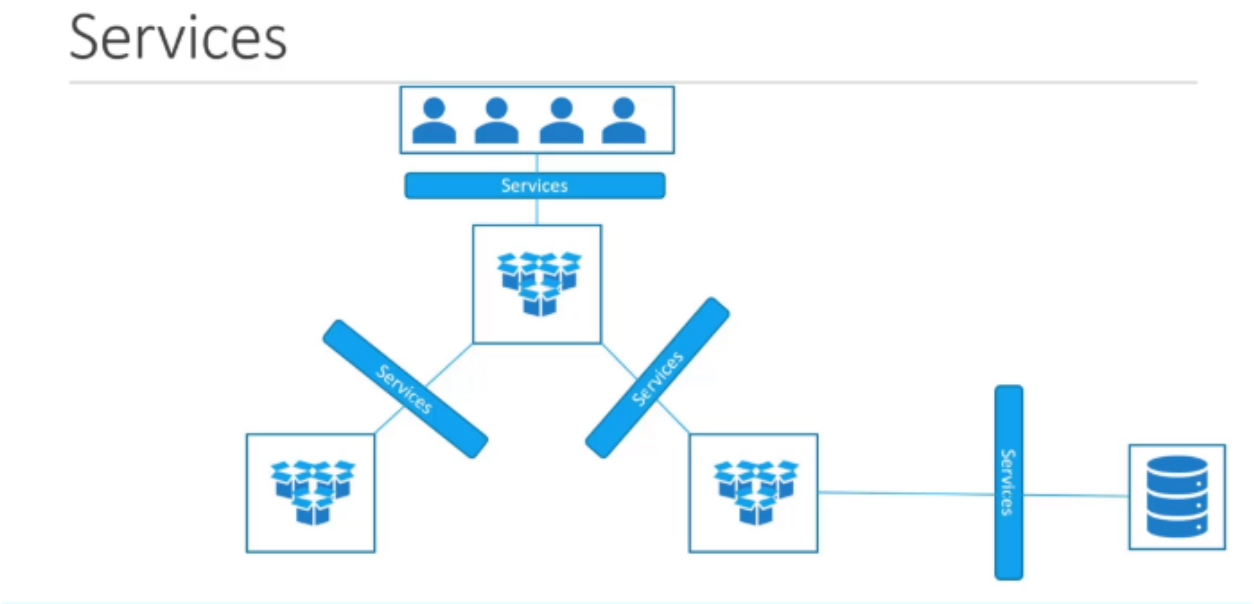
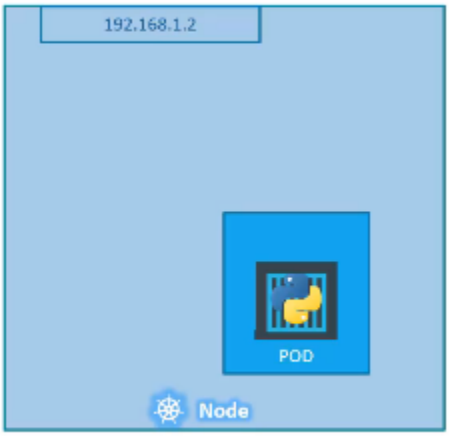
Ks enables communication b/w various components within and outside of application
Means
Within the cluster == components can talk to each other
And from outside user can access or one application talk to other application





