Network

논리적 주소(3계층주소)

- IP address 구성

  Network ID + Host ID

- Subnet Mask 기능

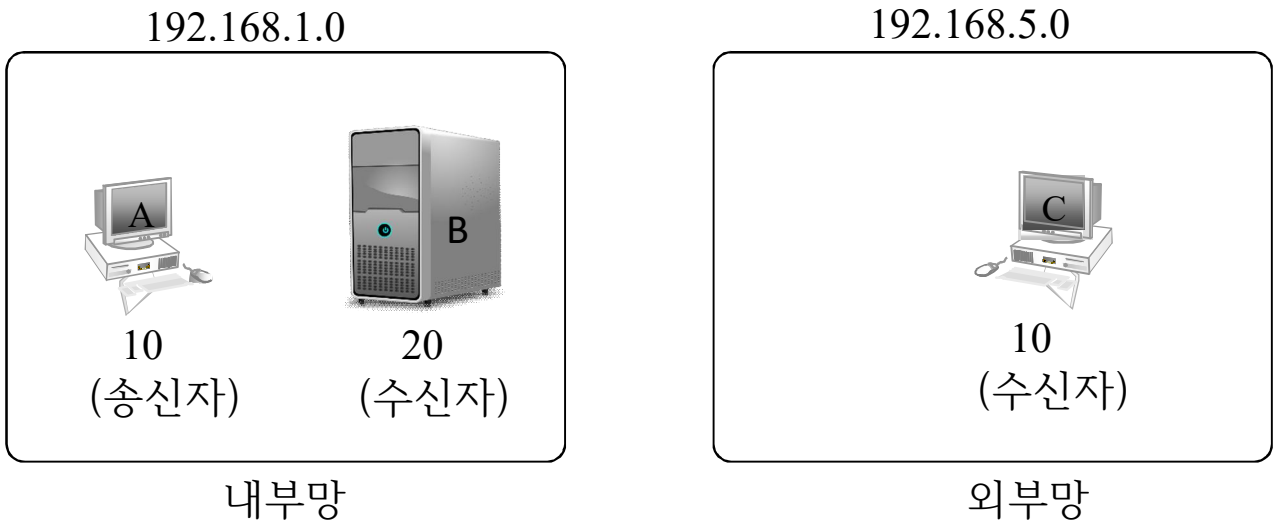
  - IP address의 Network ID와 Host ID 구분

IP address	192.168.1.10
& Subnet mask	& 255.255.255.0
Network ID	192.168.1. 0

255==1
& == X

내부망과 외부망

- 송신자 : 데이터를 보내는 측 / 수신자 : 데이터를 받는 측

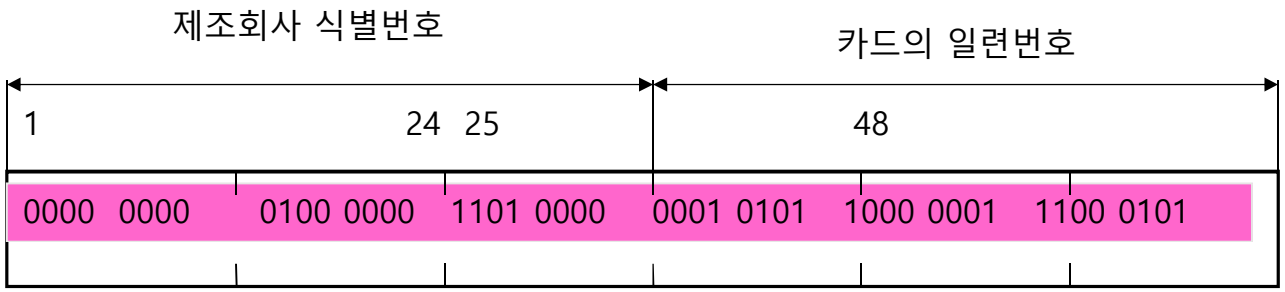


- 내부망 : 송수신자가 동일한 네트워크 ID를 사용
- 외부망 : 송수신지가 서로 다른 네트워크 ID를 사용

물리적 주소(2계층주소)

- Network Interface Card (NIC) 또는 Ethernet Card
- 데이터링크계층의 MAC 계층에 의해 사용되는 48비트의 하드웨어 주소

