Exposing Kubernetes Pods Using Services

Your company, Beebox, is in the process of building a microservice application for Kubernetes.

They have a backend database of user information and a web frontend, both of which are managed using deployments. Both of these applications are also still being developed and are currently just using simple Nginx containers for testing. You have been asked to set up appropriate Kubernetes Services for these application components.

The user database is a backend service that should only be accessible by other components within the cluster. The web frontend needs to be accessible by users outside the cluster. Locate the existing deployments and create the necessary Services to expose them. There is an existing Pod called busybox which you can use to test Services.

Reading above we have 2 task

- 1st task is to expose the pods from the user-db component as an internal service. Which this service will be designed to be consumed by internal clients within the K8s cluster.
- 2nd task is to expose the web front end pods using an external service. Of course this will be designed to be consumed by external clients (outside the cluster)

Expose the Pods from the User-db Deployment as an Internal Service

1. So we need to inspect the properties of the user-db deployment

```
kubectl get deployment user-db -o yaml
```

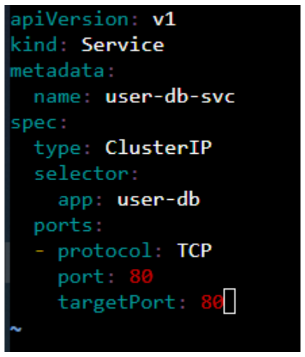
In here we want to pay attention to the label "app: user-db". Because we are going to need this label that's attached to all of the pods in order to create a service that is going to select the pods that are associated with this deployment.

Another think we want to pay attention to is the container port. In this lab it is showing 80. Which we probably need to map to that port using our service.

2. Create a yml file named user-db-svc

```
vi user-db-svc.yml
```

Figure 1-1



```
apiVersion: v1
kind: Service
metadata:
  name: user-db-svc
spec:
  type: ClusterIP
  selector:
    app: user-db
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
~
```

Include that into the yml file

We have the spec type: ClusterIP because this must be an internal service according to the instructions. We also included the selector and our app: user-db from step 1

3. Once that is saved create the service

Before we go any further we should test the service just to make sure that it works.

4. Test the user-db-svc

```
kubectl exec busybox -- curl user-db-svc
```

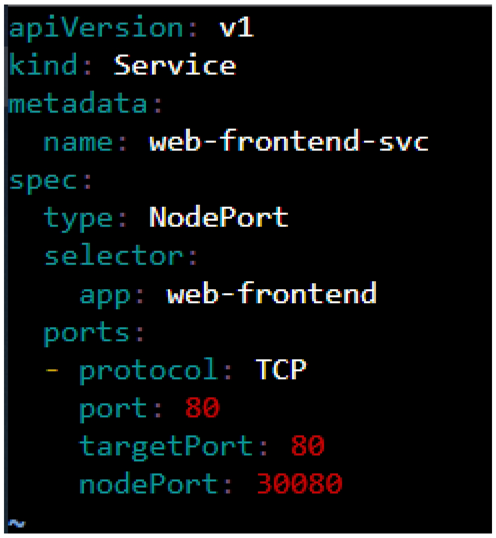
We should see the nginx here. For our first time doing this quickly. It may not appear, give it a few mins and it should be up and running.

Expose the Pods from the Web-frontend Deployment as an External Service

Similar to the previous steps we done. Except this is for external.

5. Create a yml file called based one what you find in the description
6. Define the service

Figure 1-2



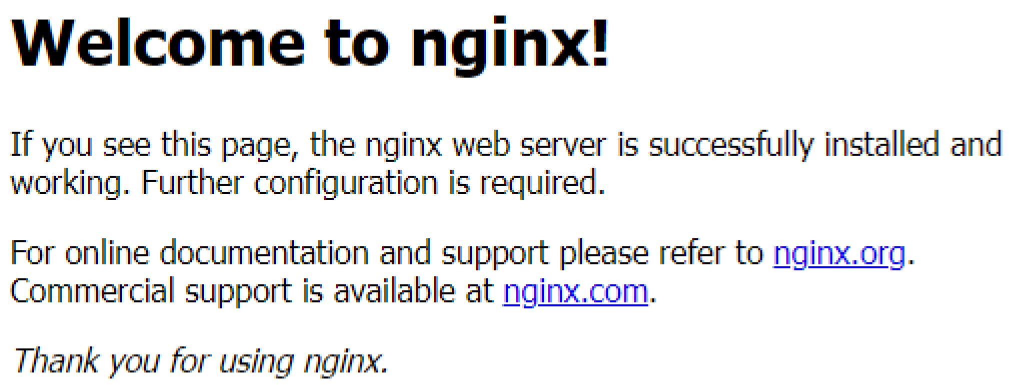
```
apiVersion: v1
kind: Service
metadata:
  name: web-frontend-svc
spec:
  type: NodePort
  selector:
    app: web-frontend
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30080
~
```

7. And then create the service
8. Once that is created we should be able to access this in a new browser tab

http://3.80.222.215:30080/

This is my public IP address and remember the nodePort which is 30080

Figure 1-3



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

So that means we can access this external service from outside the cluster