Exposing Services in Kubernetes: ClusterIP, NodePort & LoadBalancer

Step 1: Deploy a Sample Application

Let's use the Nginx demo app for illustration:

```
kubectl create deployment demo-app --image=nginxdemos/hello
```

Step 2: Expose Service using ClusterIP (Default)

```
kubectl expose deployment demo-app --type=ClusterIP --port=80 --target-port=80
```

Verify the service:

```
kubectl get svc
```

Notes:

- Default service type.
- Accessible **only inside the cluster** (via DNS or IP).
- Useful for internal communication between pods.

Step 3: Expose Service using NodePort

```
kubectl expose deployment demo-app --type=NodePort --port=80 --target-port=80
```

Get the assigned NodePort:

```
kubectl get svc demo-app
```

Look for a port number between **30000-32767** in the PORT(S) column, e.g.:

```
80:31234/TCP
```

Access the app:

```
http://<Node-IP>:<NodePort>
```

If you're using Minikube:

```
minikube service demo-app --url
```

Notes:

- Exposes app to outside world via a port on each node.
- Works on AKS, Minikube, or other clusters.

Step 4: Expose Service using LoadBalancer

```
kubectl expose deployment demo-app --type=LoadBalancer --port=80 --target-port=80
```

Check external IP:

```
kubectl get svc demo-app
```

Wait for the EXTERNAL-IP to be assigned.

Access the app:

http://<EXTERNAL-IP>

Notes:

- Best for **production-ready access** via cloud load balancer.
- Supported by cloud providers like Azure, AWS, GCP.

Step 5: Clean Up

```
kubectl delete service demo-app
kubectl delete deployment demo-app
```

Summary of Service Types

Type	**Visibility**	**Description**
ClusterIP	Internal Only	Default; accessible only within cluster
NodePort	External via Node IP	Exposes service on a static port on each node
LoadBalancer	External via LB IP	Exposes externally via cloud provider load balancer