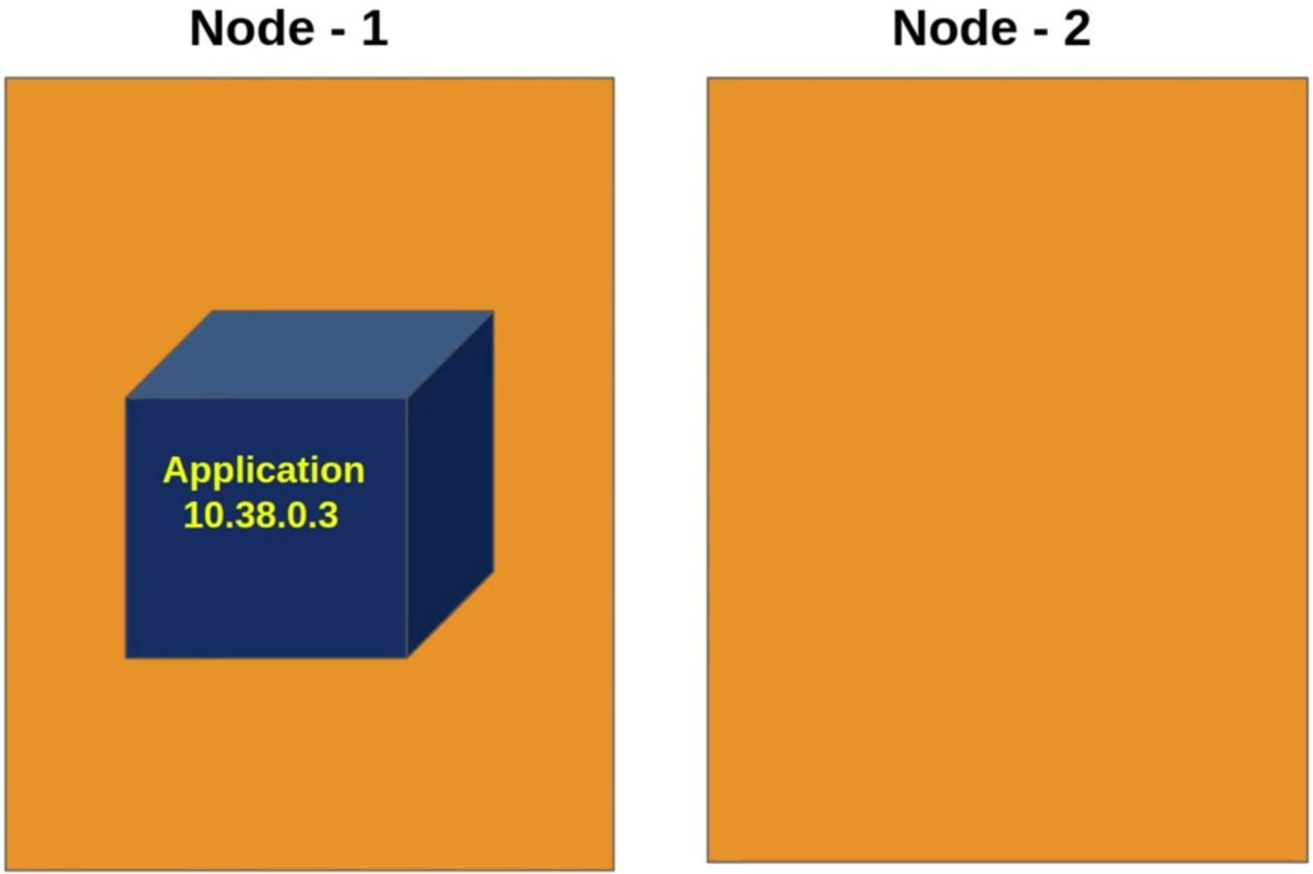
