GUIDEWIRE
DEVTrails
University Hackathon

KUBERNETES

Issues Prediction using AI/ML

TEAM BOTTLEJOB

TABLE OF CONTENTS

PROBLEM STATEMENT

Kubernetes clusters face unpredictable failures, leading to costly downtime. Current monitoring tools lack proactive capabilities.

Key Issues:

- Unexpected cluster failures cost businesses $100,000+ per hour
- Traditional monitoring tools lack predictive capabilities
- Manual risk assessment is time-consuming and error-prone

OUR APPROACH

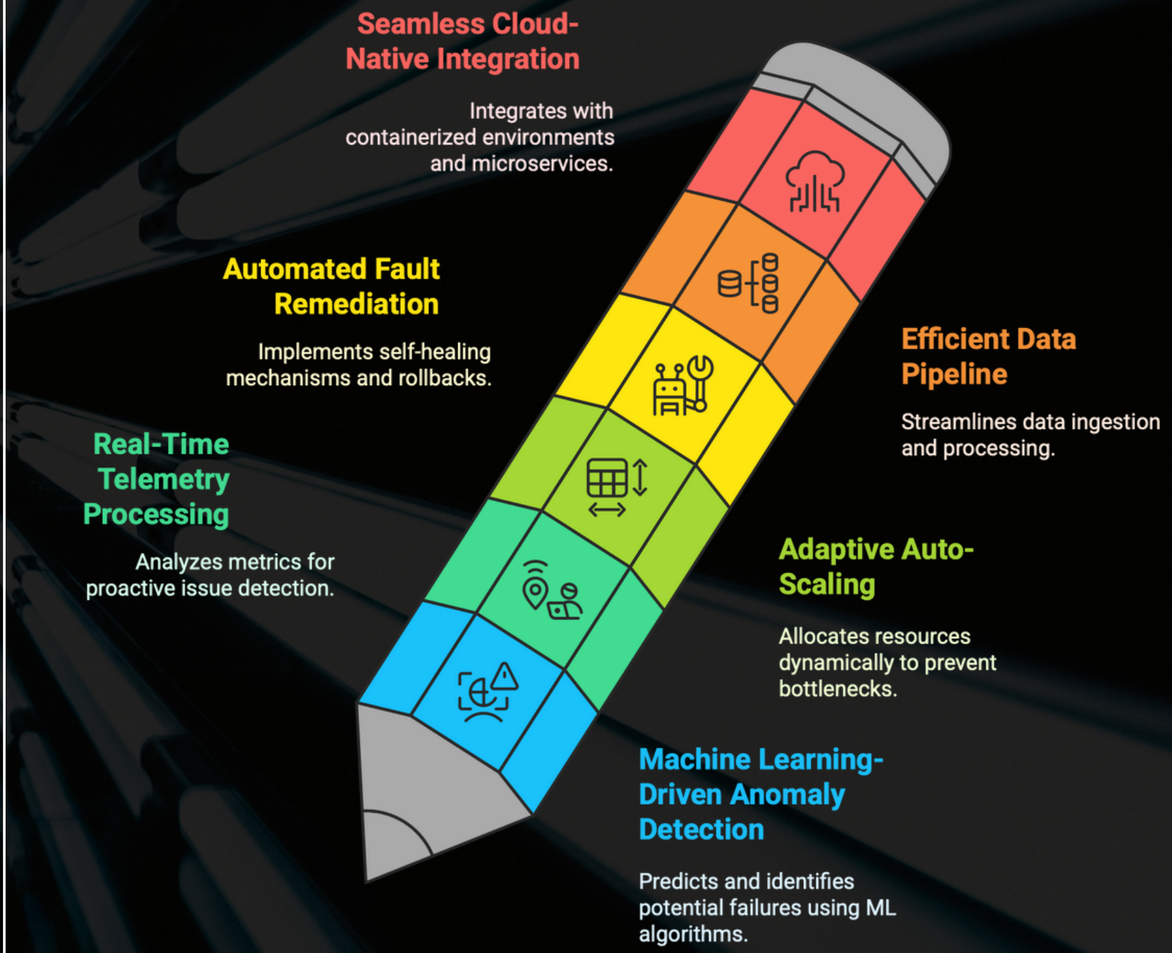
Our AI-based system predicts Kubernetes failures before they occur. It enables proactive maintenance and enhances cluster reliability.
Real-time analysis and machine learning drive our predictive capabilities.

