AI-Driven Prediction of Kubernetes Cluster Failures

Phase 1 - Prediction Model

Presented By Capital W

Introduction

Kubernetes clusters experience failures such as pod crashes, resource exhaustion, and network issues. These failures can lead to downtime, increased costs, and reduced performance. The challenge is to build an AI/ML model that can accurately predict these failures before they occur by analyzing historical and real-time cluster metrics.

Phase 1 - Overview

- **Goal:** Develop an AI/ML model to predict Kubernetes failures before they occur.
- **Approach:** Use anomaly detection and time-series analysis on cluster metrics.
- **Deliverables:** Trained model, codebase, documentation, and test data.

Data Collection

Data Sources:
https://drive.google.com/drive/folders/1FkeDE2eBsoe8D0U2h0dqoVpsgz0EfW9M?usp=sharing

Metrics Considered:

- CPU & Memory Utilization
- Pod/Node Status
- Network Latency & Throughput
- Disk I/O & Logs

Data Collection

Resource Utilization

- CPU Usage (%)
- Memory Usage (%, MB)
- Network I/O (Bytes, Packets)
- Filesystem Operations (Reads/Writes)

Data Collection

Pod Status

- Current Status (e.g., Running)
- Restarts
- Ready/Total Containers

Data Collection

Events

- Pod Events (Type, Reason, Age, Message)
- Node Events (Reason, Age, Message)

Data Collection

Resource Allocation

- Memory Requests/Limits
- Memory Utilization vs Limits

MODEL ARCHITECTURE

ML Techniques Used

- LSTM-Based Time-Series Forecasting for anomaly detection.
- Anomaly Classification to categorize failure types (CPU overload, memory exhaustion, etc.).
- Confidence Scoring Mechanism for accurate predictions and early warnings.

LIBRARIES & TOOLS

- **Deep Learning**: TensorFlow, PyTorch for LSTM model training.
- **Traditional ML:** Scikit-learn for feature engineering & preprocessing.
- Kubernetes real-time metrics collection.
- Prometheus & Grafana for visualization and monitoring.