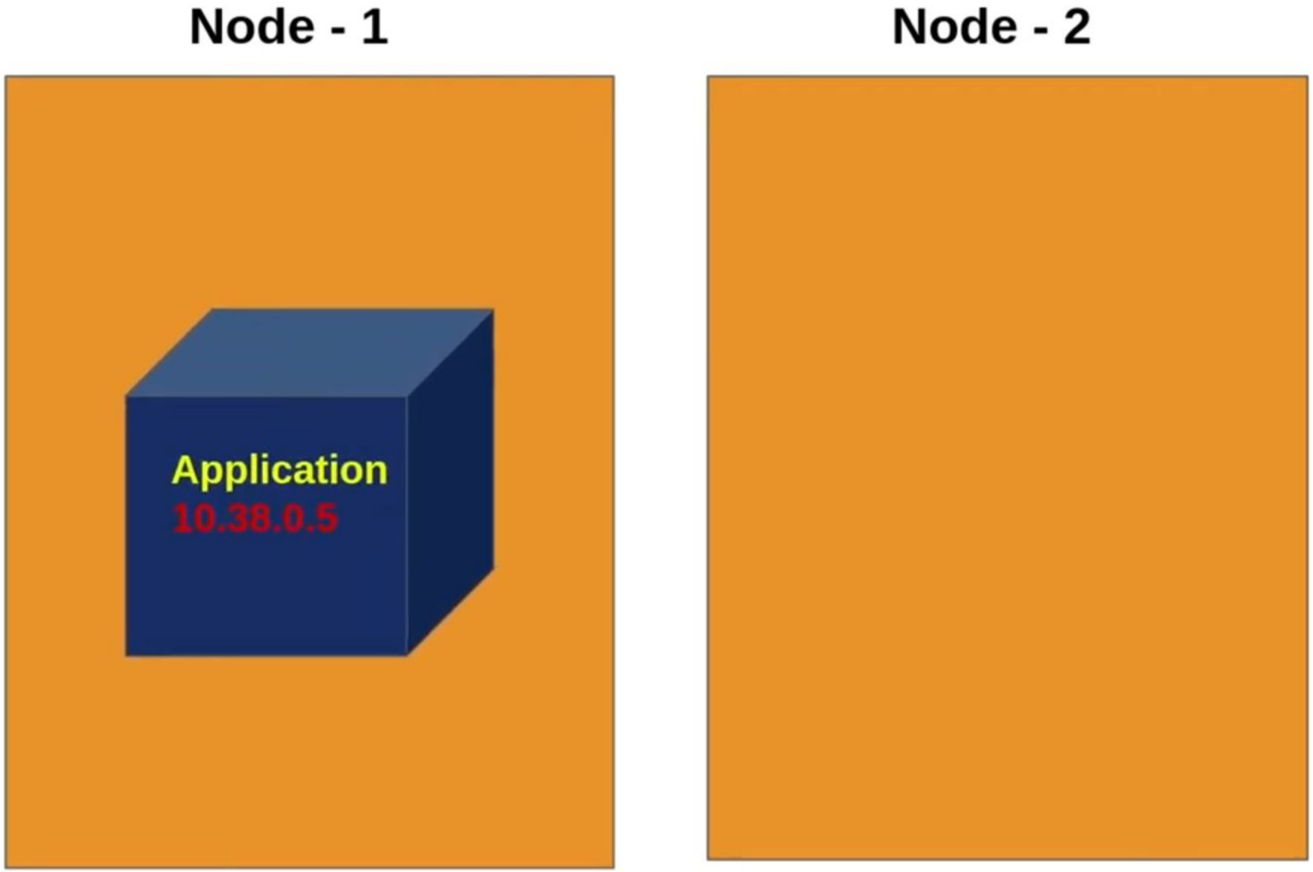


