# AI-Driven Kubernetes Health Monitoring System

Phase 1 Project Report  
  
Submitted By: error\_makers  
  
Date: March 23, 2025

## Abstract

The AI-Driven Kubernetes Health Monitoring System is designed to enhance monitoring and management of Kubernetes clusters by integrating AI-powered anomaly detection models with Prometheus and Grafana for real-time visualization. This system aims to predict potential cluster failures, automate issue resolution, and improve overall system health. Phase 1 focused on Kubernetes setup, MongoDB integration, and API creation.

## Table of Contents

1. 1. System Overview
2. 2. System Architecture
3. 3. Kubernetes and MongoDB Setup Steps
4. 4. Challenges Faced
5. 5. Future Work Plan
6. 6. References

## 1. System Overview

The AI-Driven Kubernetes Health Monitoring System is built to monitor Kubernetes clusters, predict anomalies using machine learning models, and provide real-time health data to administrators. This system automates monitoring tasks, tracks cluster status, and stores data in MongoDB for further analysis.

## 2. System Architecture

The system architecture involves Kubernetes Pods, Services, ConfigMaps, and a MongoDB database. AI/ML-based anomaly detection models collect metrics through Prometheus and visualize the cluster's health using Grafana. Below is a high-level diagram of the architecture (to be included).

\*\*Key Components:\*\*

* 1. \*\*Kubernetes Pods:\*\* Running Flask-based APIs to interact with MongoDB.
* 2. \*\*MongoDB:\*\* Used as the primary database to store user and health-related data.
* 3. \*\*ConfigMap:\*\* Contains sensitive data such as MongoDB connection strings.
* 4. \*\*Prometheus and Grafana (planned for Phase 2):\*\* For metrics collection and visualization.

## 3. Kubernetes and MongoDB Setup Steps

Below are the steps that were followed during the Phase 1 Kubernetes and MongoDB setup. This includes key commands and expected outputs.

### Kubernetes Setup Commands:

* `kubectl apply -f deployment.yaml` – Deploy the AI-K8s Health Monitor and MongoDB pods.
* `kubectl get pods -o wide` – Verify pod status and IP addresses.
* `kubectl describe configmap mongo-config` – View MongoDB ConfigMap settings.
* `kubectl logs <pod\_name>` – Check pod logs for debugging.

### MongoDB Shell Commands:

* `mongosh --host localhost --port 27017 --username admin --password password --authenticationDatabase admin`
* `show dbs;` – List all databases.
* `use ai\_health\_monitor;` – Switch to the primary database.
* `db.users.find().pretty();` – Display user information stored in MongoDB.

## 4. Challenges Faced

During Phase 1, several challenges were encountered, primarily with MongoDB authentication. Despite creating users and setting up ConfigMaps with the correct URI, the pods faced `AuthenticationFailed` errors while connecting to MongoDB. Multiple troubleshooting steps were performed, including:

* Verifying MongoDB users and roles using the MongoDB shell.
* Checking Kubernetes pod logs using `kubectl logs`.
* Editing and reapplying ConfigMaps with the correct MongoDB URI.
* Restarting pods and verifying their status using `kubectl get pods`.

However, the issue persisted due to possible misconfigurations in Kubernetes or MongoDB’s authentication mechanism. This will be addressed in Phase 2.

## 5. Future Work Plan

The following tasks are planned for Phase 2 of the project to address the current challenges and enhance the system's functionality:

* 1. Resolve MongoDB authentication issues by revisiting ConfigMap and user role configurations.
* 2. Integrate Prometheus and Grafana for real-time metrics collection and visualization.
* 3. Deploy AI/ML-based anomaly detection models to identify potential Kubernetes cluster failures.
* 4. Optimize the Kubernetes deployment configuration to improve performance and scalability.

## 6. References

- Kubernetes Documentation: https://kubernetes.io/docs/  
- MongoDB Documentation: https://www.mongodb.com/docs/  
- Prometheus Documentation: https://prometheus.io/docs/  
- Grafana Documentation: https://grafana.com/docs/

## 7. Project Screenshots