KEY IMPLEMENTATIONS

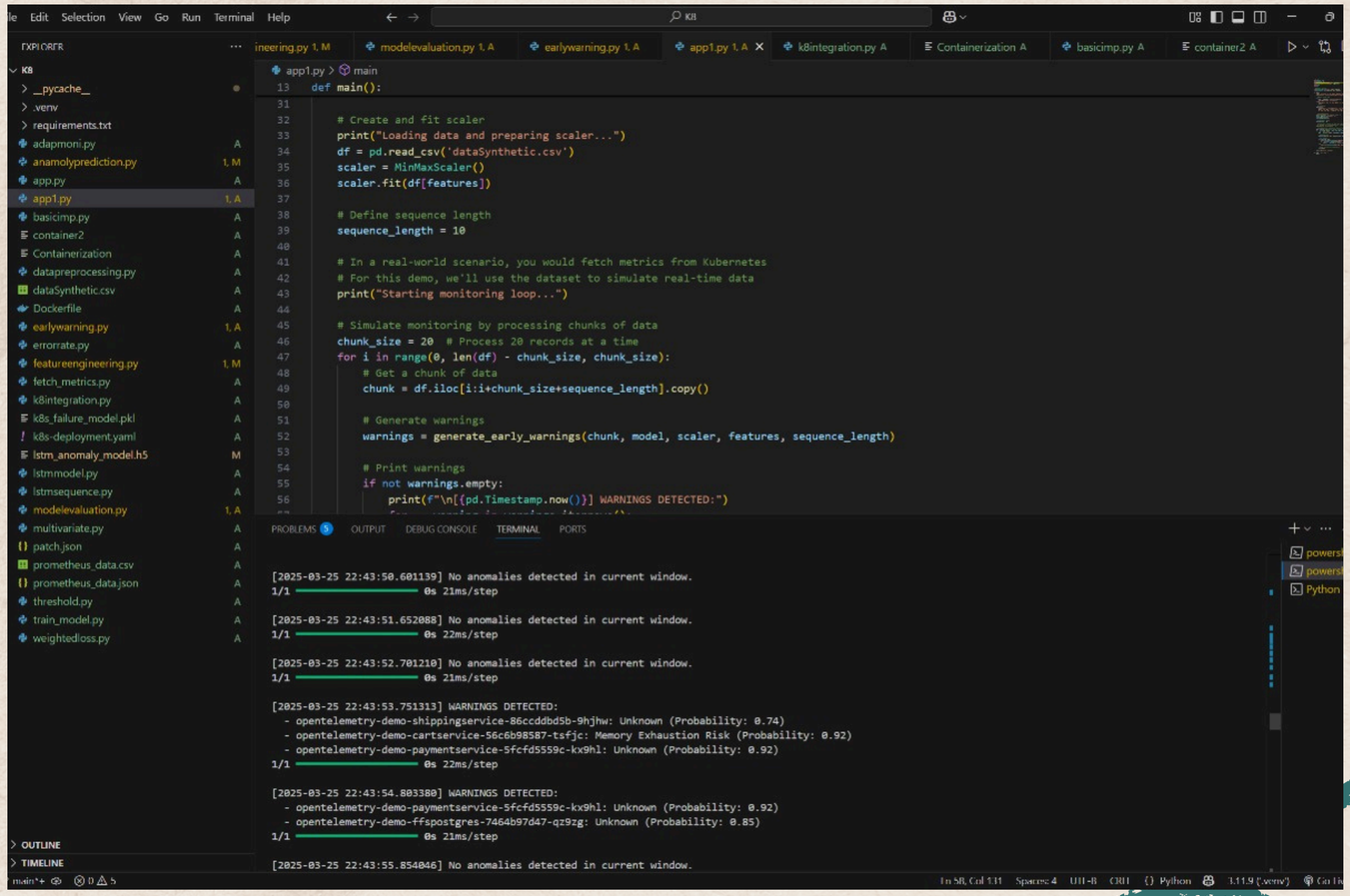
- Rolling Window Statistics & Rate-of-Change Features for better anomaly detection.
- Adaptive Thresholding Mechanism to refine anomaly alerts dynamically.
- Real-Time Model Deployment Strategy for integration with Kubernetes clusters.

Results #1

LSTM-Based Model For Prediction

Trained an LSTM-based model to predict anomalies using historical and real-time metrics.

DEMO:
https://drive.google.com/file/d/1_d42KJCyPbpIZhF-diaoI99gHWLneAl2/view?usp=drive_link



Results #2

Real-time Kubernetes Data Collection

```
PS C:\Kube\W\k8s-failure-dataset-generator> python3 dataset-generator1.py
Fetching data for pods in namespace monitoring
```

- we attempted to collect metrics from a Kubernetes cluster (ran on a local machine) using Minikube .