Node - 1

Node - 2

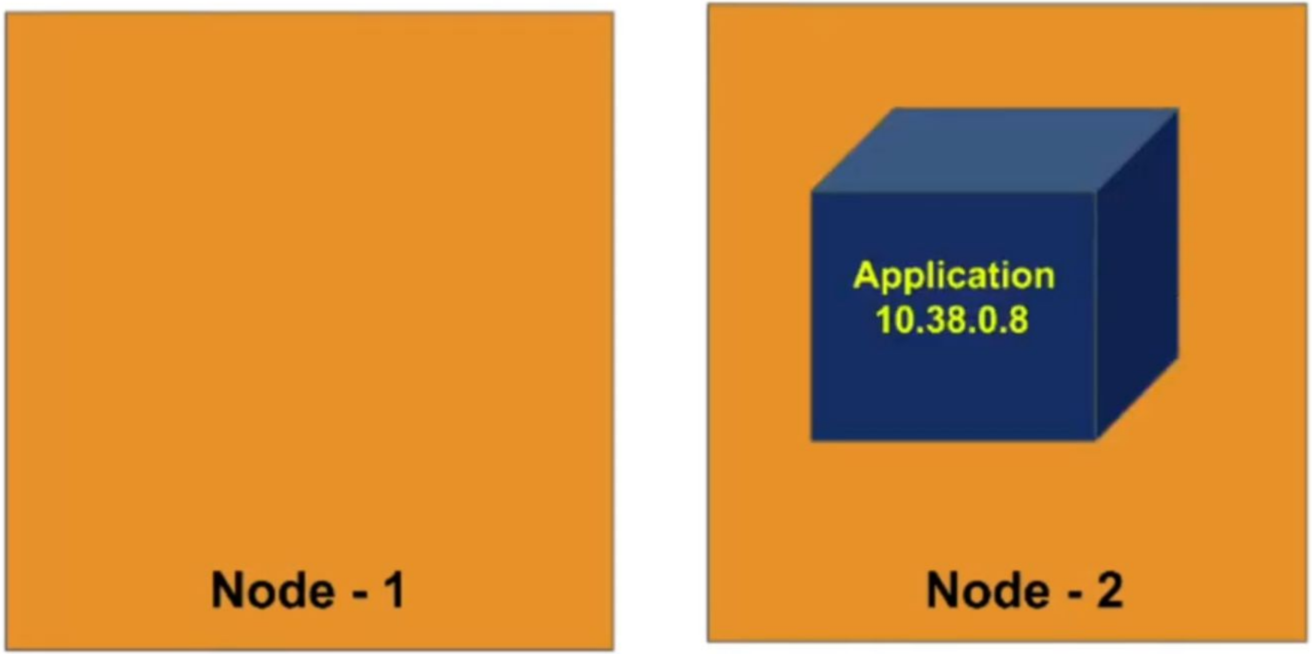
Application
10.38.0.3



Node - 1

Node - 2

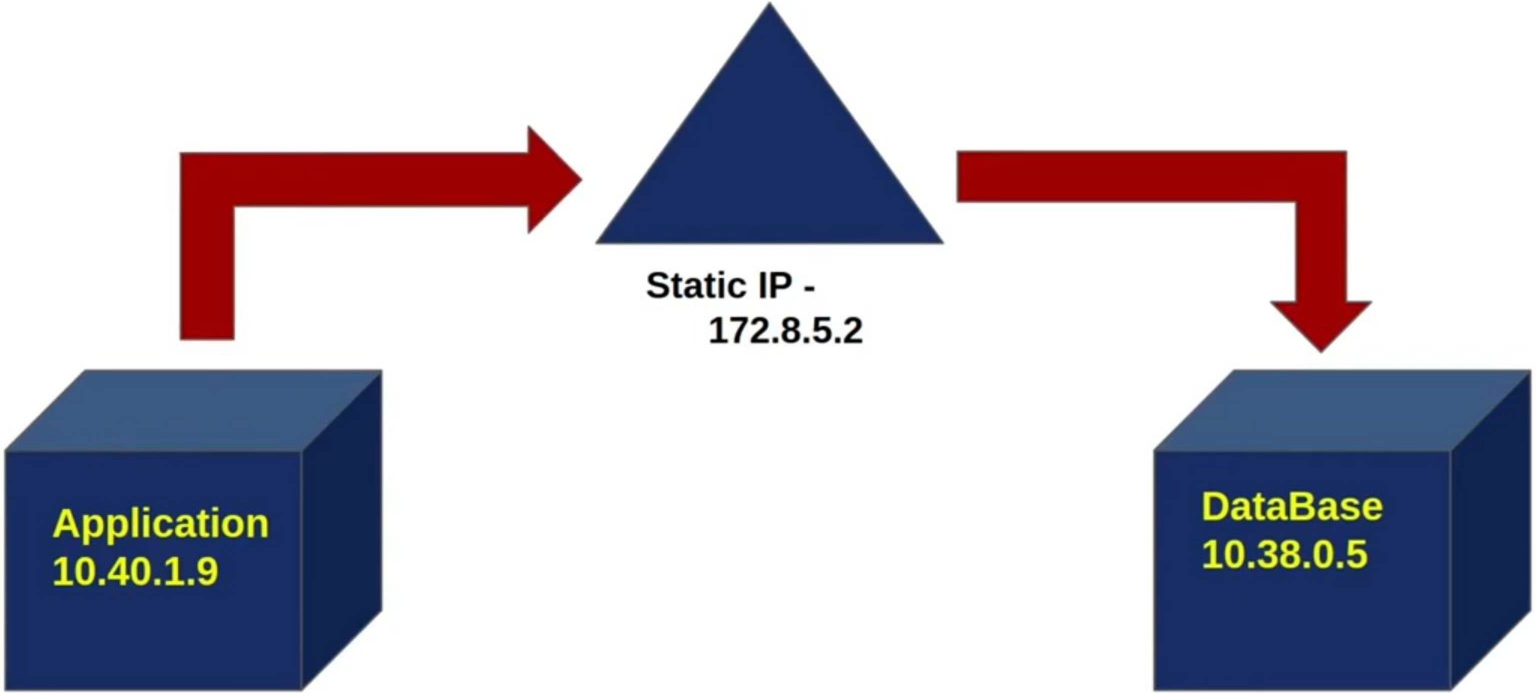
Application
10.38.0.5


Application
10.38.0.8
Node - 1
Node - 2

How to access the application running in the pod?