MAC 주소(16진수 표현) : 00-40-D0-15-81-C5



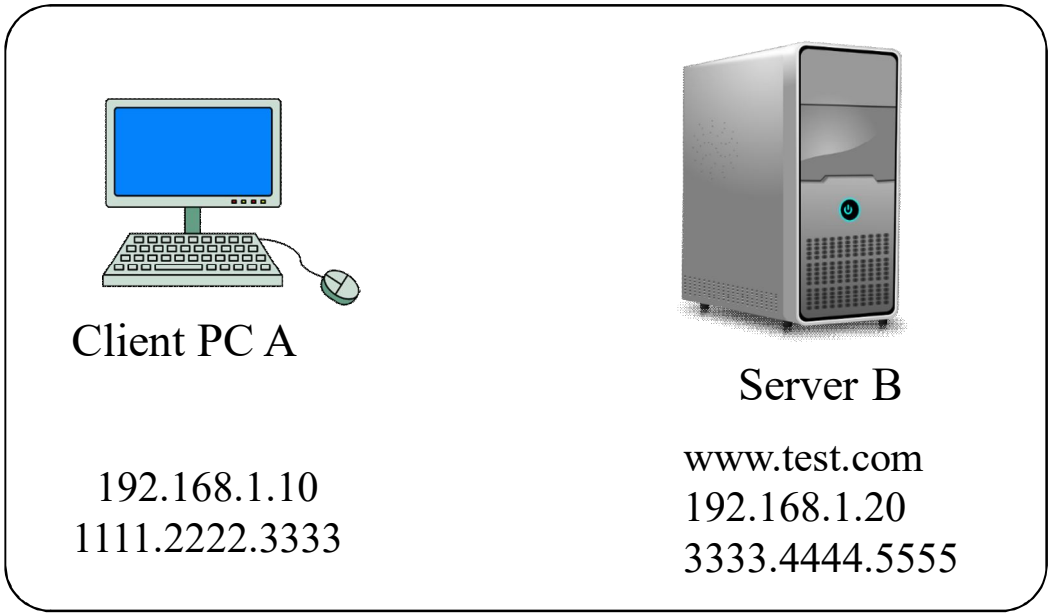
대표적인 제조회사 식별 번호의 예

- Intel: 00-A0-C9
- 3Com: 00-50-DA
- Realtek: 00-40-D0

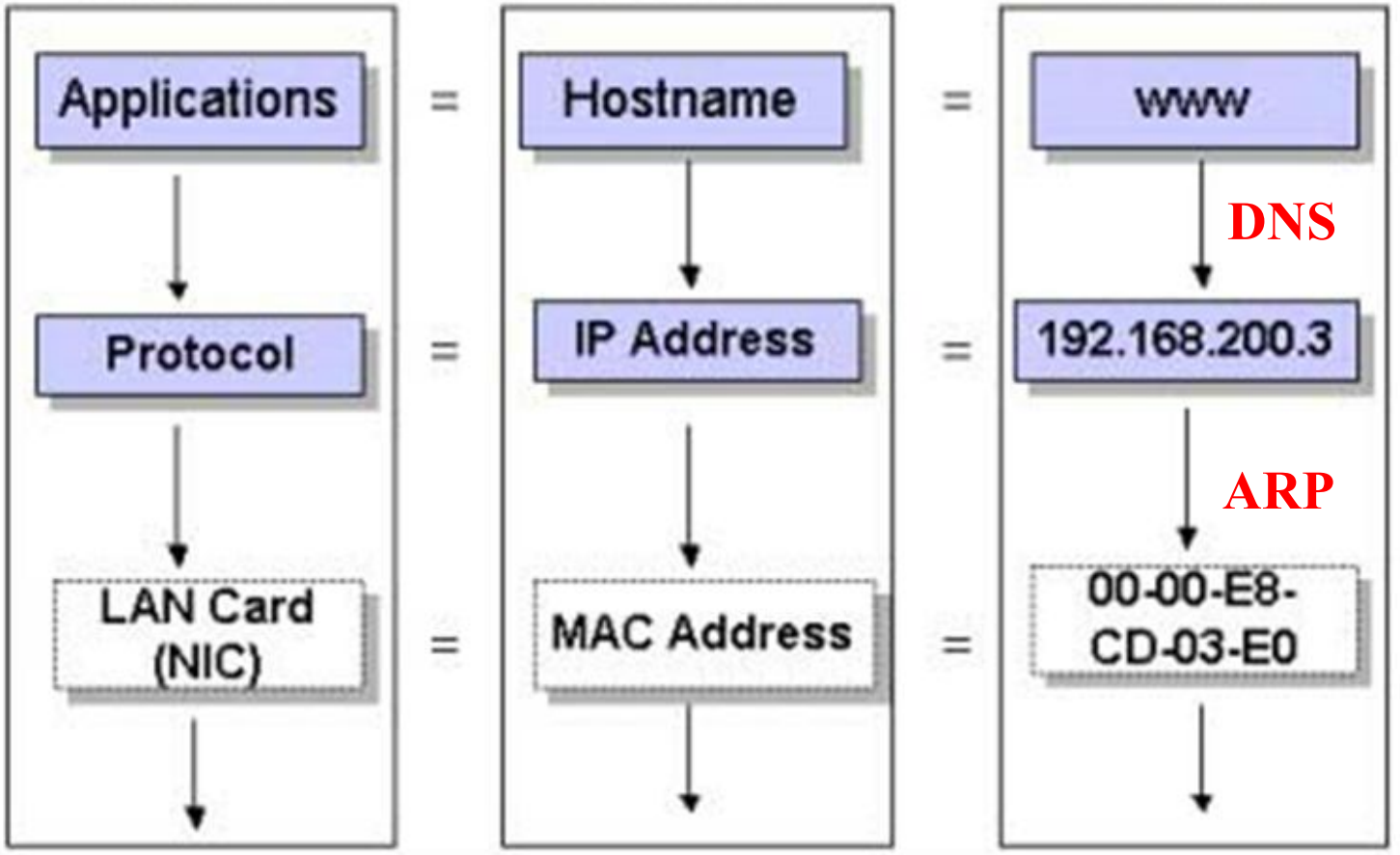
MAC/IP/FQDN Address

- MAC 주소 구성=제조회사+일련번호
- IP 주소 구성=네트워크ID+호스트ID
- FQDN 주소 구성=호스트명+도메인명

그룹주소+고유번호(명)



DNS & ARP

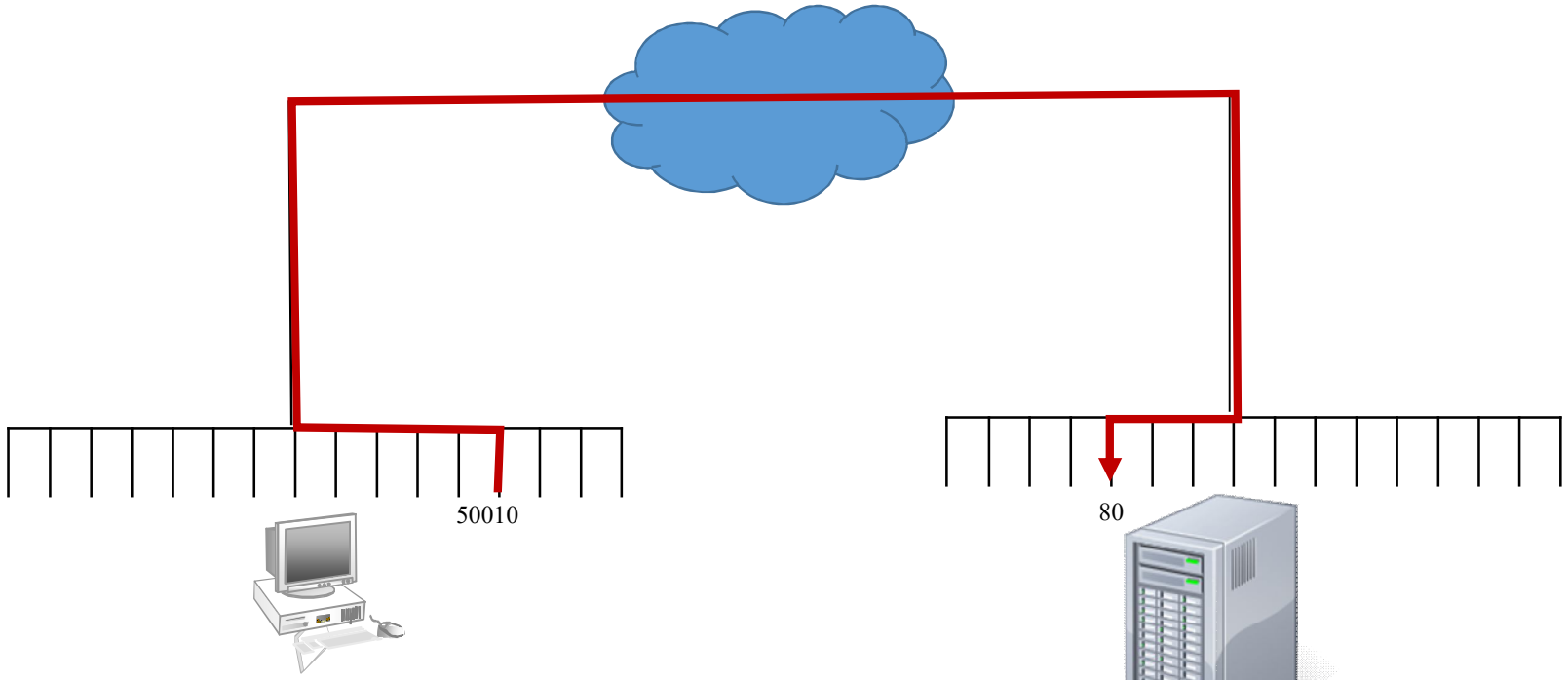

Applications
=
Hostname
=
www
DNS
Protocol
=
IP Address
=
192.168.200.3
ARP
LAN Card
(NIC)
=
MAC Address
=
00-00-E8-
CD-03-E0

포트번호(4계층 주소)

- 데이터 송수신 번호
- 서비스 번호 또는 애플리케이션 번호
- 애플리케이션에서 부착해 전송

  - Well-Known Port : 1-1023
  - Registered Port : 1024-49151
  - Dynamic Port : 49152-65535

