Tutorial: Deploying Apache Web Server in Kubernetes with Hostname Display on the Home page

Introduction:

In this tutorial, we will learn how to deploy an Apache web server in Kubernetes that displays the hostname of the pod it's running on. We will achieve this by using Kubernetes resources such as ConfigMap, Deployment, and Service. By the end of this tutorial, you will have a clear understanding of how to deploy applications in Kubernetes and customize their behavior using these resources.

Working yaml is present in this location:

https://raw.githubusercontent.com/devopsmay24batch/mydevopsrepo/main/kubernetes/apache-hostname.yaml

Use kubectl apply -f apache-hostname.yaml to deploy the objects.

Explanation of Kubernetes Resources:

1. Pod: The smallest and simplest Kubernetes object. A pod represents a single instance of a running process in the cluster, which can contain one or more containers.
2. ConfigMap:
    - A ConfigMap is a Kubernetes resource used to store configuration data in key-value pairs.
    - It allows you to decouple configuration from the container image, making it easier to manage and update configuration settings.
    - In this tutorial, we will use a ConfigMap to store a custom `index.html` file that includes a placeholder for the container hostname.
3. Deployment:

- A Deployment is a Kubernetes resource used to manage the lifecycle of a replicated application.
- It ensures that a specified number of replica Pods are running and handles updates and rollbacks gracefully.
- In this tutorial, we will use a Deployment to manage the Apache web server Pods and ensure that there are always 5 replicas running.

4. Service:
    - A Service is a Kubernetes resource used to expose an application running in a set of Pods as a network service.
    - It provides a stable endpoint for accessing the application and enables load balancing across multiple Pods.
    - In this tutorial, we will use a Service to expose the Apache web server Pods to external traffic and route requests to them.

Explanation of YAML Files:

1. ConfigMap for Apache custom index.html (apache-config.yaml):
    - This YAML file defines a ConfigMap named `apache-config`.
    - It contains a custom `index.html` file with HTML code to display the hostname of the container using the `$HOSTNAME` environment variable.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: apache-config
data:
  index.html: |
    <html>
    <head><title>Apache Pod</title></head>
    <body>
      <h1>Welcome to Apache - This is the custom page example to display container
hostname!</h1>
      <p> Container Host Name: $HOSTNAME </p>
    </body>
    </html>
```

2. Apache Deployment (apache-deployment.yaml):
    - This YAML file defines a Deployment named `apache-homepage-deployment`.
    - It specifies 5 replicas of the Apache web server Pods using the `replicas` field.
    - The Deployment mounts the ConfigMap volume containing the custom `index.html` file into the Apache containers.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-homepage-deployment
  labels:
    app: apache-homepage
spec:
  replicas: 5
  selector:
    matchLabels:
      app: apache-homepage
  template:
    metadata:
      labels:
        app: apache-homepage
    spec:
      initContainers:
      - name: init-apache
        image: alpine:3.10
        command:
          - /bin/sh
          - -c
          - |
            apk add --no-cache gettext
            echo "Copying index.html"
            cat /config/index.html
            envsubst < /config/index.html >
/usr/local/apache2/htdocs/index.html
            ls -l /usr/local/apache2/htdocs
            cat /usr/local/apache2/htdocs/index.html
        volumeMounts:
```

```
    - name: apache-config-volume
      mountPath: /config
    - name: apache-web-content
      mountPath: /usr/local/apache2/htdocs
    env:
    - name: HOSTNAME
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
  containers:
  - name: apache
    image: httpd:2.4
    ports:
    - containerPort: 80
    volumeMounts:
    - name: apache-web-content
      mountPath: /usr/local/apache2/htdocs
  volumes:
  - name: apache-config-volume
    configMap:
      name: apache-config
      items:
      - key: index.html
        path: index.html
  - name: apache-web-content
    emptyDir: {}
  tolerations:
  - key: "node-role.kubernetes.io/control-plane"
    operator: "Exists"
    effect: "NoSchedule"
```

3. Apache Service (apache-service.yaml):
    - This YAML file defines a Service named `apache-homepage-service`.
    - It exposes the Apache web server Pods on port 80 within the Kubernetes cluster.
    - The Service is of type `NodePort`, allowing external access to the Apache web server using a randomly assigned NodePort.

```
apiVersion: v1
kind: Service
metadata:
  name: apache-homepage-service
  labels:
    app: apache-homepage
spec:
  selector:
    app: apache-homepage
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30081
  type: NodePort
```

Conclusion:

In this tutorial, we have learned how to deploy an Apache web server in Kubernetes and customize its behavior to display the hostname of the container it's running on. By leveraging Kubernetes resources such as ConfigMap, Deployment, and Service, we can easily manage and expose our applications in Kubernetes, enhancing their scalability and reliability.