Pod has ip and node has ip

Pod is in different network

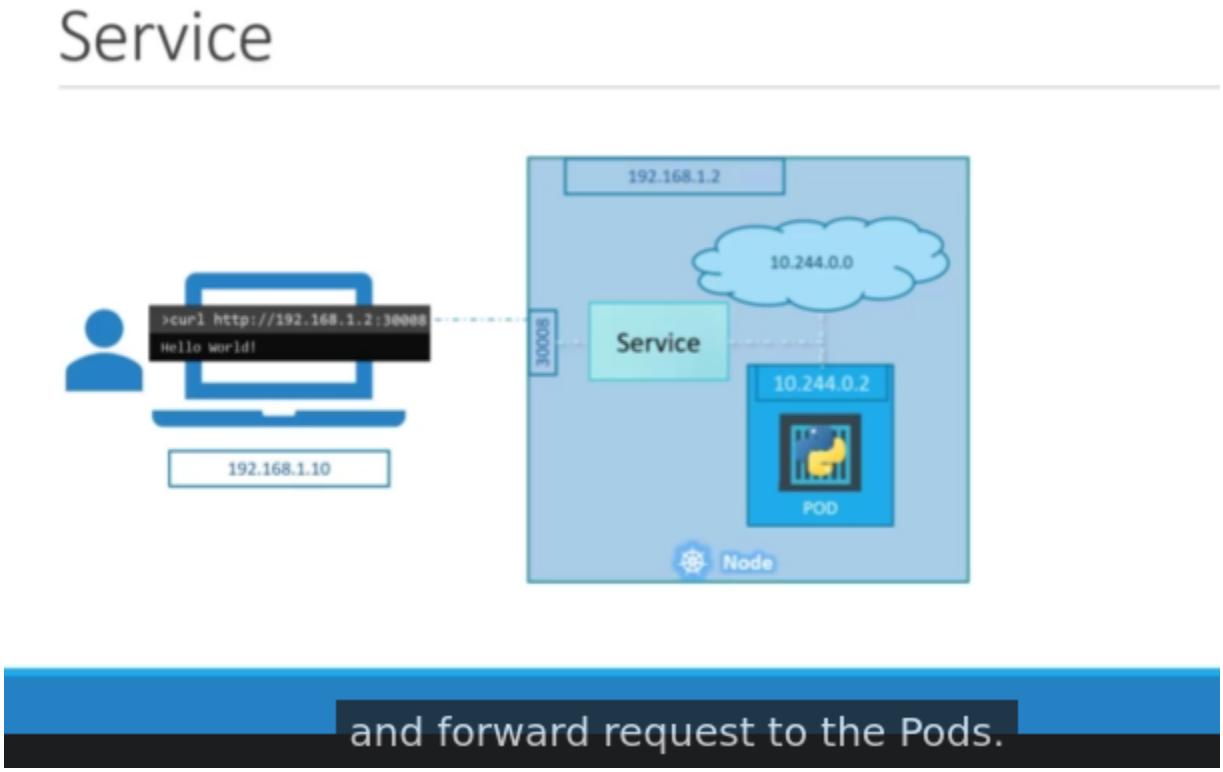
Pod ip changes every time i=it recreate

Clearly I cannot ping or access the Pod

But we want to access application by simply using ip of node
From user side

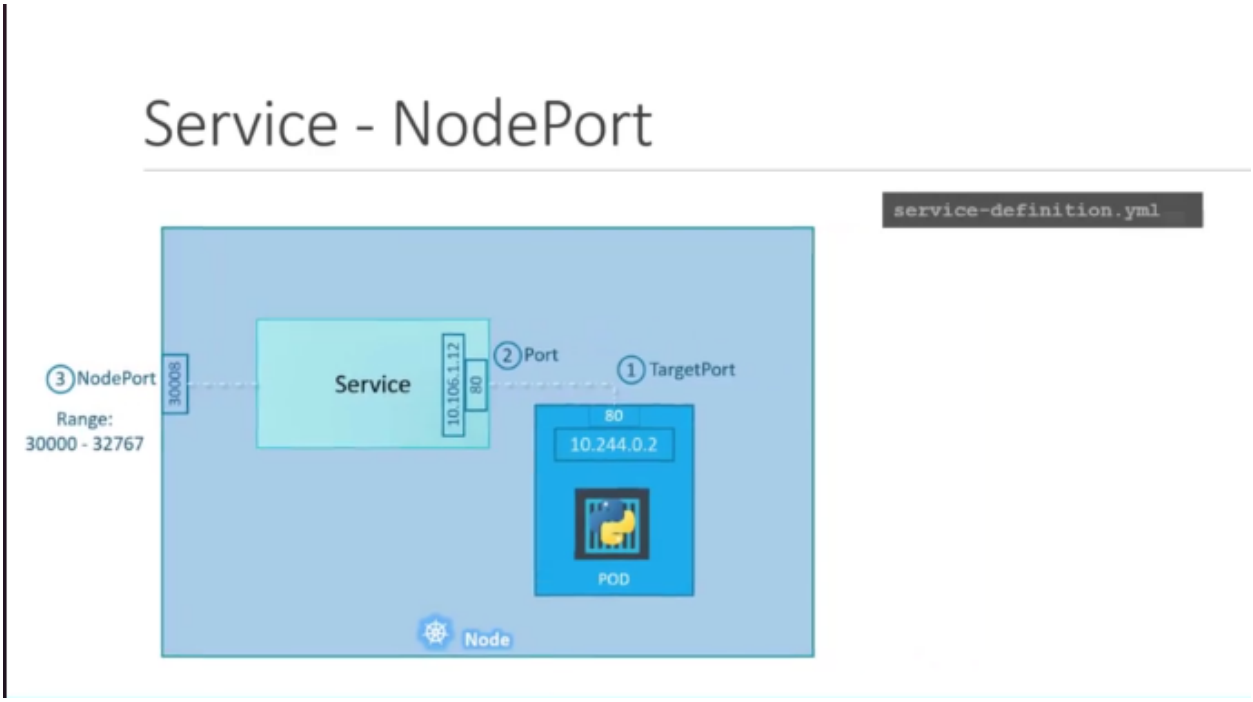
Service is a object , it has
IP == virtual ip == cluster ip
Port = where we expose our service
We can do multiple port mapping

Listen to a port on the node and forward request on that
port to a port
on the pod running the web application



Nodeport

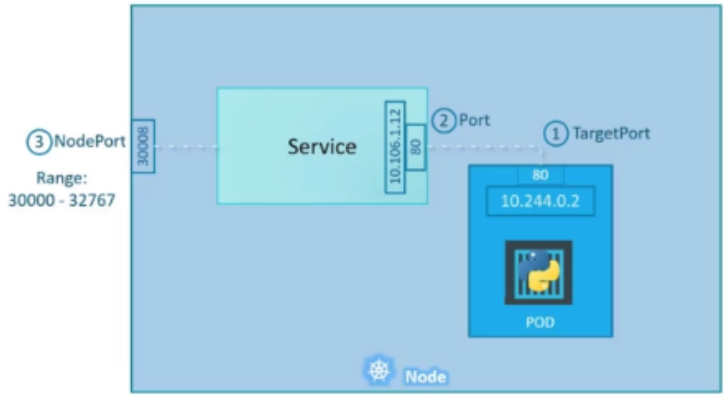
Port on service compulsory
Target port on pod optional is equal to port
Nodeport on node optional default from 30000 to 32676



apiVersion: v1
Kind: Service
Metadata:
  Name: myapp-service
Spec: //here we define actual services
  Type: NodePort or ClisterIP or LoadBalancer
  Ports:
    - targetPort: 80
      port: 80
      nodePort: 30008

  Selector: //list of labels
      App: label1
      Type : front end

Service - NodePort



service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
```

You can have multiple such port mappings within a

Service - NodePort

service-definition.yml

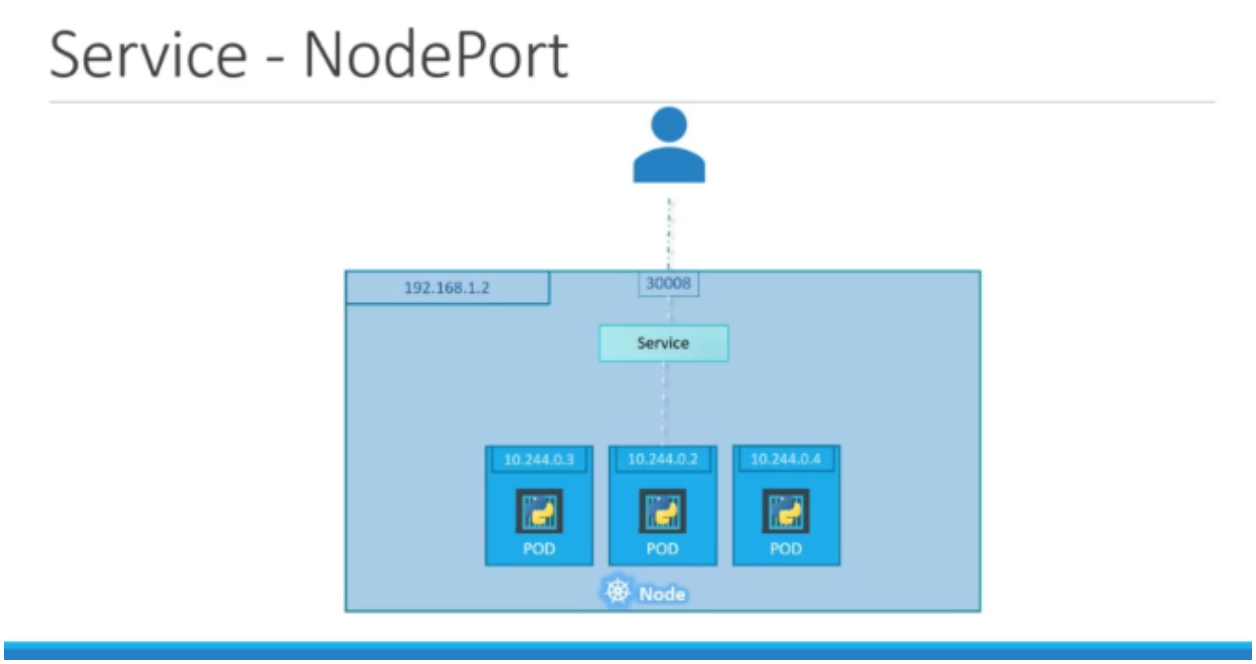
```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:


