Task-8: Deploying a Java App to a Local Kubernetes Cluster Using Minikube in Google Cloud Shell

The following sequence steps are carried out in this task.

1. Created a simple Java application.
2. Built a Docker image for the application.
3. Loaded the image into Minikube.
4. Deployed the application using a Kubernetes Deployment.
5. Exposed the application using a Kubernetes Service.
6. Accessed the application and verified its logs.

Step 1: Create a Simple Java Application

1. Create a directory for your Java application:

```
$mkdir java-app

$cd java-app
```

2. Create a **`HelloWorld.java`** file:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
        while (true) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

3. Compile the Java application:

```
$javac HelloWorld.java
```

Step 2: Create a Dockerfile

1. Create a **`Dockerfile`** in the same directory:

```
FROM openjdk:17
# Set the working directory
WORKDIR /app

# Copy the compiled Java application to the container
COPY HelloWorld.class .

# Run the Java application
CMD ["java", "HelloWorld"]
```

2. Build the Docker image:

```
$docker build -t java-hello-world:1.0 .
```

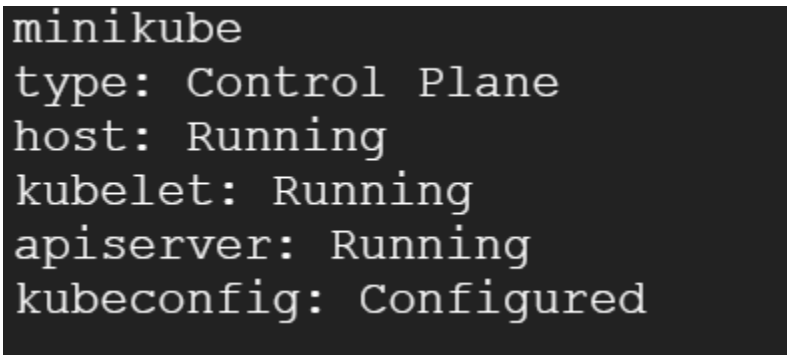
Step 3: Start Minikube

1. Start Minikube:

```
$minikube start
```

2. Verify Minikube is running:

```
$minikube status
```



Step 4: Load the Docker Image into Minikube

```
$minikube image load java-hello-world:1.0
```

- **Verify the image is loaded:**

```
$minikube image ls
```

```
registry.k8s.io/pause:3.10
registry.k8s.io/kube-scheduler:v1.32.0
registry.k8s.io/kube-proxy:v1.32.0
registry.k8s.io/kube-controller-manager:v1.32
registry.k8s.io/kube-apiserver:v1.32.0
registry.k8s.io/etcd:3.5.16-0
registry.k8s.io/coredns/coredns:v1.11.3
gcr.io/k8s-minikube/storage-provisioner:v5
docker.io/library/java-hello-world:1.0
```

Step 5: Create a Kubernetes Deployment

1. Create a `**deployment.yaml**` file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: java-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: java-app
  template:
    metadata:
      labels:
        app: java-app
    spec:
      containers:
      - name: java-app
```

```
            image: java-hello-world:1.0
            imagePullPolicy: IfNotPresent
```

2. Apply the deployment:

```
kubectl apply -f deployment.yaml
```

```
deployment.apps/java-app created
```

3. Verify the deployment:

```
$kubectl get deployments
```

```
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
java-app   3/3     3            3           49m
```

```
$kubectl get pods
```

```
NAME                        READY   STATUS    RESTARTS   AGE
java-app-69755d76f8-8fvnc   1/1     Running   0          50m
java-app-69755d76f8-cz572   1/1     Running   0          50m
java-app-69755d76f8-vwwh2   1/1     Running   0          50m
```

Step 6: Expose the Application as a Service

1. Create a `**service.yaml**` file:

```
apiVersion: v1
kind: Service
metadata:
  name: java-app-service
spec:
  selector:
    app: java-app
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 8080
```

```
    type: NodePort
```

2. Apply the service:

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

```
service/java-app-service created
```

3. Verify the service:

```
$kubectl get services
```

```
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
java-app-service   NodePort    10.101.55.61    <none>        8080:31614/TCP   36s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          74m
```

Step 7: Access the Application

```
$kubectl logs java-app-69755d76f8-8fvnc
```

Output:

```
Hello, World!
```

```
$kubectl logs java-app-69755d76f8-cz572
```

Output:

```
Hello, World!
```

```
$kubectl logs java-app-69755d76f8-vwwh2
```

Output:

```
Hello, World!
```

Step 8: Clean Up

1. Delete the deployment and service:

```
$kubectl delete -f deployment.yaml
```

```
deployment.apps "java-app" deleted
```

```
$kubectl delete -f service.yaml
```

```
service "java-app-service" deleted
```

2. Stop Minikube:

$minikube stop

```
* Stopping node "minikube"  ...
* Powering off "minikube" via SSH ...
* 1 node stopped.
```

3. Delete Minikube cluster (optional):

$minikube delete

```
* Deleting "minikube" in docker ...
* Deleting container "minikube" ...
* Removing /google/minikube/.minikube/machines/minikube ...
* Removed all traces of the "minikube" cluster.
```