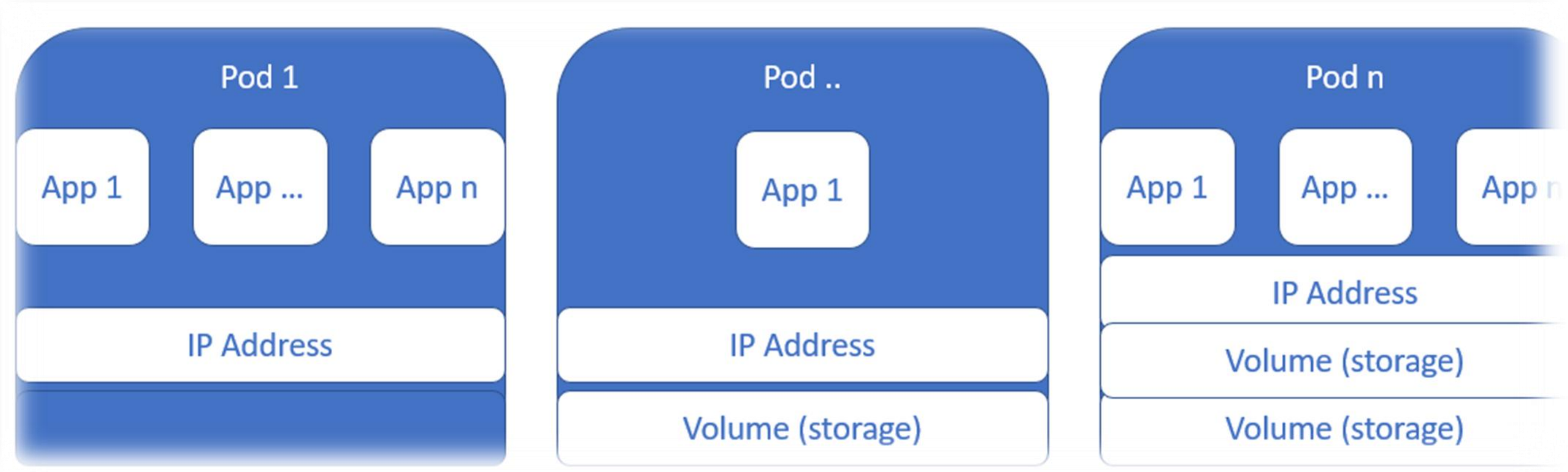
Networking

Pods and services

Kubernetes concept of a pod, which is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed



Services

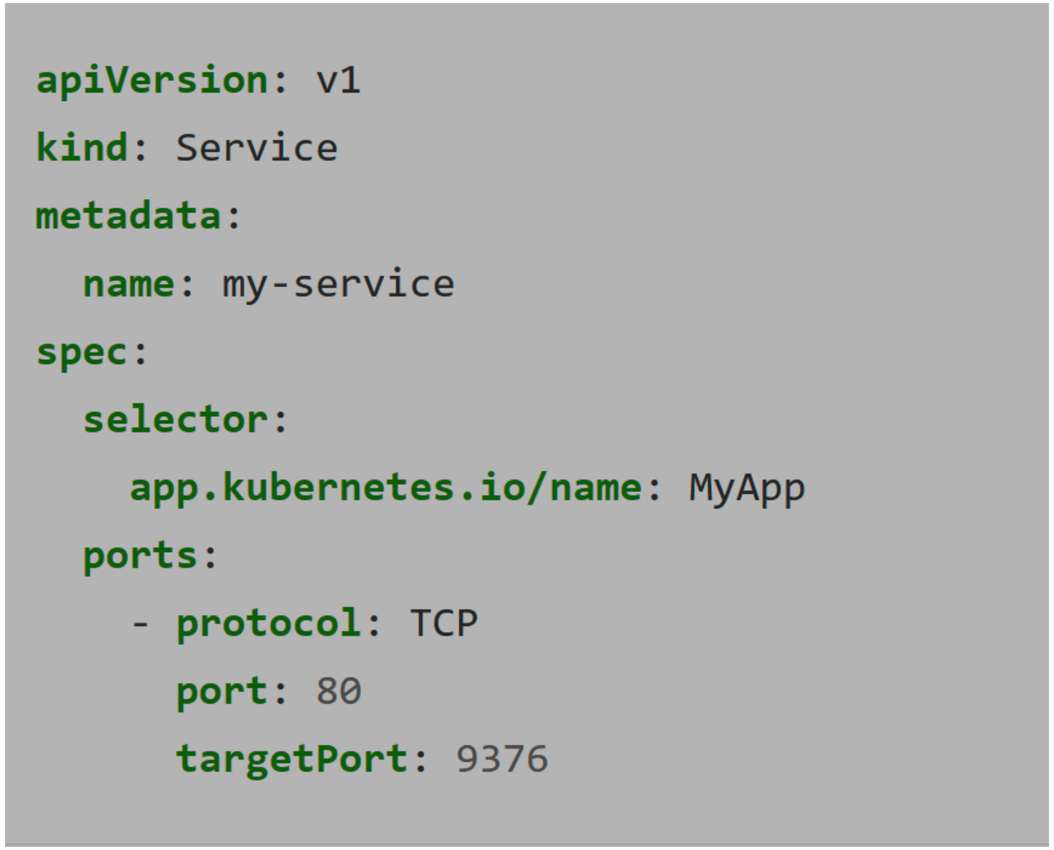
ClusterIP
Allows to connect to port on cluster-internal IP