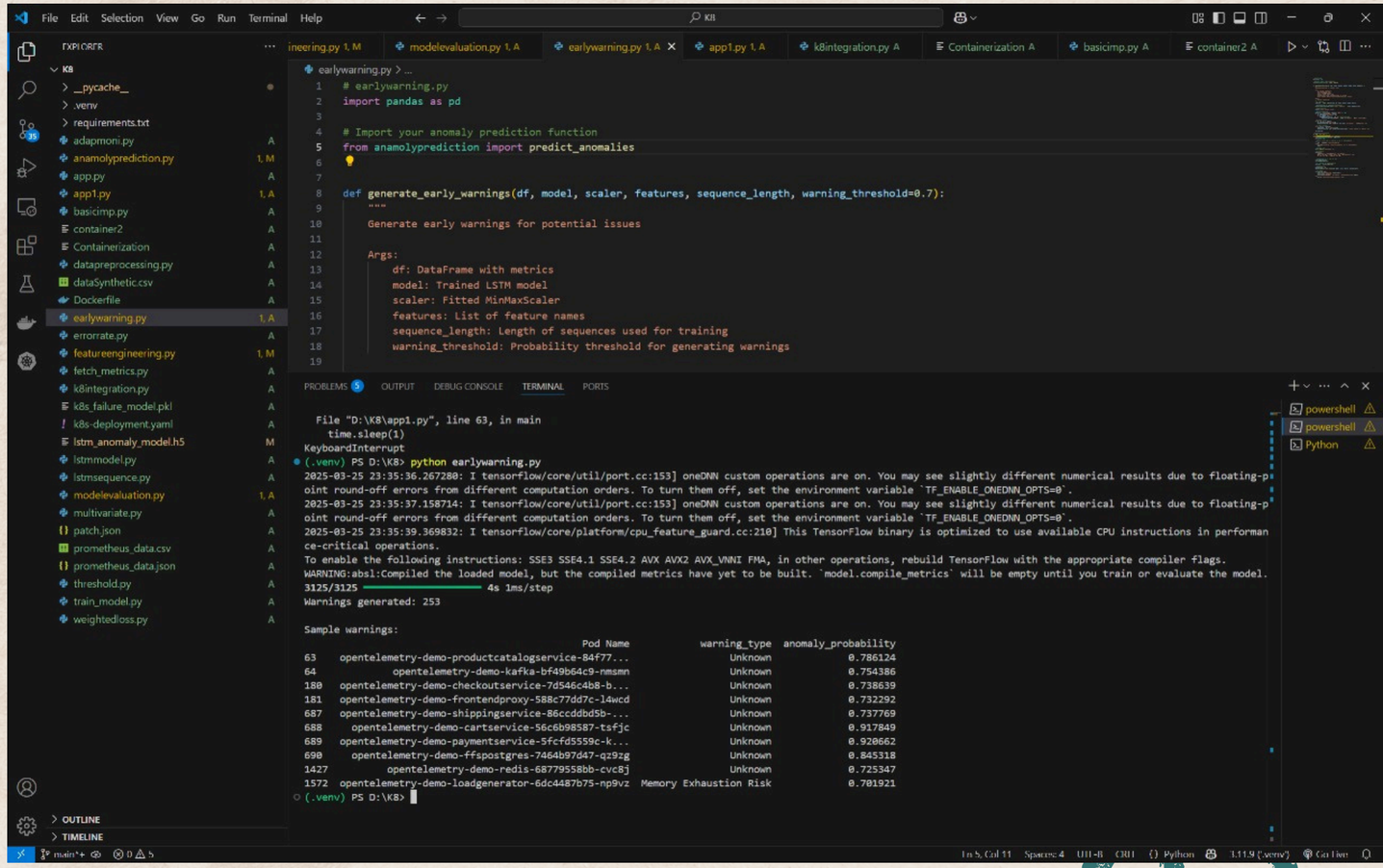
- Minikube is a tool that allows you to run a single-node Kubernetes cluster locally for development and testing purposes.

Results #3

Alert System For Anomaly Detection

Developed a proactive alert system that categorizes warnings based on thresholds and generates actionable insights.

DEMO:
https://drive.google.com/file/d/1_d42KJCyPbpIZhF-diaol99gHWLneAl2/view?usp=drive_link



Conclusion

- Successfully processed Kubernetes metrics and engineered key features.
- Trained an LSTM-based anomaly detection model for early failure prediction.
- Developed a proactive alert system for anomaly classification.
- Integrated real-time Kubernetes data collection for monitoring.

Q & A

Our Approach

1. LSTM-based anomaly detection for Kubernetes failures.
2. Confidence-based classification of anomalies (e.g., CPU overload, memory exhaustion).
3. Real-time integration with Kubernetes APIs for continuous monitoring.
4. Proactive alert system with adaptive thresholding.

Q & A

Are We Going in the Right Direction?

1. Does our approach align with best practices for AI-driven Kubernetes monitoring?
2. Are there any gaps or areas for refinement in our methodology?
3. How can we further optimize our prediction accuracy and response time?
4. Should our Kubernetes Anomaly Detection System be developed as a service-based or product-based solution?

Thank You