SAMARTH

Date _____ Page _____

SERVICES : [SECTION 41]

1. Service Type :-
   - (a) Cluster IP
   - (b) NodePort
   - (c) Load Balancer.
   - (d) External Name
2. Discover Kubernetes Services.
   - (a) DNS Names.
   - (b) DNS & Namespace.
3. Ingress
4. Ingress Controllers.
5. Named Port. (Routing to Named Port).

SECTION 11 :- SERVICES.

[260] :- How to use K8s Services.

1. Service Types
2. ClusterIP Service
3. NodePort Service
4. LoadBalancer Service
5. Hand-on-Demo.

What is Service Type :-

1. Each Service has a Type. Service Type define how & where Service will Expose the Application.

   By default the Service type is ClusterIP.

   So basically Service type defines the set of rules that how & where the Service will be Exposed.

2. There are four type of Services :
   - (a) ClusterIP.
   - (b) NodePort
   - (c) LoadBalancer } not part of this course.
   - (d) External Name

Cluster IP Service :-

Service is to Expose the application to the Outer world.

1) Cluster IP Service expose application within the Cluster network.

So whenever we have a need to expose the application within the cluster network, not outside the Cluster n/w.

2) So what is the use of Cluster IP Service ?

use ClusterIP, whenever the client is Other Pods within the Cluster

it means whenever a pod want to Communicate with the other pod, you can use the Cluster IP Service for that Communication.

Now we can also understand that the Communication blw the pods are by default Enabled.

* So why we want the Services to access the pods from a pod ?

(a) we are just dealing with the single namespace so that the Communication blw pod is easily available.

But what if your pods are belong to the different namespaces?

(b) in that Case, you need to use the Service Right ?

the pods which are part of NameSpace A & if they want to Communicate to the pod of NameSpace B, then they need to use the Cluster IP

to Communicate with the other pod in namespace.

3] There are multiple Scenario's :

if you have a simple application of frontend & backend. Then frontend want to Communicate with the backend. and you can expose that particular Communication by the cluster IP services

NodePort Service

① NodePort Service expose Application outside Cluster N/w.

② Use NodePort, when client is accessing the service from outside the cluster N/w

Load Balancer Service :-

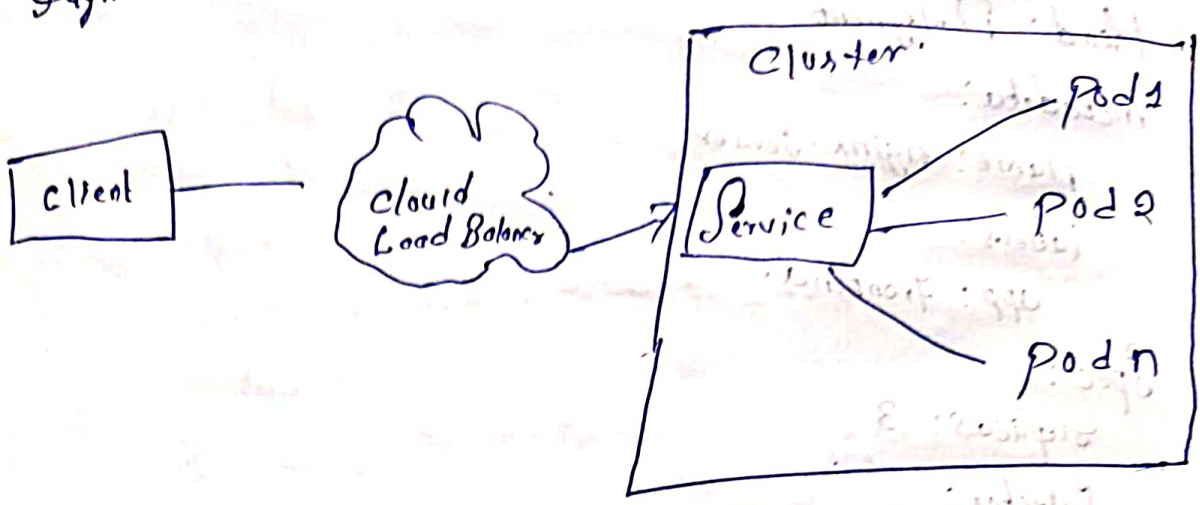
① Load Balancer service also expose Application to Outer world but Cloud ILB is required.

