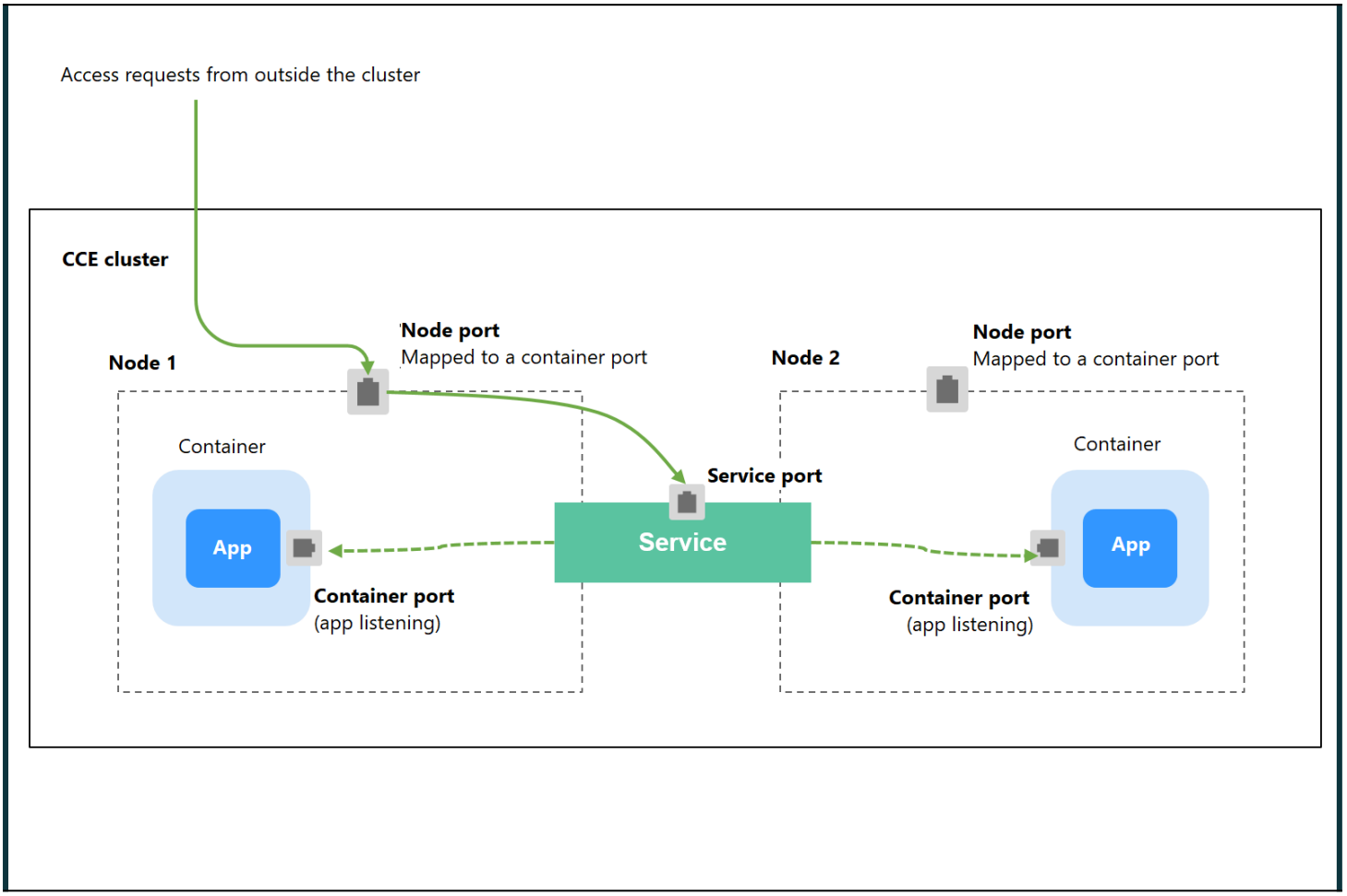
KUBERNETES

NodePort Service :

To access a web page externally in Kubernetes using the NodePort concept, follow these step-by-step instructions. I will also explain the purpose of each step to help you understand how the NodePort works.