spec:
  containers:
  - name: nginx-container
    image: nginx
```

Kubectl creare -f service-file-name.yaml
Kubectl get services / sv

In case of multiple pod



For multiple pod we use labels and selector so that services

Services transfer request to all pods using a algorithm

Service select all the pod which have mapping labels as **endpoints** to forward the external request

Services useds a random algorithm to balance the load

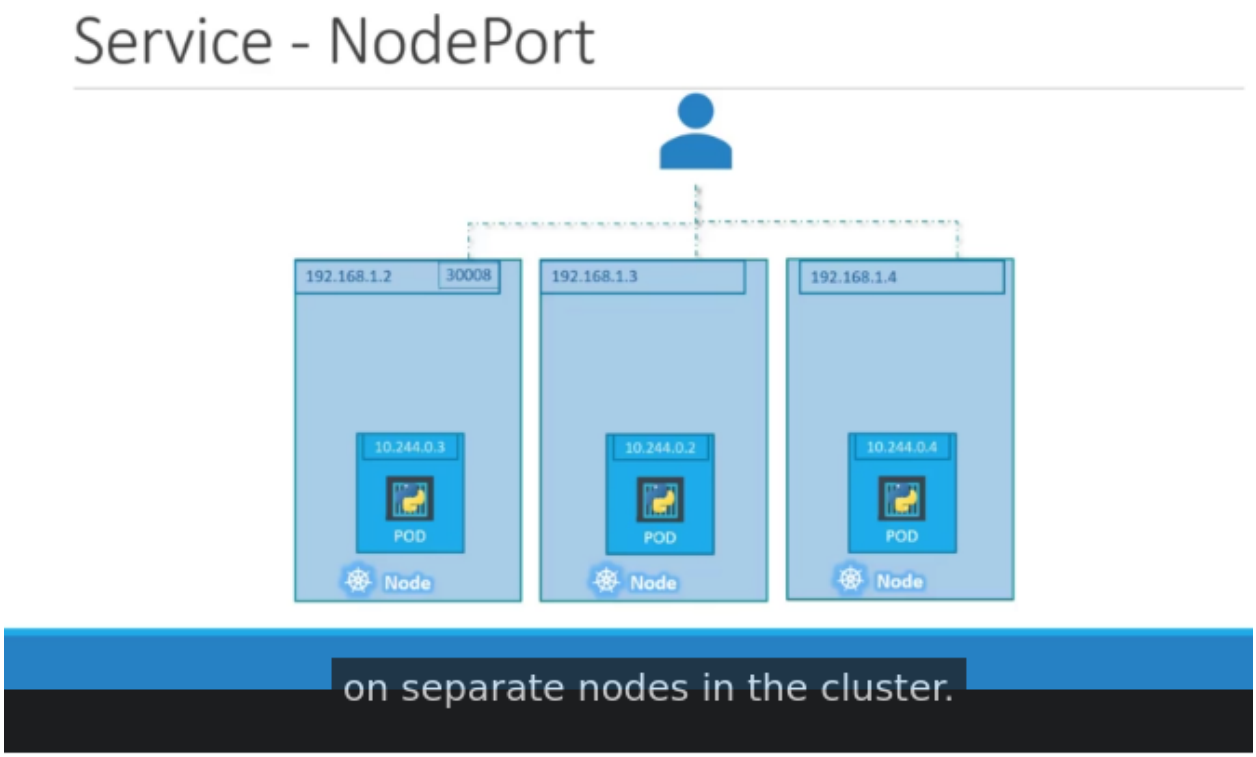
In services we don't need to do any additional configuration

Services auto do all thing

Services act as built in load balancer

Also if pod are in different node then

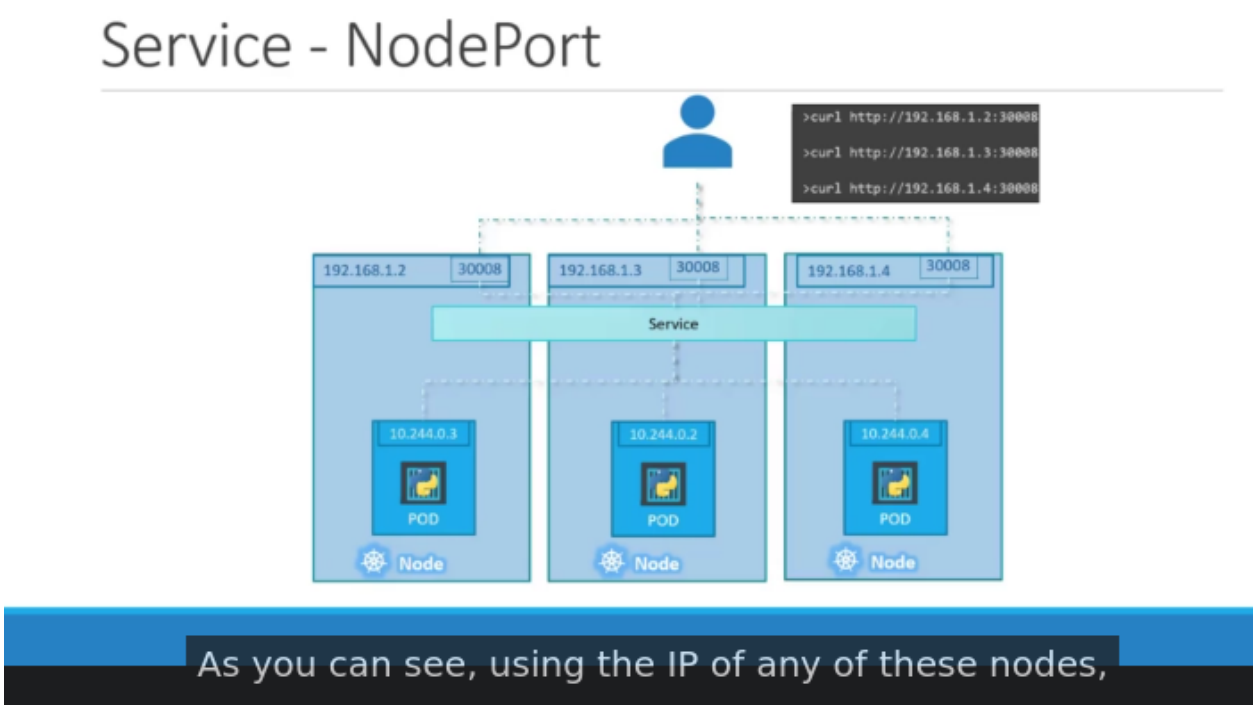
Services have the ability to span over all the nodes and we can access application By using any node ip and port



We dont need to do any additional configuartion

Services do auto done

j



In any case

Single pod on single node
Multiple pod on single node
Multiple pod on multiple node

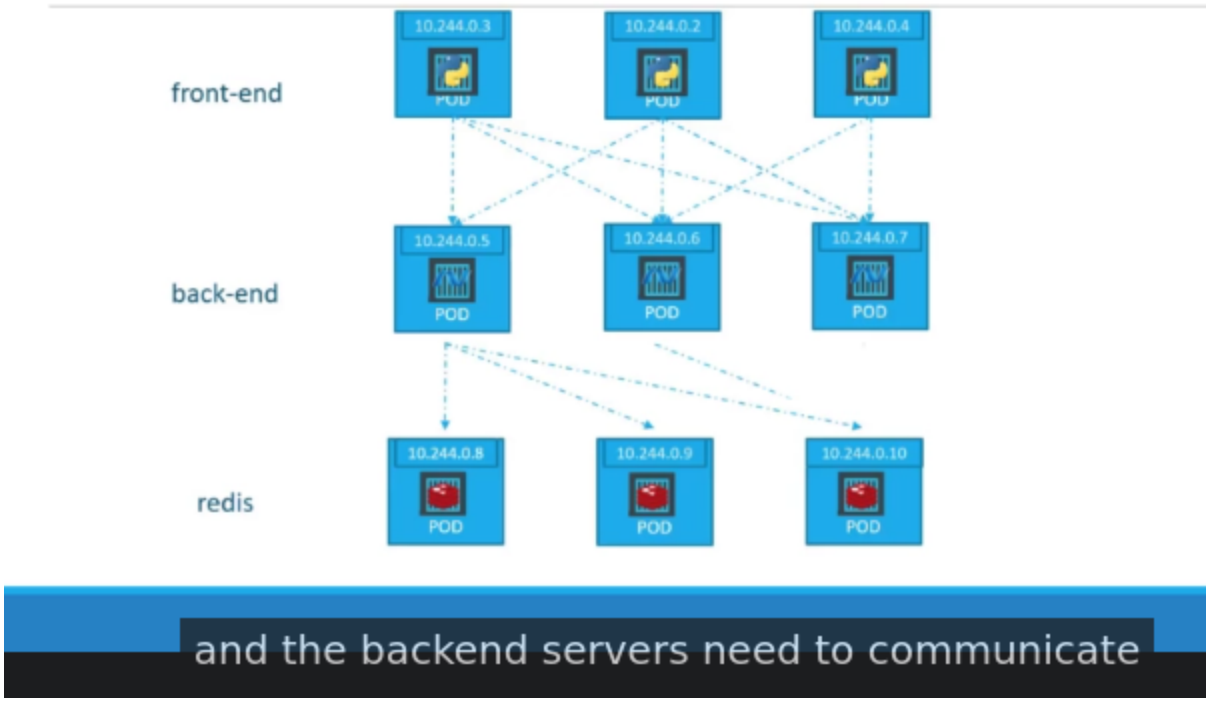
Service is crteated exactly the same without doing additional configuration

When pods are removed or added
Services is automatically updated

Cluter ip

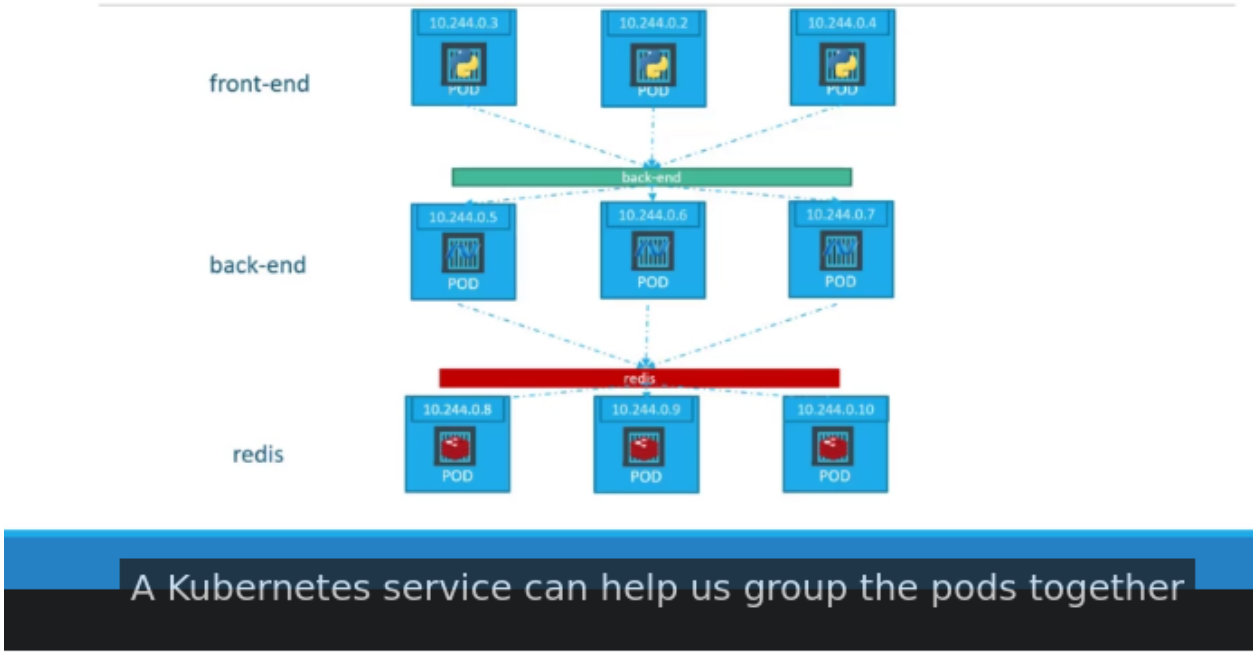
Nnnshhdd
Hsh

Scenrio



Solution

Pod ip is not static
So to talk to other pod we create service of type cluster ip
That has static ip



Services group pods together and provide a single interface to access the pods

Services have name and ip and name is used by other pods to access the service

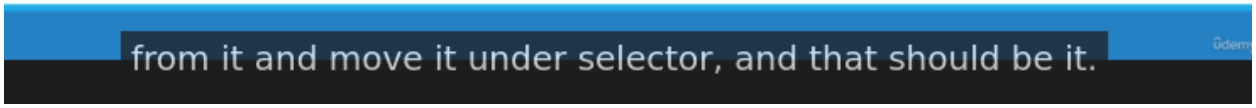
```
apiVersion: v1
Kind: Service
Metadata:
  Name: back-end
Spec:
  Type: ClusterIP == default
  Ports:
    -   targetPort: 80
        Port : 80
  Selector:
        Label1: value1
```

service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: back-end

spec:
    type: ClusterIP
    ports:
     - targetPort: 80
       port: 80

    selector:
       app: myapp
       type: back-end
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:


spec:
  containers:
  - name: nginx-container
    image: nginx
```

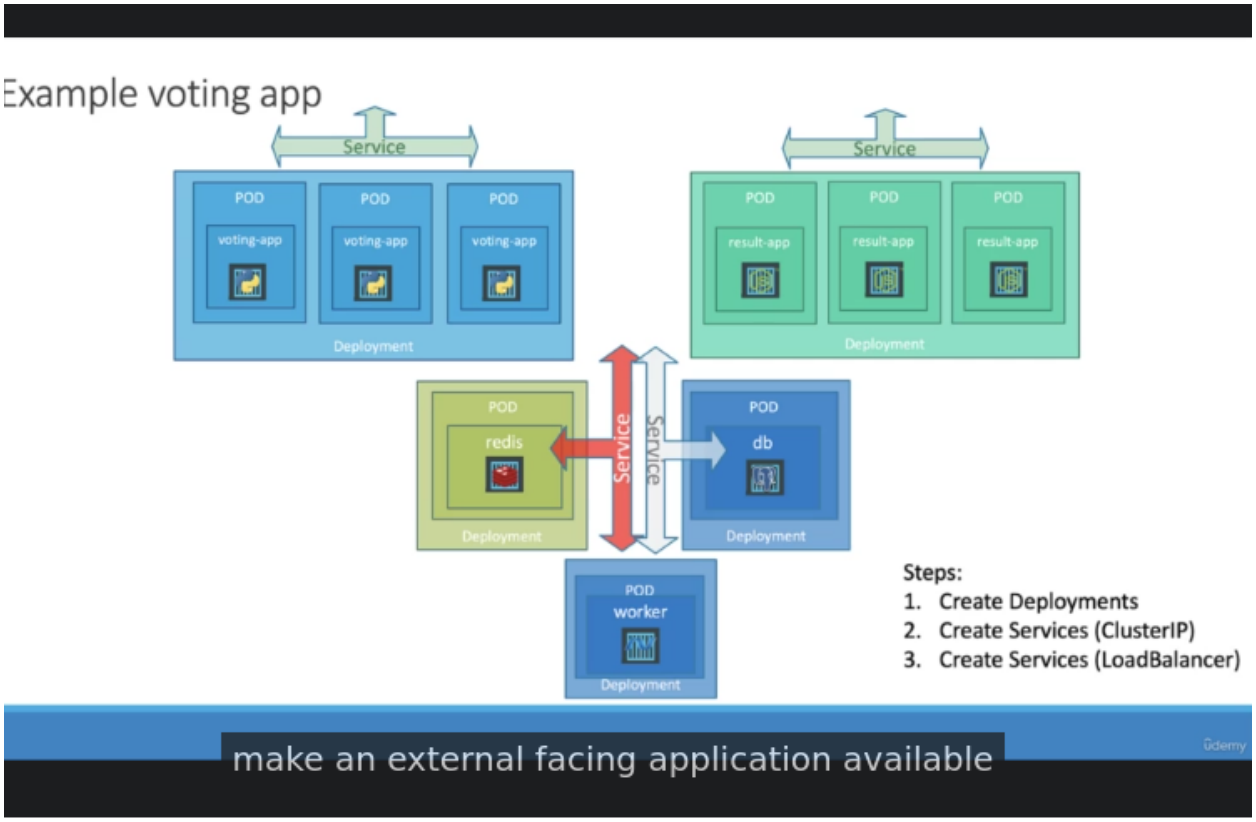


Kubectl create -f cluster-file-name.yaml
Kubectl get services

Service is access by other pods by using name or IP

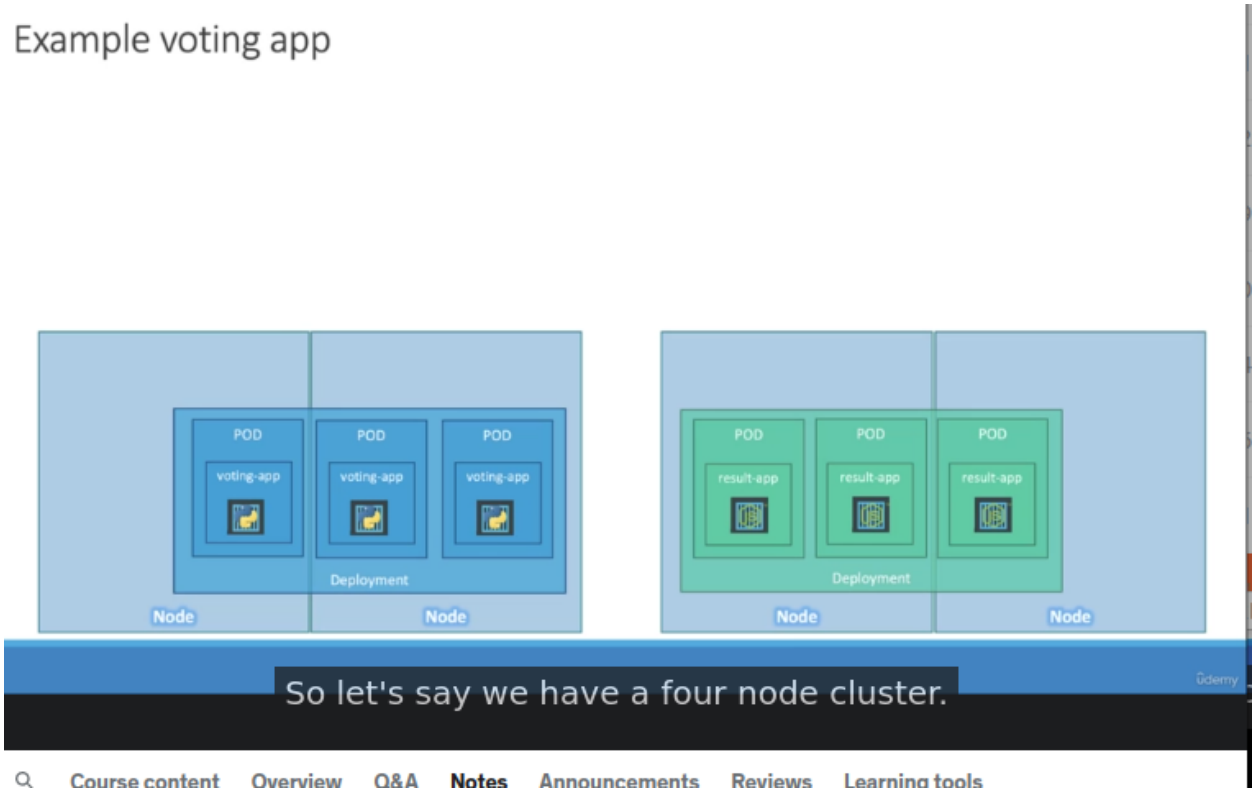
LoadBalancer

This is service is supported only by load balancer providers like cloud proviser



Course content | Overview | Q&A | **Notes** | Announcements | Reviews | Learning tools

Create deployment
Craete services (Clusrt IP)
Create services (Load Balacer)



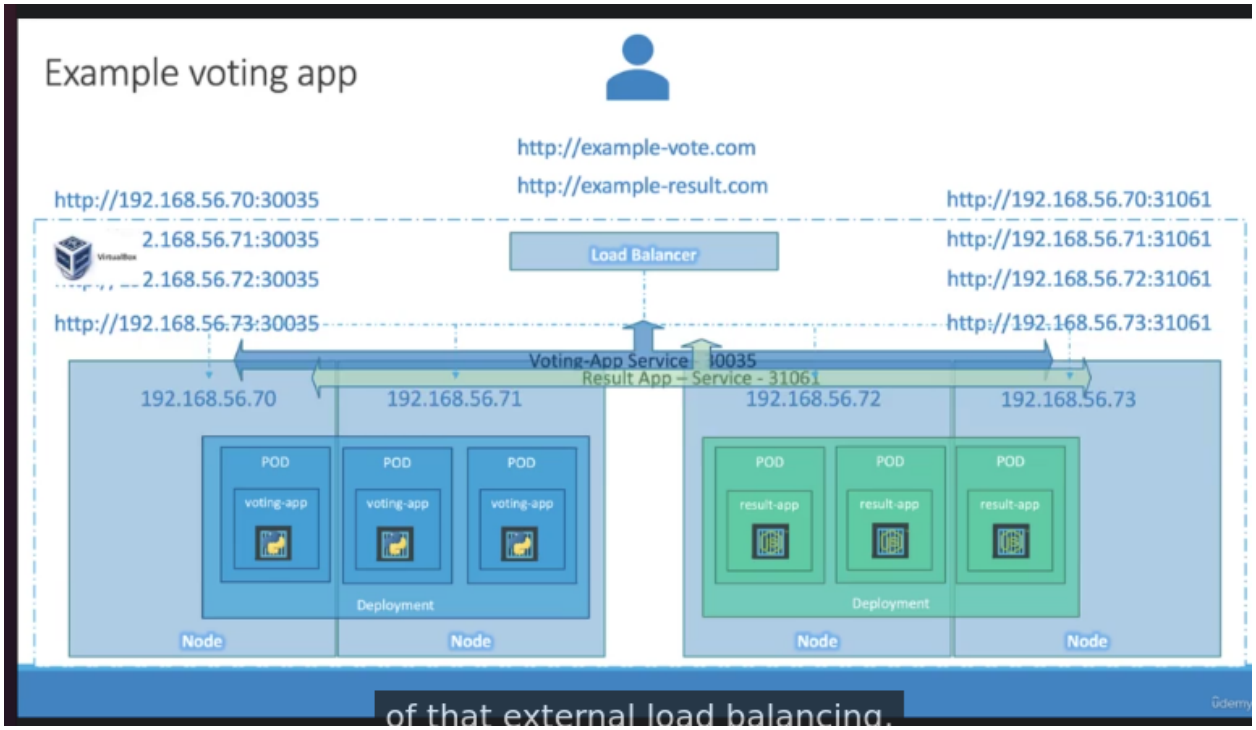
If we create service of type node port but we want to give url to our users to access the application

Because in nodeport we can access pod by node ip and ode port

End users want a single url to access the pod

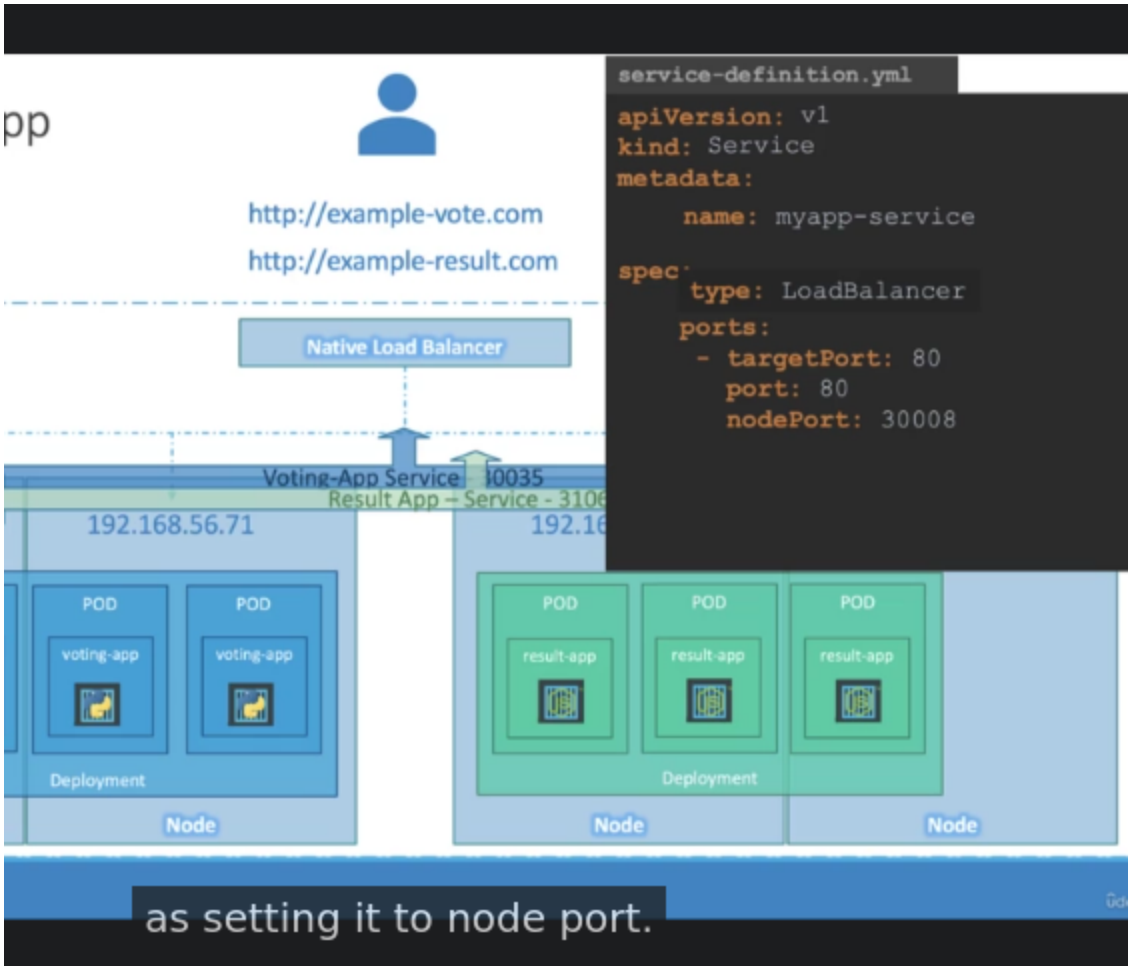
Craete a vm and install and configure a suitable load balancer
Like nginx or HA proxy

Then configure load balancer to route traffic ==== but all are tediuoys job



Kubernetes has the ability to integrate load balancer of cloud provider
We only need to setup service type of Load Bakacer

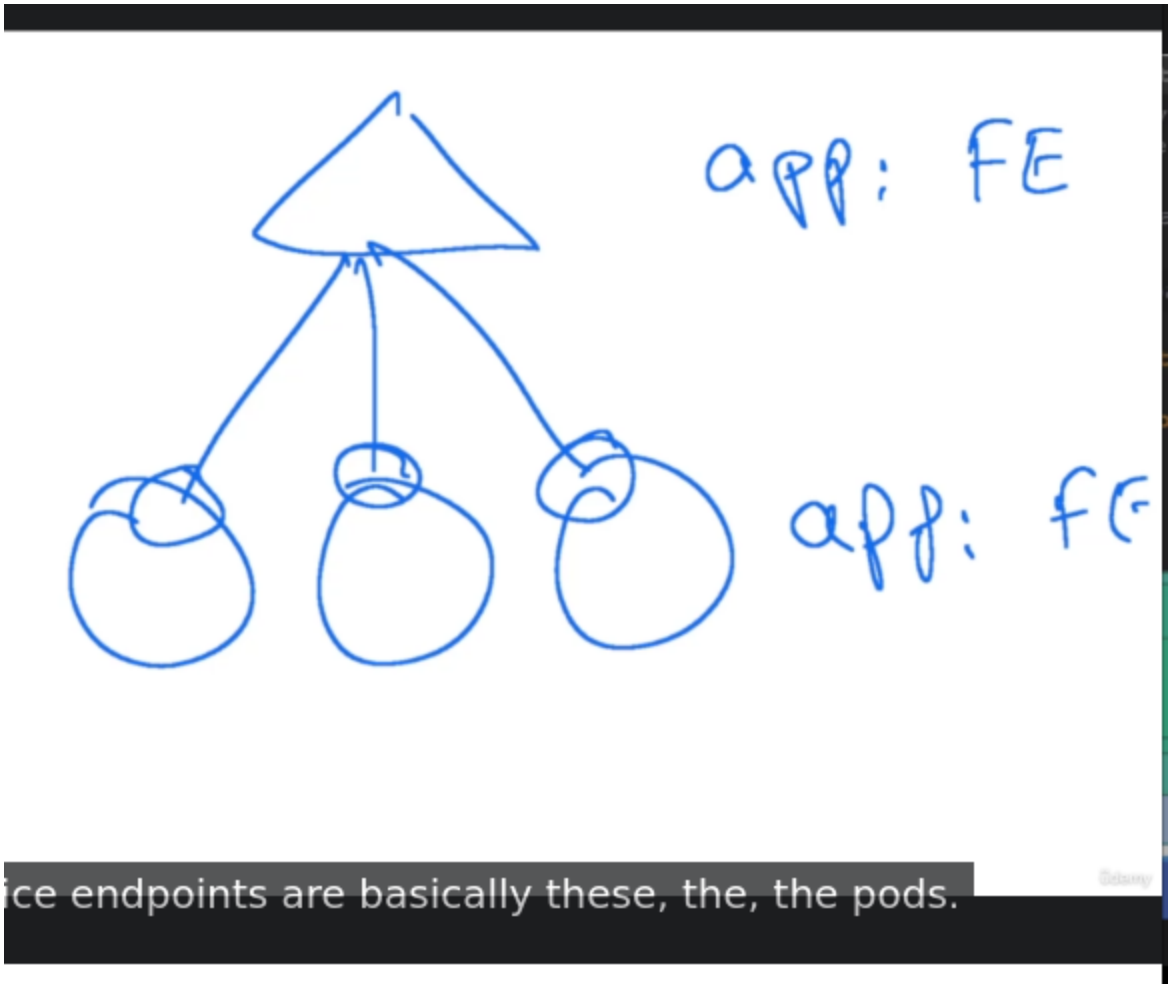
If we setup type as load balancer in un supported environment
=== Node Port



We dont need to do any configuration

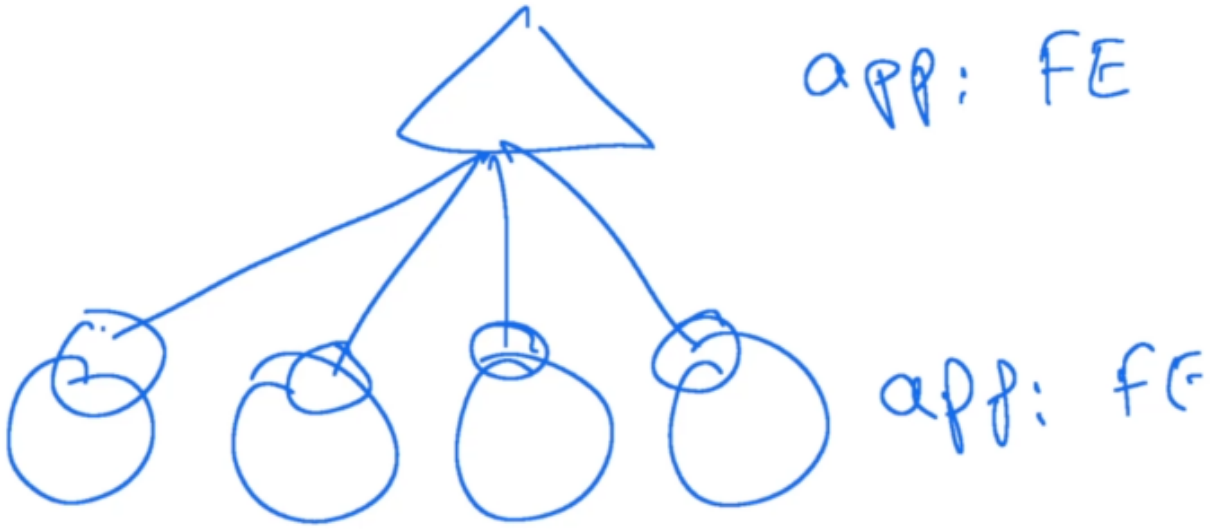
Kbectlr get services or svc
Kbectlr describe services or svc service-name

Endpoints are basically pods ip meanse where services direct traffic



Endpoints are used to identify that whether service route traffic to only thise pods which are specified

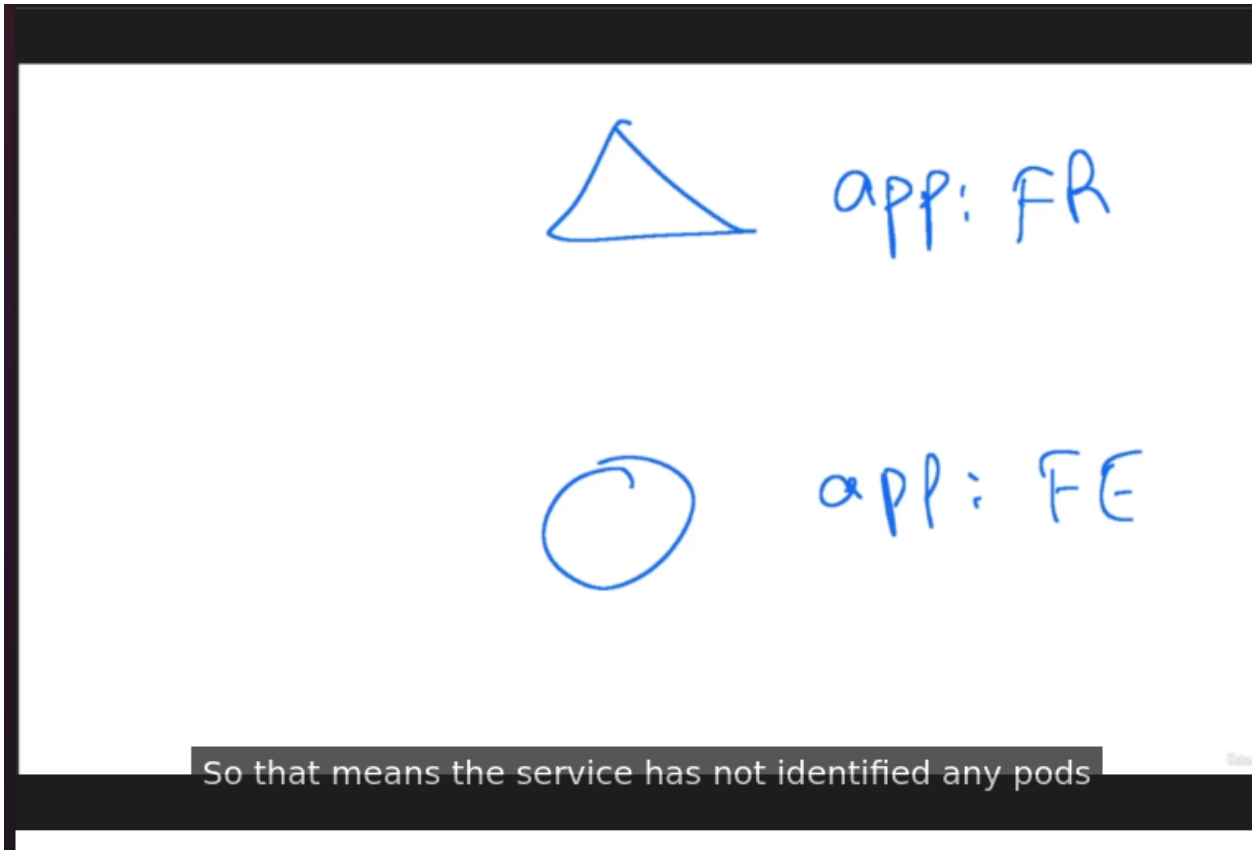
Not to other pods which have same labels

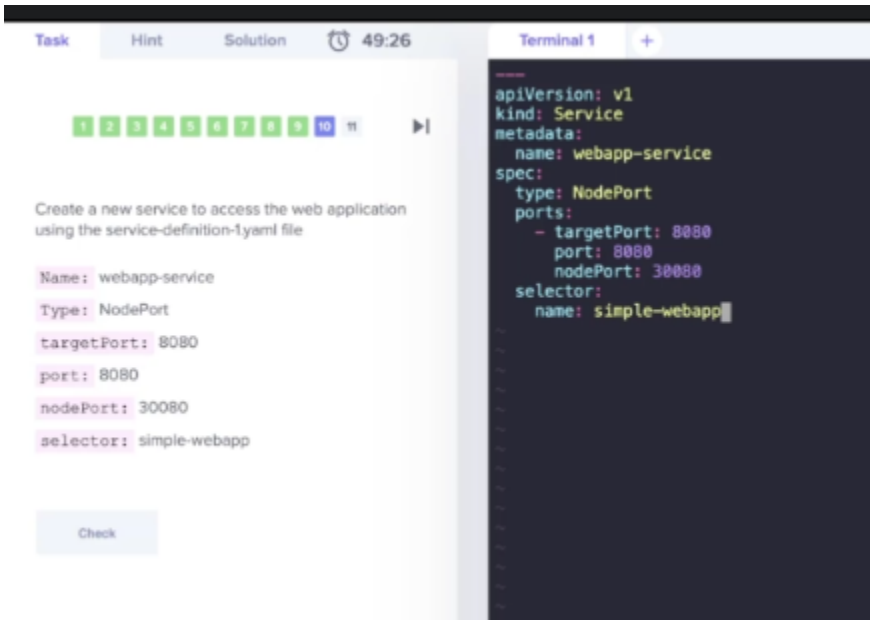


that we can identify the additional endpoints apart

rse content | Overview | Q&A | **Notes** | Announcements | Reviews | Learning tools

Single pod service
Endpoints also identify that we mention the corect endpoint


app: FR
app: FE
So that means the service has not identified any pods


Task
Hint
Solution
49:26
Terminal 1
Create a new service to access the web application using the service-definition-1.yaml file
Name: webapp-service
Type: NodePort
targetPort: 8080
port: 8080
nodePort: 30080
selector: simple-webapp
Check
apiVersion: v1
kind: Service
metadata:
name: webapp-service
spec:
type: NodePort
ports:
- targetPort: 8080
port: 8080
nodePort: 30080
selector:
name: simple-webapp