Service

Help to communicate



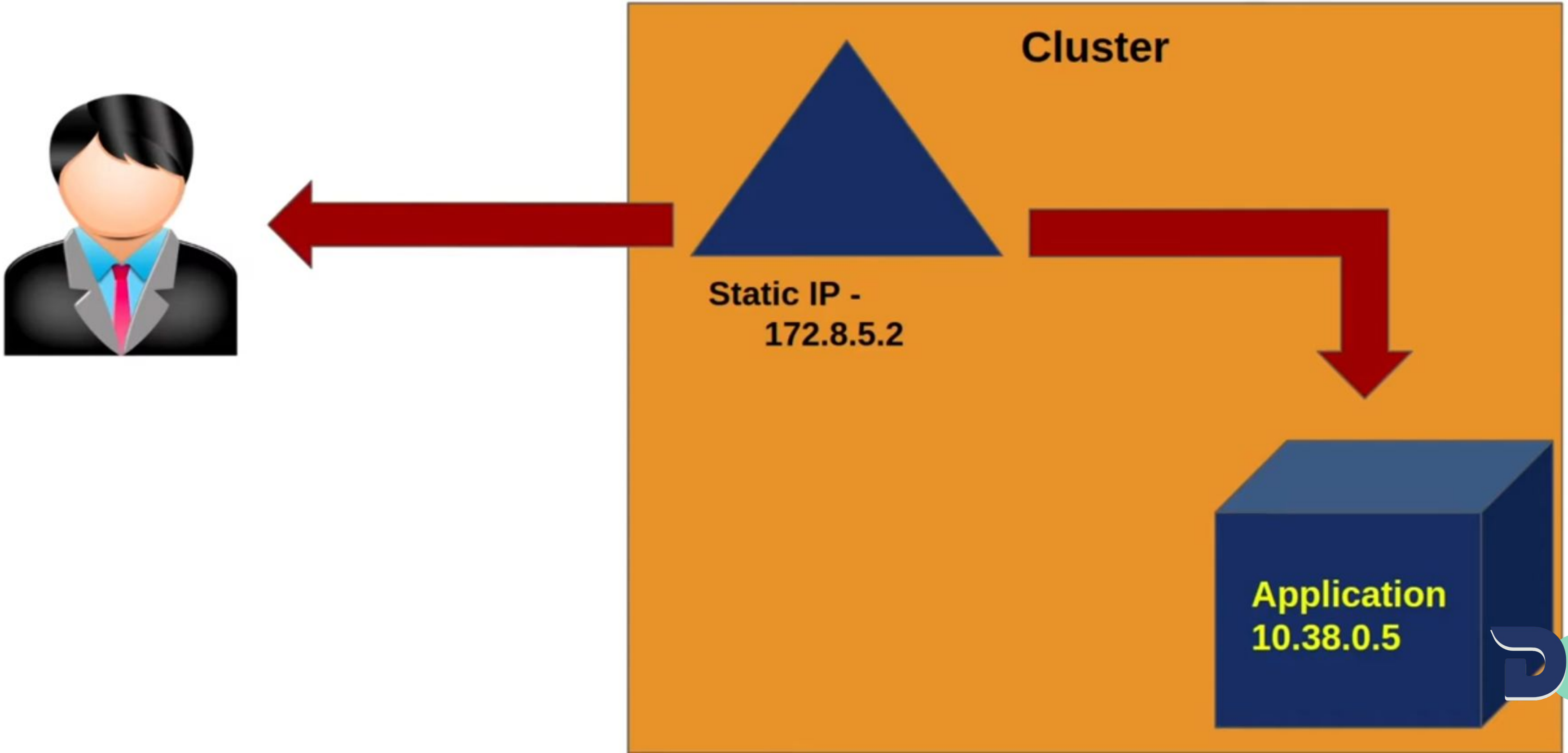
Static IP -
172.8.5.2

Application
10.40.1.9

DataBase
10.38.0.5

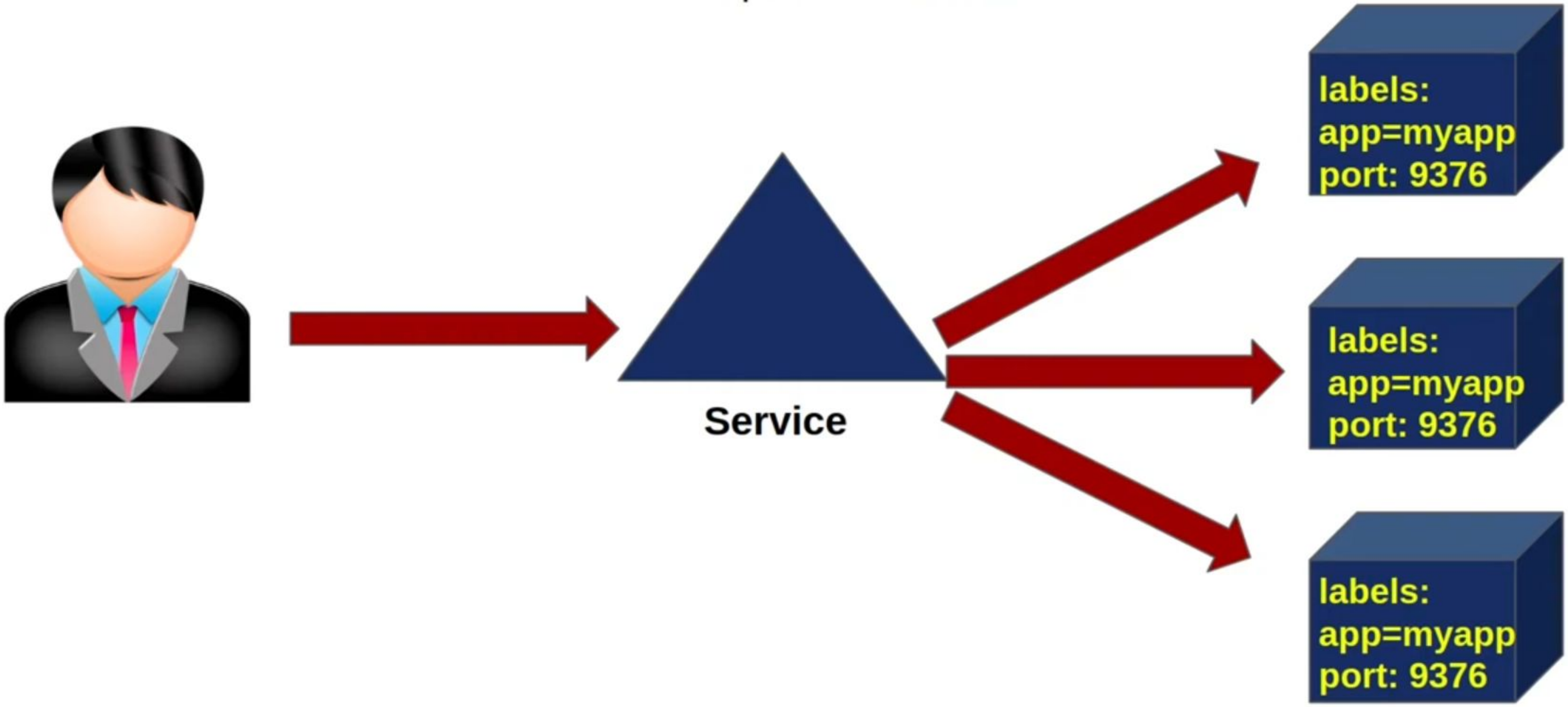
