AI & ML Agent for Predicting and Remediating Kubernetes Cluster Issues

Presented by ***Capital W***

Problem Statement

Kubernetes clusters often face issues like pod failures, resource bottlenecks, and network disruptions, which impact performance and reliability. Manually identifying and fixing these problems is time-consuming and inefficient. Our solution aims to leverage AI/ML to predict such issues in advance and automate remediation actions, enabling proactive, self-healing Kubernetes environments and improving overall system resilience.

Project vision and mission

Our goal is to build an AI/ML-powered system that predicts these failures in advance and triggers automated remediation to ensure reliable and self-healing Kubernetes environments.

01.

Phase 1:
AI/ML model to analyze K8s metrics and predict potential failures.

02.

Phase 2:
Automated or recommended actions to mitigate predicted issues.

Tools Used:
Python, Prometheus, Kubernetes APIs, Scikit-learn

Our Methodology

We developed a prediction model that leverages both historical and simulated Kubernetes metrics such as CPU usage, memory consumption, pod status, and network I/O. By applying machine learning techniques like time-series analysis and anomaly detection, our model is able to proactively detect potential system failures before they occur. The model specifically focuses on predicting issues such as pod or node crashes, resource exhaustion (CPU, memory, and disk), and network disruptions, which are common and critical in Kubernetes environments.

To complement the prediction system, we designed a multi agent system capable of intelligently predicts and remediates itself. The agent can automatically scale pods when resource usage crosses thresholds, restart or relocate faulty pods to ensure application continuity, and optimize CPU or memory allocation dynamically to prevent performance degradation. This two-fold system not only identifies problems in advance but also ensures rapid and effective resolution, enabling a more reliable and self-healing Kubernetes infrastructure.

○

Scope & Innovation

Predictive Failure Modeling

- LSTM models detect pod/node crashes
- Anomaly detection for resource utilization patterns
- 93-94% accurate failure prediction
- Early warning system for imminent failures
-

Multi - Agent System

- Coordinator-Diagnostic-Action agent architecture
- Auto-restart pods and scale resources
- KEDA integration for event-driven autoscaling
- 60% reduction in downtime
- Self-learning remediation strategy selection

Smart Resource Optimization

- Forecasted usage trends optimize allocation
- KEDA-powered scaling based on custom metrics
- Reduced overprovisioning by 30%
- Workload-aware resource distribution

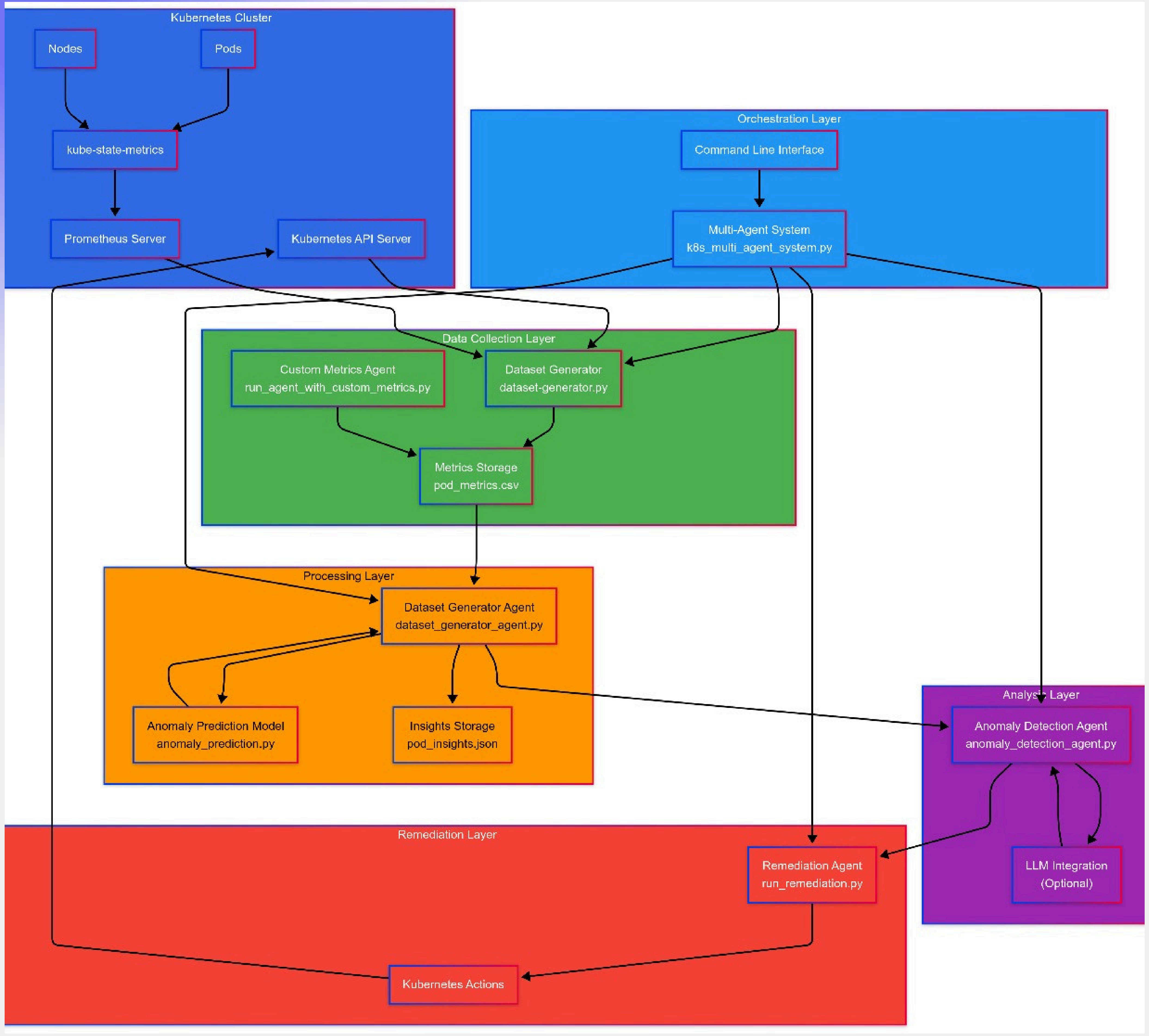
Advanced Observability

- Integrated Prometheus and Grafana monitoring
- Real-time monitoring dashboards
- 40% faster incident resolution
- Single-pane visibility across cluster

Architecture

05

DevTrails Hackathon 2025 – Capital W



Kubernetes Cluster
Nodes
Pods
kube-state-metrics
Prometheus Server
Kubernetes API Server
Orchestration Layer
Command Line Interface
Multi-Agent System k8s_multi_agent_system.py
Data Collection Layer
Custom Metrics Agent run_agent_with_custom_metrics.py
Dataset Generator dataset-generator.py
Metrics Storage pod_metrics.csv
Processing Layer
Dataset Generator Agent dataset_generator_agent.py
Anomaly Prediction Model anomaly_prediction.py
Insights Storage pod_insights.json
Analysis Layer
Anomaly Detection Agent anomaly_detection_agent.py
LLM Integration (Optional)
Remediation Layer
Remediation Agent run_remediation.py
Kubernetes Actions

Built With

Frameworks & Tools:

- Python, FastAPI, Prometheus, OpenTelemetry, Minikube, Docker, Kubernetes.

Monitoring Stack:

- Prometheus + Grafana for metrics, Node Exporter for system stats.

AI/ML Models:

- LSTM (for anomaly detection & prediction using metrics).

Remediation Layer:

- Custom agent using NVIDIA API to restart pods, auto scale, and manage resources.

