hit rs-web-fd7br
```

Now Status


Node
master
192.168.100.200
Service
10.233.53.193
80
Node
worker1
192.168.100.201
Pod
rs-web-9gp6x
10.233.110.19
container
node_web
8080
Node
worker2
192.168.100.202
Pod
rs-web-22d9l
10.233.103.67
container
node_web
8080
Pod
rs-web-fd7br
10.233.103.68
container
node_web
8080

///

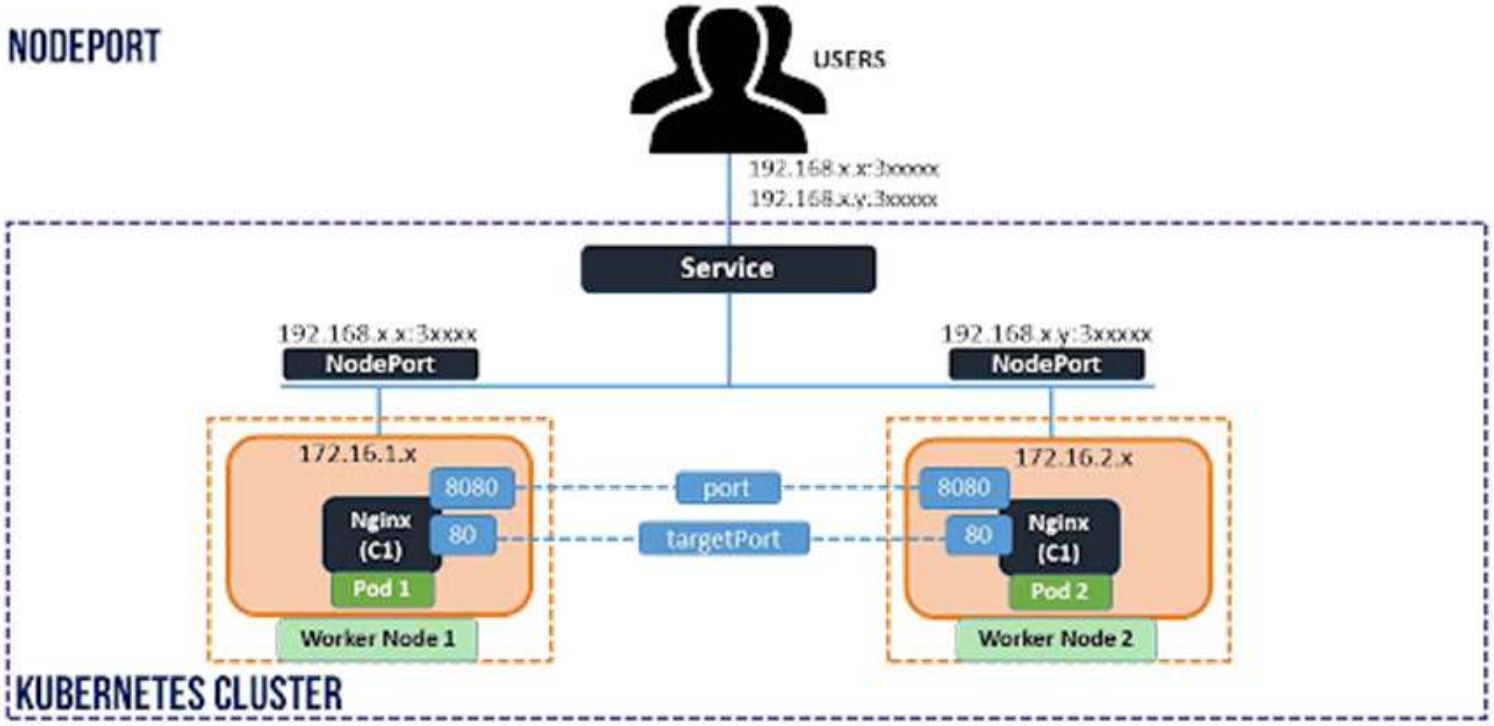
NodePort

NodePort