NodePort
Listens on specified port on all nodes and forwards to cluster-internal IP

LoadBalancer
Exposes service through an external loadbalancer. K8s does not provide this.

External Name
Maps service to externalName field which contains a hostname.

Simple ClusterIP Service



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

Service discovery

Kubernetes is quite dynamic with placing pods on nodes, horizontal autoscalers etc.

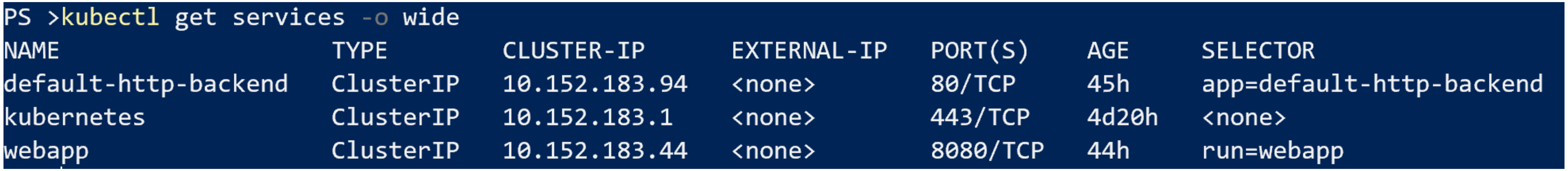
Service discovery helps us find the services in the cluster

Should resolve quickly and reliably

The Service Object

A Service Object is a way to create a named label selector

```
# Creates a service:
kubectl expose deployment nginx


# Show including the selector:
kubectl get services -o wide
```



```
PS >kubectl get services -o wide
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE     SELECTOR
default-http-backend   ClusterIP   10.152.183.94   <none>        80/TCP     45h     app=default-http-backend
kubernetes             ClusterIP   10.152.183.1    <none>        443/TCP    4d20h   <none>
webapp                 ClusterIP   10.152.183.44   <none>        8080/TCP   44h     run=webapp
```

Port-forward to service

Listen on port 8888 locally, forwarding to 5000 in the pod

```
kubectl port-forward pod/mypod 8888:5000
```

Listen on port 8888 on all addresses, forwarding to 5000 in the pod

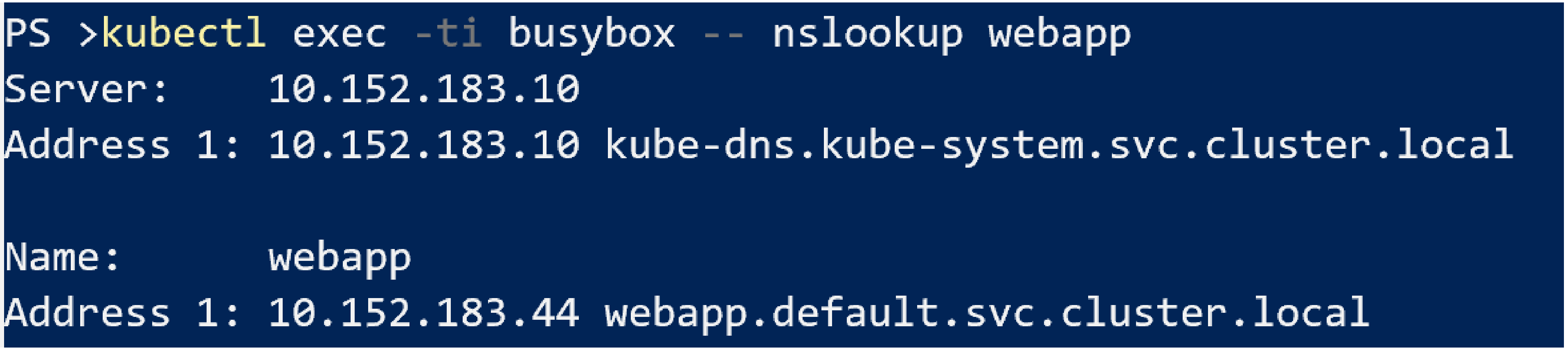
```
kubectl port-forward --address 0.0.0.0 pod/mypod 8888:5000
```