Service

Accessing application outside the cluster



Concept Of Service

Equal Distribution



Types Of Services

NodePort

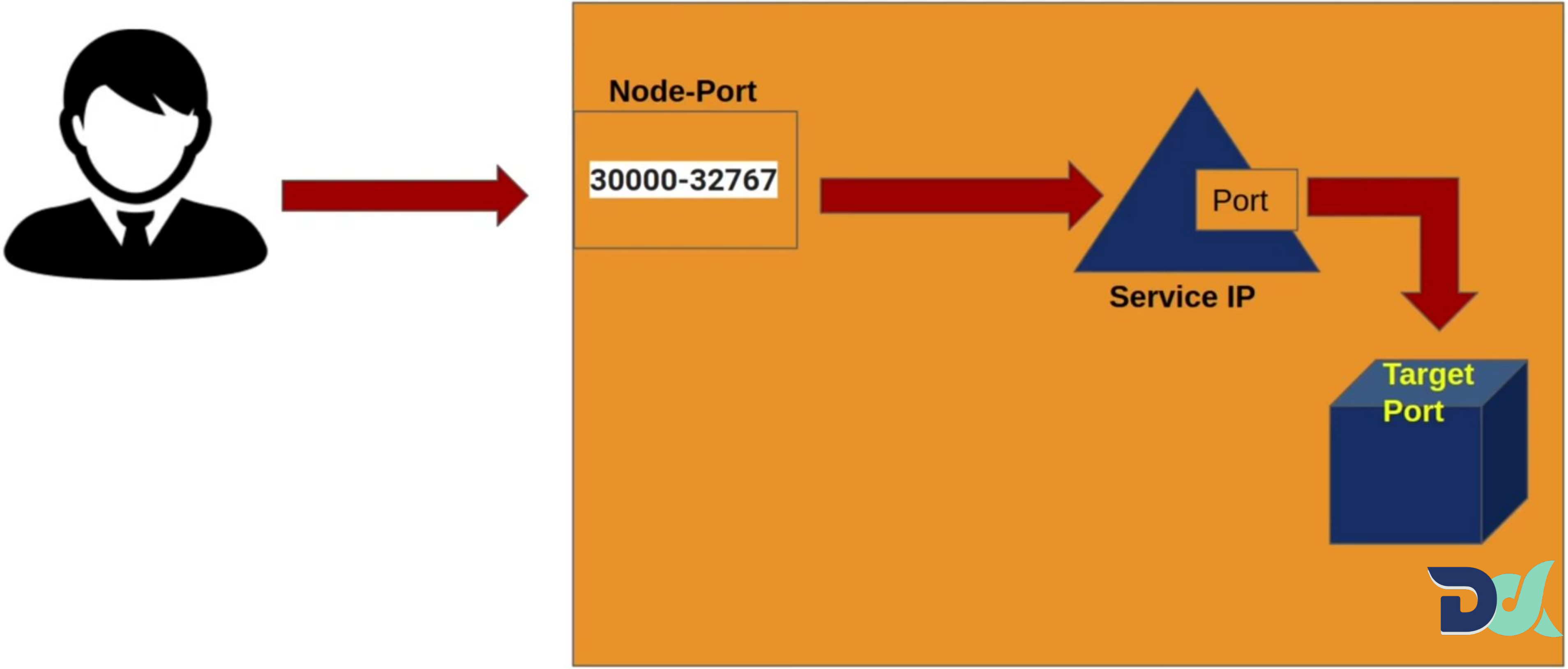
ClusterIP