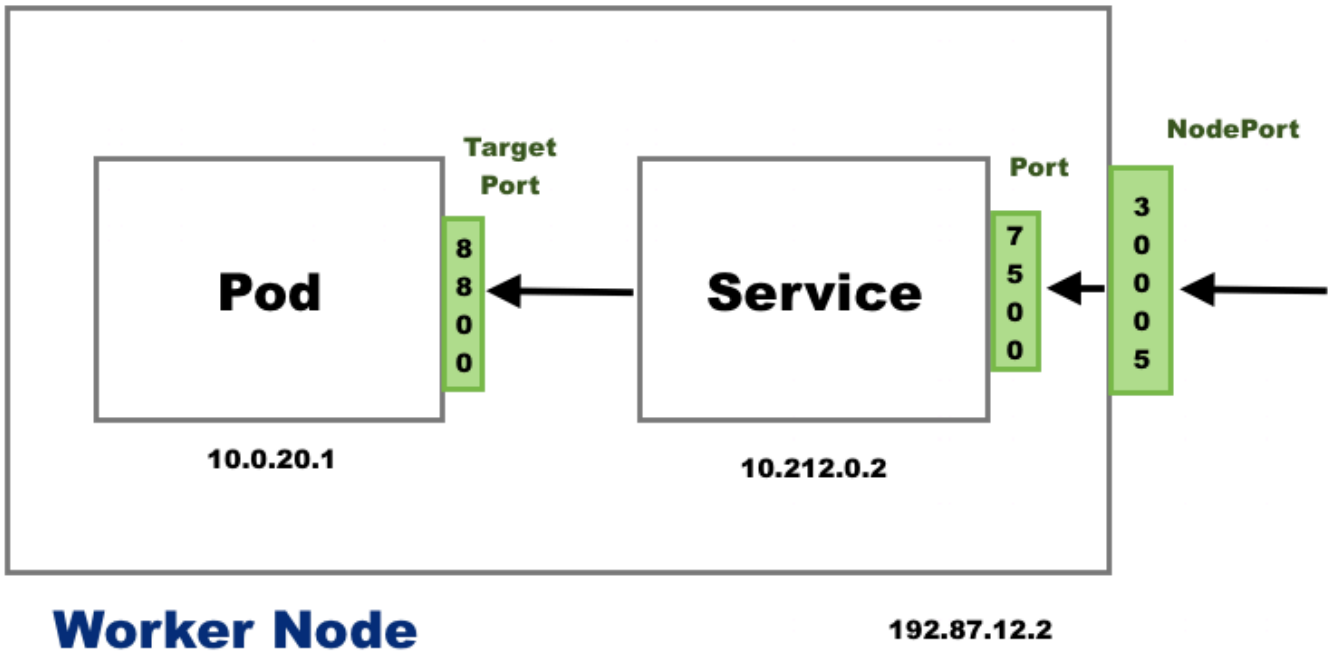


Kajal bagal

NodePort: A type of service that exposes a specific port on each node of your Kubernetes cluster. You can access the service from outside the cluster by using `<Node_IP>:<NodePort>`.

Pods: These are the smallest deployable units in Kubernetes, running your application.

Services: Kubernetes resources that allow communication between components of your application, and in this case, expose your application to the outside world using a NodePort.



NodePort Basics:

- The `NodePort` type exposes a service outside the Kubernetes cluster by allocating a port in the range 30000-32767 (by default).
- The service can be accessed from outside the cluster using `<NodeIP>:NodePort`.

Custom Port Selection:

- You can either let Kubernetes assign a random port or specify your own port using the `nodePort` field.
- If you pick a custom port, make sure it's within the NodePort range (30000-32767) and doesn't conflict with an already assigned port.

Port Allocation Policy:

- The NodePort range is split into two bands: one for dynamic (auto) assignment and another for manual assignment, reducing the chances of port collisions.

Accessing the Service:

- The service is accessible from outside the cluster via any node in the cluster using the node's IP and the NodePort.

