External Kubernetes Service For External Database

Kubernetes – Endpoints

we first learned about Kubernetes Services, we saw that we could use labels to match a frontend service with a backend pod automatically by using a selector. If any new pods had a specific label, the service would know how to send traffic to it. Well the way that the service knows to do this is by adding this mapping to an endpoint. Endpoints track the IP Addresses of the objects the service send traffic to. When a service selector matches a pod label, that IP Address is added to your endpoints and if this is all you're doing, you don't really need to know much about endpoints. However, you can have Services where the endpoint is a server outside of your cluster or in a different namespace (which we haven't covered yet).

What you should know about endpoints is that there is a list of addresses your services will send traffic and its managed through endpoints. Those endpoints can be updated automatically through labels and selectors, or you can manually configure your endpoints depending on your use case.

How about if we want to manually edit our endpoints if we don't have a selector? Maybe we're trying to have a resource to access an external service that doesn't live within our Kubernetes cluster? We could create our own endpoint to do this for us. A great example might be an external database service for our web or app containers.

Let's look at an example where we're using an endpoint to access an external resource from a container. In this case we'll access a really simple web page just for a test. For reference, I

accessed this service from my laptop first to prove that it's working. If you're doing this in your lab, you'll need to spin up a web server and modify the IP Addresses accordingly.

Next up we'll deploy our Endpoint and a service with no selector. The following manifest should do the trick. Notice the ports used and the IP Address specified in the endpoint.

Example Of Endpoint:

1. Service.yaml

```
kind: "Service"
 apiVersion: "v1"
 metadata:
   name: "external-web"
 spec:
   ports:
     - name: "apache"
       protocol: "TCP"
       port: 80
       targetPort: 80
```

2. endpoint.yaml

```
kind: "Endpoints"
apiVersion: "v1"
metadata:
  name: "external-web"
subsets:
  - addresses:
      - ip: "10.10.50.53" #The IP Address of the external web server
    ports:
      - port: 80
        name: "apache"
```

deploy the manifest file to get our service and endpoint built.

```
# kubectl apply -f service.yaml
```

```
# kubectl apply -f endpoint.yaml
```

Now get the endpoints of service

 # Kubectl get endpoint

```
Erics-MacBook-Pro-2:k8s-series eshanks$ kubectl get endpoints
NAME            ENDPOINTS                           AGE
external-web    10.10.50.53:80                      13m
ingress-nginx   10.244.1.143:80,10.244.2.204:80     1d
kubernetes      10.10.50.50:6443                    12d
```

Expected output.