

# Deploy a Web Application Using a Kubernetes Deployment and Service

---

## Table of Contents

1. **Introduction**
  2. **Problem Statement**
  3. **Prerequisites**
  4. **Setup Instructions**
    - **Step 1: Create a Deployment for NGINX**
    - **Step 2: Create a Service to Expose NGINX**
    - **Step 3: Accessing the NGINX Application**
  5. **References**
- 

## Introduction

This guide is designed to help users set up and deploy web applications in a local Kubernetes environment using Minikube on a Windows system. Minikube is an open-source tool that runs a single-node Kubernetes cluster on your local machine, providing an easy-to-use environment for Kubernetes learning, development, and testing.

**By following this guide, you will learn:**

- How to set up a Kubernetes cluster locally using Minikube.
  - How to deploy web applications using Kubernetes Deployment and manage their lifecycle.
  - How to expose your web applications to internal or external traffic using Kubernetes Service.
- 

## Problem Statement

In this lab, we will deploy an NGINX-based web application on a local Kubernetes cluster using **Minikube**. We will create a Kubernetes Deployment to manage the application's lifecycle and a Service to expose it.

---

## Prerequisites

Completion of all previous lab guides (up to Lab Guide-01) is required before proceeding with Lab Guide-02.

- Minikube is running on your Windows system.
  - `kubectl` is installed and configured to interact with your Minikube cluster.
- 

## Setup Instructions

Step 1: Create a Deployment for NGINX

We will start by deploying NGINX using a Kubernetes Deployment. The Deployment ensures that NGINX is running in a stable state with three replicas.

### 1. Create the YAML for Deployment

Create a new file called `nginx-deployment.yaml` in your working directory with the following content:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
```

#### Key points:

- **replicas:** 3 instances of NGINX will be deployed.
- **image:** NGINX is pulled from the official Docker image (`nginx:latest`).
- **ports:** Exposes port 80 inside the container.

### 2. Deploy NGINX to Kubernetes

Apply the `nginx-deployment.yaml` file by running the following command:

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

```
PS C:\Users\Administrator> kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
```

### 3. Verify the Deployment

Check the status of the deployment and ensure the pods are running:

```
kubectl get deployments
kubectl get pods
```

```
PS C:\Users\Administrator> kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment    3/3     3             3           8m43s
PS C:\Users\Administrator> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-54b9c68f67-cq4gv   1/1     Running   0          8m53s
nginx-deployment-54b9c68f67-k8zhc   1/1     Running   0          8m53s
nginx-deployment-54b9c68f67-p9d2q   1/1     Running   0          8m53s
```

You should see three pods running for the `nginx-deployment`.

## Step 2: Create a Service to Expose NGINX

Next, we will create a Service to expose the NGINX application inside the cluster. A **ClusterIP** service will be used to make the application accessible to other pods within the cluster.

### 1. Create the YAML for Service

Create a new file called `nginx-service.yaml` with the following content:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
```

#### Key points:

- **type: ClusterIP:** Exposes the service within the cluster (internal access).
- **selector:** Targets the pods labeled with `app: nginx`.
- **ports:** Exposes port 80 both internally and in the container.

### 2. Deploy the Service

Apply the `nginx-service.yaml` by running the following command:

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

### 3. Verify the Service

Check the service status and ensure it's running:

```
kubectl get services
```

```
PS C:\Users\Administrator> kubectl apply -f nginx-service.yaml
service/nginx-service created
PS C:\Users\Administrator> kubectl get services
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1     <none>       443/TCP    26m
nginx-service       ClusterIP   10.107.90.12  <none>       80/TCP     15s
PS C:\Users\Administrator>
```

You should see the **nginx-service** listed.

### Step 3: Accessing the NGINX Application

Since this service is internal to the cluster, you can use **kubectl** to forward the local port to access NGINX from your browser.

#### 1. Port-forward the Service

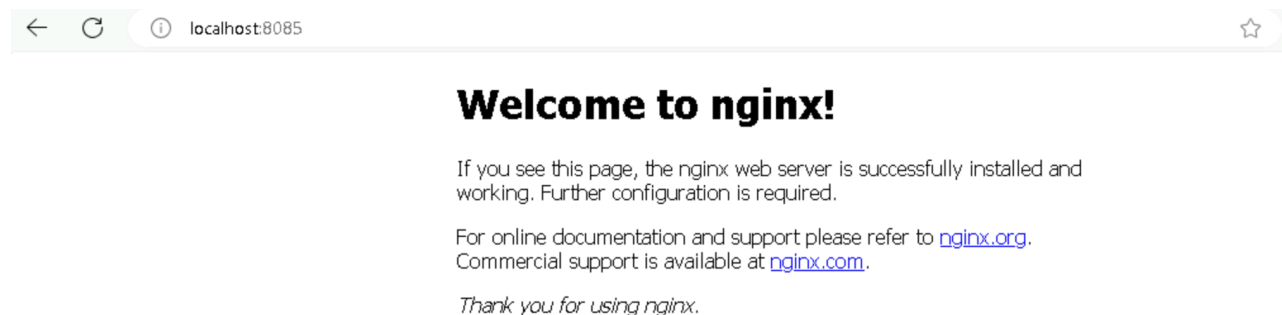
Run the following command to forward a local port (e.g., 8085) to the NGINX pod:

```
kubectl port-forward service/nginx-service 8085:80
```

```
PS C:\Users\Administrator> kubectl port-forward service/nginx-service 8085:80
Forwarding from 127.0.0.1:8085 -> 80
Forwarding from [::1]:8085 -> 80
Handling connection for 8085
Handling connection for 8085
```

#### 2. Open NGINX in Your Browser

Open your browser and navigate to <http://localhost:8085>. You should see the default NGINX welcome page.



- [Kubernetes Deployment Documentation](#)
  - [Kubernetes Service Documentation](#)
-