Lesson 6 Lesson-End Project

**Implement Ingress for Multiple Containers with AKS**

**Project agenda:** To implement Ingress for multiple containers with AKS.

**Description:** Your organization has an AKS cluster and wants to access multiple microservices using the Ingress controller. Configure AKS with Ingress to get an external URL to access applications based on the context root with the same external URL.

**Tools required:** kubeadm, kubectl, kubelet, and docker

**Prerequisites:** kubeadm, kubectl, kubelet, and docker must be installed

**Expected deliverables:** A Kubernetes cluster with high availability enabled

Steps to be followed:

1. Creating an AKS cluster
2. Deploying the Ingress controller on an AKS cluster using Helm
3. Deploying the application Deployment and Service
4. Deploying the Ingress YAML for redirecting the traffic to multiple containers in a Pod

**Step** **1**: **Creating an AKS cluster**

|  |
| --- |
| **Note:** Refer Demos 1 and 3 of Lesson 8 to create an AKS cluster |

**Step 2: Deploying the Ingress controller on an AKS cluster using Helm**

1. Create a separate namespace for the Ingress controller using the following command:

**kubectl create namespace ingress**

To check the created namespace, use the following command:

**kubectl get namespaces**

A screenshot of a computer

Description automatically generated

1. Add the **ingress-nginx** repository using the following command:

**helm** **repo add ingress-nginx https://kubernetes.github.io/ingress-nginx**

A screenshot of a computer

Description automatically generated

1. Use Helm to deploy an NGINX Ingress controller:

**helm install nginx-ingress ingress-nginx/ingress-nginx \**

**--namespace ingress \**

**--set controller.replicaCount=2 \**

**--set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \**

**--set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux**

A screenshot of a computer

Description automatically generated

**Step 3: Deploying the Application Deployment and Service**

1. Create an application Deployment manifest file using the below code:

**vi deploy.yaml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: planview-app**

**labels:**

**product: planview**

**spec:**

**replicas: 1**

**selector:**

**matchLabels:**

**app: planview**

**tier: web**

**template:**

**metadata:**

**labels:**

**app: planview**

**tier: web**

**spec:**

**containers:**

**- name: app-container**

**image: jocatalin/kubernetes-bootcamp:v1**

**ports:**

**- containerPort: 8080**

**resources:**

**limits:**

**cpu: 1000m**

**memory: 600Mi**

**requests:**

**cpu: 500m**

**memory: 300Mi**

**- name: nginx**

**image: nginx**

**ports:**

**- containerPort: 80**

**resources:**

**limits:**

**cpu: 400m**

**memory: 200Mi**

**requests:**

**cpu: 200m**

**memory: 100Mi**

**---**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: planview-service**

**labels:**

**app: planview**

**spec:**

**ports:**

**- name: bootcamp-container**

**port: 8080**

**targetPort: 8080**

**- name: nginx**

**port: 80**

**targetPort: 80**

**type: LoadBalancer**

**selector:**

**app: planview**

**tier: web**

1. Create a Service and Deployment using the following command:

**kubectl apply -f deploy.yaml**

**A screenshot of a computer

Description automatically generated**

1. To verify that the Pods, Services, and Deployments have been created, use the following commands:

**kubectl get deployments -o wide**

**kubectl get pods -o wide**

**kubectl get services -o wide**

**A screenshot of a computer

Description automatically generated**

**Step 4: Deploying ingress.YAML for redirecting the traffic to multiple containers in a Pod**

1. Create an Ingress manifest file using the below code:

**vi ingress.yaml**

**apiVersion: networking.k8s.io/v1**

**kind: Ingress**

**metadata:**

**name: planviewapp-ingress**

**annotations:**

**nginx.ingress.kubernetes.io/ssl-redirect: "false"**

**nginx.ingress.kubernetes.io/use-regex: "true"**

**nginx.ingress.kubernetes.io/rewrite-target: /$2**

**spec:**

**ingressClassName: nginx**

**rules:**

**- http:**

**paths:**

**- path: /service**

**pathType: Prefix**

**backend:**

**service:**

**name: planview-service**

**port:**

**number: 8080**

**- path: /web**

**pathType: Prefix**

**backend:**

**service:**

**name: planview-service**

**port:**

**number: 80**

1. Create an Ingress using the following command:

**kubectl apply -f ingress.yaml**

To check the created Ingress, use the following command:

**kubectl get ingress**

**A screenshot of a computer

Description automatically generated**

1. Once the Ingress is deployed, get an external IP from the NGINX Ingress controller service in the Ingress namespace using the below command:

**kubectl get services -o wide -n ingress**

A screenshot of a computer

Description automatically generated

Take the external IP and use context paths from the Ingress to redirect the traffic across Services and Pods

Open Google Chrome, type **<External IP>/service**, and press the enter key:

Graphical user interface, text, application

Description automatically generated

Open Google Chrome, type **<External IP>/web**, and press the enter key:

Graphical user interface, text, application

Description automatically generated

You have successfully implemented the Ingress controller.