# Service - Ingress 생성하기

인그레스 컨트롤러(ingress controller)는 인그레스(Ingress) 객체에 정의된 규칙을 참조하여 클러스터 외부에서 내부 서비스로 HTTP와 HTTPS 요청을 라우팅하고 SSL/TSL 종료, 가상 호스팅 등을 제공합니다.
※ 인그레스 컨트롤러에 대한 자세한 내용은 아래 인그레스 컨트롤러 문서를 참고하시기 바랍니다.



## Ingress 생성하기

### Step 1. NGINX Ingress Controller 설치하기

NGINX Ingress Controller는 많이 사용되는 인그레스 컨트롤러 중 하나입니다. 필요한 자원을 바로 생성할 수 있도록 미리 정의한 매니페스트 파일을 제공합니다. 이 매니페스트를 이용하면 쉽게 필요한 자원을 생성할 수 있습니다.

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/mandatory.yaml
namespace/ingress-nginx created
configmap/nginx-configuration created
configmap/tcp-services created
configmap/udp-services created
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created
deployment.apps/nginx-ingress-controller created
limitrange/ingress-nginx created
```

### Step 2. LoadBalancer 서비스 생성하기

LoadBalancer 서비스 생성인그레스 컨트롤러 역시 파드로 생성되기 때문에 외부에 공개하기 위해서는 LoadBalancer 서비스 또는 NodePort 서비스를 만들어야 합니다. 다음과 같이 HTTP와 HTTPS를 처리할 수 있는 LoadBalancer 서비스 매니페스트를 정의합니다.

```
apiVersion: v1
kind: Service
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
```

```
spec:
  type: LoadBalancer
  selector:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
    - name: https
      port: 443
      targetPort: 443
      protocol: TCP
  selector:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
```

· 서비스 객체를 생성하고 외부 로드 밸런서가 연결되어 있는지 확인합니다. EXTERNAL-IP 필드에는 플로팅 IP 주소가 설정되어 있어야 합니다.

```
$ kubectl apply -f service.yaml
service/nginx-svc created

$ kubectl get service
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP PORT(S)          AGE
nginx-svc   LoadBalancer   10.254.134.18   <pending>   8080:30013/TCP   11s

.............................................
$ kubectl get service
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
nginx-svc   LoadBalancer   10.254.134.18   123.123.123.30  8080:30013/TCP
3m13s
```

### Step 3. 서비스와 파드 생성하기

다음과 같이 서비스와 파드를 생성하기 위한 매니페스트를 작성합니다.
tea-svc 서비스에는 tea 파드를 연결하고, coffee-svc 서비스에는 coffee 파드를 연결합니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
spec:
  replicas: 3
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
```

```
        app: coffee
    spec:
      containers:
      - name: coffee
        image: nginxdemos/nginx-hello:plain-text
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
    name: http
  selector:
    app: coffee
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 2
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: nginxdemos/nginx-hello:plain-text
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
spec:
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
    name: http
  selector:
    app: tea
```

· 매니페스트를 적용하고 디플로이먼트, 서비스, 파드가 생성되었는지 확인합니다. 파드는 Running 상태여야 합니다.

```
$ kubectl apply -f cafe.yaml
deployment.apps/coffee created
service/coffee-svc created
deployment.apps/tea created
service/tea-svc created

$ kubectl get deploy,svc,pods
NAME             READY   UP-TO-DATE  AVAILABLE   AGE
deployment.extensions/coffee    3/3 3       3       18s
deployment.extensions/tea   2/2 2       2       18s

NAME                TYPE            CLUSTER-IP      EXTERNAL-IP PORT(S) AGE
service/coffee-svc  ClusterIP       10.254.51.117   <none>      80/TCP  18s
service/tea-svc     ClusterIP       10.254.210.170  <none>      80/TCP  18s

NAME                          READY   STATUS      RESTARTS    AGE
pod/coffee-67c6f7c5fd-98vh5 1/1     Running     0           18s
pod/coffee-67c6f7c5fd-c58l2 1/1     Running     0           18s
pod/coffee-67c6f7c5fd-dmxf6 1/1     Running     0           18s
pod/tea-7df475c6-gtlx5      1/1     Running     0           18s
pod/tea-7df475c6-lxqsx      1/1     Running     0           18s
```

### Step 4. 인그레스(Ingress) 생성하기

요청 경로에 따라 서비스를 연결하는 인그레스 매니페스트를 작성합니다.
엔드포인트가 /tea인 요청은 tea-svc 서비스에 연결하고 /coffee인 요청은 coffee-svc 서비스에 연결합니다

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress-uri
spec:
  rules:
  - http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

· 인그레스를 생성하고 잠시 후 확인했을 때 ADDRESS 필드에 IP가 설정되어 있어야 합니다.

```
$ kubectl apply -f cafe-ingress-uri.yaml
ingress.extensions/cafe-ingress-uri created

$ kubectl get ingress cafe-ingress-uri
NAME                HOSTS   ADDRESS          PORTS   AGE
cafe-ingress-uri    *       123.123.123.44   80      88s
```

## Step 5. HTTP 요청 전송하기

외부 호스트에서 ingress의 ADDRESS 필드에 설정된 IP 주소로 HTTP 요청을 전송해 인그레스가 올바르게 설정되었는지 확인합니다.
엔드포인트 /coffee에 대한 요청은 coffee-svc 서비스에 전달되어 coffee 파드가 응답합니다.
응답의 Server name 항목을 보면 coffee 파드들이 라운드-로빈 방식으로 번갈아 응답하는 것을 확인할 수 있습니다.

```
$ curl http://123.123.123.44/coffee
Server address: 10.100.3.48:8080
Server name: coffee-67c6f7c5fd-c58l2
Dat#e: 07/Apr/2020:08:24:27 +0000
URI: /coffee
Request ID: e831901e441303ad59fb02214c49d84a

$ curl http://123.123.123.44/coffee
Server address: 10.100.2.23:8080
Server name: coffee-67c6f7c5fd-98vh5
Date: 07/Apr/2020:08:24:28 +0000
URI: /coffee
Request ID: e78427e68a1cd61ec633b9328359874e
```

· 마찬가지로 엔드포인트 /tea에 대한 요청은 tea-svc 서비스에 전달되어 tea 파드가 응답합니다.

```
$ curl http://123.123.123.44/tea
Server address: 10.100.2.24:8080
Server name: tea-7df475c6-lxqsx
Date: 07/Apr/2020:08:25:03 +0000
URI: /tea
Request ID: 59303a5a5baa60802b463b1856c8ce8d
```