

* ClusterIP
  + ClusterIP is the default and most common service type.
  + Kubernetes will assign a cluster-internal IP address to ClusterIP service. This makes the service only reachable within the cluster.
  + You cannot make requests to service (pods) from outside the cluster. You can optionally set cluster IP in the service definition file.
  + Use Cases
    - Inter-service communication within the cluster. For example, communication between the front-end and back-end components of your app.
* NodePort
  + NodePort service is an extension of ClusterIP service. A ClusterIP Service, to which the NodePort Service routes, is automatically created.
  + It exposes the service outside of the cluster by adding a cluster-wide port on top of ClusterIP.
  + NodePort exposes the service on each Node’s IP at a static port (the NodePort). Each node proxies that port into your Service. So, external traffic has access to fixed port on each Node. It means any request to your cluster on that port gets forwarded to the service.
  + You can contact the NodePort Service, from outside the cluster, by requesting :.
  + Node port must be in the range of 30000–32767. Manually allocating a port to the service is optional. If it is undefined, Kubernetes will automatically assign one.
  + If you are going to choose node port explicitly, ensure that the port was not already used by another service.
  + Use Cases
    - When you want to enable external connectivity to your service. Using a NodePort gives you the freedom to set up your own load balancing solution, to configure environments that are not fully supported by
    - Kubernetes, or even to expose one or more nodes’ IPs directly. Prefer to place a load balancer above your nodes to avoid node failure.
* LoadBalancer
  + LoadBalancer service is an extension of NodePort service. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created.
  + It integrates NodePort with cloud-based load balancers.
  + It exposes the Service externally using a cloud provider’s load balancer.
  + Each cloud provider (AWS, Azure, GCP, etc) has its own native load balancer implementation. The cloud provider will create a load balancer, which then automatically routes requests to your Kubernetes Service.
  + Traffic from the external load balancer is directed at the backend Pods. The cloud provider decides how it is load balanced.
  + The actual creation of the load balancer happens asynchronously.
  + Every time you want to expose a service to the outside world, you have to create a new LoadBalancer and get an IP address.
  + Use Cases
    - When you are using a cloud provider to host your Kubernetes cluster.
* ExternalName
  + Services of type ExternalName map a Service to a DNS name, not to a typical selector such as my-service.
  + You specify these Services with the spec.externalName parameter. It maps the Service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record with its value.
  + No proxying of any kind is established.
  + Use Cases
    - This is commonly used to create a service within Kubernetes to represent an external datastore like a database that runs externally to Kubernetes.
    - You can use that ExternalName service (as a local service) when Pods from one namespace talk to a service in another namespace.

apiVersion: v1  
kind: Service  
metadata:   
name: my-internal-service  
spec:  
selector:   
app: my-app  
type: ClusterIP  
ports:   
- name: http  
port: 80  
targetPort: 80  
protocol: TCP

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-service"

namespace: "default"

spec:

ports:

- port: 80

type: ClusterIP

selector:

app: "nginx"

apiVersion: v1  
kind: Service  
metadata:   
 name: my-nodeport-service  
spec:  
 selector:   
 app: my-app  
 type: NodePort  
 ports:   
 - name: http  
 port: 80  
 targetPort: 80  
 nodePort: 30036  
 protocol: TCP

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-service"

namespace: "default"

spec:

ports:

- port: 80

nodePort: 30001

type: NodePort

selector:

app: "nginx"

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-service"

namespace: "default"

spec:

ports:

- port: 80

type: LoadBalancer

selector:

app: "nginx"

---

apiVersion: v1

kind: Service

metadata:

name: "google-service"

namespace: "default"