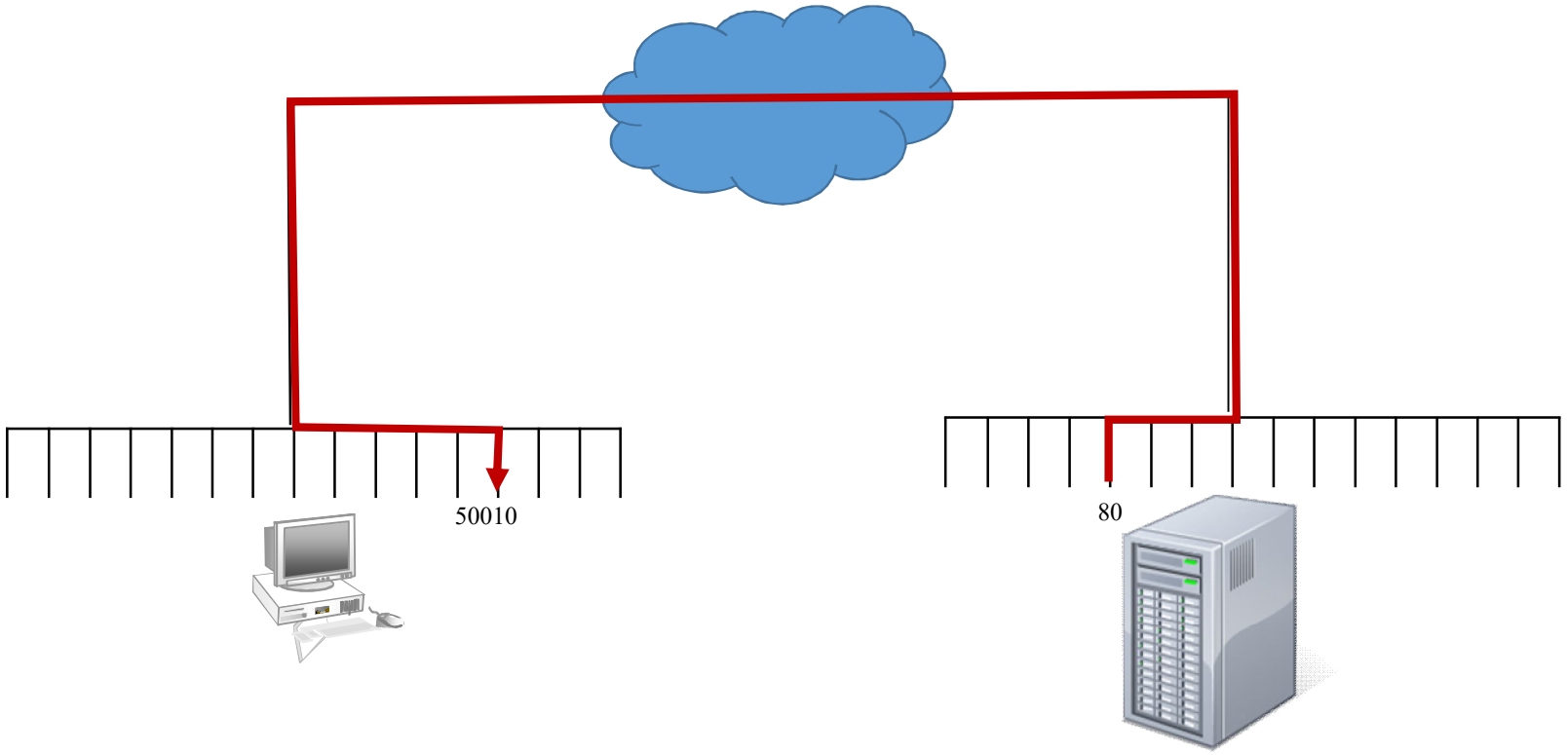
포트주소 (송신)



송신지 192.168.10.10 : 50010
수신지 200.10.10.10 : 80

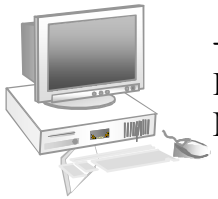
Web Server
200.10.10.10

포트주소(수신)



송신지 200.10.10.10 : 80
수신지 192.168.10.10 : 50010

Unicast 전송모드

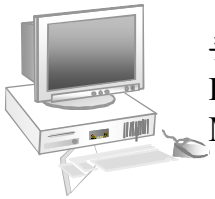


클라이언트(송신지)
IP 192.168.1.10
MAC 1111.2222.1111

서버(수신지)
www.test.com
IP 192.168.1.20
MAC 1111.2222.2222

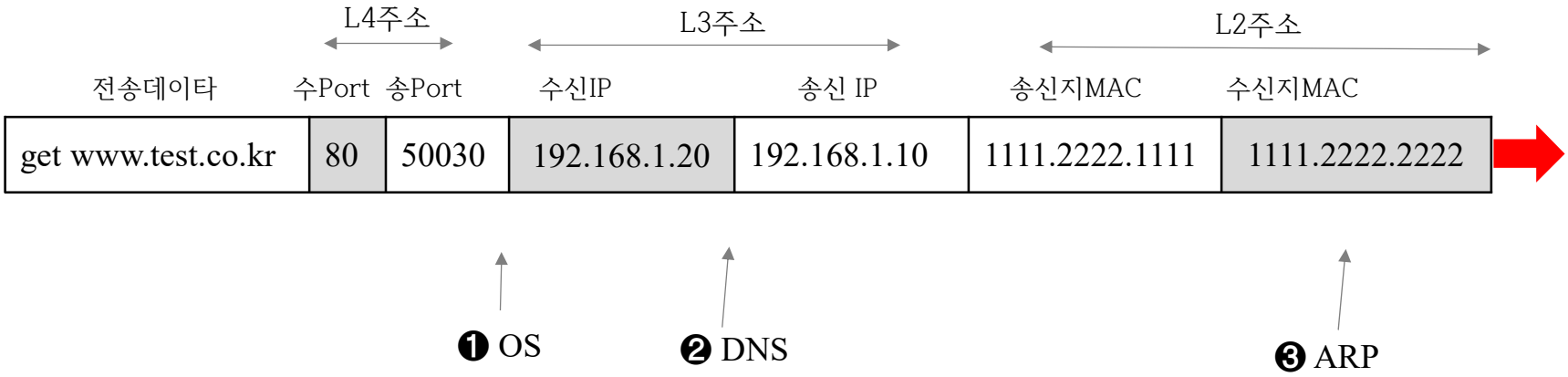
	L4주소		L3주소		L2주소	
전송데이타	수Port	송Port	수신IP	송신 IP	송신지MAC	수신지MAC
get www.test.co.kr	80	50030	????	192.168.1.10	1111.2222.1111	????

Unicast 전송모드



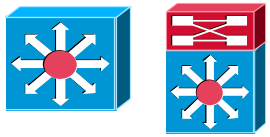
클라이언트(송신지)
IP 192.168.1.10
MAC 1111.2222.1111