Service DNS

Kubernetes provides a DNS service exposed to Pods

System component running in and managed by K8s

```
kubectl run busybox --image=busybox --restart=Never \
-- /bin/sh -c "while true; do sleep 6000; done"
```

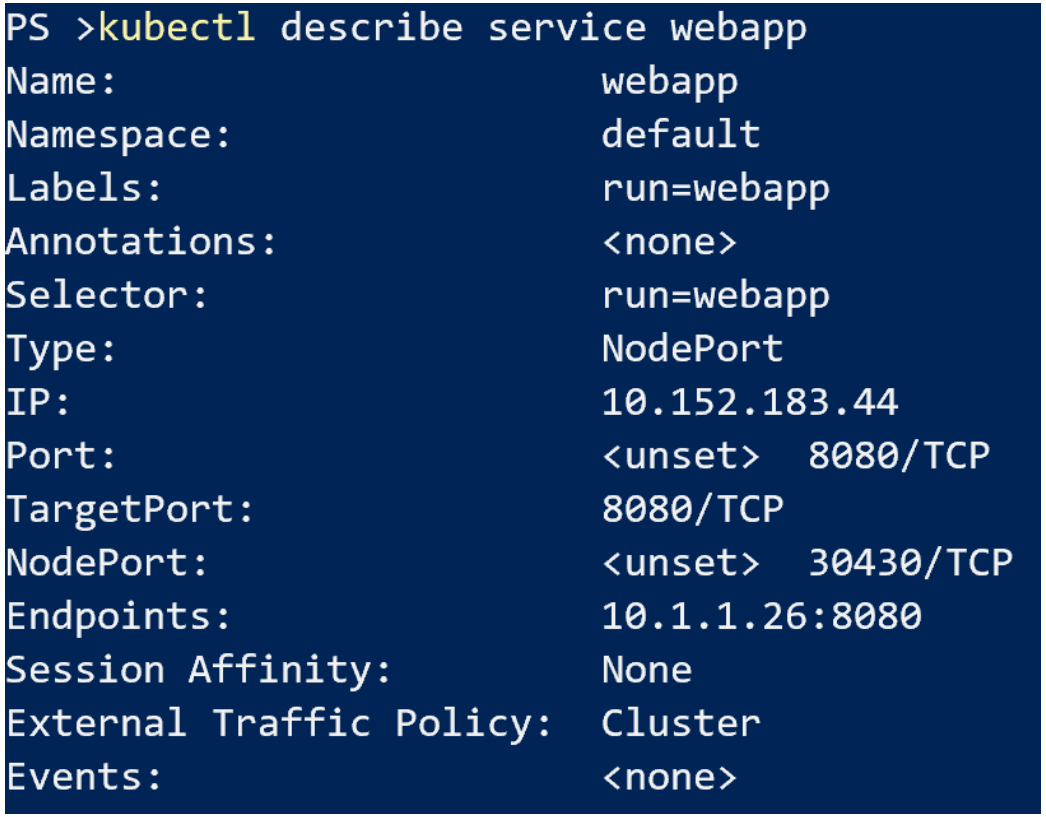


```
PS >kubectl exec -ti busybox -- nslookup webapp
Server:    10.152.183.10
Address 1: 10.152.183.10 kube-dns.kube-system.svc.cluster.local

Name:      webapp
Address 1: 10.152.183.44 webapp.default.svc.cluster.local
```

