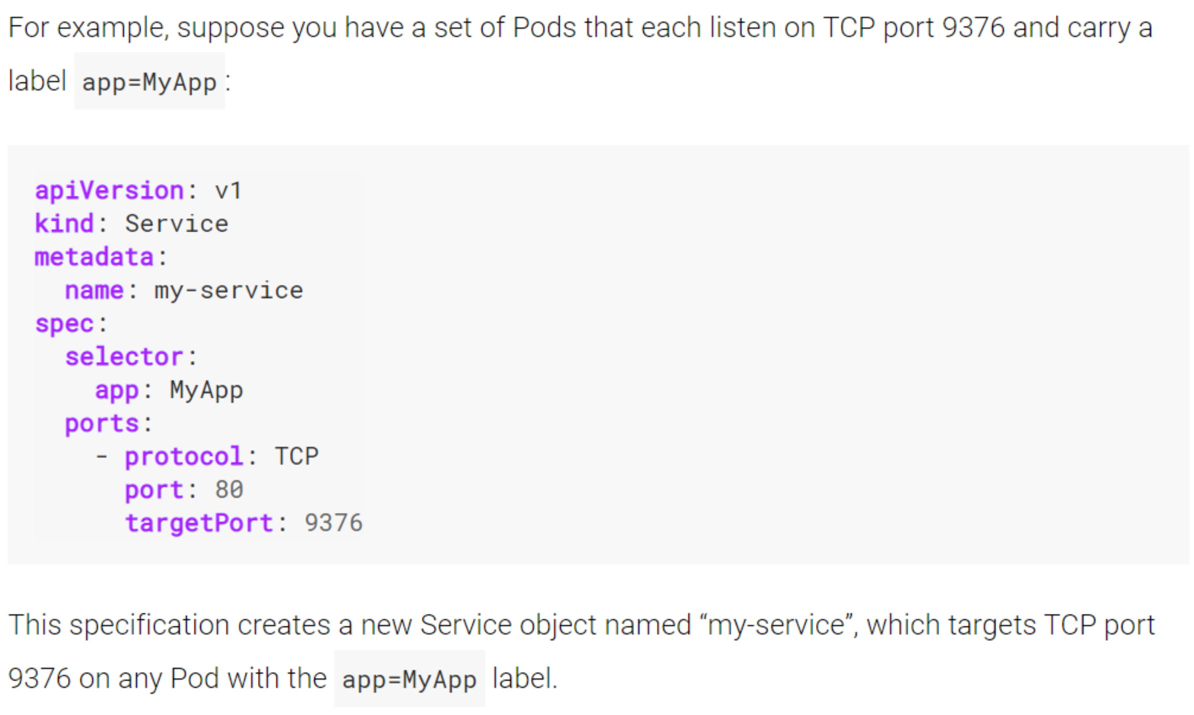
Kubernetes hands-on

Core Concepts:

CKA Certification:

Services

A Service in Kubernetes is a REST object, similar to a Pod. Like all of the REST objects, you can POST a Service definition to the API server to create a new instance. The name of a Service object must be a valid DNS label name.

