Node.js Application Configuration, Dockerization, Kubernetes Provisioning, and Monitoring Project

Purpose

The purpose of this project is to efficiently configure, provision, and monitor a Node.js application, ensuring its smooth deployment, scalability, and observability, while adhering to modern best practices and technologies.

Objectives

Configure Node.js application for various environments (e.g., development, staging, production) with well-defined settings and environment variables.

Dockerize the Node.js application for easy deployment and portability.

Provision the application using Kubernetes for efficient management and scalability.

Set up advanced monitoring and alerting solutions using Prometheus and Grafana.

Node.js Application Description

For this project, we have created a simple RESTful API using Node.js and Express.js. It implements endpoints for basic CRUD operations (Create, Read, Update, Delete) on a resource (e.g., users, products), including validation, error handling, and authentication using middleware.

Project Structure

markdown

/ project-root

|- node-app/

|- ... (Node.js application files)

|- prometheus/

|- prometheus.yml

|- Dockerfile

|- kubernetes/

|- deployment.yaml

|- service.yaml

|- ingress.yaml

|- grafana/

|- dashboard.json

|- README.md

Step-by-Step Instructions

Node.js Application Configuration

Create a directory structure for different environments (e.g., development, staging, production).

Create configuration files (e.g., JSON or YAML) for each environment, specifying environment-specific settings.

Manage sensitive information like API keys and database credentials securely using environment variables or configuration management tools.

Dockerize Node.js Application

Specify the base image (e.g., Node.js image) in the Dockerfile and include any necessary dependencies.

Copy your application code into the Docker image.

Set environment variables and configure the container to run the Node.js application.

Build the Docker image using the docker build command.

Test the Docker image by running it locally using docker run.

Provisioning with Kubernetes

Install and configure Minikube for local Kubernetes cluster provisioning.

Develop Kubernetes YAML files for deploying the Node.js application.

Define a Kubernetes Deployment resource to manage the application's pods.

Specify a Kubernetes Service resource to expose the application within the cluster.

Set up Ingress to enable external access to the Node.js application.

Deploy the application to the Minikube cluster using kubectl apply.

Monitoring Setup

Install and configure Prometheus to collect metrics from the Node.js application.

Create Prometheus configuration files to define scraping targets and alerting rules.

Set up Grafana and configure it to visualize Prometheus data.

Design a Grafana dashboard that displays key performance metrics.

Test the monitoring setup by verifying that metrics are collected and displayed in Grafana./ project-root

|- node-app/

|- ... (Node.js application files)

|- prometheus/

|- prometheus.yml

|- Dockerfile

|- kubernetes/

|- deployment.yaml

|- service.yaml

|- ingress.yaml

|- grafana/

|- dashboard.json

|- README.md