Kubectl edit service webapp

Change type to NodePort

Kubectl describe service webapp



```
PS >kubectl describe service webapp
Name:                     webapp
Namespace:                default
Labels:                   run=webapp
Annotations:              <none>
Selector:                 run=webapp
Type:                     NodePort
IP:                       10.152.183.44
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30430/TCP
Endpoints:                10.1.1.26:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

LoadBalancer

Only available if cloud supports it

Builds on NodePorts

Creates a loadbalancer in the cloud and direct it at nodes in cluster

Kubectl edit service webapp

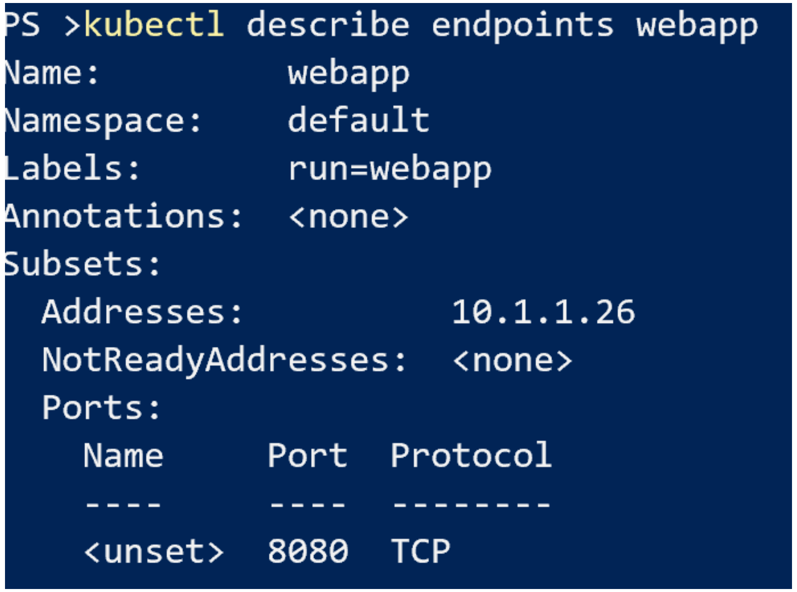
Change spec.type to LoadBalancer

Kubectl get services now gets an external ip (pending)

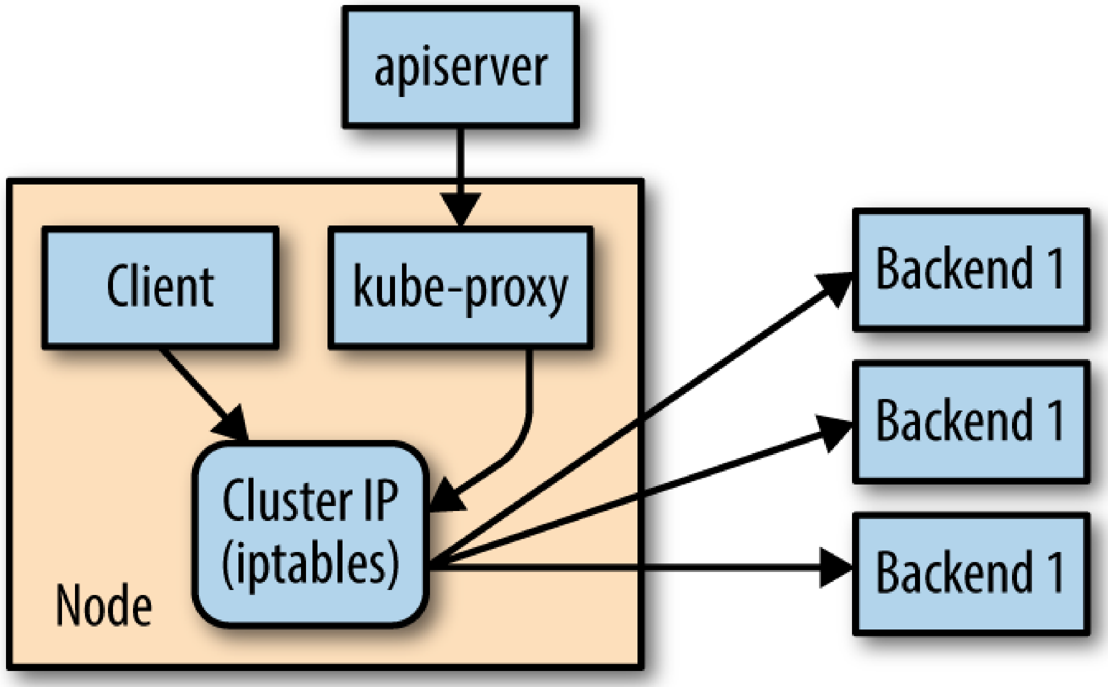
Endpoints

Endpoints are created as a pair with services

Kubectl describe endpoints webapp



