# Kubernetes Hello World Demo

This project is a simple demo application that connects a React frontend with a Java Spring Boot backend to deploy and expose services on a local Kubernetes cluster (via Minikube). - The frontend has a single button labeled Create A.
- When clicked, it sends a request to the backend.
- The backend creates a new Kubernetes Deployment and Service (using NGINX as the container).
- The backend automatically sets up a kubectl port-forward and returns a localhost endpoint.
- Visiting that endpoint shows the default NGINX page, confirming the pod is running in Kubernetes.

## How to Run the Application

Start Minikube:
minikube start --driver=docker

Run the backend:
cd hello-backend && mvn spring-boot:run

Run the frontend:
cd hello-frontend && npm install && npm start

Open the frontend in the browser at http://localhost:3000 and click Create A.

The UI will display the Kubernetes service endpoint (e.g., http://127.0.0.1:55685).

Open that endpoint in a browser to see the NGINX welcome page.

## Challenges Faced

- The original NodePort URLs (192.168.49.x:) often hung on macOS with Docker/Minikube.
- To fix this, the backend was updated to run a kubectl port-forward automatically and return a localhost link instead.
- kubectl port-forward is a command that forwards traffic from your local machine to a specific pod or service in the cluster. It allows you to access applications running inside Kubernetes using a normal localhost address. - This change made the endpoint reliable and fast, so the demo works smoothly.

## What I Learned

- How to integrate a React frontend with a Spring Boot backend.
- How to programmatically create Kubernetes Deployments and Services from Java.
- How to fix networking issues on macOS by switching from NodePort to port-forwarding and learning how kubectl port-forward works.