Experiment Optional: Ingress with k3d

In this experiment, we will deploy a simple nginx webserver deployment and make it accessible via ingress. Therefore, we have to create the cluster in a way, that the internal port 80 (where the traefik ingress controller is listening on) is exposed on the host system.

Create a cluster, mapping the ingress port 80 to localhost:8081

$ **k3d cluster create ingress-demo --api-port 6550 -p 8081:80@loadbalancer --agents 2**

Notes:

- --api-port 6550 is not required for the example to work. It's used to have k3s's API-Server listening on port 6550 with that port mapped to the host system.
- the port-mapping construct 8081:80@loadbalancer means
  - map port 8081 from the host to port 80 on the container which matches the nodefilter loadbalancer
- the loadbalancer nodefilter matches only the serverlb that's deployed in front of a cluster's server nodes
  - all ports exposed on the serverlb will be proxied to the same ports on all server nodes in the cluster

Setup your KUBECONFIG environment variable as noted in the other experiments

Create the nginx deployment

$ **kubectl create deployment nginx --image=nginx**

Create a ClusterIP service for it

$ **kubectl create service clusterip nginx --tcp=80:80**

Create an ingress object for it with kubectl apply -f *Note*: k3s deploys traefik as the default ingress controller

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
  annotations:
    ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
  - http:
      paths:
      - path: /
        backend:
```

```
        serviceName: nginx
        servicePort: 80
```

$ **kubectl apply -f simple-ingress.yaml**

Curl it via localhost

curl localhost:8081/

$ **k3d cluster delete ingress-demo**