↓
L cloud load Balancer

Aws has Load Balancer, Azure has Load Balancer etc based on public cloud you are using, use the respective Load Balancer.

Load Balancer will directly Connect to the service.

Although in the backend, load balancer will Connect to the node Port & node ports will Connect to the pods ought.



External Name :- also we the load Balancer Service type.

The only & only difference in the External name Service. is you need to Create the Cname which will directly acces the Service in K8S.

[26] Lab Demo. Kubernetes Services.

In these Lab we will Create the Pods and we will Create the application running in a pod by a Service.

we are using Singlenode cluster :- minikube.

① Go to the directory where we have store the Script.

CD ---

② Vi svc-pod.yml:

Then deployment manifest.Yml file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-Server
  labels:
    app: frontend.
Spec:
  Replicas: 3
  Selector:
    matchLabels:
      app: frontend.
  template:
    metadata:
      labels:
        app: frontend
    Spec:
      Containers:
        - name: nginx
          image: nginx
          ports:
            - ContainerPort: 80
```

Save & Exit.

These file we Created 3 Replica.

(3) Kubectl apply -f svc-pod.Yml.
Copy the Entire name Shown in Output.

(4) Kubectl describe deployment.apps/nginx-Server.

5) Kubectl get pods -o wide

6) now we need to Create a Service. (Cluster IP Service)

Vi clusterip-svc.yml.

```
apiVersion: v1
Kind: Service
metadata:
  name: nginx-service
Spec:
  type: ClusterIP
  Selector:
    app: frontend.
  ports:
  - name:
    protocol: TCP
    port: 8080
    targetPort: 80
```

Save & Exit

7) Kubectl apply -f clusterip-svc.yml.

8) Kubectl get pods -o wide --show-labels

9) Kubectl Describe Service/nginx-service.

10) access Service.

Curl nginx-service: 8080.

get error Could not Resolve host:

because we are accessing via host machine, as we have used ClusterIP Type. it is accessible only in pods.

11] Let's Create another temporary Pod.

12] Vi temp-pod.yml:

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-svc-test
Spec:
  Containers:
  - name: busybox
    image: radial/busyboxplus:curl
    Commands:
      - sleep
      - '3600'
```

Save & Exit.

13] Kubectl apply -f temp-pod.yml

14] Kubectl exec pod-sv-test -- Curl nginx-Service:8080

we are also to accer the nginx.

Letus See how NodePort Service

15] vi nodeport-svc.yml.

```
apiVersion: v1
Kind: Service
metadata:
    name: nginx-Service-nodeport
Spec:
```

```
type: NodePort
selector:
  app: frontend
ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodeport: 30099
```

16] kubectl apply -f nodeport-svc.yml.

17] kubectl describe service (nginx-service-nodeport.

18] Let access your service.

curl localhost:30099.

[262] DISCOVER KUBERNETES SERVICE

How we can discover the service.

Suppose you have a need that some of your pods want to access your service. Then how these pods can discover the service, how these pods or the services can communicate to the other services. → Let's discuss.

1. Service DNS Names.
2. Service DNS & Namespace
3. Hands-on-Demo.

Service DNS Names :-

(1) Kubernetes DNS assign DNS Names to Services, allow applications within cluster to easily locate the Service.

So each & every Service, whatever you are Creating in your Communities, that must have the DNS Name. Given it is by default, Define the DNS for your Service.

DNS means the domain name Service that allow. application with in the Cluster to easily locate the Service.