```
PS >kubectl describe endpoints webapp
Name:         webapp
Namespace:    default
Labels:       run=webapp
Annotations:  <none>
Subsets:
  Addresses:          10.1.1.26
  NotReadyAddresses:  <none>
  Ports:
    Name     Port  Protocol
    ----     ----  --------
    <unset>  8080  TCP
```

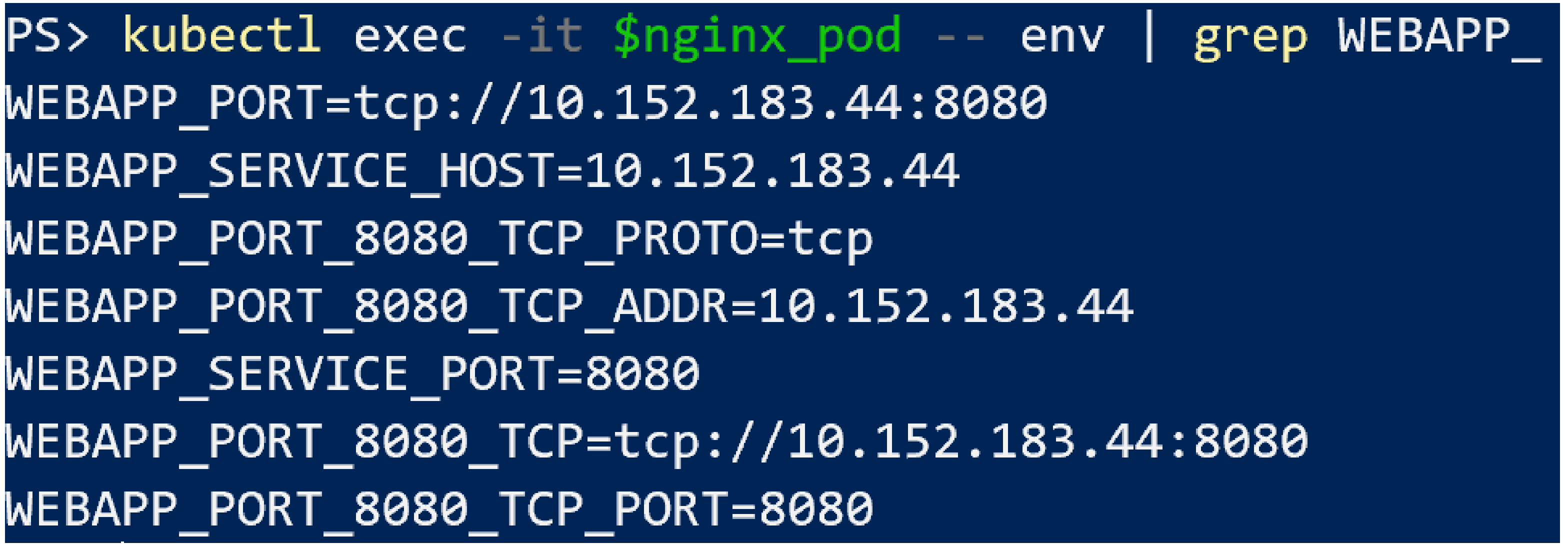


Kube-proxy and Cluster IPs

Cluster Ips are stable virtual IPs,

Load balance traffic across all endpoints of a service

Looking at injected environment



```
PS> kubectl exec -it $nginx_pod -- env | grep WEBAPP_
WEBAPP_PORT=tcp://10.152.183.44:8080
WEBAPP_SERVICE_HOST=10.152.183.44
WEBAPP_PORT_8080_TCP_PROTO=tcp
WEBAPP_PORT_8080_TCP_ADDR=10.152.183.44
WEBAPP_SERVICE_PORT=8080
WEBAPP_PORT_8080_TCP=tcp://10.152.183.44:8080
WEBAPP_PORT_8080_TCP_PORT=8080
```

Ingress

Ingress

Exposing service with http or https

Other services use:

- NodePort
- LoadBalancer

Uses an ingress controller

- Nginx-ingress
- HAProxy, Traefik, Istio, Kong,

Single service

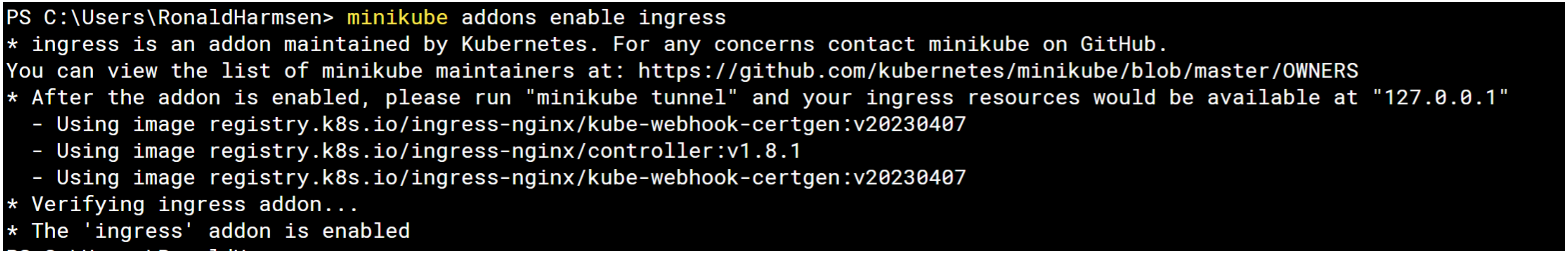
```yaml
apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