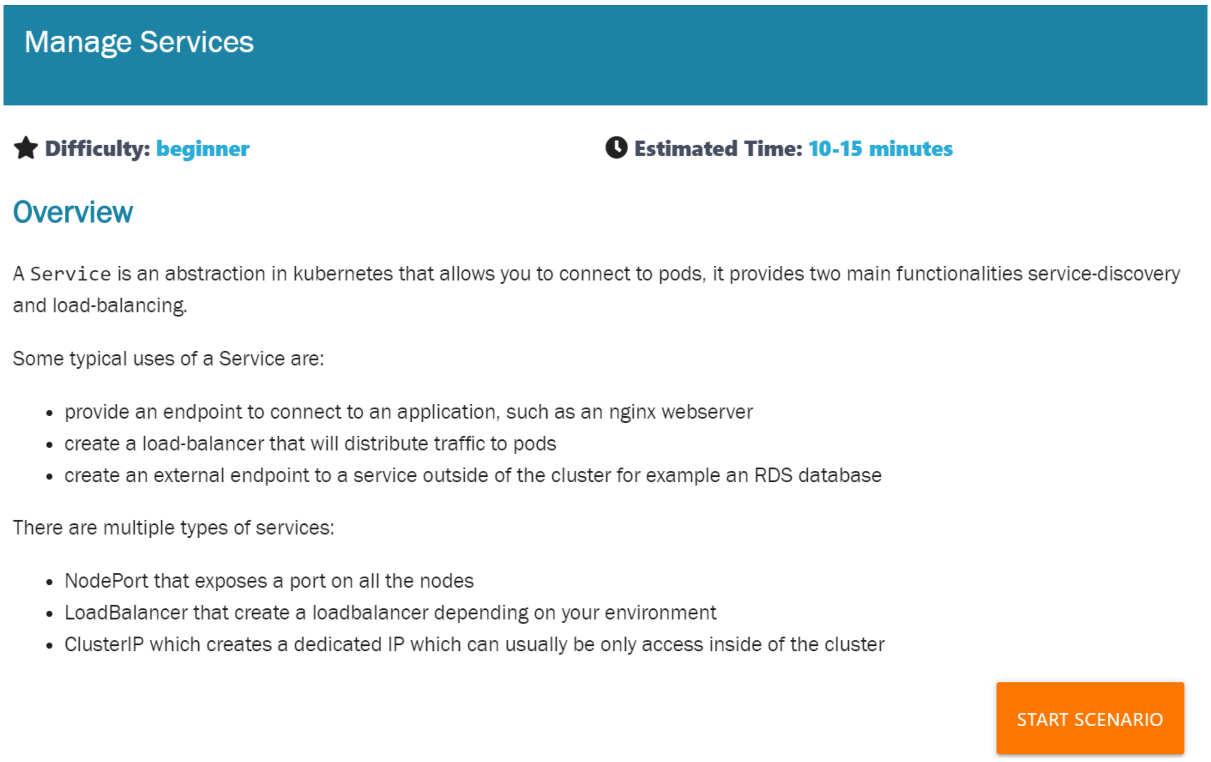


For example, suppose you have a set of Pods that each listen on TCP port 9376 and carry a label `app=MyApp`:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

This specification creates a new Service object named "my-service", which targets TCP port 9376 on any Pod with the `app=MyApp` label.

The kubernetes service is an object just like PODs, Replicasetor Deployments that we worked with before. One of its use case is to listen to a port on the Node and forward requests on that port to a port on the POD running the web application.



Manage Services

Difficulty: **beginner** **Estimated Time:** **10-15 minutes**

Overview

A `Service` is an abstraction in kubernetes that allows you to connect to pods, it provides two main functionalities service-discovery and load-balancing.

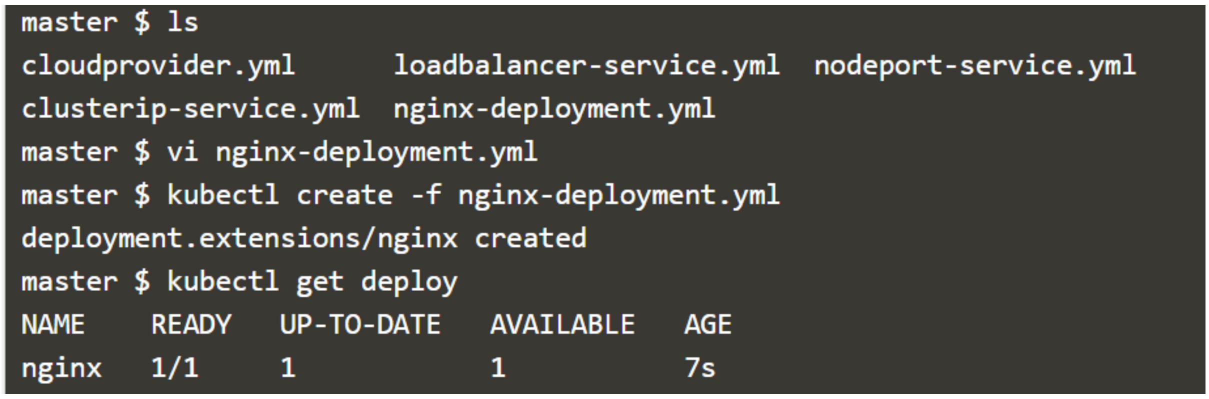
Some typical uses of a Service are:

- provide an endpoint to connect to an application, such as an nginx webserver
- create a load-balancer that will distribute traffic to pods
- create an external endpoint to a service outside of the cluster for example an RDS database

There are multiple types of services:

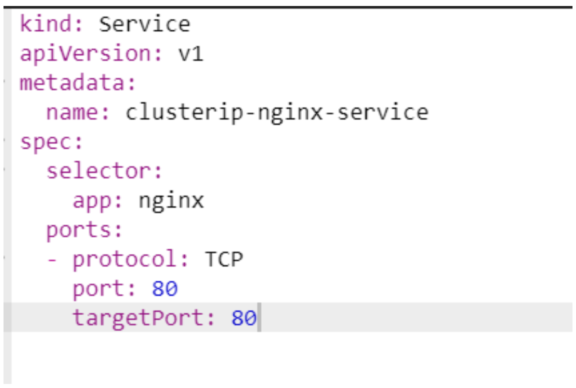
- NodePort that exposes a port on all the nodes
- LoadBalancer that create a loadbalancer depending on your environment
- ClusterIP which creates a dedicated IP which can usually be only access inside of the cluster

let's create a deployment that we will use to learn the various service types.

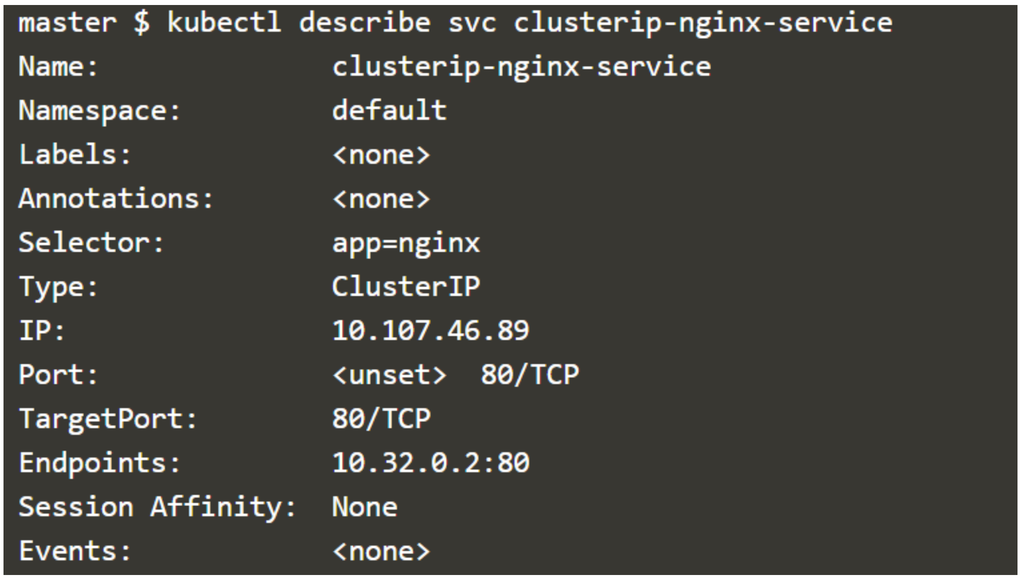


```
master $ ls
cloudprovider.yml       loadbalancer-service.yml  nodeport-service.yml
clusterip-service.yml  nginx-deployment.yml
master $ vi nginx-deployment.yml
master $ kubectl create -f nginx-deployment.yml
deployment.extensions/nginx created
master $ kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           7s
```

Now that we have a working deployment, lets expose it to the cluster so that other deployments can access it too.