Because if each & every Service have their own Specific DNS, Then that will be easy for the Other K8S object to Communicate via the Services.

(2) Service Fully Qualified Name has the following format :-

Service-name.namespace-name.svc.cluster-domain.example.

this will be the domain-name if you have provide during Kubernetes cluster installation.

(3) if not given domain name during installation, by default Cluster Domain is cluster.local

what is the Service Relation between.

Service DNS & Namespaces.

① Service Fully Qualified Domain Name can be used to reach Service from within any Namespace in Cluster.

"Service-name.namespace-name.svc.cluster.local."

it means if you are using the Service Fully Qualified Domain name to access the Service, then from any namespace you can access your Service. it doesn't matter that you are trying to access the Service from the Cross Namespace.

② Pods within the same Namespace can use the Service Name Only.

"Service-name"

They can use the "Service name" Only to access the Services, but the K8s object or the pods that belong to the different namespaces.

They must need to use the Fully Qualified Service Domain name to access the Service & K8s.

[263] Lab : Discover Kubernetes Services :-

We will see a short lab on the service discovery. that how we can access the service object from within the same namespace or the cross namespace.

(1) Go To Singlenode Cluster.

(2) To check : How many Services are running in cluster.

Kubectl : get services -o wide

(3) Kubectl get pods -o wide --show-labels.

in last lab we have created a temporary pod. to check service is accessible with in Cluster IP.

-- pod-svc-test"

(4) and also step 2 :- last lab we have created a "Service nginx-service" Service.

(4) Check if Service is accesible from "pod-svc-test"

Kubectl exec pod-svc-test -- curl nginx-service:80

(5) Let's Create one more pod.

Vi dns-pod.yml.

```
apiVersion: v1
Kind : pod
metadata :
   Name: svc-tst-dns
   Namespace: Service-namespace
```

Spec:
  Containers:
    - name : busybox-svc
      image : radial/busyboxplus:curl
      Command:
        - sleep
        - '3600'

Save & Exit.

(6) Kubectl get namespace --show-labels.
we see that namespace specified in step 5 is not present, before creating pod. we need to create namespace in cluster.

(7) Kubectl Create namespace service-namespace.

(8) Kubectl apply -f dns-pod.yml.
pod is created

(9) Kubectl get pods -o wide --show-labels.
we cannot see the pod as we have created in different name space.

(10) Kubectl get pods -o wide --show-labels -n service-namespace.

we can see the pod.

(11) Let Try to access.

Kubectl exec ~~svc-test~~ -n Service-namespace

Svc-test-dns -- curl nginx-Service:8080

we can see that it is not able to Resolve the Service.

because "nginx-Service" in not running in same.

Namespace as earlier Command ~~did~~ Step 4 was Creating.

(12) To access these Service we need to define fully Qualified Domain Name of Service.

Kubectl exec -n Service-namespace Svc-test-dns

-- curl nginx-Service.default.svc.cluster.local:8080

Now it is working.

[264] Manage Access via Ingress Controller.

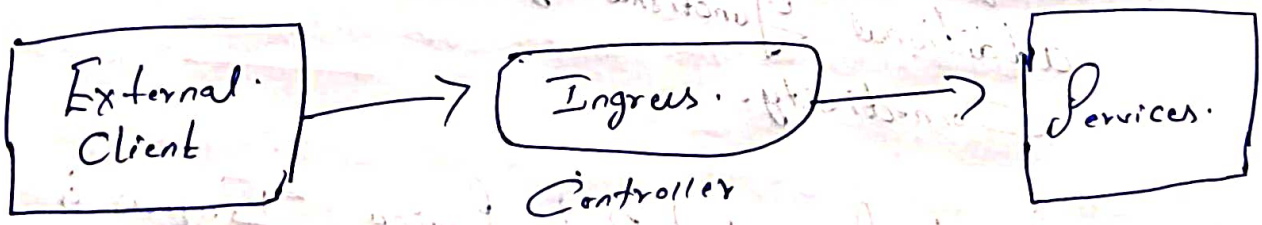
1. what is an Ingress.
2. ingress Controllers
3. Routing to a Service
4. Routing to Service with Named Port

what is an Ingress:-

1. Ingress in Kubernetes manage the External Access to Service.

   Ingress → incoming traffic to the pod.

There is One more thing Called ingress Controller & that is used to manage the External access to the Service in K8s.



a. we assume here we have an External Client which want to access my Services.

b. External client will send the Requests, if ingress Controller is in place. That Request is directly Received by the ingress Controller.

(c) Then ingress Controller will send this request further to the Service or Services.

when I'm saying Service or Services means ingress Controller will send the request to a single Service or multiple Services

(d) In last lecture/lesson we ~~are~~ were not using ingress Controller, instead we were using Specific port for ingress flow.

now the ingress Controller in place. External client is not directly dealing with the Service, they are dealing Service via Ingress Controller.

(e) Ingress Controller is an object b/w the client & the Service & it is provind a abstract interface to manage the more secure traffic & provide some additional functionality b/w the client & Service Connectivity.

(2) Apart from NodePort Service, Ingress is capable of many more.

(a) Provide the SSL Termination at ingress level.

(b) load balancing features.

(c) Name Base Virtual Hosting.

Name Base Virtual Hosting means you can define the different different Services for different different Resource path.

Ingress Controllers.

(1) In order for the Ingress resource to work, the cluster must have an ingress Controller running.

(2) Variety of Ingress Controller available in. K8s to provide the multiple mechanism for External access of Service.

(3) you can deploy any number of ingress Controller. There's no limitations. you can deploy a number of ingress Controller in your K8s.

(4) Ingress define a set of Routing Rules.

(5) Each rule will have a Specific path which is Connected to a Specific backend, & Request matching to a path will be routed to the associated backend.

for Example:

```
apiVersion : networking.k8s.io/v1
kind : Ingress.
metadata:
  name: sample-ingress
spec:
  rules: -> we can define multiple rules. this is one rule
  -http:
    paths:
    -path: /testpath. -> multiple path. this is one path
     pathType: Prefix
     backend:
       service:
         name : test
         port:
           number : 80
                    -> service
```

for Example : if we have the url.

http://www.testSample.com/testpath -> path

if user enter testpath prefix in url. then the bakend block with Execute the service mentioning manifest.

Named Port :-

① if Service use Named Port, ingress can also use the Port's name to choose to which port it will route.

Service manifest.

```
apiVersion: V1
Kind: Service
metadata:
  name: nginx-Service
Spec:
  Selector:
    app: frontend.
  ports:
    - name: nginx-port
      protocol: TCP
      port: 80.
      targetPort: 80
```

Ingress Controller.

```
apiVersion: networking.k8s.io/v1
Kind: Ingress.
metadata:
  name: sample-ingress.
Spec:
  rules:
    - http:
        paths:
          - path: /testpath.
            pathType: Prefix
            backend:
              Service:
                name: nginx-Service
                port:
                  name: nginx-port
```

So we can use name of the port defined in Service manifest. which we have used to create Service.

Benefits :- If the port of your Service is going to change but the name remains the same.. then you don't need to change the ingress rule.

[265] Lab:- Manage Access via Ingress Controller.

Ingress Controller.yml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-rules
spec:
  rules:
  - host: nginx-offical.example.com
    http:
      paths:
        - path: /
          pathType: Exact
          backend:
            service:
              name: nginx-offical-service
              port:
                number: 80
  - host: magical-nginx.example.com
    http:
      paths:
       - path: /
         pathType: Exact
         backend:
           service:
             name: magical-nginx
             port:
               number: 80
```

for Lab please Refer the udemy Cources.
(devops kubernetes masterclass) Section 265.
Cos it has deployment, Services, and ingress Controller.