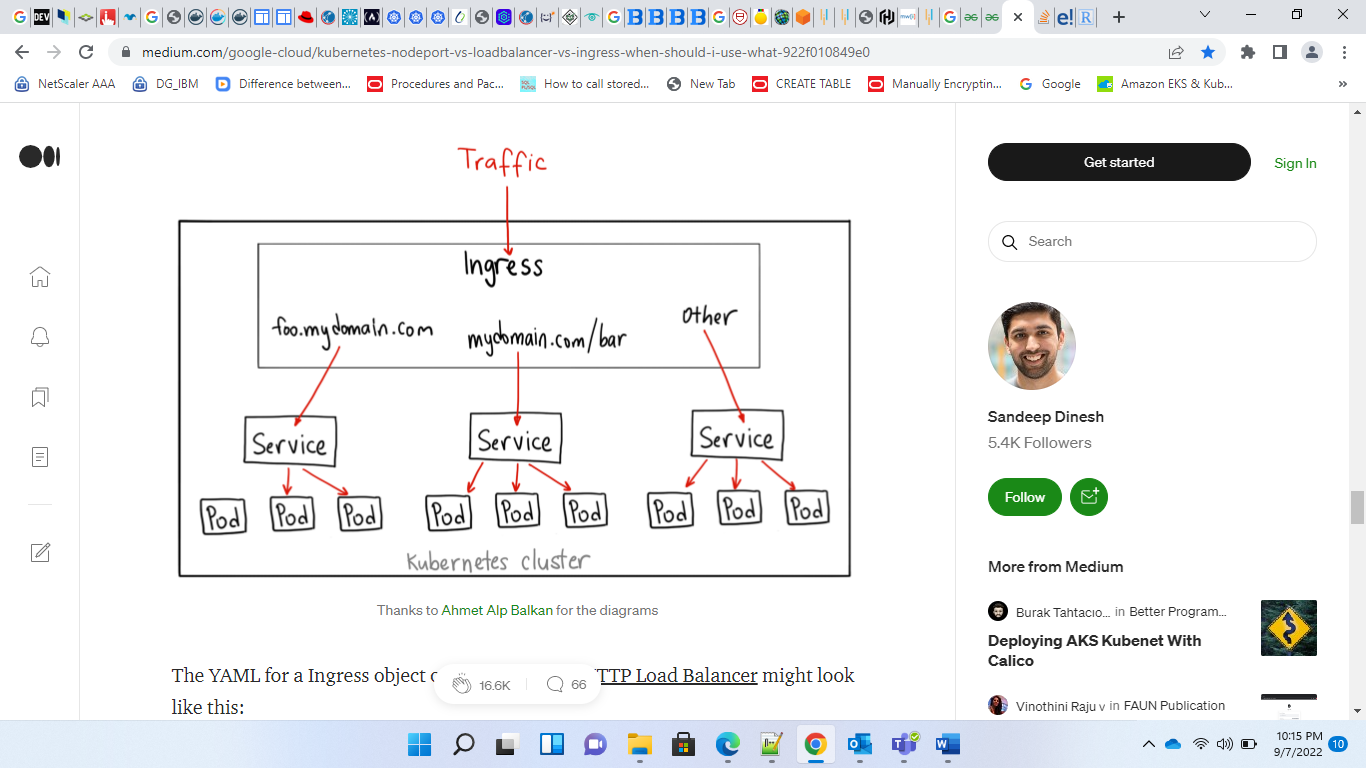
spec:

ports:

- port: 80

type: ExternalName

externalName: google.com



---

apiVersion: v1

kind: Service

metadata:

name: "nginx-service"

namespace: "default"

spec:

ports:

- port: 80

type: NodePort

selector:

app: "nginx"

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: "nginx-ingress"

annotations:

kubernetes.io/ingress.class: alb

alb.ingress.kubernetes.io/scheme: internet-facing

labels:

app: "nginx"

spec:

backend:

serviceName: "nginx-service"

servicePort: 80

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-1-service"

namespace: "default"

spec:

ports:

- port: 80

type: NodePort

selector:

app: "nginx-1"

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-2-service"

namespace: "default"

spec:

ports:

- port: 80

type: NodePort

selector:

app: "nginx-2"

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: "nginx-ingress"

annotations:

kubernetes.io/ingress.class: alb

alb.ingress.kubernetes.io/scheme: internet-facing

labels:

app: "nginx"

spec:

rules:

- http:

paths:

- path: /svc1.html

backend:

serviceName: "nginx-1-service"

servicePort: 80

- path: /svc2.html

backend:

serviceName: "nginx-2-service"

servicePort: 80

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-1-service"

namespace: "default"

spec:

ports:

- port: 80

type: NodePort

selector:

app: "nginx-1"

---

apiVersion: v1

kind: Service

metadata:

name: "nginx-2-service"

namespace: "default"

spec:

ports:

- port: 80

type: NodePort

selector:

app: "nginx-2"

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: "nginx-ingress"

annotations:

kubernetes.io/ingress.class: alb

alb.ingress.kubernetes.io/scheme: internet-facing

labels:

app: "nginx"

spec:

rules:

- host: "svc1.example.com"

http:

paths:

- backend:

serviceName: "nginx-1-service"

servicePort: 80

- host: "svc2.example.com"

http:

paths:

- backend:

serviceName: "nginx-2-service"

servicePort: 80

- http:

paths:

- backend:

serviceName: "nginx-1-service"

servicePort: 80

apiVersion: extensions/v1beta1  
kind: Ingress  
metadata:  
name: my-ingress  
spec:  
backend:  
serviceName: other  
servicePort: 8080  
rules:  
- host: foo.mydomain.comhttp:  
paths:  
- backend:  
serviceName: foo  
servicePort: 8080- host: mydomain.com  
http:  
paths:  
- path: /bar/\*  
backend:  
serviceName: bar  
servicePort: 8080

<https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0>

<https://stackoverflow.com/questions/41509439/whats-the-difference-between-clusterip-nodeport-and-loadbalancer-service-types#:~:text=ClusterIP%20(default)%3A%20Internal%20clients,of%20a%20network%20load%20balancer>.

<https://www.edureka.co/community/19351/clusterip-nodeport-loadbalancer-different-from-each-other>

<https://rtfm.co.ua/en/kubernetes-clusterip-vs-nodeport-vs-loadbalancer-services-and-ingress-an-overview-with-examples/>