**20. Kubernetes Services – Demo**

--- **Reference** - https://github.com/stacksimplify/kubernetes-fundamentals/tree/master/05-Services-with-kubectl

**Introduction to Services**

--- **Service Types**

1. **ClusterIp**
2. **NodePort**
3. **LoadBalancer**
4. **ExternalName**

--- We are going to look in to ClusterIP and NodePort in this section with a detailed example.

--- LoadBalancer Type is primarily for cloud providers and it will differ cloud to cloud, so we will do it accordingly (per cloud basis)

--- ExternalName doesn't have Imperative commands and we need to write YAML definition for the same, so we will look in to it as and when it is required in our course.

**ClusterIP Service - Backend Application Setup**

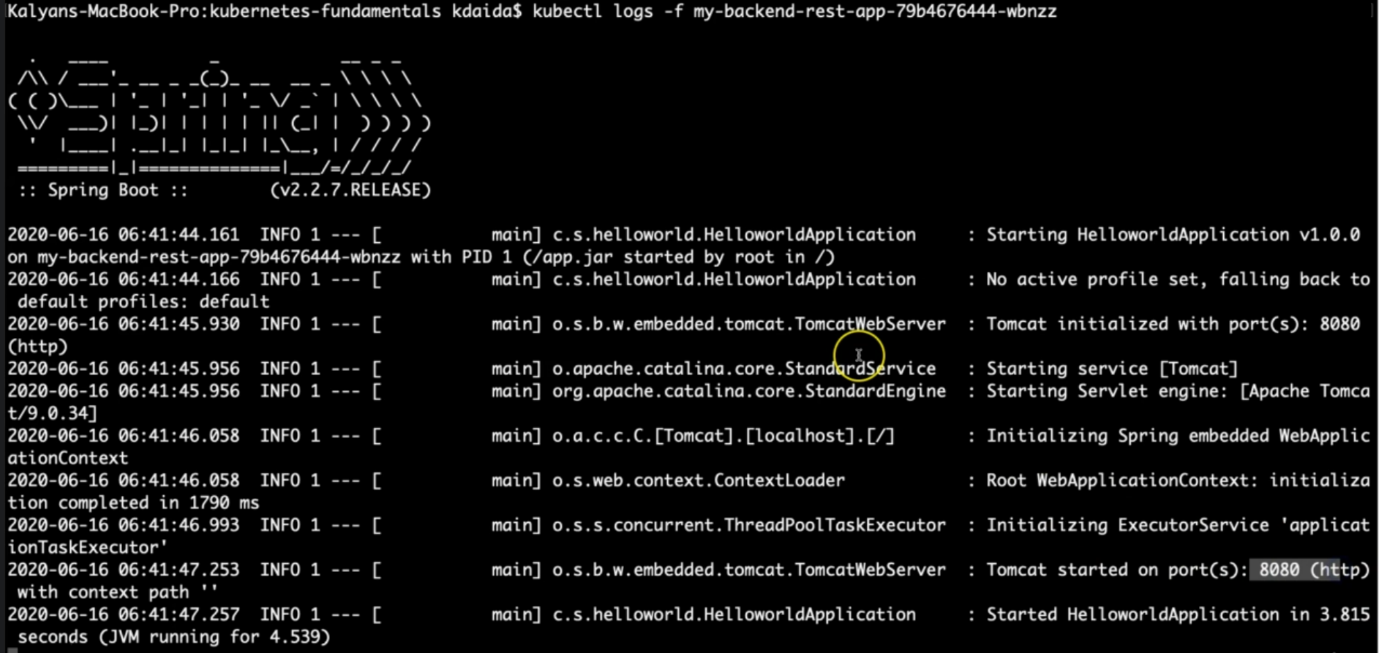
--- Create a deployment for Backend Application (Spring Boot REST Application)

--- Create a ClusterIP service for load balancing backend application.

**# Create Deployment for Backend Rest App**

--- **kubectl create deployment my-backend-rest-app --image=stacksimplify/kube-helloworld:1.0.0**

--- kubectl log -f my-backend-rest-app – you want to see the logs of your application.



--- **note** – our application started successfully.

--- **kubectl get deploy**

**# Create ClusterIp Service for Backend Rest App**

--- **kubectl expose deployment my-backend-rest-app --port=8080 --target-port=8080 --name=my-backend-service**

--- **kubectl get svc**

--- **Observation**: We don't need to specify "--type=ClusterIp" because default setting is to create ClusterIp Service.

--- **Important Note**: If backend application port (Container Port: 8080) and Service Port (8080) are same we don't need to use --target-port=8080 but for avoiding the confusion i have added it. Same case applies to frontend application and service.

--- Backend HelloWorld Application Source kube-helloworld (https://github.com/stacksimplify/kubernetes-fundamentals/tree/master/00-Docker-Images/02-kube-backend-helloworld-springboot/kube-helloworld)

**NodePort Service - Frontend Application Setup**

--- We have implemented NodePort Service multiple times so far (in pods, replicasets and deployments), even then we are going to implement one more time to get a full architectural view in relation with ClusterIp service.

--- Create a deployment for Frontend Application (Nginx acting as Reverse Proxy)

--- Create a NodePort service for load balancing frontend application.

--- **Important Note**: In Nginx reverse proxy, ensure backend service name my-backend-service is updated when you are building the frontend container. We already built it and put ready for this demo (stacksimplify/kube-frontend-nginx:1.0.0)

--- **Nginx Conf File**

server {

    listen       80;

    server\_name  localhost;

    location / {

    # Update your backend application Kubernetes Cluster-IP Service name and port below

    # proxy\_pass http://<Backend-ClusterIp-Service-Name>:<Port>;

    proxy\_pass http://my-backend-service:8080;

    }

    error\_page   500 502 503 504  /50x.html;

    location = /50x.html {

        root   /usr/share/nginx/html;

    }

}

--- Docker Image Location: https://hub.docker.com/repository/docker/stacksimplify/kube-frontend-nginx

--- Frontend Nginx Reverse Proxy Application Source kube-frontend-nginx (https://github.com/stacksimplify/kubernetes-fundamentals/tree/master/00-Docker-Images/03-kube-frontend-nginx)

**# Create Deployment for Frontend Nginx Proxy**

--- **kubectl create deployment my-frontend-nginx-app --image=stacksimplify/kube-frontend-nginx:1.0.0**

--- **kubectl get deploy**

**# Create ClusterIp Service for Frontend Nginx Proxy**

--- **kubectl expose deployment my-frontend-nginx-app --type=NodePort --port=80 --target-port=80 --name=my-frontend-service**

--- **kubectl get svc**

**# Capture IP and Port to Access Application**

--- **kubectl get svc**

--- **kubectl get nodes -o wide**

--- **http://<node1-public-ip>:<Node-Port>/hello**

**# Scale backend with 10 replicas**

--- **kubectl scale --replicas=10 deployment/my-backend-rest-app**

**# Test again to view the backend service Load Balancing**

**http://<node1-public-ip>:<Node-Port>/hello**

**Pending Topics**

--- We will look in these items when we progress in to course on that respective cloud provider

1. **LoadBalancer**
2. **ExternalName**