Custom IP Address:

- You can configure a node to use a specific IP address for serving NodePort traffic using the `--nodeport-addresses` flag in the `kube-proxy` settings.

1. Create a Deployment or Pod (Your Web Application)

First, define a YAML file (`kajal.yaml`) that will create your web application (e.g., Nginx, Apache). The purpose here is to deploy your web application as a pod in your Kubernetes cluster.

Example `kajal.yaml` (for an Nginx web server):

Editor Tab 1 + 55 min

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webpod1
  labels:
    role: myrole
spec:
  containers:
    - name: webapp
      image: nginx
      ports:
        - containerPort: 80
---
apiVersion: v1
kind: Pod
metadata:
  name: webpod2
  labels:
    role: myrole
spec:
  containers:
    - name: webapp
      image: nginx
      ports:
        - containerPort: 80
---
```

Purpose:

- **Deployment**: Manages the creation and scaling of your Pods.
- **ContainerPort 80**: Exposes port 80 inside the container, where the web server listens.

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: webpod3
  labels:
    role: myrole
spec:
  containers:
    - name: webapp
      image: nginx
      ports:
        - containerPort: 80
```

2. Apply the YAML file to Create the Deployment

After defining the YAML file, apply it to create the deployment.

kubectl create -f kajal.yaml

```
controlplane $ vi kajal.yaml
controlplane $ kubectl create -f kajal.yaml
pod/webpod1 created
pod/webpod2 created
pod/webpod3 created
```

You can check if the pod is running with:

```
controlplane $ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
webpod1   1/1     Running   0          10s
webpod2   1/1     Running   0          10s
webpod3   1/1     Running   0          10s
```

3. Create a Service Using NodePort

Next, create a service that exposes your pod to external traffic using the NodePort type. You'll define this in `svc.yaml`.

Apply the Service YAML

Apply the service configuration to create the NodePort service:

kubectl create -f svc.yaml

```
controlplane $ vi svc.yaml
controlplane $ kubectl create -f svc.yaml
service/my-service created
controlplane $ kubectl get svc
```

Editor Tab 1 + 53 mi

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    role: myrole
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Alternatively, you can use this code as well, so there's no need to manually edit the service file.

```yaml
apiVersion: v1

kind: Service

metadata:

 name: my-service

spec:

 type: NodePort

 selector:

   myrole: MyApp

 ports:

   - port: 80

     # By default and for convenience, the `targetPort` is set to
```

```
    # the same value as the `port` field.
    targetPort: 80
    # Optional field
    # By default and for convenience, the Kubernetes control plane
    # will allocate a port from a range (default: 30000-32767)
    nodePort: 30007
```

You can verify the service with:

```
controlplane $ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP   21d
my-service   ClusterIP   10.111.71.217   <none>        80/TCP    10s
```

Editing the Service

If you need to change the configuration of your service, such as the `nodePort` or other settings, you can edit the service using:

```
controlplane $ kubectl edit svc my-service
service/my-service edited
```

```
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - nodePort: 31416
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    role: myrole
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
"/tmp/kubectl-edit-2826935944.yaml" 32L, 761C
```

You can verify the service with:

```
controlplane $ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        21d
my-service   NodePort    10.111.71.217   <none>        80:31416/TCP   70s
```

Check Service and Pod Details

You can check the labels and describe the service for more details:

To show the pod labels:

```
controlplane $ kubectl get pod --show-labels
NAME      READY   STATUS    RESTARTS   AGE     LABELS
webpod1   1/1     Running   0          7m55s   role=myrole
webpod2   1/1     Running   0          7m55s   role=myrole
webpod3   1/1     Running   0          7m55s   role=myrole
controlplane $
```

To describe the service:

kubectl describe svc my-service

```
controlplane $ kubectl describe svc my-service
Name:                     my-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 role=myrole
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.111.71.217
IPs:                      10.111.71.217
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31416/TCP
Endpoints:                192.168.1.4:80,192.168.1.5:80,192.168.1.6:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
controlplane $
```

Final result: