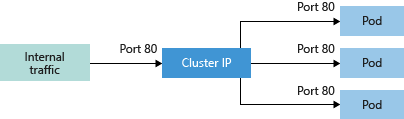
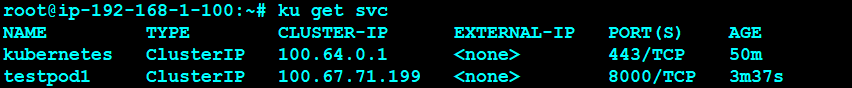
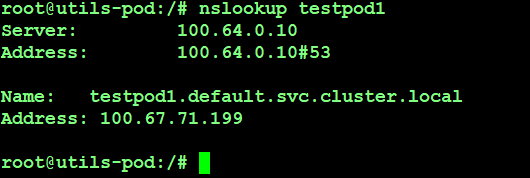
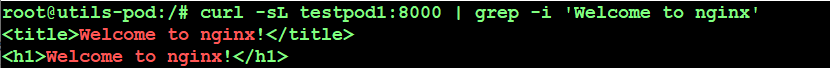
**CLUSTERIP SERVICE:**

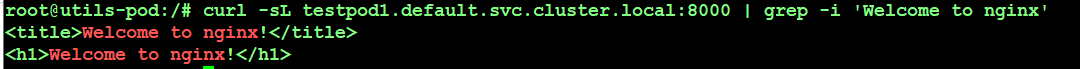
ClusterIP: Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default ServiceType.











---

#Cluster-IP Service

apiVersion: v1

kind: Service

metadata:

labels:

run: testpod1

name: testpod1

spec:

ports:

- port: 8000 #This is the port the service listens.

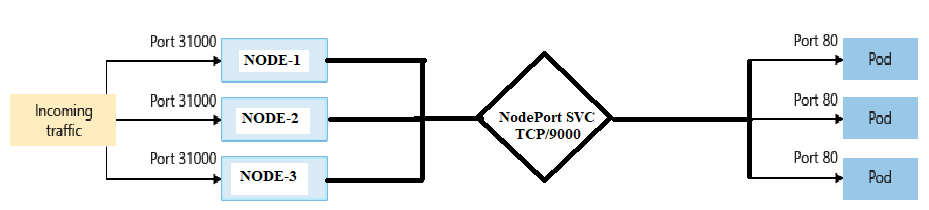
protocol: TCP

targetPort: 80 #Application Port Running on the container.

selector:

run: testpod1

**NODEPORT SERVICE:**

****

NodePort - Creates a port mapping on the underlying node that allows the application to be accessed directly with the node IP address and port.

---

#NodePort Service

apiVersion: v1

kind: Service

metadata:

labels:

run: deploy01

name: deploy01

spec:

ports:

- port: 9000

protocol: TCP

targetPort: 80

nodePort: 31000 #This can be deleted when u want K8S to assign random port.

selector:

run: deploy01

type: NodePort

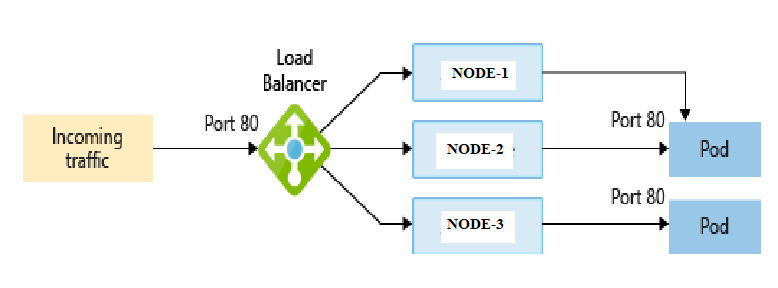
**LOADBALANCER SERVICE:**

LoadBalancer - Creates an Azure/AWS/GCP load balancer resource, configures an external IP address, and connects the requested pods to the load balancer backend pool. To allow customers' traffic to reach the application, load balancing rules are created on the desired ports.

Exposes the Service externally using a cloud provider's load balancer.

Using External Load Balancers per service can become costly and it's not a recommended practise.

The recommended practise is using a Ingress Controller.You can also use [Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/) to expose your Service. Ingress is not a Service type, but it acts as the entry point for your cluster. It lets you consolidate your routing rules into a single resource as it can expose multiple services under the same IP address



---

#Below code will deploy aws classic LB with K8s version v1.18.12

apiVersion: v1

kind: Service

metadata:

labels:

run: deploy01

name: deploy01

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

run: deploy01

type: LoadBalancer

---

#Below code will deploy aws Network LB with K8s version v1.18.12

apiVersion: v1

kind: Service

metadata:

labels:

run: deploy01

name: awsnlb-deploy01

annotations:

service.beta.kubernetes.io/aws-load-balancer-type: nlb

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

run: deploy01

type: LoadBalancer

**EXTERNALNAME SERVICE:**

ExternalName - Creates a specific DNS entry for easier application access. 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 set up.

apiVersion: v1

kind: Service

metadata:

name: google-svc

spec:

type: ExternalName

externalName: www.google.com

ports:

- port: 80

name: http

targetPort: 80

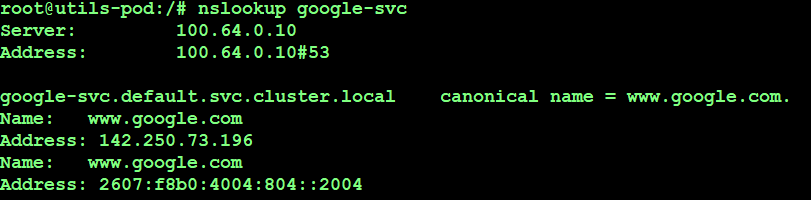
protocol: TCP

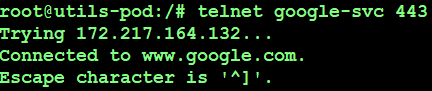
- port: 443

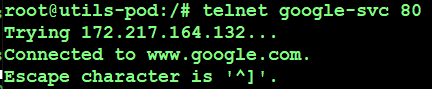
name: htts

targetPort: 443

protocol: TCP







**SERVICE FOR EXTERNAL IP ADDRESS:**

---

apiVersion: v1

kind: Service

metadata:

name: mysql

spec:

clusterIP: None

selector:

app: mysql

ports:

- name: mysql

protocol: TCP

port: 3306

targetPort: 3306

---

kind: Endpoints

apiVersion: v1

metadata:

name: mysql

subsets:

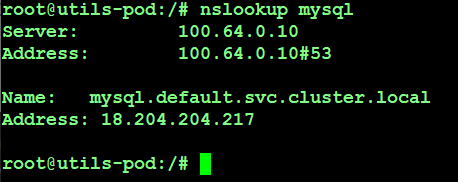
- addresses:

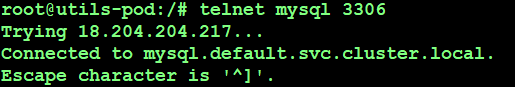
- ip: 18.204.204.217

ports:

- port: 3306

name: mysql





**HEADLESS ADDRESS:**

Sometimes you don't need load-balancing and a single Service IP. In this case, you can create what are termed "headless" Services, by explicitly specifying "None" for the cluster IP (.spec.clusterIP).

Headless Service in general used for stateful sets in Kubernetes.

---

#Headless Service

apiVersion: v1

kind: Service

metadata:

labels:

run: deploy01

name: deploy01

spec:

clusterIP: None

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

run: deploy01