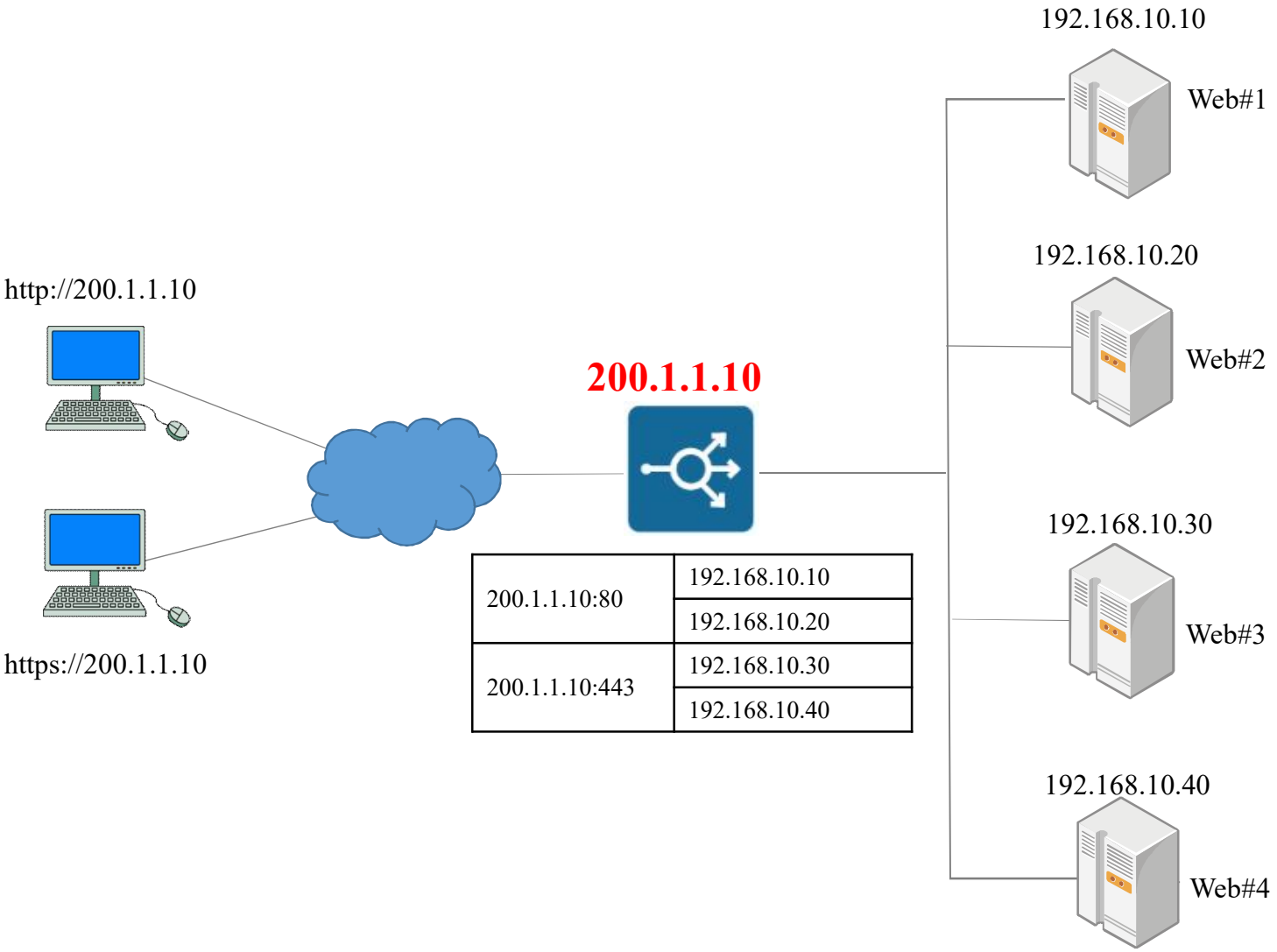
서버(수신지)
IP 192.168.1.20
MAC 1111.2222.2222



Multilayer Switch(다계층스위치)

구분	L2 Switch	L3 Switch	L4 Switch	L7 Switch
포워딩기준	2계층 (MAC Address)	3계층 (IP Address)	4계층 (IP address + Port 번호)	7계층 (IP address + Port 번호+ Content)
기능	Switching - Learning - Forwarding - Filtering	Switching Routing	L3 Switch Load Balance	L4 Switch Security Content 인식
주요 용도	Frame 전송	Packet 전송	FLB SLB	FLB SLB Security




192.168.10.10
Web#1
192.168.10.20
Web#2
192.168.10.30
Web#3
192.168.10.40
Web#4
http://200.1.1.10
https://200.1.1.10
200.1.1.10
200.1.1.10:80
192.168.10.10
192.168.10.20
200.1.1.10:443
192.168.10.30
192.168.10.40

Service

Service 개념

- 여러 개의 파드에 접근 할 수 있는 하나의 IP제공
  - 파드는 컨트롤러가 관리하므로 한 곳에서 고정해 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

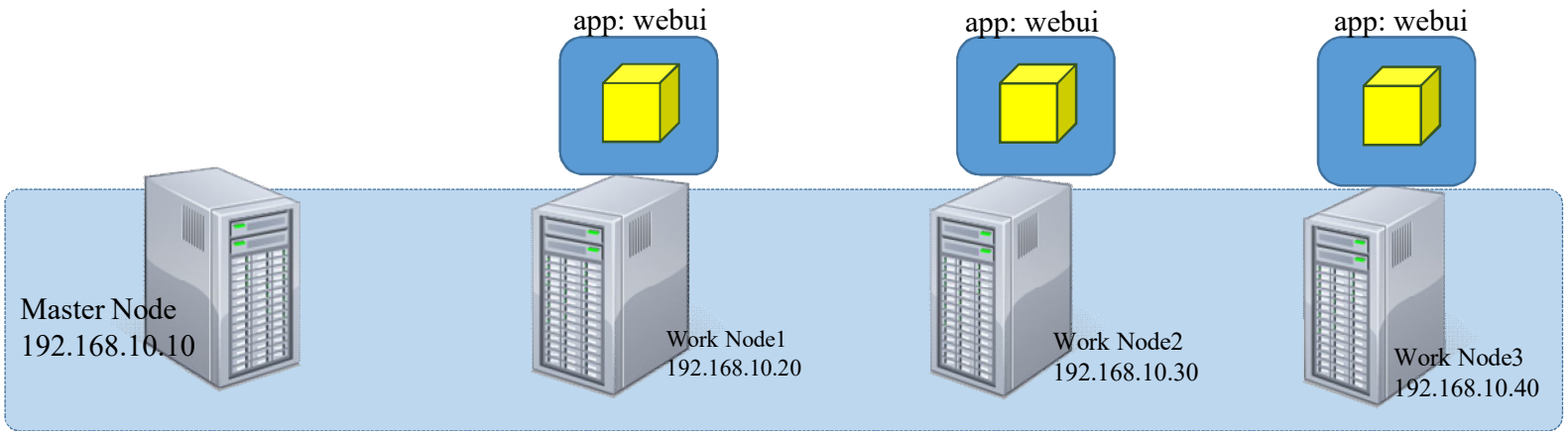
Service 타입

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 번호만 사용파면 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 장비에서 사용 • 클라우드에서 제공하는 로드밸런서와 파드를 연결 • 로드밸런서 IP를 이용해서 파드에 접근
ExternalName	• 클러스터 안에서 외부에 접근 할 때 사용

