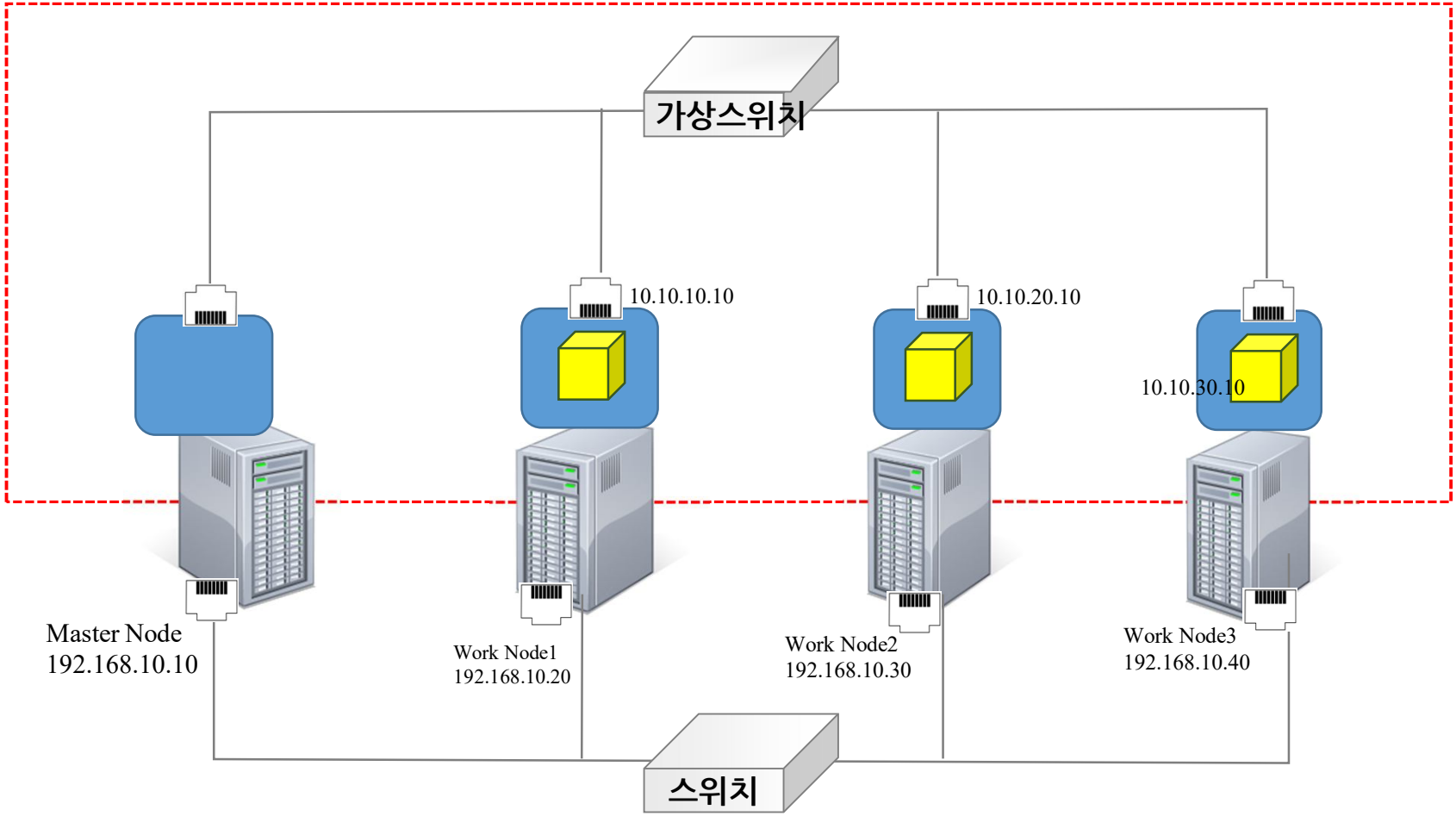
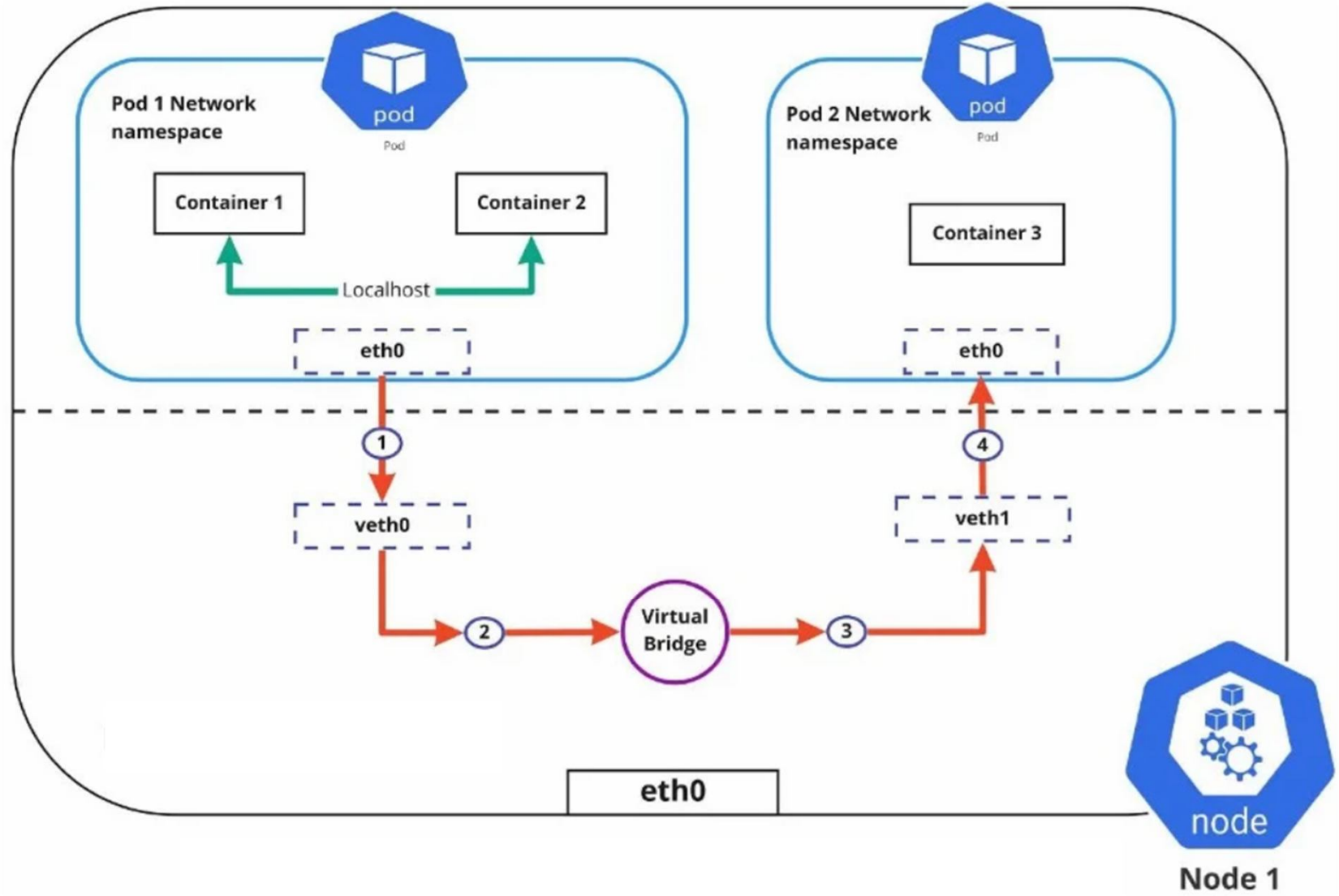
