# Java Demo App Kubernetes Deployment - Case Scenario

## Overview

This repository contains the Kubernetes manifests and configuration files to deploy the \*\*Java Demo App\*\* on a local Minikube cluster.

## Resources

- \*\*Java Demo App Source:\*\* [https://github.com/benc-uk/java-demoapp/](https://github.com/benc-uk/java-demoapp/)

- \*\*Kubernetes:\*\* Minikube (local cluster)

- \*\*Workspace:\*\* Local machine or VM (any cloud provider)

---

## Scenario Requirements

### 1. Application Deployment

- Deploy the Java Demo App on a Minikube Kubernetes cluster using deployment and service manifests.

### 2. Ingress Setup

- Configure Ingress on Minikube to expose the application over \*\*HTTP\*\* and \*\*HTTPS\*\*.

- Use a \*\*self-signed SSL certificate\*\* to enable HTTPS access.

### 3. Health Checks

- Implement \*\*readiness\*\* and \*\*liveness probes\*\* in the deployment manifest to monitor app health and improve reliability.

### 4. Optional Features

- Configure \*\*Horizontal Pod Autoscaler (HPA)\*\* to dynamically scale the app based on CPU usage.

- Set up \*\*Ingress redirect\*\* to automatically forward HTTP traffic to HTTPS for secure communication.

### 5. DNS and Access

- No DNS records are required.

- Use the `/etc/hosts` file (or `C:\Windows\System32\drivers\etc\hosts` on Windows) to map Minikube’s IP address to a hostname, e.g., `local.java-demo.com`.

---

## Deployment and Setup Steps

1. \*\*Initialize and Start Minikube\*\*

The local Kubernetes cluster was started using Minikube, providing a lightweight environment for deployment.

2. \*\*Build and Prepare Kubernetes Manifests\*\*

Kubernetes manifests for Deployment, Service, and Ingress resources were created to define application deployment, networking, and exposure.

3. \*\*Deploy Application to Minikube\*\*

The manifests were applied to the cluster to deploy the Java Demo App and expose it internally via a NodePort service.

4. \*\*Configure Ingress Controller\*\*

An NGINX Ingress Controller was enabled and configured on Minikube to manage inbound HTTP and HTTPS traffic.

5. \*\*Set Up Self-Signed SSL Certificate\*\*

A self-signed SSL certificate was generated and configured in Kubernetes as a TLS secret to enable HTTPS on the Ingress resource.

6. \*\*Configure Health Checks\*\*

Readiness and liveness probes were added to the Deployment manifest to ensure Kubernetes can monitor the application’s health status effectively.

7. \*\*Implement Horizontal Pod Autoscaling (Optional)\*\*

A Horizontal Pod Autoscaler was configured to monitor CPU usage and scale the number of pods between minimum and maximum limits.

8. \*\*Configure HTTP to HTTPS Redirection (Optional)\*\*

Annotations were added to the Ingress resource to automatically redirect HTTP traffic to HTTPS for enhanced security.

9. \*\*Update Hosts File for Local DNS Resolution\*\*

The Minikube IP address was mapped to a custom hostname in the local hosts file to allow easy access via `local.java-demo.com`.

10. \*\*Verification and Testing\*\*

The deployment was tested by accessing the application over both HTTP and HTTPS, verifying health check behavior, autoscaling functionality, and redirect rules.

---

## Challenges and Resolutions

- \*\*Minikube Tunnel Requirement for HTTPS Access:\*\*

To expose the Ingress controller's HTTPS endpoint externally, `minikube tunnel` was used to enable proper routing and load balancing on local Windows environment.

- \*\*Git Repository Management:\*\*

Managed branch naming and remote URL configuration carefully to push Kubernetes manifests and related files to the designated GitHub repository.

- \*\*Windows Environment Specific Adjustments:\*\*

Special care was taken to update Windows hosts file and run Minikube tunnel as a background service for seamless local development.

---

## Summary

This case scenario demonstrates how to deploy a Java web application on Kubernetes using Minikube, with a focus on secure ingress exposure, application health monitoring, and scalability. The setup provides a practical example suitable for interviews or demos showcasing Kubernetes deployment skills in a local development environment.

##Evidence:

HTTPS LOCAL Control :

A screenshot of a computer

AI-generated content may be incorrect.

##POD Control:

A screenshot of a computer

AI-generated content may be incorrect.

##SERVICE Control :



##INGRESS Control :



A black screen with white text

AI-generated content may be incorrect.

##HPA Test:



Readiness Test:



