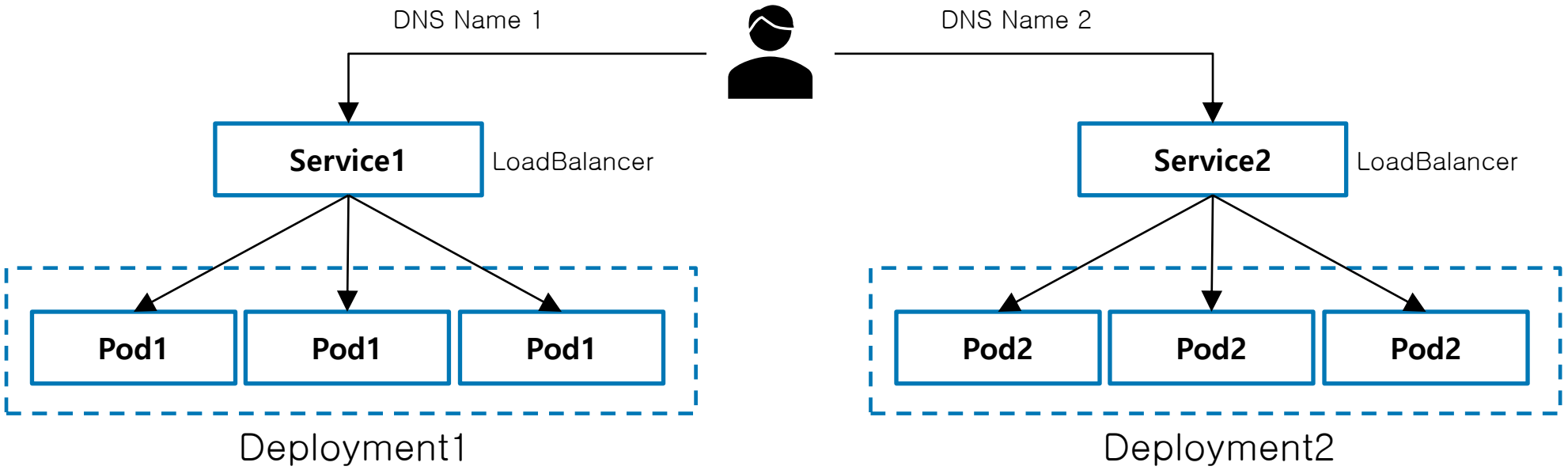
7.4 Ingress

❖Ingress란?

- k8s 클러스터 내부의 서비스에 대한 외부 접근을 관리하는 k8s API 리소스
- 부하 분산, TLS 종료, 이름 기반 가상 호스팅 제공

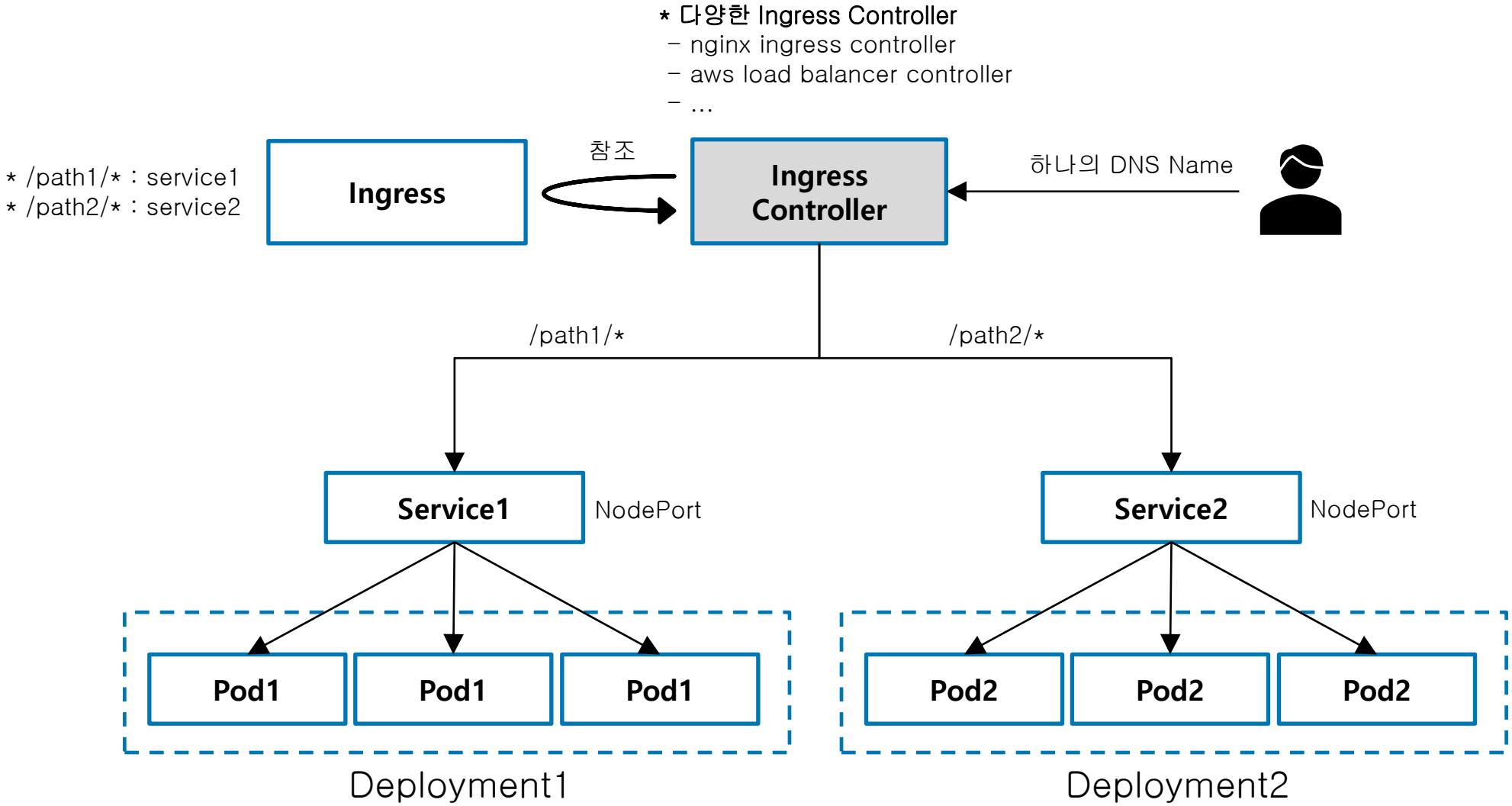
❖Ingress를 사용하지 않는다면?

- LoadBalancer 서비스마다 각각의 External IP 또는 Host 명을 이용해야 함



7.4 Ingress

❖Ingress 아키텍처



7.4 Ingress

❖Nginx Ingress Controller 실습

- 사전 조건 : metalLB가 미리 설치되어 있어야 함
- 다음 명령어를 이용해 ingress-nginx-controller 설치

```
# helm repo 추가 후 업데이트
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

# ingress-nginx 네임스페이스 ingress-controller 설치
# service 타입을 LB로 설정하고 LB의 IP 주소를 192.168.56.60으로 설정. 이를 위해 metalLB가 미리 설정되어야 함
helm install ingress-nginx ingress-nginx/ingress-nginx ₩
  --namespace ingress-nginx --create-namespace ₩
  --set controller.service.type=LoadBalancer ₩
  --set controller.service.loadBalancerIP=192.168.56.60
  --set controller.progressDeadlineSeconds=600
```

7.4 Ingress

❖ ingress-nginx-controller 설치 확인

```
stepano@minipc:~ $ kubectl get all -n ingress-nginx
NAME                                            READY   STATUS    RESTARTS   AGE
pod/ingress-nginx-controller-7f49467bd9-rff79   1/1     Running   0          2m42s

NAME                                         TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
service/ingress-nginx-controller             LoadBalancer   10.97.168.240   192.168.56.60   80:30396/TCP,443:31738/TCP   2m42s
service/ingress-nginx-controller-admission   ClusterIP      10.107.11.229   <none>          443/TCP                      2m42s

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ingress-nginx-controller   1/1     1            1           2m42s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/ingress-nginx-controller-7f49467bd9   1         1         1       2m42s
```

7.4 Ingress

❖deployment, service 설치 : nodeapp1.yaml, nodeapp2.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: svc-nodeapp1
  namespace: default
  labels:
    app: nodeapp1
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      protocol: TCP
  selector:
    app: nodeapp1
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodeapp-path1
  namespace: default
  labels:
    app: nodeapp1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nodeapp1
  template:
    metadata:
      name: nodeapp-path1
      labels:
        app: nodeapp1
    spec:
      containers:
      - name: nodeapp1
        image: stepanowon/nodeapp-path1:1.0.0
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
```

nodeapp2는 볼드체인 부분만 차이가 있음

7.4 Ingress

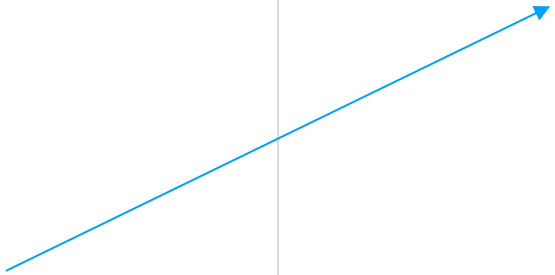
- nodeapp-path1 이미지 코드

```
'use strict';

const express = require('express');
const os = require('os');
//
const PORT = 8080;
const HOST = '0.0.0.0';

//
const app = express();
app.get('/:reqpath', (req, res) => {
  return res.status(200).send(`
  <div style="background-color:aqua">
    <h2> nodeapp-path1</h2>
    <h2> 호스트명 : ${os.hostname()} </h2>
    <h2> 요청경로 : ${req.params.reqpath} </h2>
  </div>
  `);
});

app.listen(PORT, HOST);
console.log(`Running on http://${HOST}:${PORT}`);
```



* nodeapp-path1 : aqua 색
* nodeapp-path2 : yellow 색

7.4 Ingress

- ingress-nginx-controller 의 호스트명을 반영하여 다음 yaml 작성
  - kubectl apply -f nodeapp-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
  ingressClassName: nginx
  rules:
  - host: demo.192.168.56.60.nip.io
    http:
      paths:
      - pathType: ImplementationSpecific
        path: /path1(/|$)(.*)
        backend:
          service:
            name: svc-nodeapp1
            port:
              number: 8080
      - pathType: ImplementationSpecific
        path: /path2(/|$)(.*)
        backend:
          service:
            name: svc-nodeapp2
            port:
              number: 8080
```

7.4 Ingress

❖참고 : nip.io 주소

- Wild card DNS 서비스
- DNS 서버에 호스트를 등록하지 않고, hosts 파일을 변경하지 않고도 유연성있는 host 명을 사용할 수 있도록 함
  - 테스트시에 편리함
- 규칙 : 다음의 모든 호스트명은 192.168.56.60에 매핑됨
  - 192.168.56.60.nip.io
  - 192-168-56-60.nip.io
  - test.com.192.168.56.60.nip.io
  - app-192-168-56-60.nip.io
  - test1.app.192.168.56.60.nip.io
  - test2-app-192-168-56-60.nip.io

7.4 Ingress

- 기능 테스트 : 호스트명으로 요청하기
  - curl http://demo.192.168.56.60.nip.io/path1/abc
  - curl http://demo.192.168.56.60.nip.io/path2/abc

```
stepano@minipc:~/k8s/ingress $ curl http://demo.192.168.56.60.nip.io/path1/abc

  <div style="background-color:aqua">
    <h2> nodeapp-path1</h2>
    <h2> 호스트명 : nodeapp-path1-84c8cb66df-pvbbx </h2>
    <h2> 요청경로 : abc </h2>
  </div>
  stepano@minipc:~/k8s/ingress $
stepano@minipc:~/k8s/ingress $ curl http://demo.192.168.56.60.nip.io/path2/abc

  <div style="background-color:yellow">
    <h2> nodeapp-path2</h2>
    <h2> 호스트명 : nodeapp-path2-7c48f69bb5-xr2k6 </h2>
    <h2> 요청경로 : abc </h2>
  </div>
  stepano@minipc:~/k8s/ingress $
```

7.4 Ingress

❖ 리소스 삭제

```
kubectl delete -f nodeapp1.yaml
kubectl delete -f nodeapp2.yaml
kubectl delete -f nodeapp-ingress.yaml

helm uninstall ingress-nginx -n ingress-nginx
kubectl delete ns ingress-nginx
```