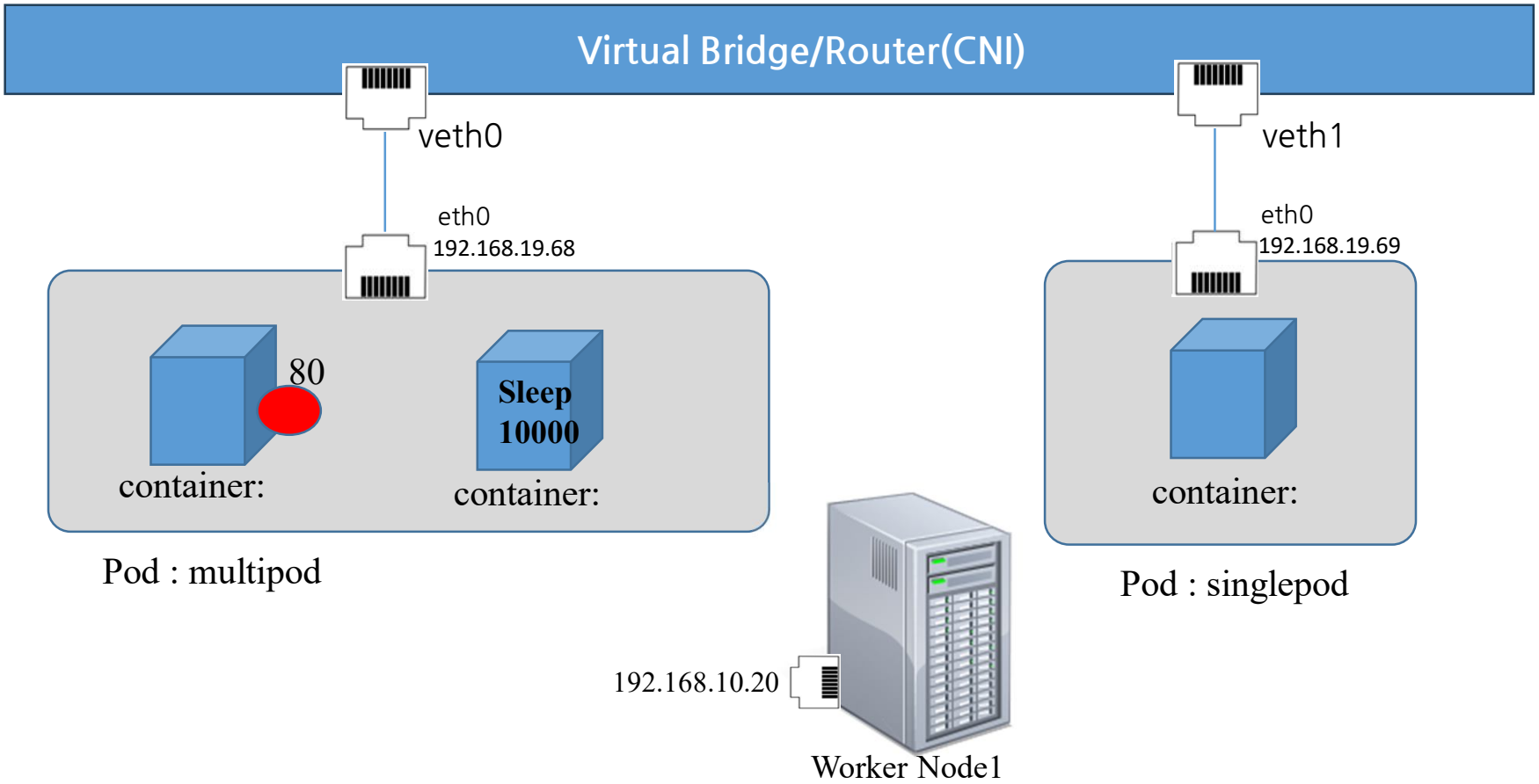
CNI (Container Network Interface)