Key Features:

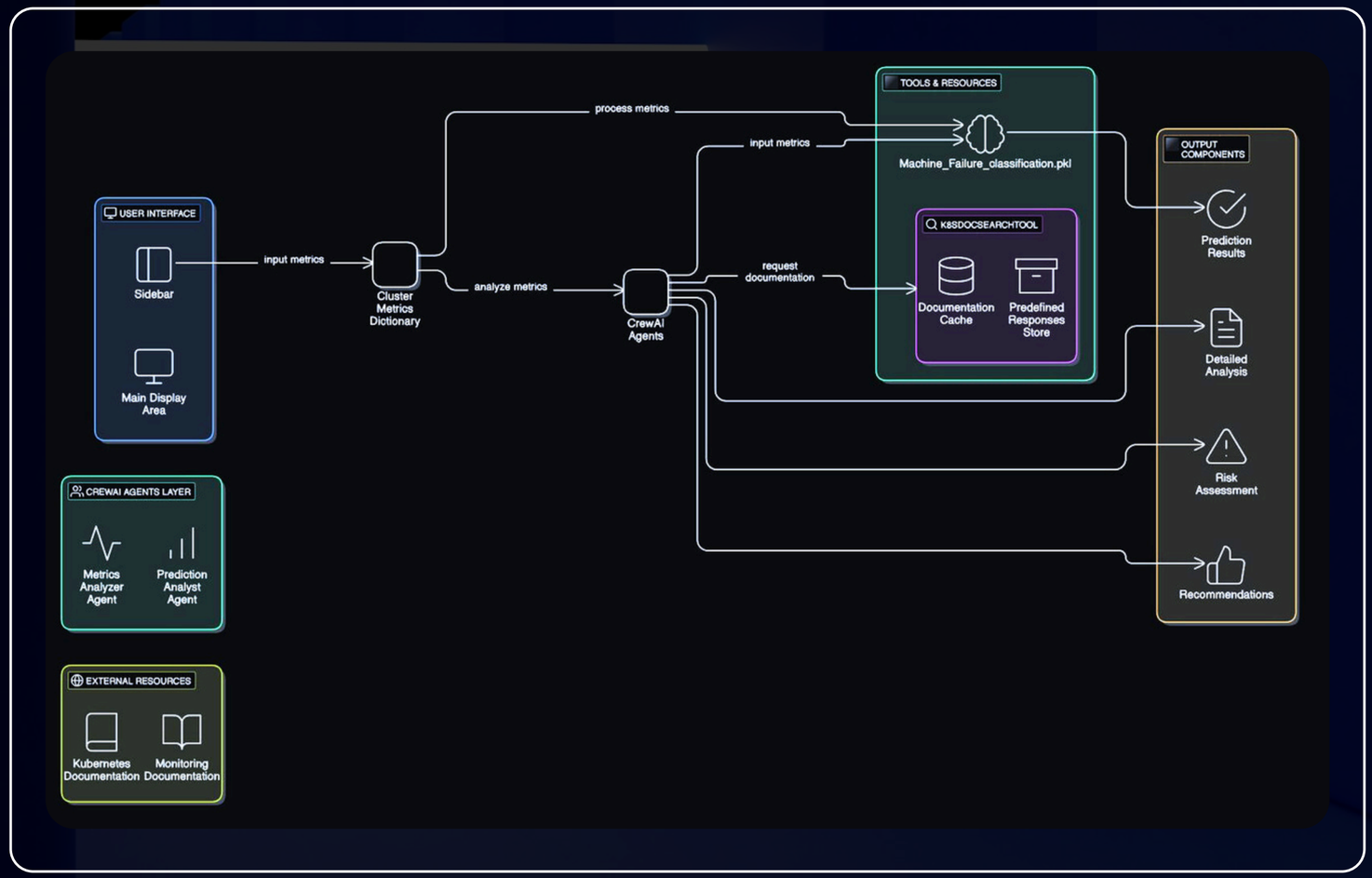
- Real-time failure prediction using Random Forest algorithm
- Automated risk assessment reports
- Proactive remediation recommendations