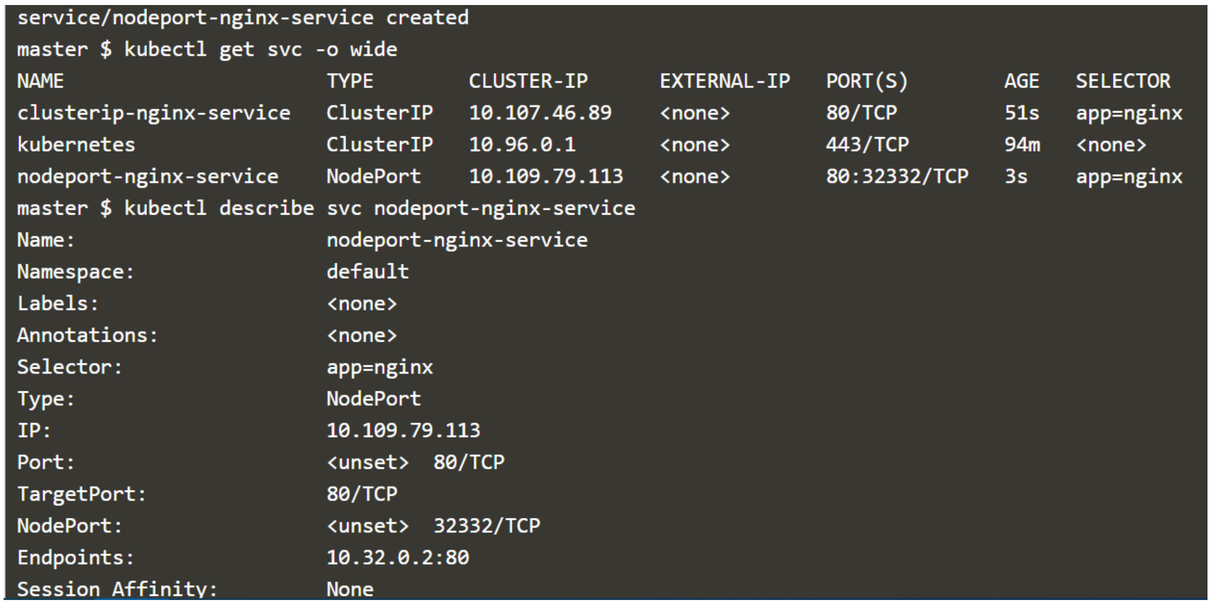


```
kind: Service
apiVersion: v1
metadata:
  name: clusterip-nginx-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```



```
master $ kubectl describe svc clusterip-nginx-service
Name:              clusterip-nginx-service
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=nginx
Type:              ClusterIP
IP:                10.107.46.89
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.32.0.2:80
Session Affinity:  None
Events:            <none>
```

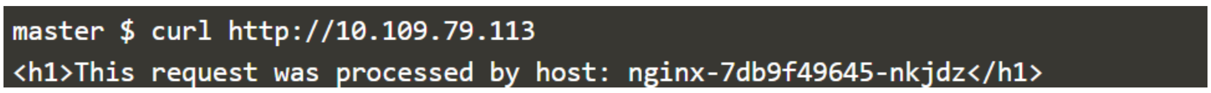
What if we wanted to expose our service outside of the cluster? This is where NodePort comes in. NodePort is one the most often utilized service types in kubernetes.