- 클러스터 IP로만 접근이 가능한 것이 아니라, 모든 노드의 IP와 포트를 통해서도 접근이 가능하게 된다.

- 예를 들어 hello-node-svc 라는 서비스를 NodePort 타입으로 선언을 하고 nodePort를 30036으로 설정하면,

- 설정에 따라 클러스터 IP의 80포트로도 접근이 가능하지만, 모든 노드의 30036 포트로도 서비스를 접근할 수 있다.



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

Service Create - NodePort

앞에서 살펴본 ClusterIP와의 차이점을 잘 살펴보도록 하자.

svc-web-node.yaml

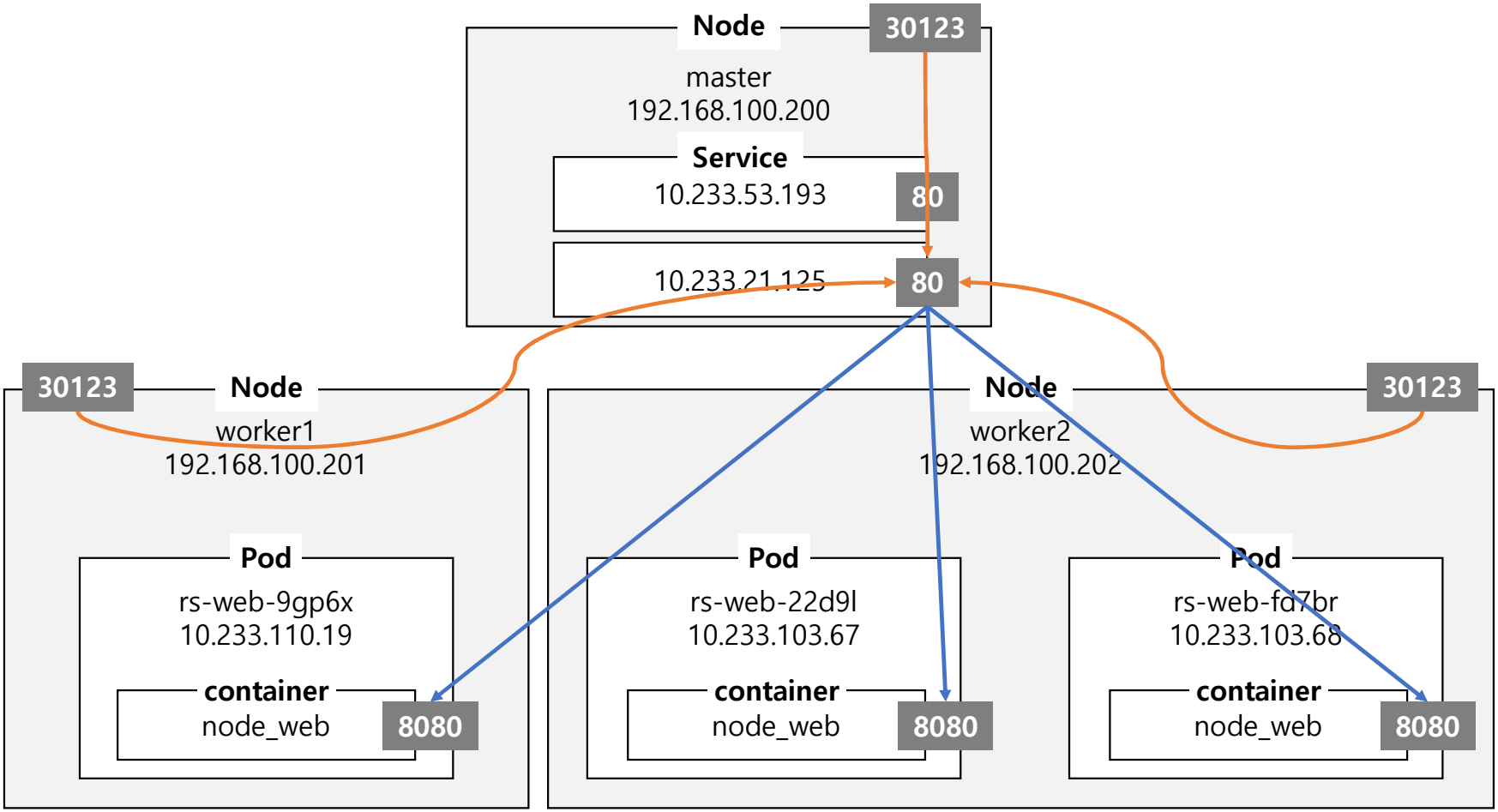
```
apiVersion: v1
kind: Service

metadata:
  name: svc-web-node

spec:
  type: NodePort

  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30123

  selector:
    app: node-web
```

```
remote ❯ cd kubernetes/04-ClusterIP-NodePort-ExternalName/hands-on

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

service/svc-web-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        20d   <none>
svc-web        ClusterIP   10.233.53.193   <none>        80/TCP         37m   app=node-web
svc-web-node   NodePort    10.233.21.125   <none>        80:30123/TCP   6s    app=node-web
```

바로 접근해볼 수 있다.

```
remote ❯ curl http://192.168.100.200:30123

You've hit rs-web-fd7br


remote ❯ curl http://192.168.100.201:30123

You've hit rs-web-22d9l
```

Now Status


Node
30123
master
192.168.100.200
Service
10.233.53.193
80
10.233.21.125
80
30123
Node
worker1
192.168.100.201
Node
worker2
192.168.100.202
30123
Pod
rs-web-9gp6x
10.233.110.19
container
node_web
8080
Pod
rs-web-22d9l
10.233.103.67
container
node_web
8080
Pod
rs-web-fd7br
10.233.103.68
container
node_web
8080

///

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, google.com. 을 입력할지, www.google.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 (옛날 이야기!?)

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

NAME                                   READY   STATUS    RESTARTS        AGE   IP               NODE      NOMINATED NODE   READINESS GATES
calico-kube-controllers-6dd874f784-rjxtz   1/1     Running   11 (32m ago)    20d   192.168.100.202   worker2   <none>           <none>
calico-node-599ck                      1/1     Running   8 (32m ago)     20d   192.168.100.200   master    <none>           <none>
calico-node-qlhvf                      1/1     Running   8 (32m ago)     20d   192.168.100.202   worker2   <none>           <none>
calico-node-tpwvg                      1/1     Running   18 (32m ago)    20d   192.168.100.201   worker1   <none>           <none>
coredns-76b4fb4578-kc7vm               1/1     Running   8 (32m ago)     20d   10.233.70.18      master    <none>           <none>
coredns-76b4fb4578-zbtvb               1/1     Running   7 (32m ago)     20d   10.233.110.20     worker1   <none>           <none>
dns-autoscaler-7979fb6659-p5fpm        1/1     Running   8 (32m ago)     20d   10.233.70.17      master    <none>           <none>
kube-apiserver-master                  1/1     Running   9 (32m ago)     20d   192.168.100.200   master    <none>           <none>
kube-controller-manager-master         1/1     Running   9 (32m ago)     20d   192.168.100.200   master    <none>           <none>
...
```

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

NAME                  READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS            IMAGES                                                  SELECTOR
calico-kube-controllers   1/1     1            1           20d   calico-kube-controllers   quay.io/calico/kube-controllers:v3.22.3                 k8s-app=calico-kube-controllers
coredns               2/2     2            2           20d   coredns               k8s.gcr.io/coredns/coredns:v1.8.6                       k8s-app=kube-dns
dns-autoscaler        1/1     1            1           20d   autoscaler            k8s.gcr.io/cpa/cluster-proportional-autoscaler-amd64:1.8.5 k8s-app=dns-autoscaler
```

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

[반복] Ready ... ReplicaSet

Service 가 필요한 Pod를 앞에서 공부했던 ReplicaSet을 통해 생성해 놓자

rs-web.yaml

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

metadata:
  name: rs-web

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 kubernetes/03-RS-DS-JOB-CJ/hands-on

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

replicaset.apps/rs-web created


remote ❯ kubectl get replicasets -o wide

NAME     DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                   SELECTOR
rs-web   3         3         3       2d    node-web     whatwant/node-web:1.0    app=node-web


remote ❯ kubectl get pods -o wide

NAME           READY   STATUS    RESTARTS       AGE     IP              NODE      NOMINATED NODE   READINESS GATES
rs-web-22d9l   1/1     Running   2 (114m ago)   2d23h   10.233.103.70   worker2   <none>           <none>
rs-web-9gp6x   1/1     Running   2 (115m ago)   2d23h   10.233.110.21   worker1   <none>           <none>
rs-web-fd7br   1/1     Running   2 (114m ago)   2d23h   10.233.103.69   worker2   <none>           <none>


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   19d   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>                 19d   v1.23.7   192.168.100.201   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
worker2  Ready    <none>                 19d   v1.23.7   192.168.100.202   <none>        Ubuntu 20.04.4 LTS   5.4.0-122-generic   containerd://1.6.4
```