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
```

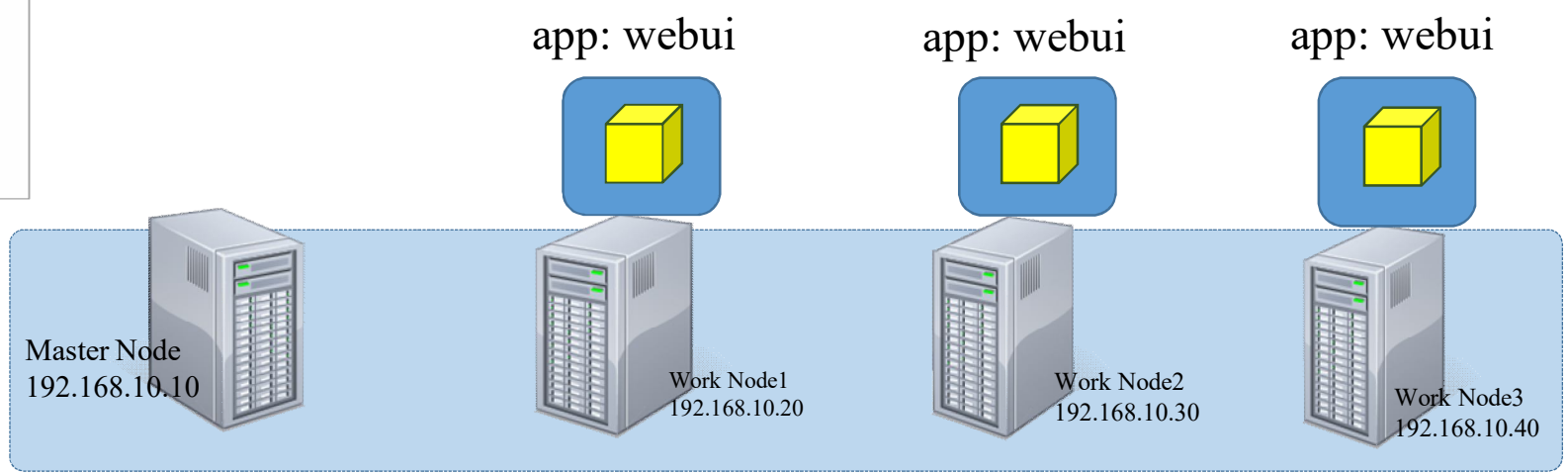


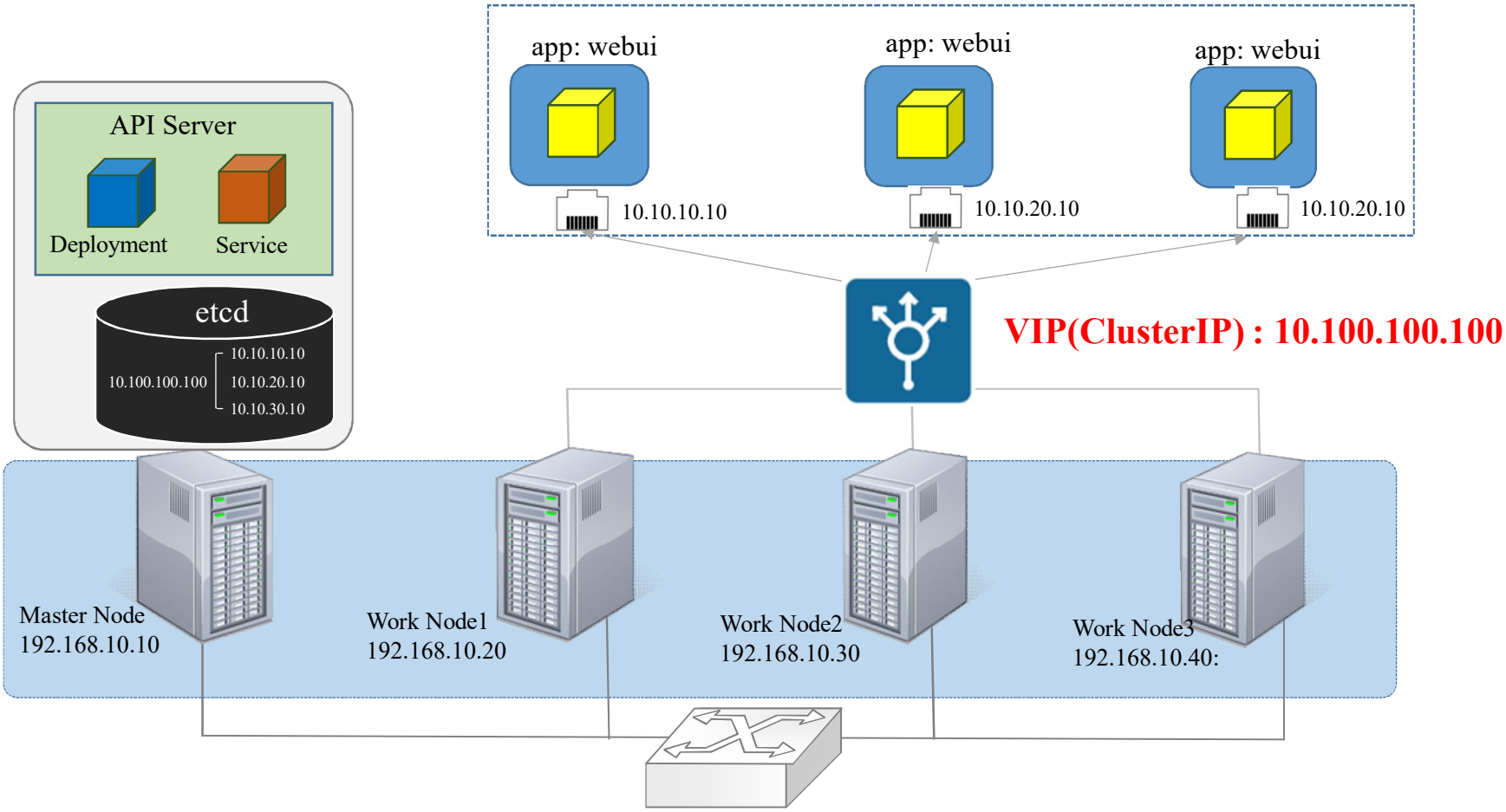
- 묶여진 그룹에 ClusterIP(Virtual IP) 주소 생성
- ClusterIP를 이용해서 서비스에 연결된 파드에 접근
- Type 생략 시 default로 10.96.0.0/12이 할당
- 클러스터 내에서만 사용할 수 있음

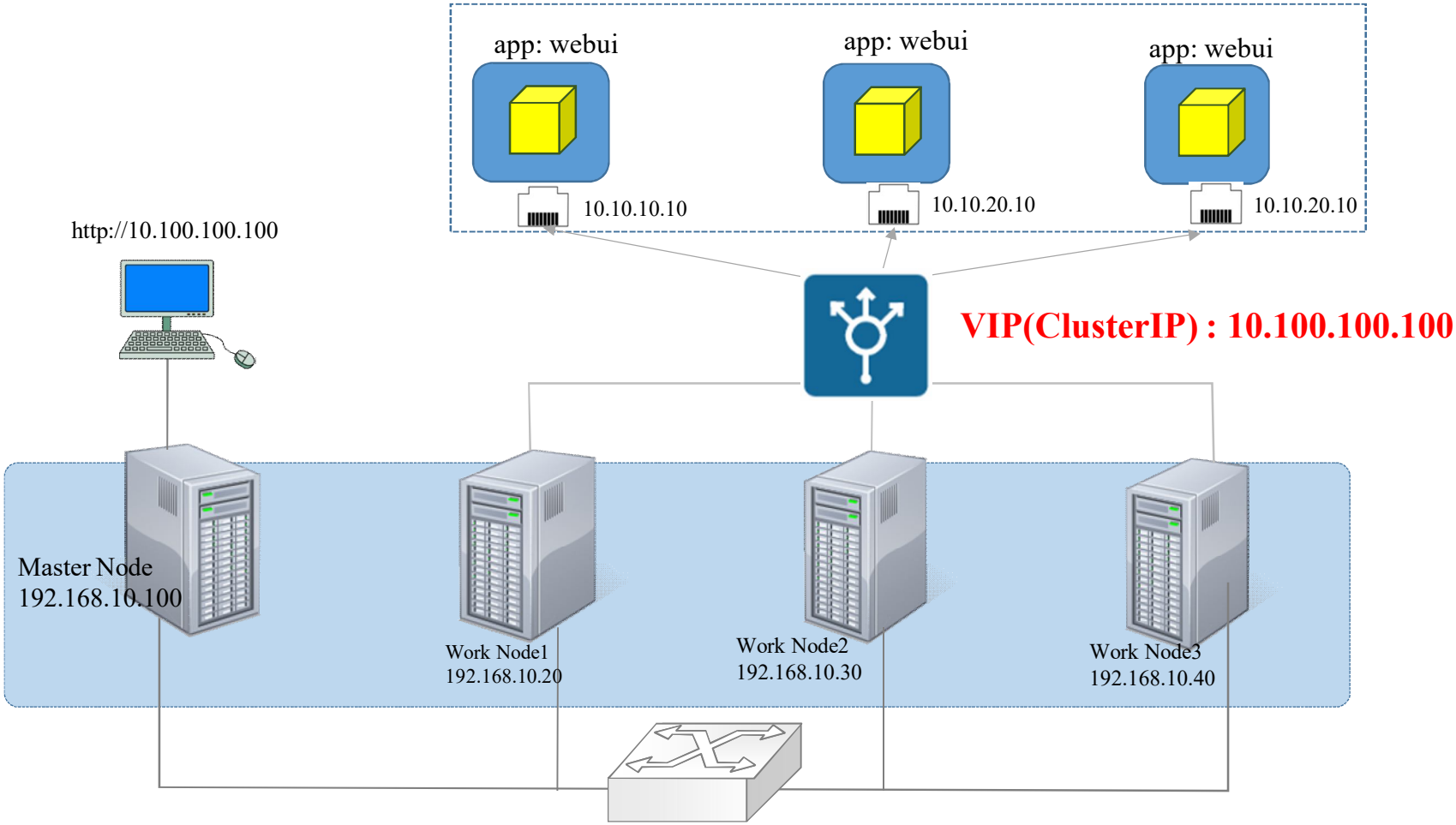
```
apiVersion: v1
kind: Service
metadata:
  name: clusterip-service
spec:
  type: ClusterIP
  clusterIP: 10.100.100.100
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```yaml
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
```

<<deploy-nginx.yaml>>




app: webui
app: webui
app: webui
10.10.10.10
10.10.20.10
10.10.20.10
API Server
Deployment
Service
etcd
10.100.100.100
10.10.10.10
10.10.20.10
10.10.30.10
VIP(ClusterIP) : 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:


app: webui
app: webui
app: webui
10.10.10.10
10.10.20.10
10.10.20.10
http://10.100.100.100
VIP(ClusterIP) : 10.100.100.100
Master Node
192.168.10.100
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 LAB(2)

kubectl get pod –o wide

webui-~~

curl 10.100.100.100

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

kubectl exec ***webui-~~*** -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

ClusterIP LAB(3)

kubectl scale deployment webui --replicas=5

kubectl describe svc clusterip-service

kubectl get nodes
kubectl get nodes –o wide