Load-Balancer

ExternalName

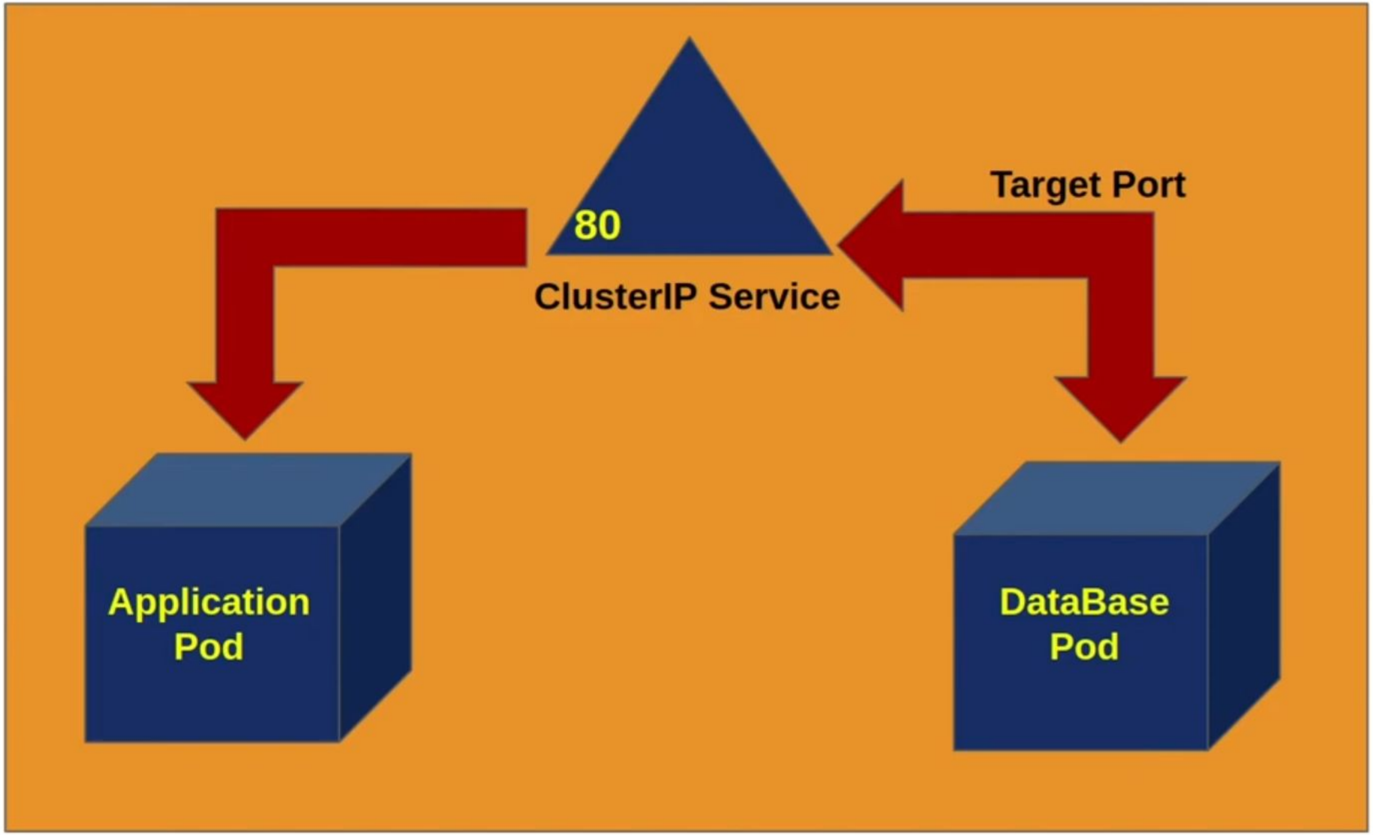
NodePort Service

Accessible from the outside but only in same network



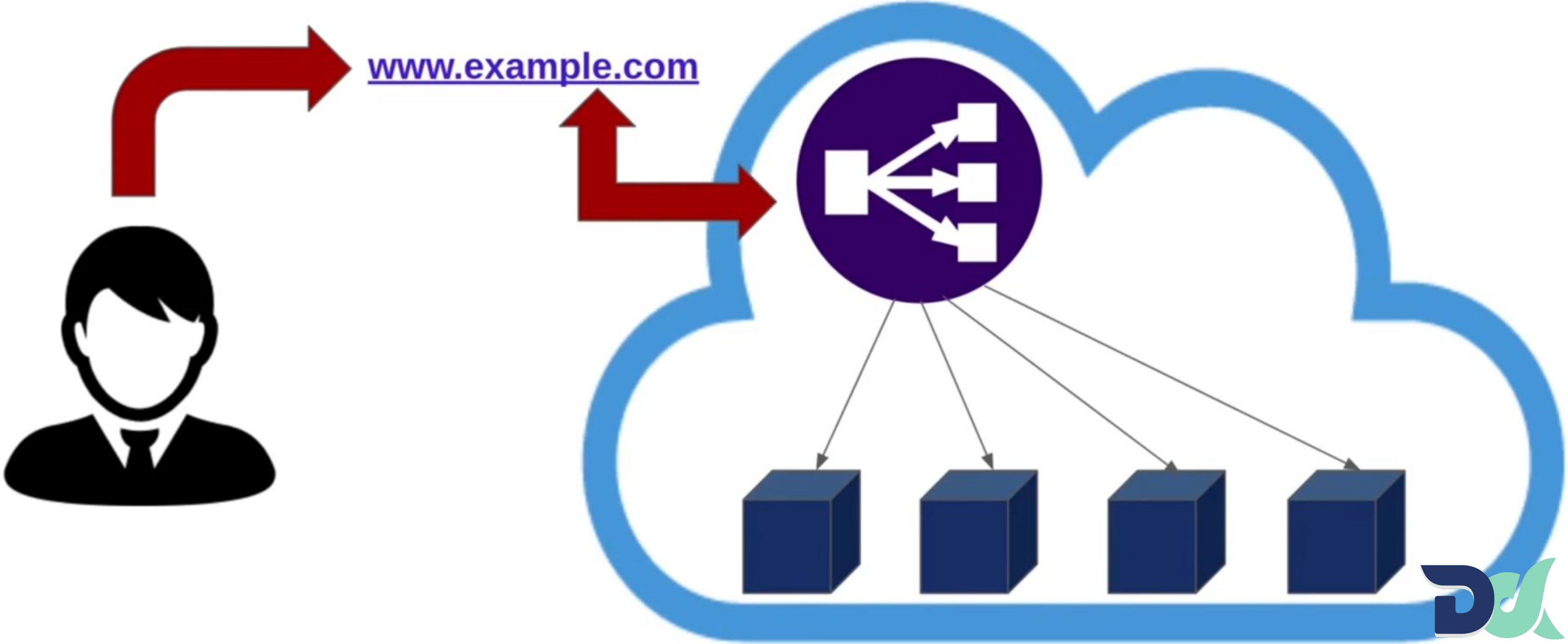
ClusterIp Service

Only Accessible within the Cluster



Load Balancer Service

- On cloud providers which support external load balancers
- Have to Provision a Load Balancer

