SOFTWARE ENGINEERING LAB

EXERCISE – 7

TOPIC – 5

DEPLOYING AND SCALING APPLICATIONS USING MINIKUBE

How to Set Up, Scale, and Stop Nginx Using Kubernetes (Minikube)

What Are We Doing?

In this guide, we will:

1. **Set up a mini-Kubernetes system on your computer** using a tool called Minikube. This is like creating a small virtual computer where we can run apps.
2. **Create a Nginx app** (Nginx is a simple web server that shows websites).
3. **Make the Nginx app accessible** from your browser so you can see it on your screen.
4. **Create more copies of the Nginx app** (this helps it handle more users or traffic).
5. **Clean up everything** when we're done so nothing is left running.

This process will work the same on **Windows, macOS, and Linux**, with only a few small differences in how Minikube is set up on each system.

Step 1: Setting Up Minikube and Kubernetes

1.1 Start Minikube

Run this command to start the Minikube cluster:

```
minikube start
```

- **Minikube** creates a **local Kubernetes cluster**. This cluster helps us manage and run applications on our machine. Think of it as a small **virtual computer** running in the background that Kubernetes will use to manage apps.

Step 2: Create and manage the Nginx Deployment

2.1 Create the Nginx Deployment

Run the following command to create a new deployment (which is like creating an app) for **Nginx**:

```
kubectl create deployment mynginx --image=nginx
```

- **Explanation of the command**:
  - `kubectl`: This is the tool we use to talk to Kubernetes.
  - `create deployment`: This tells Kubernetes to create a **Deployment**, which is a way of managing and running apps.
  - `mynginx`: This is the name we give to our app. You can call it anything you want, but we are calling it **mynginx**.
  - `--image=nginx`: This tells Kubernetes to use the **nginx image** (a ready-made version of Nginx) to create the app. An **image** is like a blueprint for creating an app.

You can verify if the Nginx deployment was created correctly by running this command:

```
kubectl get deployments
```

You should see **mynginx** listed as a deployment.

Step 3: Expose the Nginx Deployment to the Outside World

3.1 Expose the Service

We will use the following command to **expose** the Nginx app to the outside world:

```
kubectl expose deployment mynginx --type=NodePort --port=80 --target-port=80
```

- **Explanation of the command**:
  - `kubectl expose`: This tells Kubernetes to create a **Service**. A Service is a way to expose your app to the outside world.

- `deployment mynginx`: This specifies which app (Deployment) we want to expose.
- `--type=NodePort`: This makes the service accessible **outside the Kubernetes cluster** by opening a specific port on your computer.
- `--port=80`: This is the port we'll use to access Nginx from outside the cluster (port 80 is the default for web servers).
- `--target-port=80`: This is the port inside the pod where Nginx is running (also port 80).

Step 4: Scale the Deployment (Make More Copies)

4.1 Scale the Deployment

To scale Nginx to **4 replicas** (running 4 copies of the app), run the following command:

```
kubectl scale deployment mynginx --replicas=4
```

- **Explanation of the command**:
  - `kubectl scale`: This tells Kubernetes to **increase or decrease** the number of **pods** running.
  - `--replicas=4`: This specifies that we want **4 copies** (pods) of the Nginx app running at the same time.

Step 5: Accessing the Nginx App

5.1 What is Port Forwarding?

Port forwarding is a way to **forward traffic** from a port on your computer (like **8081**) to a port inside the Kubernetes cluster (like **port 80** in the Nginx pod).

We need **port forwarding** because **Jenkins is already using port 8080** on your computer. That's why we forward traffic from **8081** to **port 80** on Nginx.

5.2 Run Port Forwarding

Run this command to forward port **8081** on your local machine to port **80** inside the Nginx container:

```
kubectl port-forward svc/mynginx 8081:80
```

- **Explanation of the command**:
  - `kubectl port-forward`: This command forwards traffic from one port to another.
  - `svc/mynginx`: This refers to the **mynginx service** we created earlier.
  - `8081:80`: This forwards **port 8081** on your local machine to **port 80** inside the Nginx container.

- Open your browser and go to:

```
http://localhost:8081
```

You should see the **Nginx welcome page**, confirming that the app is running!

Step 6: Alternative Method – Minikube Tunnel

If **port forwarding** doesn't work for you, or if you prefer an alternative method, you can use **Minikube Tunnel**.

6.1 Start Minikube Tunnel

Run this command to create a tunnel that lets you access the service directly:

```
minikube tunnel
```

- **Explanation**: This command creates a **tunnel** between your local machine and the Kubernetes cluster, allowing you to access the service.

6.2 Get the Minikube Service URL

Once the tunnel is running, get the URL of the service using:

```
minikube service mynginx --url
```

- **What happens**: This command gives you a URL (like `http://192.168.x.x:80`) that you can open in your browser to access Nginx.

6.3 Open the URL in Your Browser

Copy and paste the provided URL into your browser to access Nginx directly.

Step 7: Stopping and Cleaning Up Everything

7.1 Stop the Nginx Deployment and Service

To delete the **Nginx deployment** and **service**, run:

```
kubectl delete deployment mynginx
kubectl delete service mynginx
```

7.2 Stop Minikube (Optional)

If you're done with Minikube, stop it to free up system resources:

```
minikube stop
```

7.3 Delete the Minikube Cluster (Optional)

To delete the entire Minikube cluster, run:

```
minikube delete
```