To access your **pod** ***webapp-6f7dc9dcbf-hpbsc*** from outside the Kubernetes cluster, you need to expose it through a service. Here are the general steps for achieving that:

**1. Create a Kubernetes Service**

You can expose the pod using a Service. If you want to make it accessible outside the cluster, you will need a LoadBalancer, NodePort, or Ingress depending on your infrastructure.

**a. Service with NodePort (for basic external access) – It is free**

This method is simpler and works if you just need to access the pod externally using the node's IP and a specific port.

1. **Create a NodePort service** to expose your pod's port (8080 in this case):

apiVersion: v1

kind: Service

metadata:

name: webapp-service

spec:

selector:

app: webapp

ports:

- protocol: TCP

port: 8080 # The port inside the cluster

targetPort: 8080 # The port on the container

nodePort: 30007 # The external port to access

type: NodePort

Save this YAML file and apply it:

kubectl apply -f webapp-service.yaml

Now, you can access your pod at:

<NodeIP>:30007

Replace <NodeIP> with the IP address of one of your nodes, and you should be able to access the webapp on port 8080 through the external port 30007.

**b. Service with LoadBalancer (if on cloud provider) – It is Paid**

If you're using a cloud provider like AWS, GCP, or Azure, you can create a LoadBalancer service to automatically provision an external load balancer.

1. **Create a LoadBalancer service:**

apiVersion: v1

kind: Service

metadata:

name: webapp-service

spec:

selector:

app: webapp

ports:

- protocol: TCP

port: 8080 # The port inside the cluster

targetPort: 8080 # The port on the container

type: LoadBalancer

Apply it:

kubectl apply -f webapp-service-lb.yaml

After a few moments, you can check the external IP assigned to the service:

kubectl get svc webapp-service

The output will show an external IP in the EXTERNAL-IP column. You can access your app at that IP on port 8080.

**c. Service with Ingress (if using Ingress Controller) – It is free**

If you're using an Ingress controller in your cluster, you can set up an Ingress resource to expose your application. This is typically useful when you need to manage multiple services with custom domain routing.

1. **Create an Ingress resource:**

First, install Ingress NGINX on OCI.

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml

Here's a basic example assuming you're using an Ingress controller like NGINX:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: webapp-ingress

spec:

rules:

- host: webapp.example.com # Set your domain name here

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: webapp-service

port:

number: 8080

Apply it:

kubectl apply -f webapp-ingress.yaml

Ensure you have configured DNS to point to your Ingress controller's external IP.

**2. Verify and Test Access**

Once you have exposed the service:

* For NodePort: Access the app at http://<NodeIP>:<NodePort>.
* For LoadBalancer: Access the app at http://<ExternalIP>:8080.
* For Ingress: Access it at http://webapp.example.com/ (after DNS is configured).

Let me know if you need help with any of the steps!