[반복+New] Service Create - ClusterIP

우리가 원하는 것은 하나의 접점으로 Pod들 중 하나에 접근할 수 있도록 하는 것이다.

svc-web.yaml

```yaml
apiVersion: v1
kind: Service

metadata:
  name: svc-web

spec:
  ports:
  - port: 80
    targetPort: 8080

  selector:
    app: node-web
```

```
remote ❯ cd kubernetes/04-ClusterIP-NodePort-ExternalName/hands-on

remote ❯ kubectl create -f svc-web.yaml

service/svc-web 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   20d   <none>
svc-web      ClusterIP   10.233.53.193   <none>        80/TCP    21h   app=node-web
```

Pod를 이용해서 curl 실행을 해보자

```
remote ❯ kubectl exec -it rs-web-22d9l -- curl http://10.233.53.193

You've hit rs-web-22d9l


remote ❯ kubectl exec -it rs-web-22d9l -- curl http://10.233.53.193

You've hit rs-web-fd7br
```

FQDN in K8s

- Kubernetes는 각 container의 /etc/resolv.conf 파일을 직접 관리 (resolv.conf : 시스템의 DNS 설정)

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

```
remote ❯ kubectl exec -it rs-web-22d9l -- 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-web-22d9l -- curl -s http://svc-web

You've hit rs-web-fd7br

remote ❯ kubectl exec -it rs-web-22d9l -- curl -s http://svc-web.default

You've hit rs-web-fd7br

remote ❯ kubectl exec -it rs-web-22d9l -- curl -s http://svc-web.default.svc

You've hit rs-web-9gp6x

remote ❯ kubectl exec -it rs-web-22d9l -- curl -s http://svc-web.default.svc.cluster

command terminated with exit code 6

remote ❯ kubectl exec -it rs-web-22d9l -- curl -s http://svc-web.default.svc.cluster.local

You've hit rs-web-22d9l
```