```
[node1 ~]$
[node1 ~]$ kubectl get nodes
NAME     STATUS   ROLES                  AGE     VERSION
node1    Ready    control-plane,master   3m10s   v1.20.1
node2    Ready    <none>                 84s     v1.20.1
node3    Ready    <none>                 51s     v1.20.1
node4    Ready    <none>                 23s     v1.20.1
[node1 ~]$
```

kubectl create –f deploy-nginx.yaml
kubectl get pod –o wide

```
[node1 ~]$ kubectl create -f deploy-nginx.yaml
deployment.apps/webui created
[node1 ~]$ kubectl get pod -o wide
NAME                     READY   STATUS              RESTARTS   AGE   IP       NODE    NOMINATED NODE   READINESS GATES
webui-6d4c4cc4b8-g76pg   0/1     ContainerCreating   0          9s    <none>   node2   <none>           <none>
webui-6d4c4cc4b8-rfwfc   0/1     ContainerCreating   0          9s    <none>   node3   <none>           <none>
webui-6d4c4cc4b8-xbtwr   0/1     ContainerCreating   0          9s    <none>   node4   <none>           <none>
[node1 ~]$ kubectl get pod -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
webui-6d4c4cc4b8-g76pg   1/1     Running   0          15s   10.5.1.2    node2   <none>           <none>
webui-6d4c4cc4b8-rfwfc   1/1     Running   0          15s   10.5.2.2    node3   <none>           <none>
webui-6d4c4cc4b8-xbtwr   1/1     Running   0          15s   10.5.3.2    node4   <none>           <none>
[node1 ~]$
```

kubectl create –f clusterip-nginx.yaml

kubectl get service

```
[node1 ~]$ kubectl create -f clusterip-nginx.yaml
service/clusterip-service created
[node1 ~]$ kubectl get service
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
clusterip-service   ClusterIP   10.100.100.100   <none>        80/TCP    5s
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP   6m53s
[node1 ~]$
```

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>
```

curl 10.100.100.100

```
[node1 ~]$ curl 10.100.100.100
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
```

kubectl exec ***webui-~~*** -it -- /bin/bash

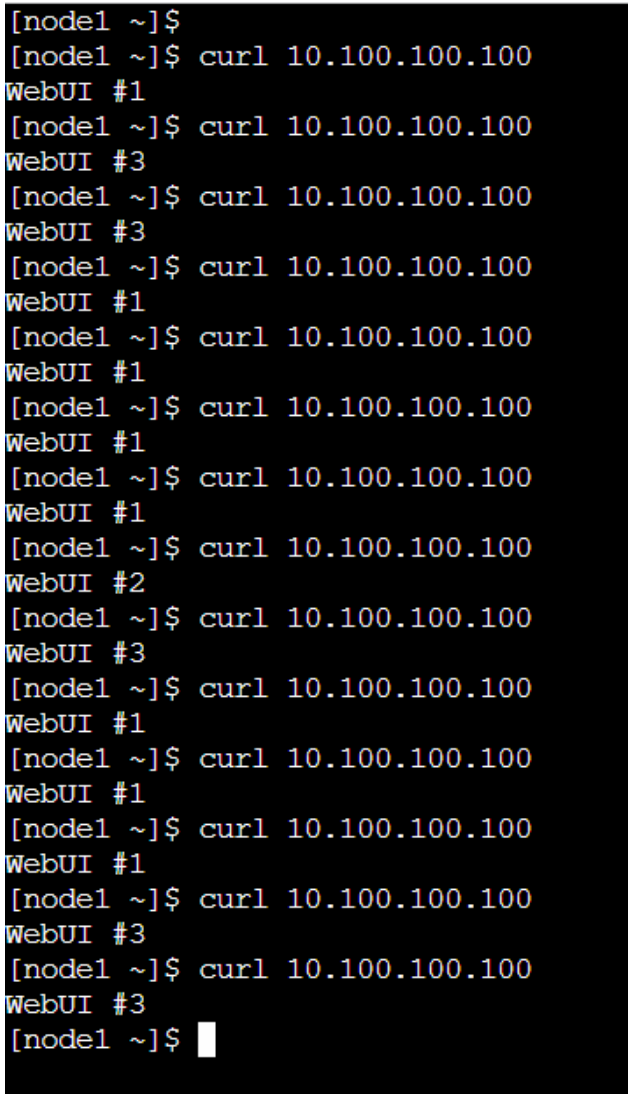
cd /usr/share/nginx/html

ls

echo "webui #1" > index.html

exit

```
[node1 ~]$
[node1 ~]$ kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE     IP          NODE    NOMINATED NODE   READINESS GATES
webui-6d4c4cc4b8-g76pg   1/1     Running   0          4m18s   10.5.1.2    node2   <none>           <none>
webui-6d4c4cc4b8-rfwfc   1/1     Running   0          4m18s   10.5.2.2    node3   <none>           <none>
webui-6d4c4cc4b8-xbtwr   1/1     Running   0          4m18s   10.5.3.2    node4   <none>           <none>
[node1 ~]$
[node1 ~]$ kubectl exec webui-6d4c4cc4b8-g76pg -it -- /bin/bash
root@webui-6d4c4cc4b8-g76pg:/# cd /usr/share/nginx/html
root@webui-6d4c4cc4b8-g76pg:/usr/share/nginx/html# ls
50x.html  index.html
root@webui-6d4c4cc4b8-g76pg:/usr/share/nginx/html# echo "WebUI #1" > index.html
root@webui-6d4c4cc4b8-g76pg:/usr/share/nginx/html# cat index.html
WebUI #1
root@webui-6d4c4cc4b8-g76pg:/usr/share/nginx/html# exit
exit
[node1 ~]$
[node1 ~]$ kubectl exec  webui-6d4c4cc4b8-rfwfc  -it -- /bin/bash
root@webui-6d4c4cc4b8-rfwfc:/# echo "WebUI #2" > /usr/share/nginx/html/index.html
root@webui-6d4c4cc4b8-rfwfc:/# cat /usr/share/nginx/html/index.html
WebUI #2
root@webui-6d4c4cc4b8-rfwfc:/# exit
exit
[node1 ~]$ kubectl exec webui-6d4c4cc4b8-xbtwr  -it -- /bin/bash
root@webui-6d4c4cc4b8-xbtwr:/# echo "WebUI #3" > /usr/share/nginx/html/index.html
root@webui-6d4c4cc4b8-xbtwr:/# cat /usr/share/nginx/html/index.html
WebUI #3
root@webui-6d4c4cc4b8-xbtwr:/# exit
exit
[node1 ~]$
```