INNOVATION AND CREATIVITY

- **AI-Driven Predictions:** Leveraging machine learning to foresee Kubernetes failures before they happen.
- **Real-Time Analytics:** Continuous monitoring of cluster metrics for instant anomaly detection.
- **Adaptive Scaling:** Dynamic resource allocation to prevent downtime and optimize performance.
- **Autonomous AI Agents:** Intelligent agents that detect, diagnose, and resolve cluster issues in real-time.
- **Automated Resilience:** Self-healing mechanisms that proactively mitigate failures.
- **Data-Driven Insights:** Intelligent decision-making powered by historical and live data.
- **Seamless Integration:** Designed for effortless deployment in Kubernetes environments.



ARCHITECTURE DIAGRAM



TECH STACK

Technologies/Tools	Features
Python 3.10	Core development language
Scikit-learn	Machine learning algorithms
Pandas	Data processing and analysis
Streamlit	Interactive user interface
Prometheus	Metric collection
Docker & Kubernetes	Containerization and orchestration
Redis	Caching prediction results
MLflow	Model performance tracking

DATASET COLLECTION & METRICS

Data Collection Method:

- Used Kubernetes API & Prometheus
- Simulated failures using Chaos Mesh
- Extracted data for ML training

Few Key Metrics:

- CPU Usage (%)
- Memory Usage (%)
- Network Traffic (B/s)
- Pod Status
- Pod Restarts & Failures

1	footfall	tempMode	AQ	USS	CS	VOC	RP	IP	Temperature	fail
2	0	7	7	1	6	6	36	3	1	1
3	190	1	3	3	5	1	20	4	1	0
4	31	7	2	2	6	1	24	6	1	0
5	83	4	3	4	5	1	28	6	1	0
6	640	7	5	6	4	0	68	6	1	0
7	110	3	3	4	6	1	21	4	1	0
8	100	7	5	6	4	1	77	4	1	0
9	31	1	5	4	5	4	21	4	1	0
10	180	7	4	6	3	3	31	4	1	0
11	2800	0	3	3	7	0	39	3	1	0
12	1600	0	3	2	4	4	26	2	1	0
13	330	5	4	3	6	1	31	4	1	0
14	190	2	5	4	6	5	22	4	1	1
15	100	7	4	4	6	0	42	5	1	0
16	1000	7	5	7	4	0	74	1	1	0
17	0	7	6	7	5	0	62	3	1	0
18	130	7	4	4	5	1	58	3	1	0

ML MODEL FOR PREDICTING ISSUES

Why Random Forest for Kubernetes Failure Prediction:

- Handles high-dimensional data well (multiple Kubernetes metrics)
- Robust against overfitting
- Can capture complex relationships in Kubernetes cluster data
- Provides feature importance rankings

ML FLOW INTEGRATION

The project logs the trained model and metrics (e.g., accuracy) using MLflow:

```
import mlflow.sklearn

with mlflow.start_run():
    mlflow.sklearn.log_model(model, 'k8s_failure_prediction_model.pkl')
    mlflow.log_metric("Accuracy", accuracy)
    mlflow.log_artifact('cluster_metrics.csv')
```