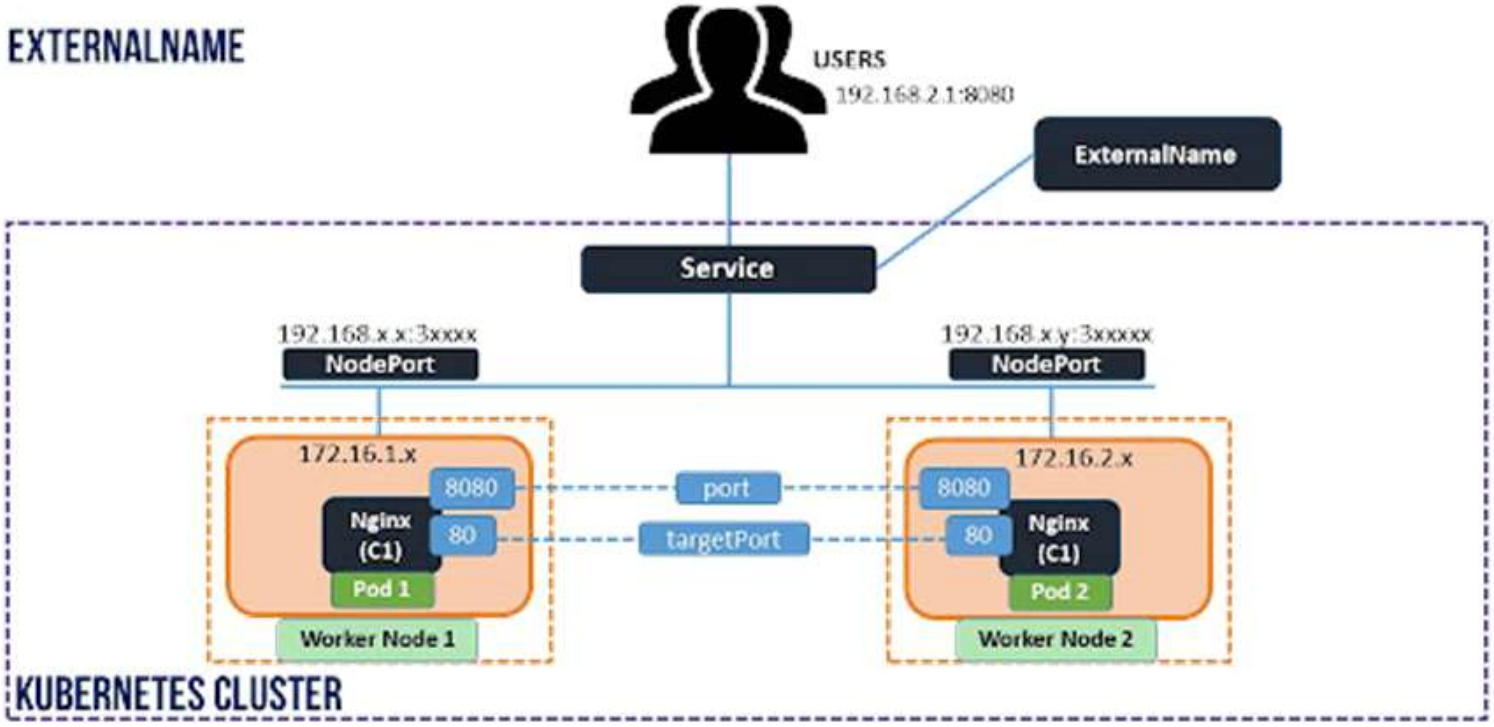
///

ExternalName

ExternalName

- 외부 서비스를 쿠버네티스 내부에서 호출하고자 할 때 사용할 수 있다.

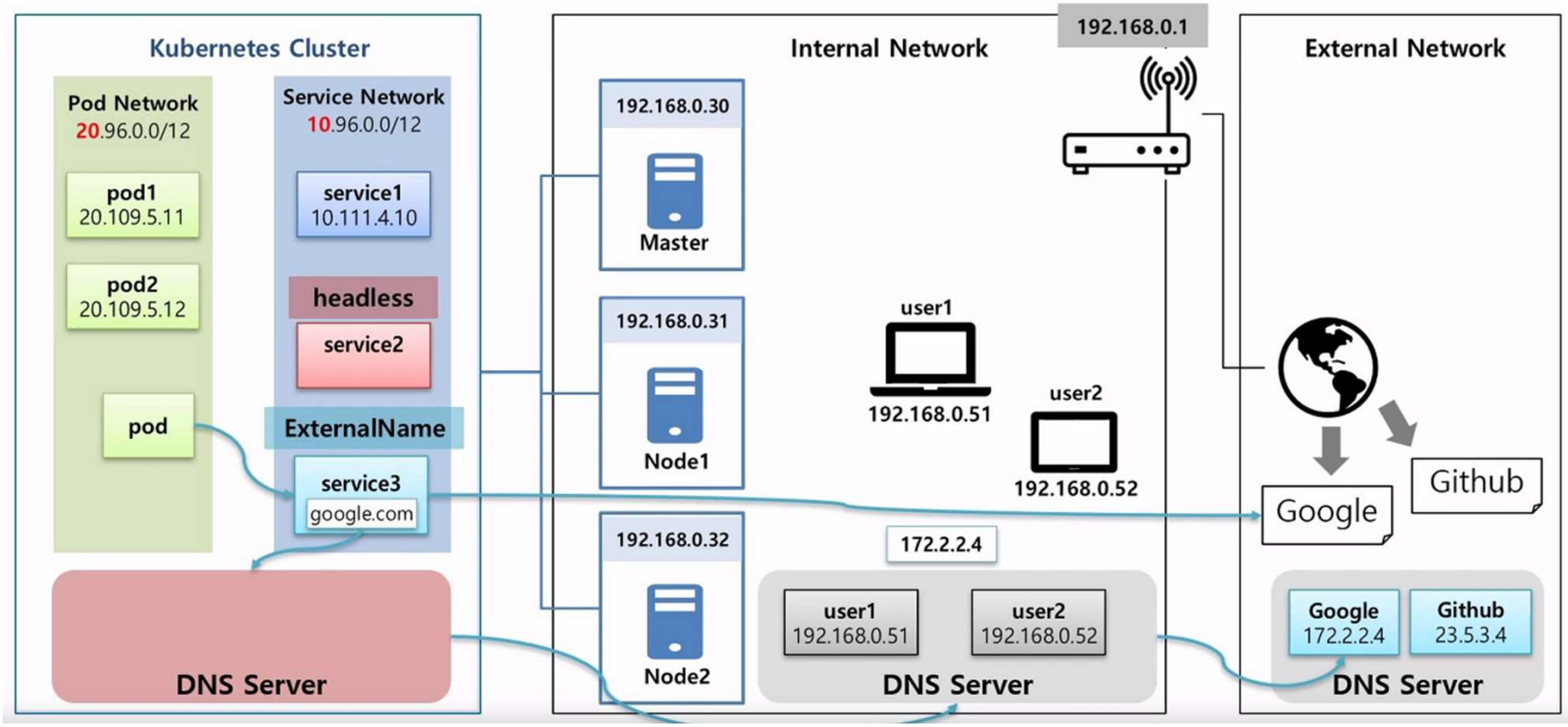
- Pod들은 ClusterIP를 가지고 있기 때문에 ClusterIP 대역 밖의 서비스를 호출하고자 할 때 사용하면 된다.



