Kubernetes: Create Pod with kubectl and expose it

1. Create a simple Pod (Nginx example)

kubectl run nginx-pod --image=nginx:alpine --restart=Never

Explanation:

- kubectl run : quick way to create a Pod
- nginx-pod : name of the Pod
- --image=nginx:alpine : lightweight Nginx image
- --restart=Never : forces creation of a Pod (not a Deployment)

Verify:

kubectl get pods

2. Expose the Pod as a Service

Option A: ClusterIP (internal only)

kubectl expose pod nginx-pod --port=80 --name=nginx-service

Option B: NodePort (accessible from outside)

kubectl expose pod nginx-pod --port=80 --type=NodePort --name=nginx-service-nodeport

What the expose command does:

- Creates a Service object

- Forwards traffic to port 80 on the Pod

- NodePort assigns a high port (30000-32767) on every node

Check services:

kubectl get svc

kubectl describe svc nginx-service

Access examples:

- ClusterIP : only from inside cluster → curl http://nginx-service

- NodePort : from outside → http://<any-node-ip>:<node-port>

3. Best practice: Use YAML files (recommended for production)

pod.yaml

```
---------
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:alpine
    ports:
    - containerPort: 80
---------
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx          # matches Pod label
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: ClusterIP       # or NodePort / LoadBalancer
```

Apply:

kubectl apply -f pod.yaml

kubectl apply -f service.yaml

4. Quick one-liner summary

kubectl run nginx-pod --image=nginx:alpine --restart=Never

kubectl expose pod nginx-pod --port=80 --type=NodePort --name=nginx-svc

kubectl get pods,svc

Note

Standalone Pods are not self-healing. In real apps, use Deployment + Service instead.

Done! You now have a running Nginx Pod exposed via a Kubernetes Service.