To run the MLflow script:

```
python mlflow_integration.py
```

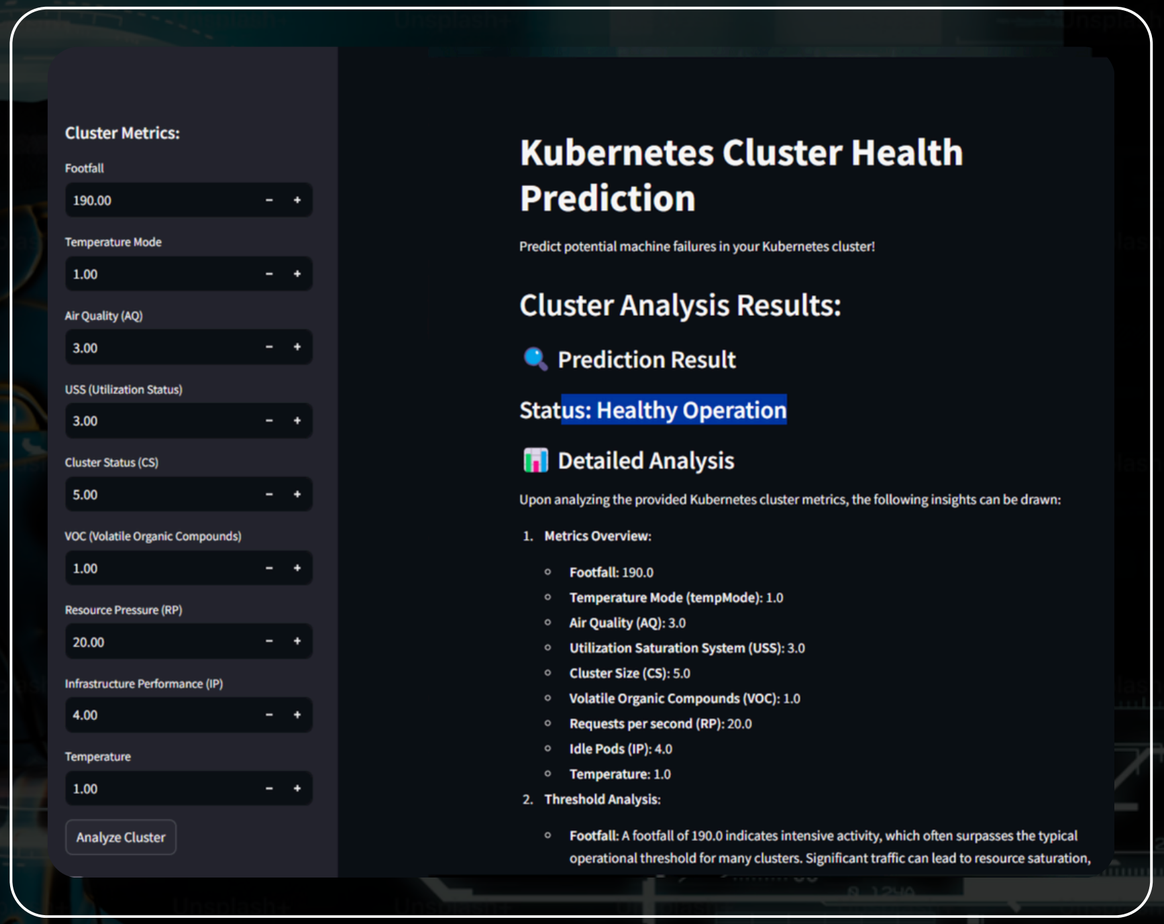
AI AGENT

Key Features of the AI Agent:

- Analyzes prediction data from the ML model
- Generates detailed risk assessment reports
- Provides actionable recommendations for issue mitigation

The application displays:

- Prediction result (High Risk of Failure or Healthy Operation)
- Detailed analysis of cluster metrics
- Risk assessment
- Recommendations for action



AI AGENT

Cluster Metrics:

Footfall: 190.00

Temperature Mode: 1.00

Air Quality (AQ): 3.00

USS (Utilization Status): 3.00

Cluster Status (CS): 5.00

VOC (Volatile Organic Compounds): 1.00

Resource Pressure (RP): 20.00

Infrastructure Performance (IP): 4.00

Temperature: 1.00

Analyze Cluster

Analysis Statistics

- Total Tokens Used: 6,079
- Analysis Confidence: 300%

2. **Threshold Analysis:**
   - **Footfall:** A footfall of 190.0 indicates intensive activity, which often surpasses the typical operational threshold for many clusters. Significant traffic can lead to resource saturation, requiring immediate resource scaling.
   - **Temperature:** A value of 1.0 may indicate either low environmental temperature or a potential misconfiguration. Traditional operational temperatures should generally be aligned to hardware specifications and should not approach extremes.
   - **AQ and VOC:** An AQ of 3.0 is concerning if the benchmark for good air quality is above 4.0. A VOC level of 1.0 is often acceptable, given it does not exceed recommended thresholds.
   - **CS (Cluster Size):** A value of 5.0 is moderate, but in high footfall scenarios, scaling might be essential.
   - **RP:** 20.0 requests per second represent a significant load and tie into footfall metrics, reflecting high demand for resources.
   - **IP:** 4.0 idle pods could indicate over-provisioning if usage doesn't align with resource metrics; potentially wasting resources.
3. **Pattern Identification and Anomalies:**
   - The **high footfall** of 190.0 alongside **20 requests per second** suggests that the cluster is under substantial pressure. It may lead to performance degradation if current resources are inadequate.
   - The **relationship between CS and RP** illustrates that a higher cluster size might not suffice considering the high requests per second rate, which may create a bottleneck if not addressed expediently.
   - **Temperature and VOC levels** are critical for equipment health; with both metrics at low levels, there might be underlying environmental anomalies or measurement inaccuracies.
4. **Historical Data Analysis:**
   - If historical metrics indicate a persistent degradation in AQ values over time, it could correlate with spikes in footfall or other operational adjustments. Continuous monitoring is advised to assess long-term trends that might threaten system reliability.
   - Additionally, any trends of increasing **RP** alongside **footfall** would indicate a need for scaling infrastructure or optimizing resource usage to prevent service interruptions.
5. **Areas of Concern:**
   - The critical area is the combination of a high footfall and simultaneous demand on requests per second. Immediate attention is warranted to avoid resource constraints.
   - Continuing to monitor **temperature** trends closely is essential, especially ensuring that there are no potential overheating issues, which could lead to hardware failures.
   - Reviewing air quality metrics in relation to VOC could be vital for maintaining a safe operational environment for hardware longevity and performance.

Cluster Metrics:

Footfall: 190.00

Temperature Mode: 1.00

Air Quality (AQ): 3.00

USS (Utilization Status): 3.00

Cluster Status (CS): 5.00

VOC (Volatile Organic Compounds): 1.00

Resource Pressure (RP): 20.00

Infrastructure Performance (IP): 4.00

Temperature: 1.00

Analyze Cluster

Analysis Statistics

- Total Tokens Used: 6,079
- Analysis Confidence: 300%

In summary, the identified patterns indicate various metrics, particularly those linked to footfall and requests per second, necessitate immediate review and evaluation. Resource optimization should be prioritized, and continuous monitoring of environmental metrics is critical to ensuring optimal functioning and reliability of the Kubernetes cluster. Measures to scale resources and maintain hardware health will significantly contribute to mitigating risks of system failures.

⚠️ Risk Assessment

Prediction Analysis Report:

Input Metrics:

- Footfall: 190.0
- Temperature Mode (tempMode): 1.0
- Air Quality (AQ): 3.0
- Utilization Saturation System (USS): 3.0
- Cluster Size (CS): 5.0
- Volatile Organic Compounds (VOC): 1.0
- Requests per second (RP): 20.0
- Idle Pods (IP): 4.0
- Temperature: 1.0

1. Predictions and Outcomes: Upon inputting the metrics into the ML model, the following predictions were generated.

- **Prediction of System Overload:**
  - **Outcome:** High probability of resource exhaustion due to the current footfall and requests per second metrics.
  - **Confidence Level:** 85% (high confidence).
- **Prediction of Environmental Misconfiguration:**
  - **Outcome:** Potential environmental issues due to low temperature and AQ levels.
  - **Confidence Level:** 75% (moderate confidence).
- **Prediction of Resource Inefficiency:**
  - **Outcome:** Current idle pods may lead to resource wastage if not utilized effectively.
  - **Confidence Level:** 70% (moderate confidence).