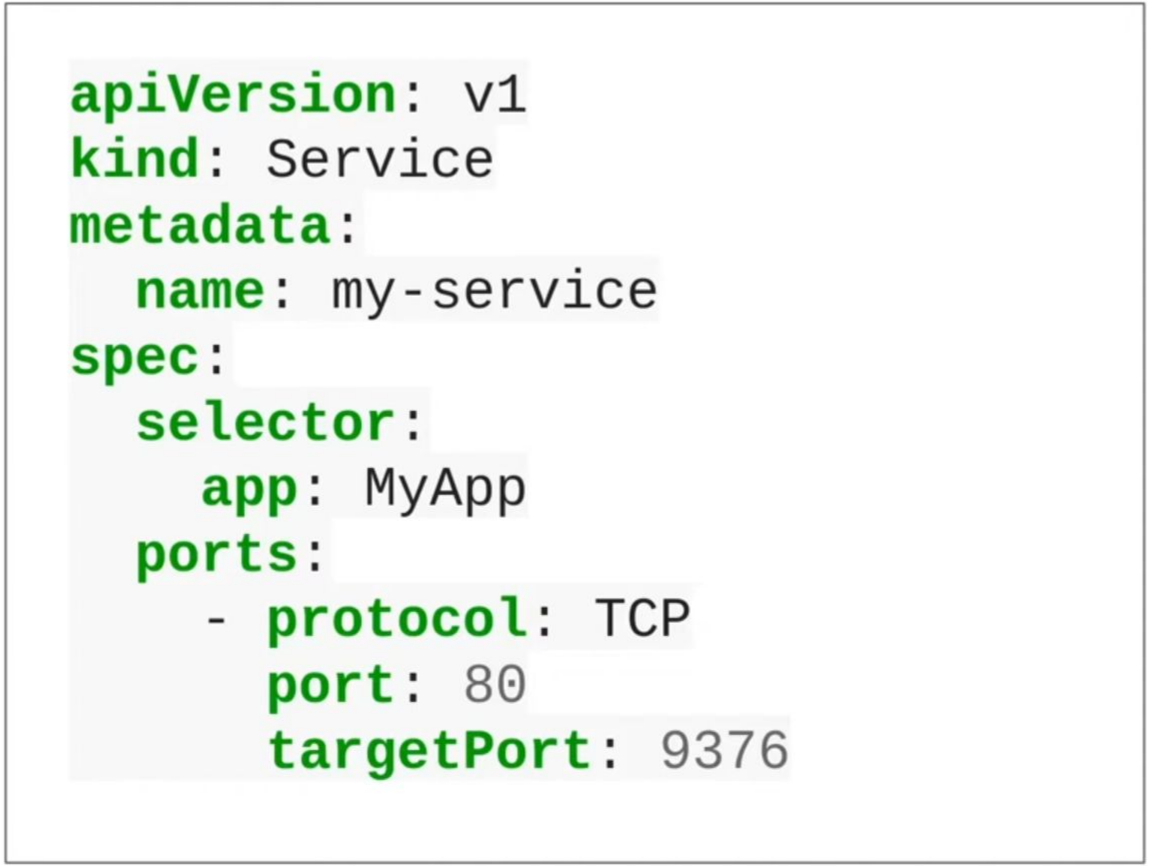


ExternalName Service

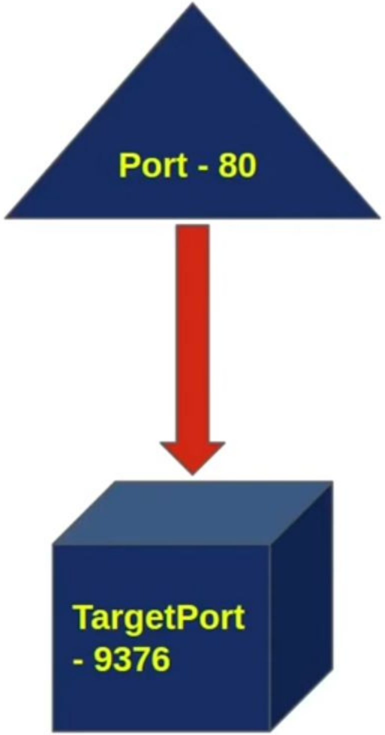
Services of type External Name map a Service to a DNS name

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: prod
spec:
  type: ExternalName
  externalName: my.database.example.com
```

Service File



```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```



How to Create Service?

Declarative Way

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: MyApp
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

Imperative Way

Using **Expose** Keyword

Questions

1. Create two pods named blue and red with image nginx and expose 80 port for blue pod.
2. Create a NodePort service for the blue pod.
3. Create a ClusterIp service for red pod.
4. Delete the nodeport service of blue pod.
5. Again create a nodeport service for blue pod and use-
   a. Port - 8080
   b. NodePort - 32711
   c. Target Port - 80