- 파드 네트워킹을 관리하고 노드 간 통신을 가능하게 함
- 파드에 IP 할당, 네트워크 플러그인 규격 준수, 클러스터 네트워크 구성
- 파드 생성 시 네트워크 설정, IP 주소 할당, 파드 간 통신 경로 설정 (라우팅) 등 파드 네트워킹의 기본 기능 구현


가상스위치
10.10.10.10
10.10.20.10
10.10.30.10
Master Node
192.168.10.10
Work Node1
192.168.10.20
Work Node2
192.168.10.30
Work Node3
192.168.10.40
스위치


Pod 1 Network namespace
pod
Pod
Container 1
Container 2
Localhost
eth0
Pod 2 Network namespace
pod
Pod
Container 3
eth0
1
4
veth0
veth1
2
Virtual Bridge
3
eth0
node
Node 1

Pod 생성 → CNI 호출 → veth 생성 → Pod에 IP 할당 → 다른 Pod들과 연결



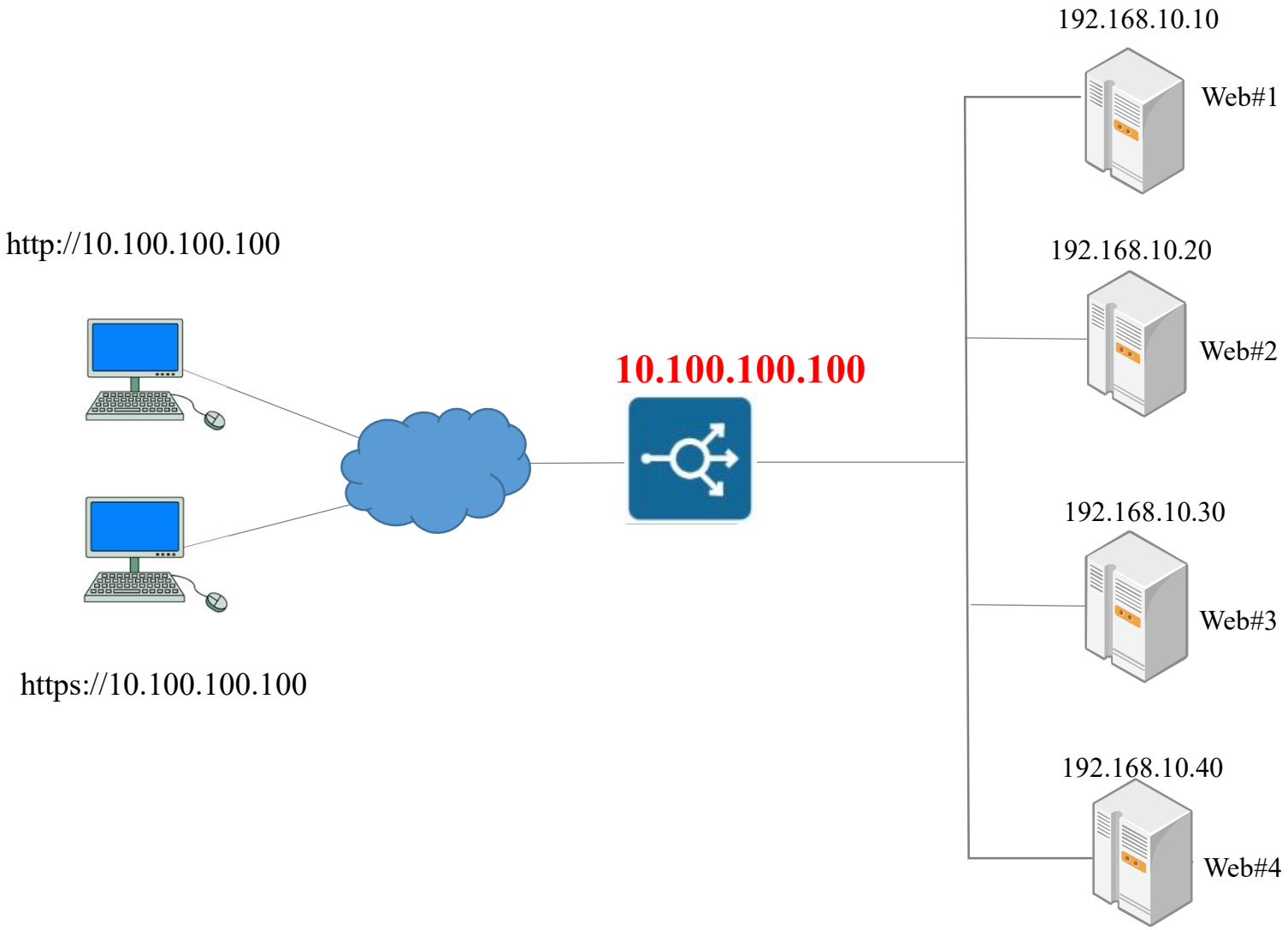
Service

서비스 개념

- 여러 개의 파드에 접근 할 수 있는 하나의 IP제공
  - 파드는 컨트롤러가 관리하므로 한 군데에 고정해서 실행되지 않음
  - 클러스터 안의 노드를 옮기면서 실행
  - 클러스터 안 파드의 IP가 변경되기도 함
  - 동적으로 변하는 파드들에 고정적으로 접근 할 때 사용하는 방법
- 서비스를 사용하면 파드가 클러스터 안 어디에 있든 고정주소를 이용해서 접근
- 클러스터 외부에서 클러스터 안 파드에 접근 할 수 있음
- 로드 밸런서 역할
  - L4영역에서 통신할 때 사용

서비스 타입

ClusterIP	• 기본 서비스 타입(default) • 클러스터 안에서만 사용할 수 있음 • ClusterIP를 이용해서 서비스에 연결된 파드에 접근
NodePort	• 모든 노드에 지정된 포트 할당 • 서비스에 지정된 포트 번호를 사용하여 파드에 접근 • 클러스터 내/외부에서도 접근 • ClusterIP가 생성된 후 모든 Worker node에 외부에서 접속 가능 한 포트가 예약
LoadBalancer	• 퍼브릭 클라우드와 프라이빗 클라우드, K8S에서 지원하는 로드밸런서 장비 사용 • 클라우드에서 제공하는 로드밸런서와 파드를 연결 • 로드밸런서 IP를 이용해서 파드에 접근
ExternalName	• 클러스터 안에서 외부에 접근 할 때 사용


192.168.10.10
Web#1
http://10.100.100.100
192.168.10.20
Web#2
10.100.100.100
192.168.10.30
Web#3
https://10.100.100.100
192.168.10.40
Web#4

1) CluseterIP Type Service

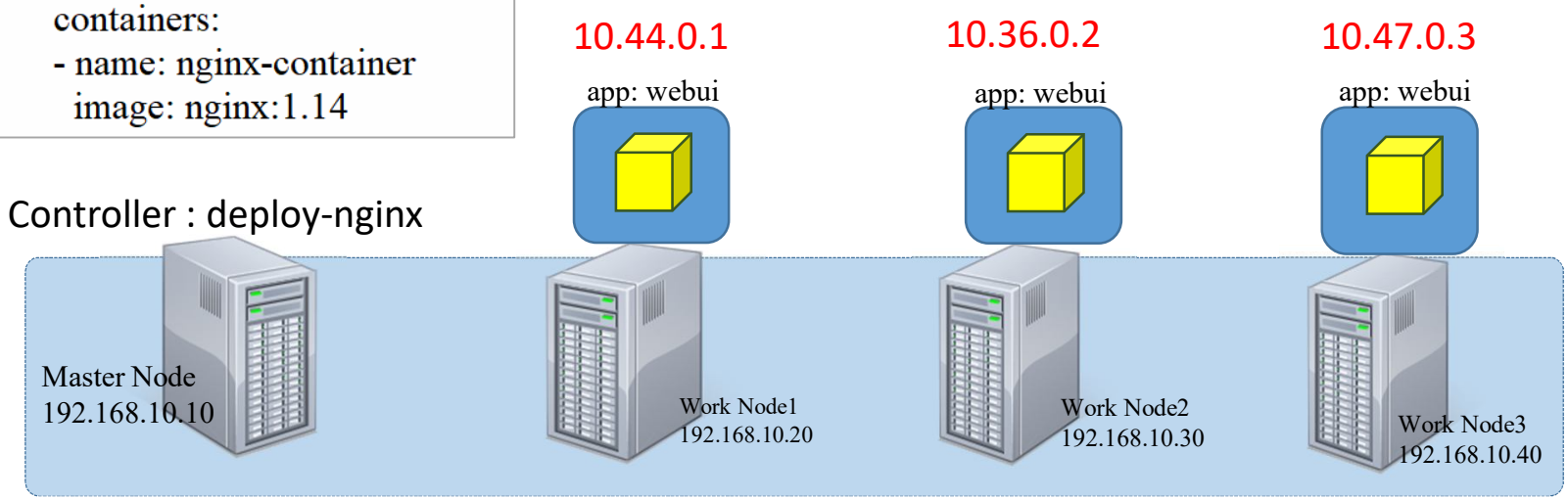
- Selector의 label이 동일한 파드들을 그룹으로 묶음

```
apiVersion: v1
kind: Service
metadata:
  name: webui-svc
spec:
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

- 묶여진 그룹에 ClusterIP(Virtual IP) 주소 생성
- ClusterIP를 이용해서 서비스에 연결된 파드에 접근
- Type 생략 시 default로 10.96.0.0/12이 할당
- 클러스터 내에서만 사용할 수 있음

```
apiVersion: v1
kind: Service
metadata:
  name: clusterip-service
spec:
  type: ClusterIP
  clusterIP: 10.100.100.100
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webui
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webui
  template:
    metadata:
      name: nginx-pod
      labels:
        app: webui
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.14
```

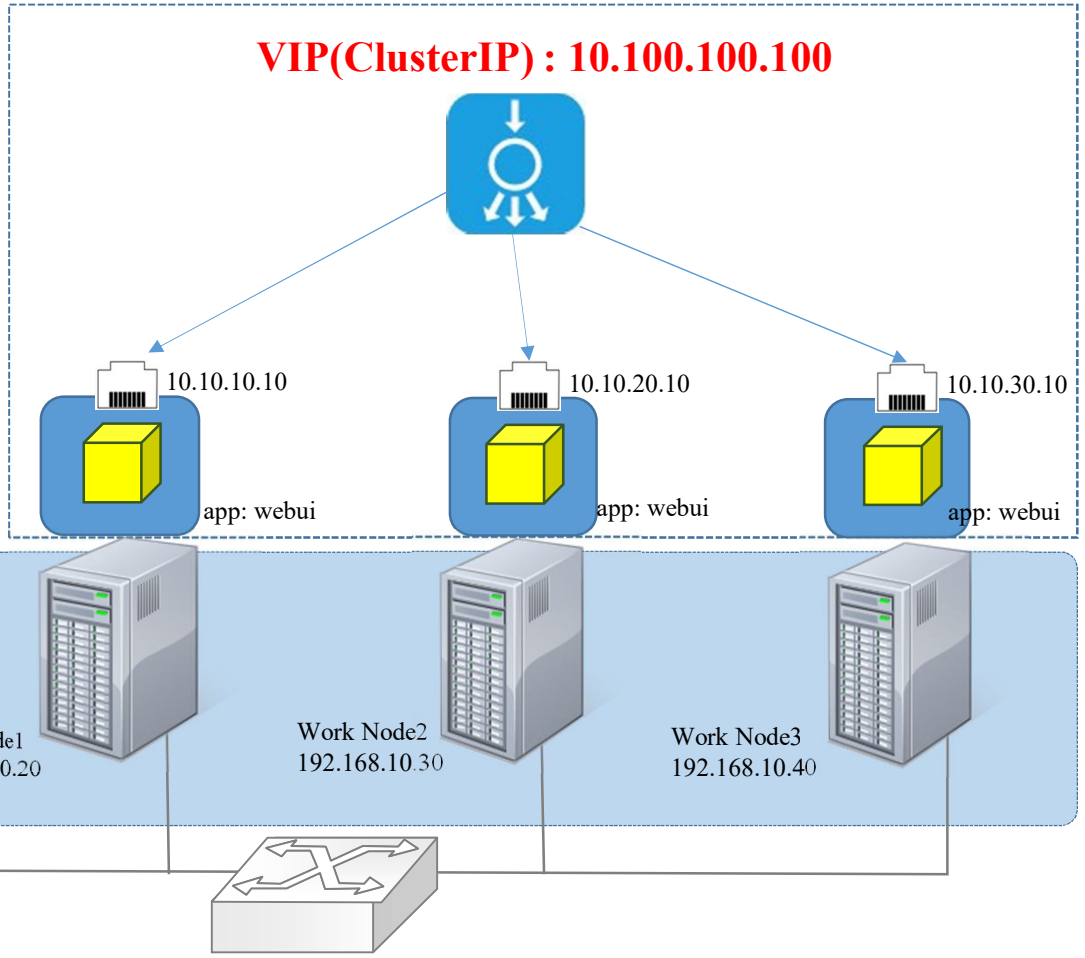


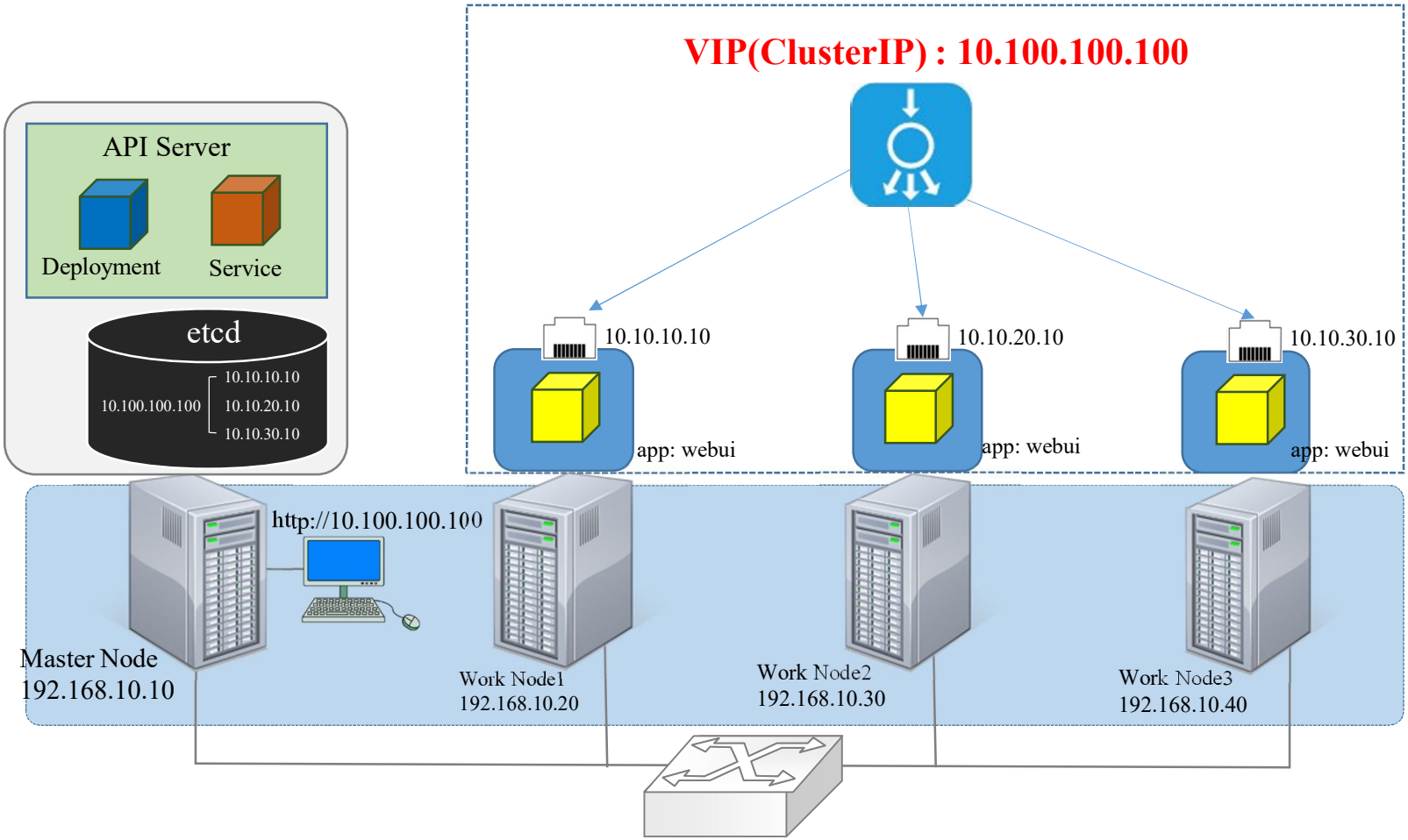
```
apiVersion: v1
kind: Service
metadata:
  name: clusterip-service
spec:
  type: ClusterIP
  clusterIP: 10.100.100.100
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

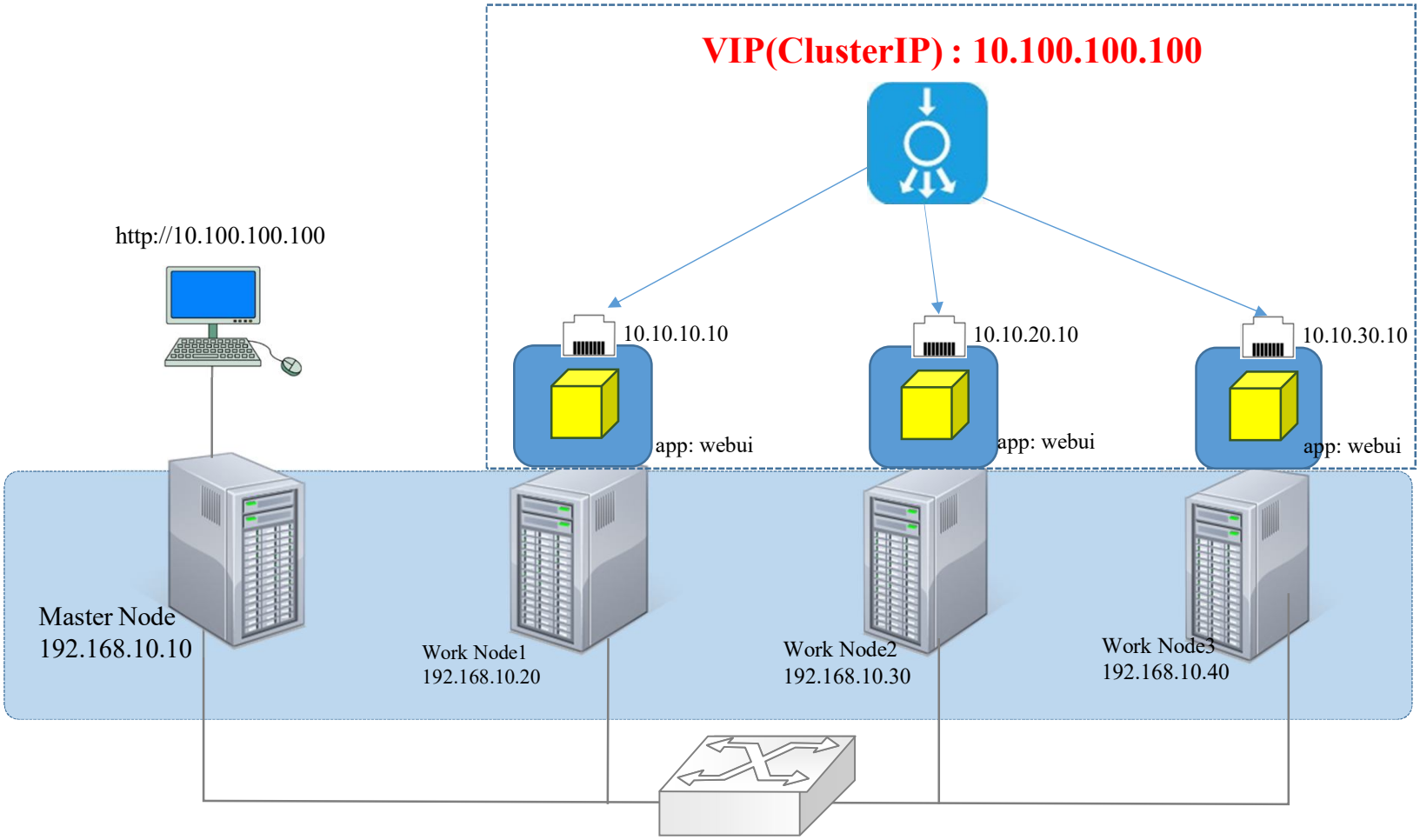
Service : clusterip-service

Controller : deploy-nginx

(ex) webui 파드들을 하나의 IP로 묶어서 관리




VIP(ClusterIP) : 10.100.100.100
API Server
Deployment
Service
etcd
10.100.100.100
10.10.10.10
10.10.20.10
10.10.30.10
10.10.10.10
10.10.20.10
10.10.30.10
app: webui
app: webui
app: webui
http://10.100.100.100
Master Node
192.168.10.10
Work Node1
192.168.10.20
Work Node2
192.168.10.30
Work Node3
192.168.10.40


VIP(ClusterIP) : 10.100.100.100
http://10.100.100.100
10.10.10.10
10.10.20.10
10.10.30.10
app: webui
app: webui
app: webui
Master Node
192.168.10.10
Work Node1
192.168.10.20
Work Node2
192.168.10.30
Work Node3
192.168.10.40

ClusterIP LAB(1)

kubectl create -f deploy-nginx.yaml

kubectl get pod -o wide

kubectl create -f clusterip-nginx.yaml

kubectl get service

kubectl describe svc clusterip-service

kubectl descirbe service clusterip-service

<<clusterip-nginx.yaml>>

```
apiVersion: v1
kind: Service
metadata:
  name: clusterip-service
spec:
  type: clusterIP
  clusterIP: 10.100.100.100
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

kubectl get service

kubectl describe svc clusterip-service

```
[node1 ~]$
[node1 ~]$ kubectl get service
NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
clusterip-service    ClusterIP   10.100.100.100   <none>        80/TCP    62s
kubernetes           ClusterIP   10.96.0.1        <none>        443/TCP   7m50s
[node1 ~]$
[node1 ~]$ kubectl describe svc clusterip-service
Name:              clusterip-service
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=webui
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.100.100
IPs:               10.100.100.100
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.5.1.2:80,10.5.2.2:80,10.5.3.2:80
Session Affinity:  None
Events:            <none>
```

ClusterIP LAB(2)

kubectl get pod –o wide
deploy-~~
curl 10.100.100.100

kubectl exec ***deploy-~~*** -it -- /bin/bash
cd /usr/share/nginx/html
ls
echo "webui #1" > index.html
exit

kubectl exec ***deploy-~~*** -it -- /bin/bash
cd /usr/share/nginx/html
ls
echo "webui #2" > index.html
exit

<<master에서 확인>>

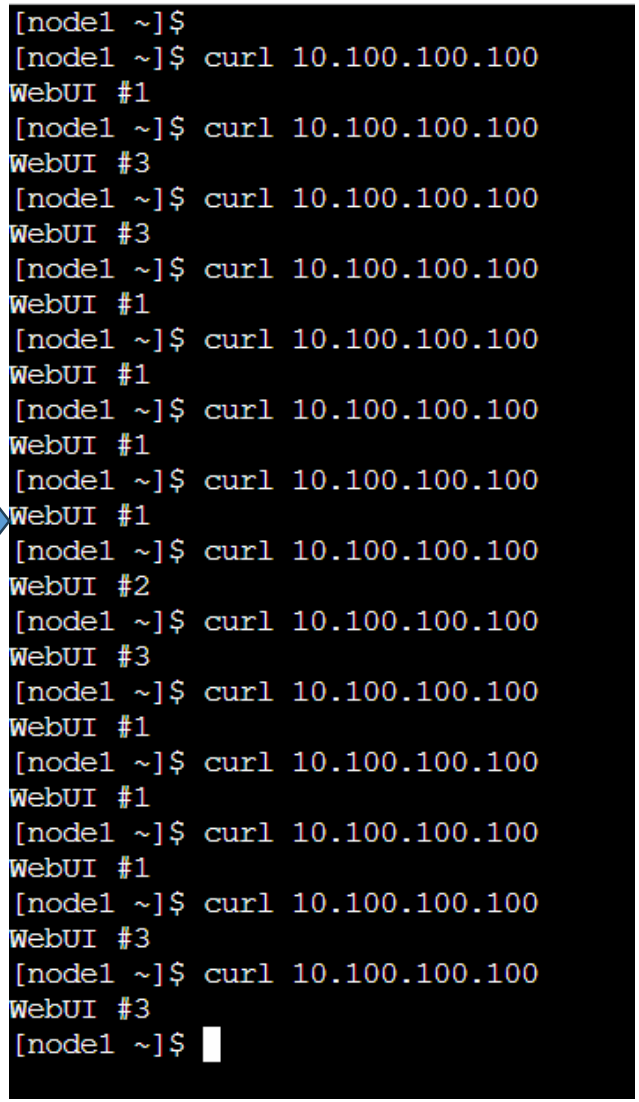
curl 10.100.100.100

curl 10.100.100.100

curl 10.100.100.100

curl 10.100.100.100

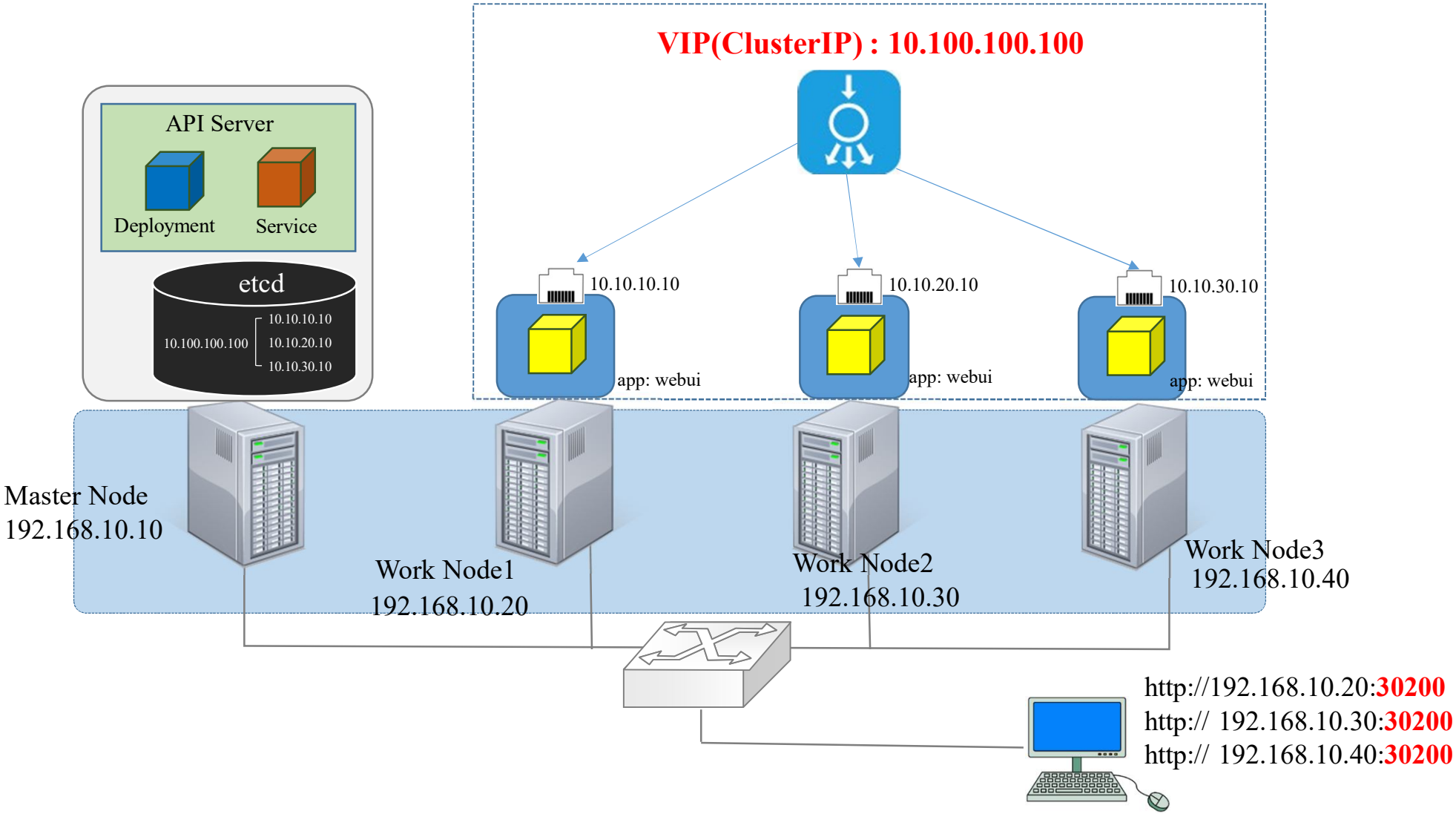
curl 10.100.100.100



2) NodePort Type 서비스 사용하기

- 모든 노드를 대상으로 외부 접속이 가능한 포트를 생성
- Default NodePort 범위 : 30000~32767
- ClusterIP를 생성 후 NodePort를 예약

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-service
spec:
  type: NodePort
  clusterIP: 10.100.100.100
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30200
```


VIP(ClusterIP) : 10.100.100.100
API Server
Deployment
Service
etcd
10.100.100.100
10.10.10.10
10.10.20.10
10.10.30.10
10.10.10.10
10.10.20.10
10.10.30.10
app: webui
app: webui
app: webui
Master Node
192.168.10.10
Work Node1
192.168.10.20
Work Node2
192.168.10.30
Work Node3
192.168.10.40
http://192.168.10.20:30200
http:// 192.168.10.30:30200
http:// 192.168.10.40:30200

NodePort LAB

<<master node>>

kubectl delete service --all

kubectl create -f nodeport-nginx.yaml

kubectl get svc

<<client에서 접속>>

curl 192.168.10.20:30200

curl 192.168.10.30:30200

curl 192.168.10.40:30200

<<크롬에서 접속>>

http://192.168.10.20:30200

http://192.168.10.30:30200

http://192.168.10.40:30200

<<nodeport-nginx.yaml>>

```
apiVersion: v1
kind: Service
metadata:
  name: nodeport-service
spec:
  type: NodePort
  clusterIP: 10.100.100.100
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30200
```

kubectl create -f nodeport-nginx.yaml

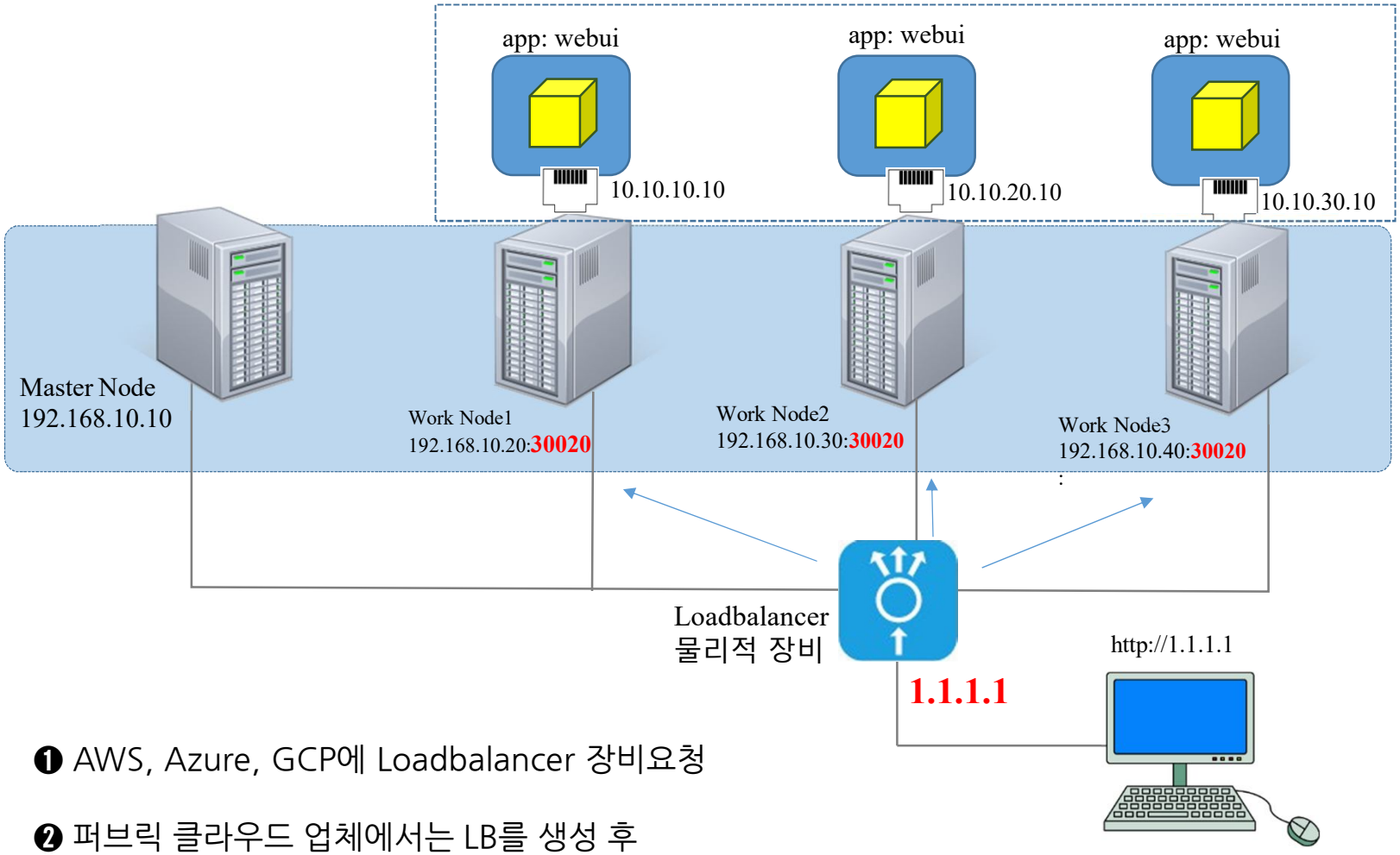
kubectl get svc

```
[node1 ~]$ kubectl create -f nodeport-nginx.yaml
service/nodeport-service created
[node1 ~]$
[node1 ~]$ kubectl get svc
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP          2m2s
nodeport-service    NodePort    10.100.100.100   <none>        80:30200/TCP     21s
[node1 ~]$
```

