# Guidewire Hackathon 2025: Predictive Failure Detection in Kubernetes Clusters

## ****Problem Statement****

Kubernetes clusters can experience critical failures, including pod crashes, resource bottlenecks, and network disruptions. This hackathon challenge aims to develop an AI/ML-powered model capable of predicting these failures before they occur by analyzing historical and real-time cluster metrics.

## ****Data Collection & Preprocessing****

To ensure accurate failure prediction, we leverage multiple data sources and refine the collected data through preprocessing techniques.

### ****🔹 Key Metrics to Monitor****

| **Failure Type** | **Key Metrics to Collect** |
| --- | --- |
| **Node & Pod Failures** | CPU & Memory Usage, Node Conditions, Pod Status, Exit Codes, Restart Counts |
| **Resource Exhaustion** | CPU/Memory Requests vs. Limits, Disk I/O Latency, Network Throughput |
| **Network Issues** | Packet Loss, DNS Failures, Latency |
| **Service Disruptions** | API Server Latency, Load Balancer Errors, Authentication Logs |
| **Scheduler Failures** | Pending Pods, Resource Fragmentation |
| **Autoscaling Issues** | HPA Logs, CPU Utilization vs. Scaling |
| **Storage Failures** | Disk Space, Read/Write Latency, PVC Errors |
| **Security Anomalies** | Unauthorized Access (kubectl auth can-i), Abnormal Network Traffic |

### ****🔹 Data Sources****

* **Prometheus Exporters** → Collect CPU, memory, disk, and network usage metrics.
* **Kubernetes API Logs** → Monitor pod restarts, scheduler failures, and resource allocation.
* **Fluentd & ELK Stack** → Ingest system logs for deep failure analysis.
* **Packet Capturing** → Analyze network anomalies and detect potential failures.
* **Public Datasets** → Kaggle datasets such as [Ceph Drive Telemetry Data](https://www.kaggle.com/datasets/chauhankaranraj/ceph-drive-telemetry-data) containing time-series performance data.

### ****🔹 Data Cleaning & Preprocessing****

* **Handle missing values**: Fill gaps in CPU/memory usage data with rolling averages.
* **Normalize data**: Apply MinMax Scaling for CPU %, memory usage, and network traffic.
* **Remove outliers**: Use IQR (Interquartile Range) or Isolation Forests to eliminate anomalies.

### ****🔹 Feature Engineering****

| **Feature** | **Computation Method** |
| --- | --- |
| **CPU Spike %** | (current\_cpu\_usage - avg\_cpu\_usage) / avg\_cpu\_usage |
| **Memory Trend** | Rolling average of memory usage over 5-minute intervals |
| **Pod Restart Rate** | restart\_count / uptime |
| **Network Latency Variance** | Standard deviation of response times |

## ****Model Selection & Training****

The AI/ML models selected are tailored for different failure types to ensure optimal accuracy.

### ****🔹 AI/ML Models for Failure Prediction****

| **Failure Type** | **Best AI/ML Models** |
| --- | --- |
| **Node & Pod Failures** | Random Forest, XGBoost (classification) |
| **Resource Exhaustion** | LSTM, ARIMA (time-series forecasting) |
| **Network Issues** | Isolation Forest (anomaly detection) |
| **Service Disruptions** | Decision Trees, Gradient Boosting |
| **Security Threats** | NLP-based Log Analysis (BERT, LSTM) |

### ****🔹 Model Training Strategy****

* **Supervised Learning**: Train models on labeled historical data from logs & metrics.
* **Time-Series Forecasting**: Use LSTM/ARIMA models to predict upcoming resource exhaustion events.
* **Anomaly Detection**: Implement Isolation Forest to detect network-based anomalies.
* **Feature Selection & Optimization**: Utilize Recursive Feature Elimination (RFE) and GridSearchCV to enhance model performance. **Deliverables for Phase 1**

**Trained ML Model**: A predictive AI/ML model capable of forecasting Kubernetes cluster issues.  
 **Codebase**: Functional code including data collection, model training, and evaluation scripts uploaded to GitHub.  
 **Documentation**: Comprehensive explanation of the approach, key metrics used, and model performance.  
 **Presentation**: A brief recorded presentation of the model, including results and potential improvements, along with the presentation file.  
**Test Data**: The dataset used for training and validation (if applicable).

## ****Expected Impact****

🔹 **Proactive Failure Mitigation**: Enables cluster administrators to address issues before they escalate.  
🔹 **Optimized Resource Allocation**: Prevents unnecessary downtime and ensures efficient Kubernetes operations.  
🔹 **Scalability & Adaptability**: The model can be expanded to other cloud-native architectures.  
🔹 **Security Enhancement**: Early detection of unauthorized access or security threats in Kubernetes environments.

## ****Next Steps****

🔹 Fine-tune models with real-time Kubernetes data.  
🔹 Integrate the model with a visualization tool (Grafana or Kibana).  
🔹 Optimize deployment strategies using Kubernetes-native tools.