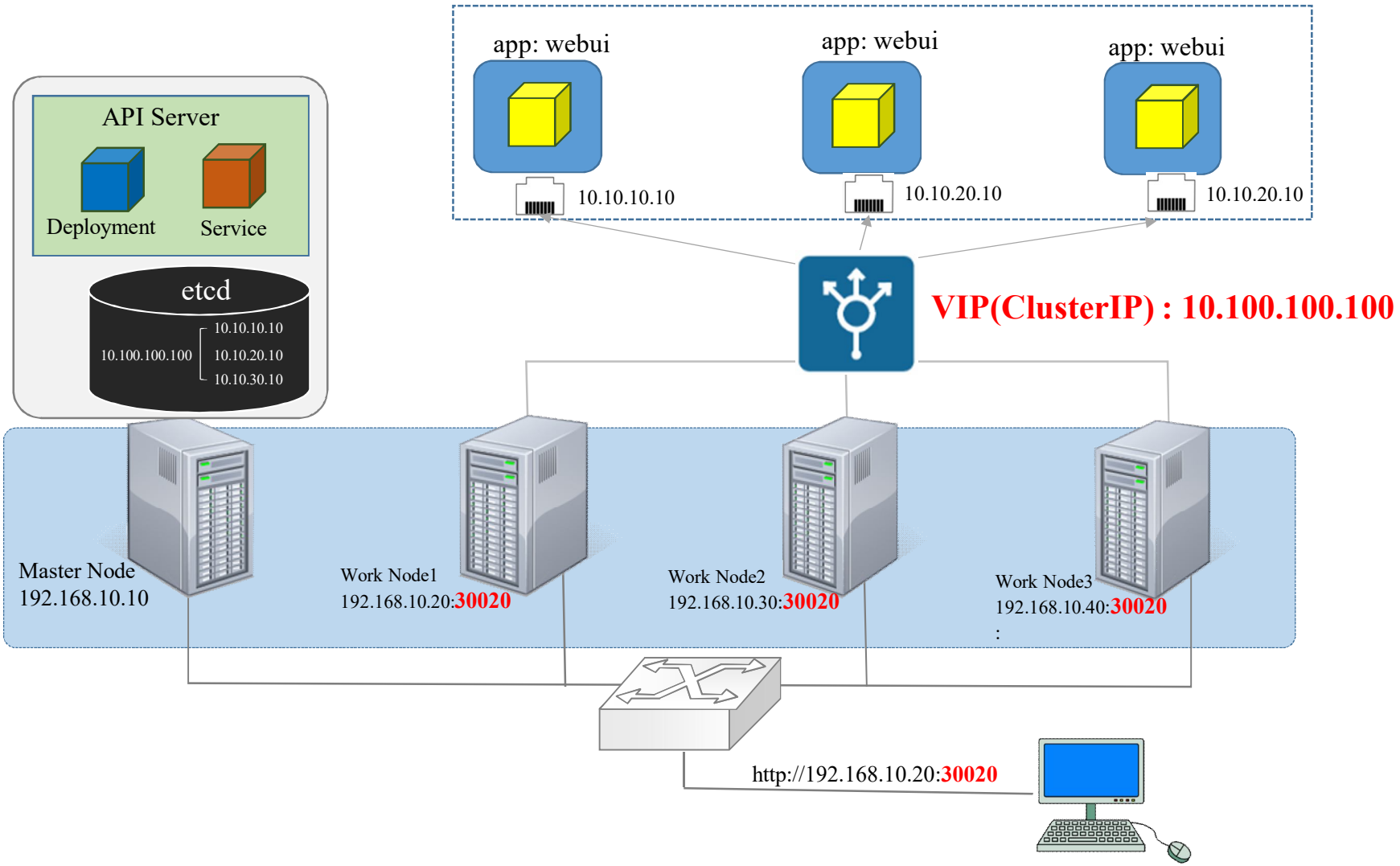

[node1 ~]$
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #3
[node1 ~]$ curl 10.100.100.100
WebUI #3
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #2
[node1 ~]$ curl 10.100.100.100
WebUI #3
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #1
[node1 ~]$ curl 10.100.100.100
WebUI #3
[node1 ~]$ curl 10.100.100.100
WebUI #3
[node1 ~]$

2) NodePort Type 서비스 사용하기

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

<<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
```


app: webui
app: webui
app: webui
10.10.10.10
10.10.20.10
10.10.20.10
API Server
Deployment
Service
etcd
10.100.100.100
10.10.10.10
10.10.20.10
10.10.30.10
VIP(ClusterIP) : 10.100.100.100
Master Node
192.168.10.10
Work Node1
192.168.10.20:30020
Work Node2
192.168.10.30:30020
Work Node3
192.168.10.40:30020
:
http://192.168.10.20:30020

NodePort LAB(1)

<<master node>>
kubectl delete service --all
kubectl create -f nodeport-nginx.yaml
kubectl get svc

<<work node1/2/3>>
netstat -napt | grep 30200
Exit

<<client에서 접속>>
curl 192.168.10.30:30200
curl 192.168.10.30:30200
curl 192.168.10.40:30200

NodePort LAB(1)

<<master node>>
kubectl delete service --all
kubectl create -f nodeport-nginx.yaml
kubectl get svc

<<work node1/2/3>>
netstat -napt | grep 30200
Exit

<<client에서 접속>>
curl 192.168.10.20:30200
curl 192.168.10.30:30200
curl 192.168.10.40:30200

ClusterIP LAB(2)

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

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

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

<<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

ClusterIP LAB(3)

kubectl scale deployment webui --replicas=5

kubectl describe svc clusterip-service

```
[node1 ~]$
[node1 ~]$ kubectl get svc
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
clusterip-service   ClusterIP   10.100.100.100   <none>        80/TCP     9m18s
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP    16m
[node1 ~]$
[node1 ~]$ kubectl delete service --all
service "clusterip-service" deleted
service "kubernetes" deleted
[node1 ~]$
[node1 ~]$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP    2s
[node1 ~]$
```

```
[node1 ~]$ vi nodeport-nginx.yaml
[node1 ~]$ kubectl create -f nodeport-nginx.yaml
service/nodeport-service created
[node1 ~]$ kebuctl get svc
bash: kebuctl: command not found
[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