3) LoadBalancer Type 서비스 사용하기

- Public 클라우드(AWS, Azure, GCP 등)에서 운영가능
- LoadBalancer를 자동으로 구성요청
- NodePort를 예약한 후 해당 NodePort로 외부접근을 허용

```
apiVersion: v1
kind: Service
metadata:
  name: loadbalancer-service
spec:
  type: LoadBalancer
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```



❶ AWS, Azure, GCP에 Loadbalancer 장비요청

❷ 퍼브릭 클라우드 업체에서는 LB를 생성 후

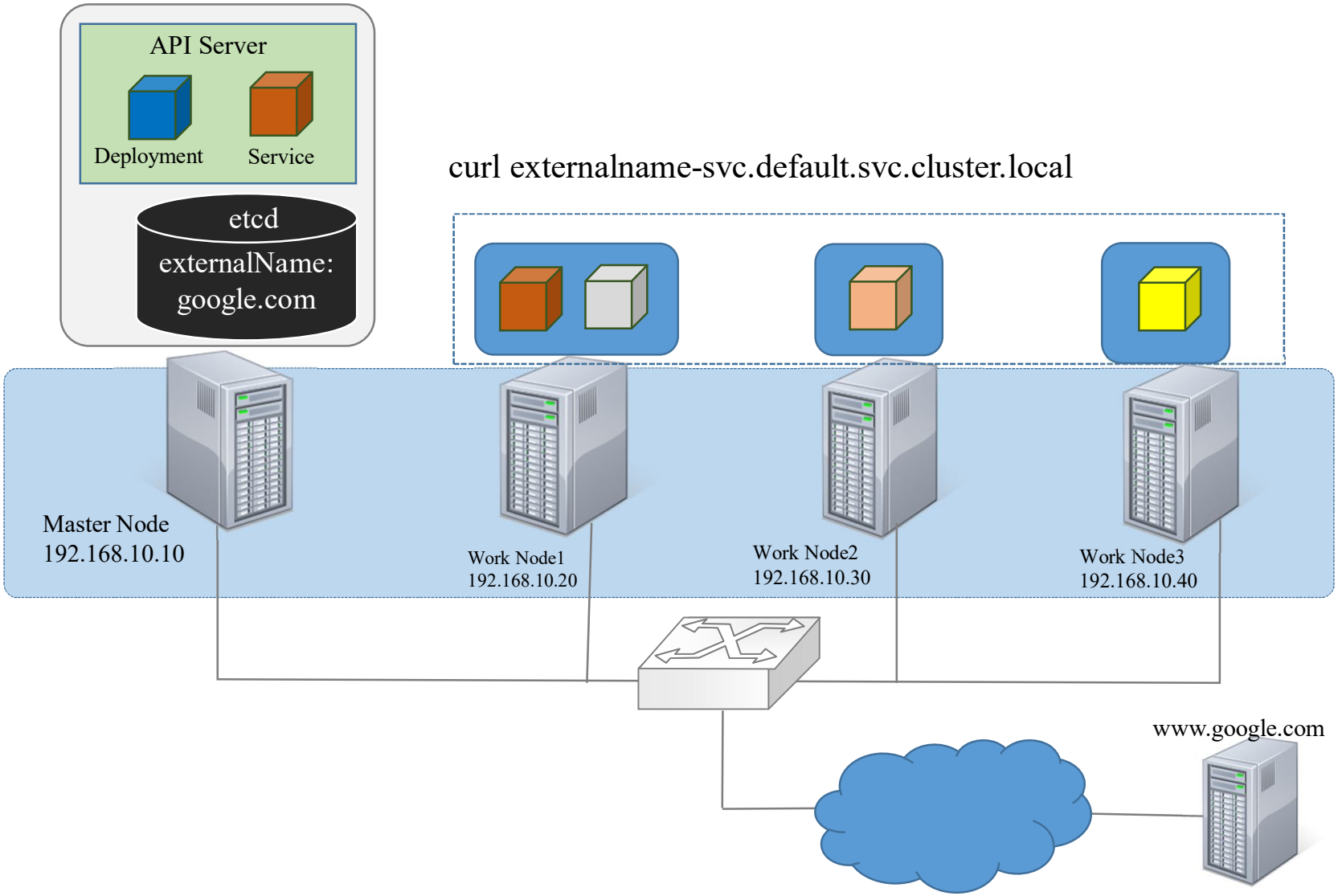
❸ LB의 IP를 서비스에 제공

❹ 제공된 IP주소와 NodePort가 매칭되어 트래픽 로드밸런싱이 진행

4) ExternalName Type 서비스 사용하기

- 클러스터 내부에서 외부의 도메인을 설정

```
apiVersion: v1
kind: Service
metadata:
  name: externalname-svc
spec:
  type: ExternalName
  externalName: google.com
```


API Server
Deployment
Service
etcd
externalName:
google.com
curl externalname-svc.default.svc.cluster.local
Master Node
192.168.10.10
Work Node1
192.168.10.20
Work Node2
192.168.10.30
Work Node3
192.168.10.40
www.google.com

```
apiVersion: v1
kind: Service
metadata:
  name: externalname-svc
spec:
  type: ExternalName
  externalName: google.com
```

```
[node1 ~]$
[node1 ~]$ kubectl create -f external-name.yaml
service/externalname-svc created
[node1 ~]$ kubectl get svc
NAME               TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
externalname-svc   ExternalName   <none>           google.com    <none>           16s
kubernetes         ClusterIP      10.96.0.1        <none>        443/TCP          16m
nodeport-service   NodePort       10.100.100.100   <none>        80:30200/TCP     14m
[node1 ~]$
[node1 ~]$ kubectl run testpod -it --image=centos:7
```

curl externalname-svc. default.svc.cluster.local

K8S에서 사용하는 도메인명