```
service/nodeport-nginx-service created
master $ kubectl get svc -o wide
NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
clusterip-nginx-service   ClusterIP   10.107.46.89    <none>        80/TCP         51s   app=nginx
kubernetes                ClusterIP   10.96.0.1       <none>        443/TCP        94m   <none>
nodeport-nginx-service    NodePort    10.109.79.113   <none>        80:32332/TCP   3s    app=nginx
master $ kubectl describe svc nodeport-nginx-service
Name:                     nodeport-nginx-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx
Type:                     NodePort
IP:                       10.109.79.113
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32332/TCP
Endpoints:                10.32.0.2:80
Session Affinity:         None
```

We can now access our service with:

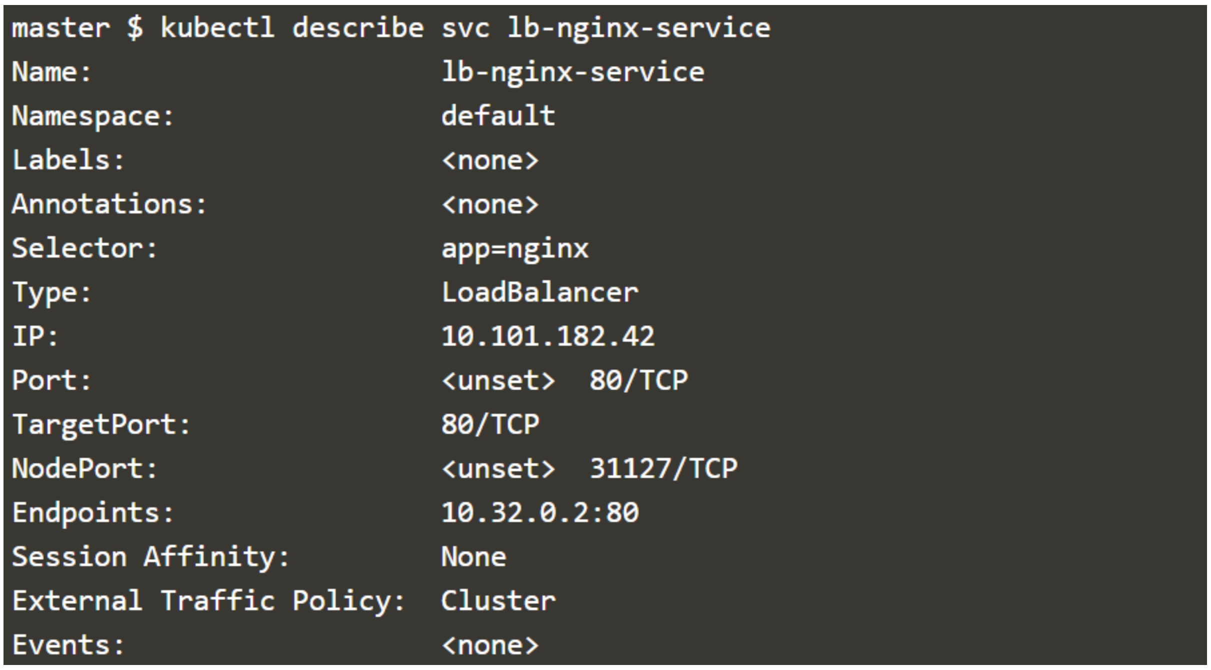
`curl http://<NODEPORT-IP>`



```
master $ curl http://10.109.79.113
<h1>This request was processed by host: nginx-7db9f49645-nkjdz</h1>
```

What if we wanted a single point of entry for our service from the oustide? For that we need a LoadBalancer type of service. If you are running on any of the major cloud providers it will be freely available for you, but if you are on-prem or in this case katacoda, then you need to make this functionality available.

kubectl describe svc lb-nginx-service



```
master $ kubectl describe svc lb-nginx-service
Name:                     lb-nginx-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx
Type:                     LoadBalancer
IP:                       10.101.182.42
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31127/TCP
Endpoints:                10.32.0.2:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

References:

https://kubernetes.io/docs/concepts/services-networking/service/#defining-a-service

https://www.katacoda.com/contino/courses/kubernetes/services#

https://www.udemy.com/course/certified-kubernetes-administrator-with-practice-tests/learn/lecture/14295512#overview

https://kodekloud.com/courses/certified-kubernetes-administrator-with-practice-tests-labs/lectures/12038870