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

ExternalName - Recap 1/3

- ExternalName service3에 Google 도메인 이름을 넣으면 DNS를 타고 타서 Google 아이피를 가져올 수 있음
- 추후 Google을 GitHub으로 service3의 ExternalName만 변경하면 google이 아닌 GitHub 아이피를 가져올 수 있음



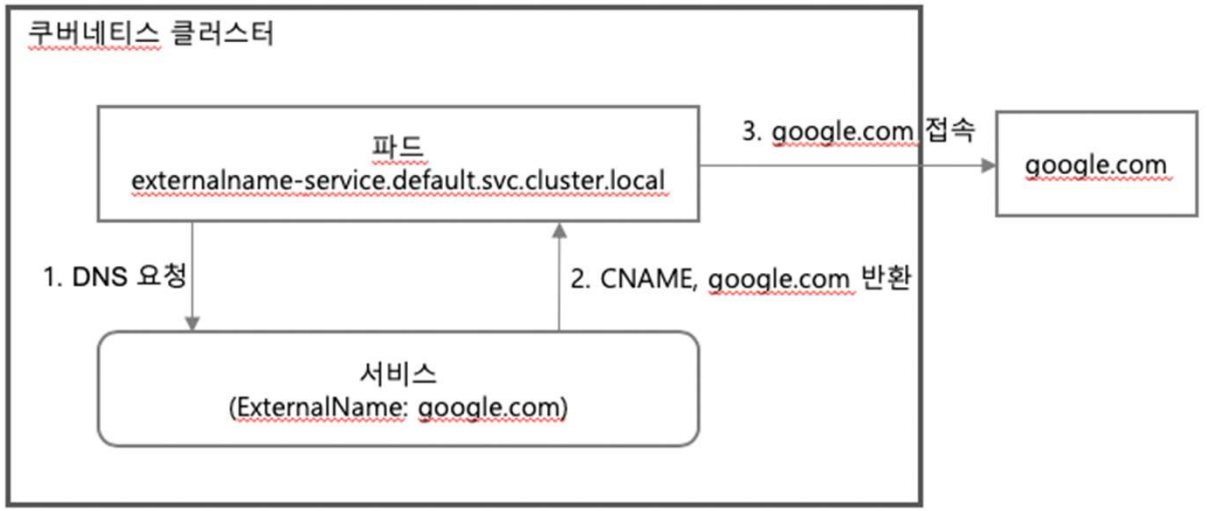
※ 참고 : https://junior-developer.tistory.com/74

ExternalName - Recap 2/3

- 외부 서비스의 별칭으로 사용되는 서비스
  . Pod는 서비스의 FQDN을 사용하는 대신 svc-ext.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