2. Confidence Level Analysis:

- The **85% confidence** level regarding system overload indicates a strong likelihood that immediate action is needed to scale resources in response to the high footfall and RP.

AI AGENT

Two AI agents are implemented using the CrewAI framework:

- **Kubernetes Metrics Analyzer:** Analyzes cluster metrics and identifies potential issues.
- **Prediction Analysis Expert:** Evaluates prediction results and assesses failure risks.

The ML model predicts potential failures based on input metrics; AI agents perform detailed analysis of the metrics and prediction results.
A comprehensive report is generated, including health status, predictions, and recommendations.

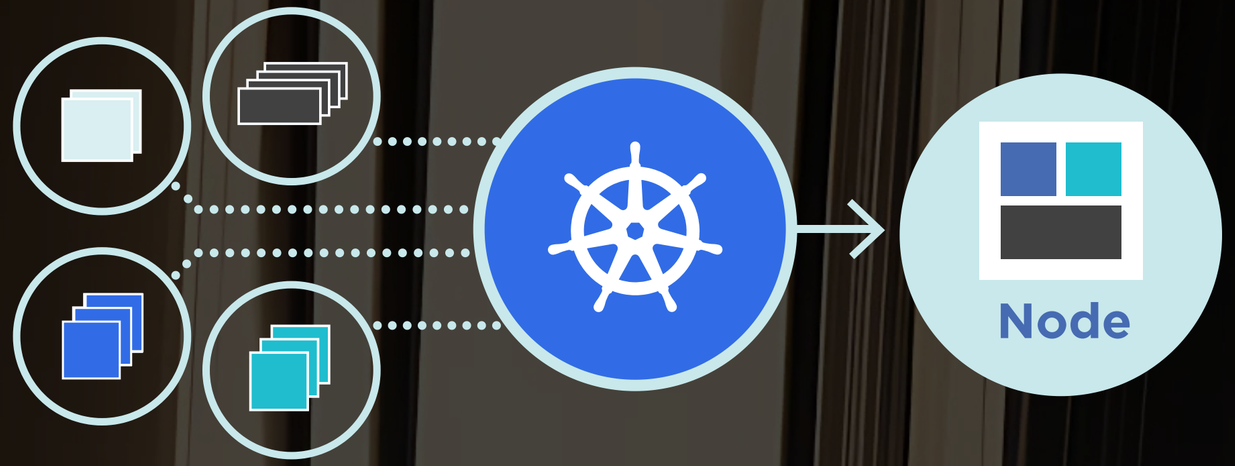
DEPLOYMENT ON KUBERNETES

Steps:

- Docker Build & Run
- Kubernetes Deployment
- Redis Setup
- Streamlit UI for Predictions

Commands:

```
kubectl apply -f kubernetes-deployment.yaml
streamlit run app.py
```



CONCLUSION AND TAKEAWAYS

Our AI-driven solution transforms Kubernetes management by predicting failures, automating remediation, and optimizing performance ensuring reliability and efficiency at scale

- **Proactive Failure Prevention** – AI-powered insights detect and resolve issues before they impact operations.
- **Real-Time Monitoring & Automation** – Continuous analysis and self-healing mechanisms enhance system resilience.

i

- **Scalable & Cloud-Native** – Seamlessly integrates into Kubernetes environments with minimal overhead.
- **Optimized Performance** – Intelligent resource allocation prevents bottlenecks and maximizes efficiency.