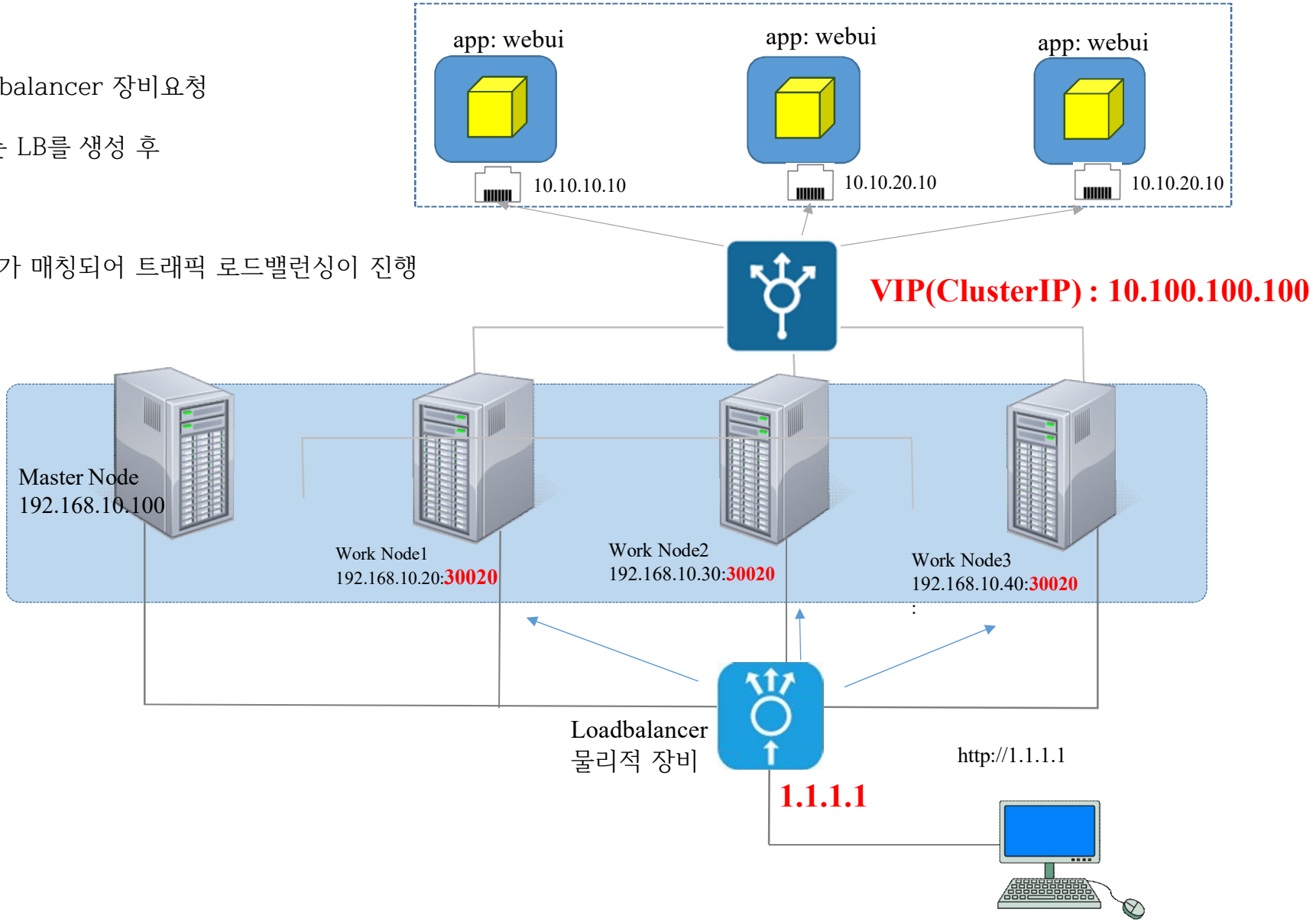
```
apiVersion: v1
kind: Service
metadata:
  name: loadbalancer-service
spec:
  type: LoadBalancer
  selector:
    app: webui
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

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



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

kubectl create –f loadbalancer-nginx.yaml

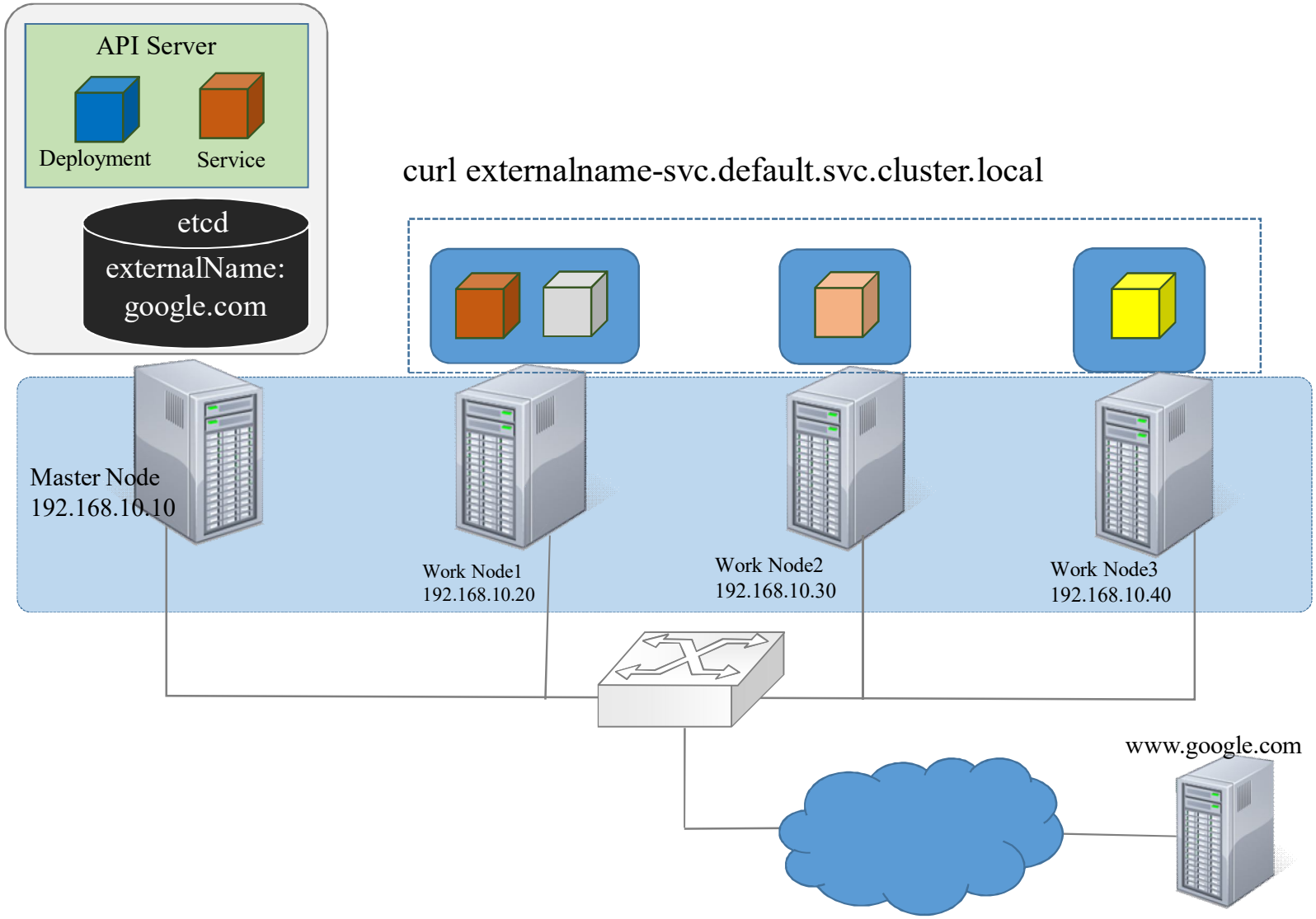
kubectl get svc

kubectl delete svc loadbalancer-serivce

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

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: test-pod
spec:
  replicas: 3
  selector:
    app: test
  template:
```