svc-ext.yaml

```
apiVersion: v1
kind: Service

metadata:
  name: svc-ext

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



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

ExternalName - Recap 3/3

- 일반적인 selector에 대한 Service가 아닌, DNS 이름에 대한 Service에 매핑

- ExternalName은 IPv4 주소 문자열 및 숫자로 구성된 DNS 이름도 허용
  . But, IPv4 주소와 유사한 ExternalName은 CoreDNS 또는 ingress-nginx에 의해 확인되지 않음
  . ExternalName은 정식(canonical) DNS 이름을 지정하기 때문 (Proxy, Forwarding이 아닌 DNS라는 점이 중요!!!)
  . IP 주소를 하드 코딩하려면, 헤드리스(headless) 서비스 사용 권장

- HTTP 및 HTTPS를 포함한, 몇몇 일반적인 프로토콜에 ExternalName을 사용하는 것은 문제점 존재
  . 클러스터 내부의 클라이언트가 사용하는 호스트 이름(hostname)이 ExternalName이 참조하는 이름과 다름
  . 호스트 이름을 사용하는 프로토콜의 경우, 이러한 차이로 인해 오류가 발생하거나 예기치 않은 응답이 발생할 수 있음
  . HTTP 요청에는 오리진(origin) 서버가 인식하지 못하는 `Host :` 헤더가 존재
  . TLS 서버는 클라이언트가 연결된 호스트 이름과 일치하는 인증서를 제공할 수 없음

※ 참고 : https://kubernetes.io/ko/docs/concepts/services-networking/service/#externalname

///

Endpoints

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-web-22d9l   1/1     Running   2 (134m ago)   2d23h   10.233.103.70   worker2   <none>           <none>
rs-web-9gp6x   1/1     Running   2 (134m ago)   2d23h   10.233.110.21   worker1   <none>           <none>
rs-web-fd7br   1/1     Running   2 (134m ago)   2d23h   10.233.103.69   worker2   <none>           <none>


remote ❯ kubectl describe service svc-web

Name:              svc-web
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=node-web
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.233.53.193
IPs:               10.233.53.193
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         10.233.103.69:8080,10.233.103.70:8080,10.233.110.21:8080
Session Affinity:  None
Events:            <none>
```

svc-web.yaml

```yaml
apiVersion: v1
kind: Service

metadata:
  name: svc-web

spec:
  ports:
  - port: 80
    targetPort: 8080

  selector:
    app: node-web
```

Endpoints

- Endpoint 리소스도 만들 수 있을까?

  . Selectors를 지정하지 않고, 직접 생성한 Endpoints 활용하기

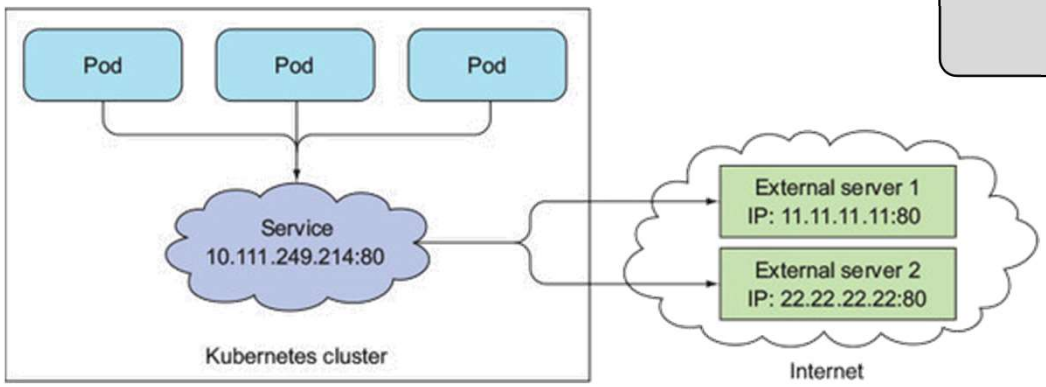
  . Service와 Endpoints 명칭 일치 필요

svc-no-selector.yaml

```
apiVersion: v1
kind: Service

metadata:
  name: example-endpoints

spec:
  ports:
  - port: 80
```

endpoints.yaml

```
apiVersion: v1
kind: Endpoints

metadata:
  name: example-endpoints

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

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 ❯ cd kubernetes/04-ClusterIP-NodePort-ExternalName/hands-on

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        21d   <none>
svc-node-web-headless   ClusterIP   None            <none>        80/TCP         14s   app=node-web
svc-web                 ClusterIP   10.233.53.193   <none>        80/TCP         24h   app=node-web
svc-web-node            NodePort    10.233.21.125   <none>        80:30123/TCP   23h   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:    10.233.110.21
Name:       svc-node-web-headless.default.svc.cluster.local
Address:    10.233.103.70
Name:       svc-node-web-headless.default.svc.cluster.local
Address:    10.233.103.69


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

Server:     169.254.25.10
Address:    169.254.25.10#53

Name:       svc-web.default.svc.cluster.local
Address:    10.233.53.193
```

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
```

///

자습 (복습)

▷ Service - ClusterIP / NodePort / ExternalName

▷ Service - DNS / FDQN / Endpoints / Headless