  name: test-ingress

spec:

  backend:

    serviceName: testsvc

    servicePort: 80
```

Ingress

```
apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

 name: test-ingress

 annotations:

   nginx.ingress.kubernetes.io/rewrite-target: /

spec:
```

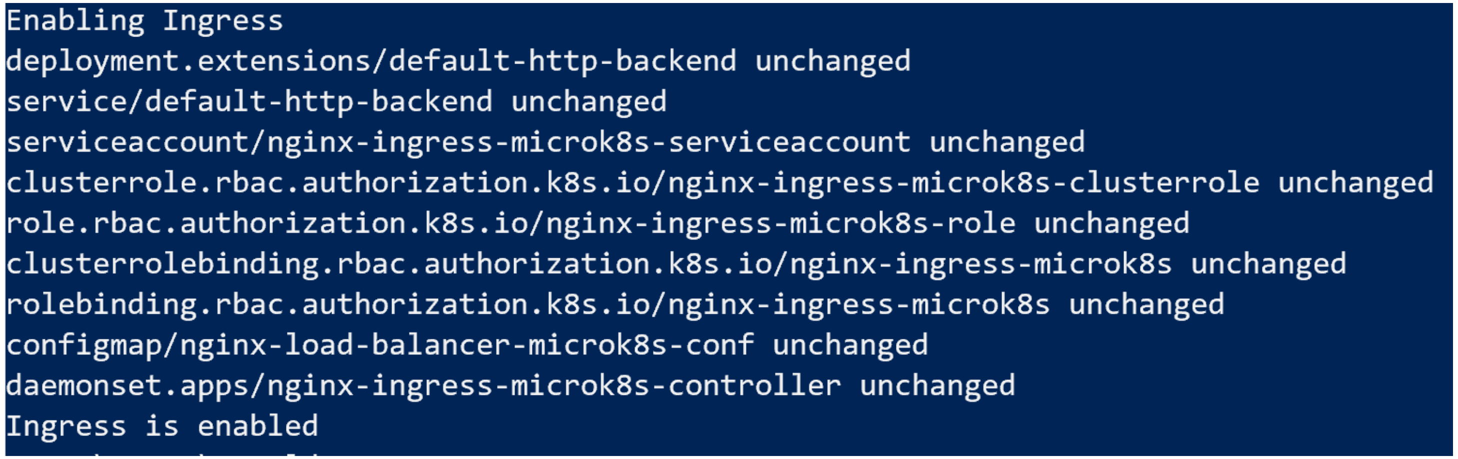
```
rules:

 - http:

   paths:

   - path: /testpath

    backend:

     serviceName: test

     servicePort: 80
```

```
minikube addons enable ingress
minikube addons enable ingress-dns
```



```
PS C:\Users\RonaldHarmsen> minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
* After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407
  - Using image registry.k8s.io/ingress-nginx/controller:v1.8.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

```
multipass exec microk8s-vm --
/snap/bin/microk8s.enable ingress
```



```
Enabling Ingress
deployment.extensions/default-http-backend unchanged
service/default-http-backend unchanged
serviceaccount/nginx-ingress-microk8s-serviceaccount unchanged
clusterrole.rbac.authorization.k8s.io/nginx-ingress-microk8s-clusterrole unchanged
role.rbac.authorization.k8s.io/nginx-ingress-microk8s-role unchanged
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-microk8s unchanged
rolebinding.rbac.authorization.k8s.io/nginx-ingress-microk8s unchanged
configmap/nginx-load-balancer-microk8s-conf unchanged
daemonset.apps/nginx-ingress-microk8s-controller unchanged
Ingress is enabled
```