07

Conclusion

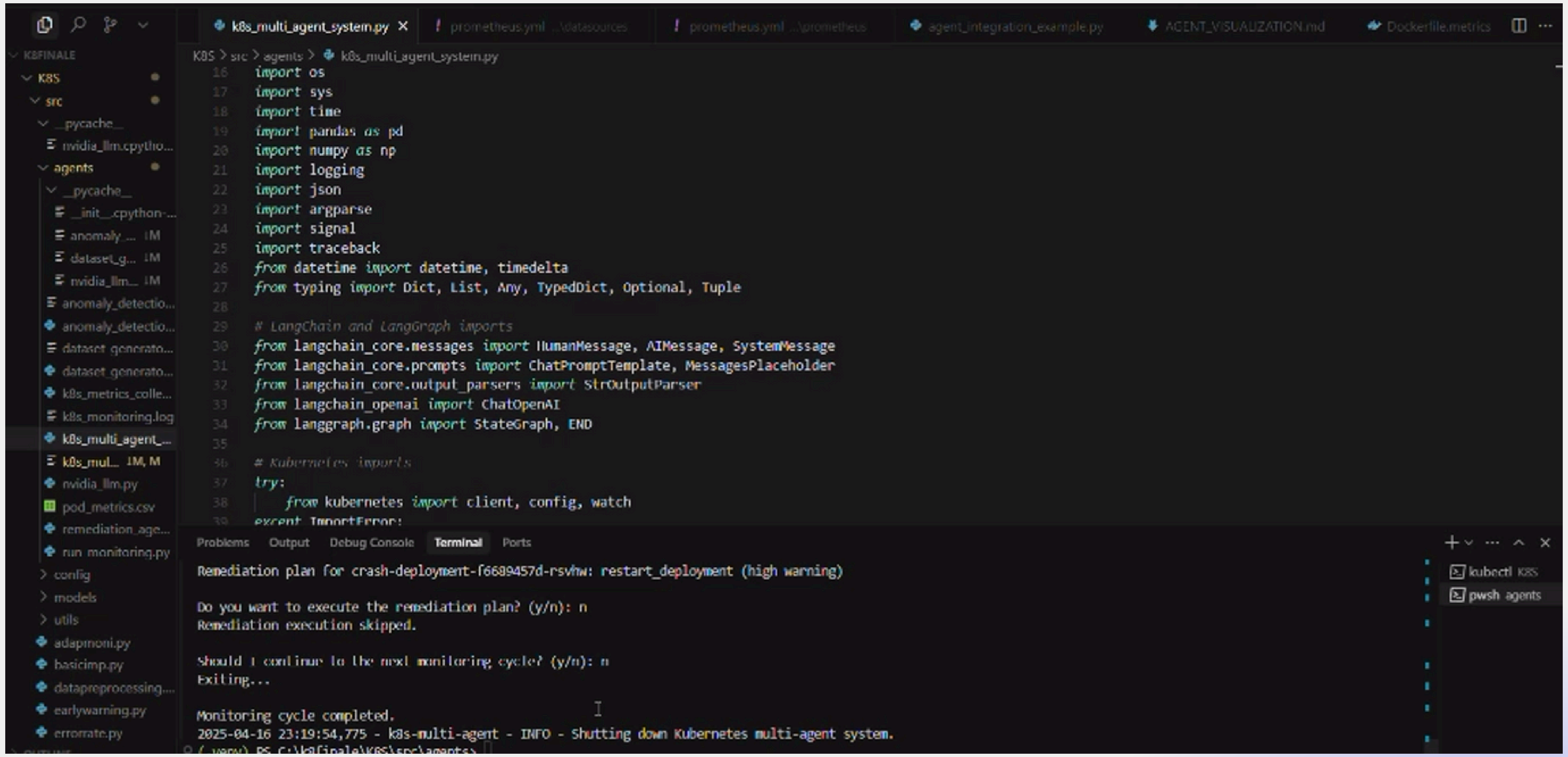
Key Benefits:

- Proactive failure detection in Kubernetes clusters.
- Automated remediation reduces manual DevOps effort.
- Scalable architecture using lightweight agents and Prometheus.
- Adaptable to various environments with minimal overhead.

Challenges & Mitigation:

- Limited hardware for real-time inference → Used lightweight ML models.
- Noisy or incomplete metrics → Injected synthetic anomalies to enrich training data.
- Frequent false alerts → Tuned model thresholds using historical cluster behavior.

Thank You



```python
import os
import sys
import time
import pandas as pd
import numpy as np
import logging
import json
import argparse
import signal
import traceback
from datetime import datetime, timedelta
from typing import Dict, List, Any, TypedDict, Optional, Tuple

# LangChain and LangGraph imports
from langchain_core.messages import HumanMessage, AIMessage, SystemMessage
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_core.output_parsers import StrOutputParser
from langchain_openai import ChatOpenAI
from langgraph.graph import StateGraph, END

# Kubernetes imports
try:
    from kubernetes import client, config, watch
except ImportError:
```

```
Remediation plan for crash-deployment-f6689457d-rsvhw: restart_deployment (high warning)

Do you want to execute the remediation plan? (y/n): n
Remediation execution skipped.

Should I continue to the next monitoring cycle? (y/n): n
Exiting...

Monitoring cycle completed.
2025-04-16 23:19:54,775 - k8s-multi-agent - INFO - Shutting down Kubernetes multi-agent system.
(.venv) PS C:\k8finale\K8S\src\agents>
```

Video Explaination