# AIOPs-Driven Automation: Enhancing Kubernetes Management with AI and Ansible

Electrical and Computer Engineering
Ricardo Godinez; Dr. Valerio Formicola

## Introduction

This project integrates **DevOps principles** with Kubernetes (K8s) to enhance system reliability and efficiency. By monitoring critical K8s metrics—such as CPU usage, memory consumption, pod status, and node health—using **Prometheus**, automated **Ansible playbooks** are triggered to maintain system stability.

Motivated by the need for scalable and self-healing infrastructure, the system will evolve to include an **AI-driven approach** that interprets real-time metrics and initiates corrective actions. This fusion of monitoring, automation, and AI supports continuous uptime and operational efficiency in containerized environments.

## Abstract

This project explores the integration of Kubernetes (K8s) environments with automated configuration management using Ansible playbooks to enhance operational efficiency, scalability, and system resilience. The core objective is to monitor critical K8s metrics—such as CPU and memory utilization, pod status, node health, and service availability—using Prometheus for data collection and Grafana for visualization. Based on predefined metric thresholds, Ansible playbooks are automatically triggered to optimize the environment by dynamically scaling workloads, addressing resource bottlenecks, and ensuring overall system stability.

Additionally, the collected metrics will contribute to building a robust dataset for training an AI model capable of automating Ansible playbook execution based on real-time system conditions. This AI-driven approach aims to streamline system management, reduce human error, and enhance the responsiveness and adaptability of Kubernetes management tasks. The project highlights the potential of combining monitoring, automation, and machine learning to advance the efficiency and reliability of containerized environments.

## Materials and Methods

To simulate and design the proposed system, the following components and methods were utilized:

### Virtualized Environment

- **VMware Workstation**: Used as the host platform for creating and managing a virtual Kubernetes cluster.
- **CentOS 7**: Deployed consistently across all virtual machines (VMs) to simulate a stable server environment.
- **Fedora Linux**: Host operating system on developer's laptop for orchestration, configuration, and system development.

### Core Technologies

- **Kubernetes (kubeadm)**: Configured to manage, deploy, and orchestrate containerized applications.
- **Prometheus and Grafana**: Used to collect, store, and visualize Kubernetes metrics (CPU usage, memory usage, pod status, node availability, and container health).
- **Ansible**: Implemented for automating infrastructure remediation tasks triggered by specific metric thresholds.

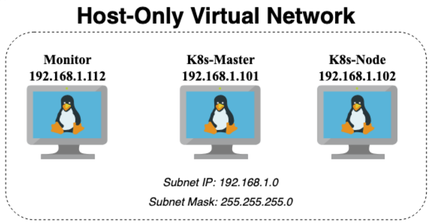
### AI Integration Workflow (ETL and Automation Pipeline)

- **Extract**: Prometheus collects real-time Kubernetes performance data through its metrics scraping process.
- **Transform**: Python-based scripts are planned to preprocess metrics, aggregating and formatting them into structured datasets suitable for machine learning analysis.
- **Load**: Datasets are intended for use in training AI models designed to detect anomalies and predict infrastructure issues.
- **Trigger**: Once implemented, trained models will analyze live metrics, triggering Ansible playbooks to proactively resolve system issues in real time.

*Due to time constraints, the complete ETL and AI pipeline implementation remains pending and will be addressed in future stages of the project.*

## System Architecture/Design Overview

This project features a simulated Kubernetes (K8s) environment built on a virtual infrastructure, consisting of two primary components—network architecture and automation workflow—designed to enable proactive infrastructure management through integrated observability and automation tools.:




*This project features a simulated Kubernetes (K8s) environment built on a virtual infrastructure, integrated with automation and observability tools to enable proactive infrastructure management.*

### Data Flow and Automation Pipeline

The designed workflow integrates monitoring, visualization, automation, and planned AI-driven analysis:

- **Prometheus**: Collects real-time performance metrics from Kubernetes master and worker nodes.
- **Grafana**: Provides visual dashboards for real-time monitoring of collected metrics.
- **Python-based ETL Process**: Planned to extract, transform, and load Prometheus data into structured datasets for AI analysis.
- **AI Algorithm**: Planned to analyze metrics, detect anomalies, and predict infrastructure needs or failures.
- **Ansible**: Executes corrective actions automatically based on AI model outputs or predefined thresholds.
- **System Administrator**: Receives alerts and visualizations, overseeing the automated environment.

### Virtual Network Architecture

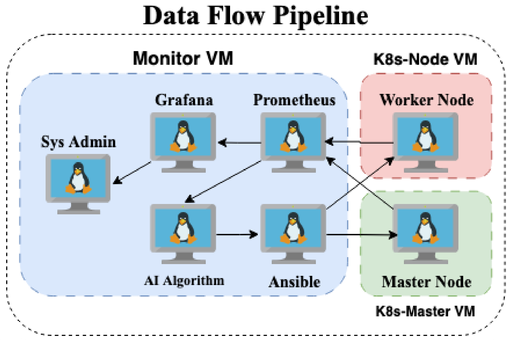
The project utilizes a host-only virtual network, configured with VMware Workstation and CentOS-based virtual machines (VMs):

#### Host-Only Virtual Network:

- Provides isolated networking among virtual machines.
- Ensures secure internal communication, validated by successful ping tests between VMs.

#### Kubernetes Cluster Virtual Machines:

- Consists of designated master and worker nodes.
- Facilitates realistic simulation of a production Kuberentes environment for testing and development.




*This diagram illustrates the end-to-end data flow from Kubernetes metric collection using Prometheus to AI-driven analysis and automated remediation via Ansible playbooks.*

Together, the virtual network and the data automation pipeline create a robust foundation to demonstrate the potential of integrating AI-driven decision-making within Kubernetes environments for enhanced operational reliability and efficiency.

## Results

Despite not completing the full system integration, a key milestone was reached in demonstrating automated service recovery using Ansible. The following results were achieved and are supported by the accompanying screenshots:

```
master@k8s-master:~$ sudo systemctl stop node_exporter.service
master@k8s-master:~$ sudo systemctl status node_exporter.service
○ node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service;
     Active: inactive (dead) since Mon 2025-04-21 22:50:54 PDT;
   Duration: 1min 3.171s
 Invocation: 820d8c0130984cb699b094d74ec68a4a
    Process: 6991 ExecStart=/opt/node_exporter/node_exporter (co
   Main PID: 6991 (code=killed, signal=TERM)
   Mem peak: 12.2M
        CPU: 518ms
```

### Simulated Failure:

- The node_exporter service was manually stopped on the Kubernetes master node.
- Running systemctl status confirmed the service had transitioned to an inactive (dead) state.
- This interruption was logged and would stop metric flow to Prometheus and Grafana.

```
monitor@monitor:~/ansible$ ansible-playbook -i hosts recover-node-exporter.yaml -kK
```

```
master@k8s-master:~$ sudo cat /var/log/recovery.log
Checked node_exporter at 2025-04-22T05:51:34Z - Status: inactive
Restarted node_exporter at 2025-04-22T05:51:34Z
master@k8s-master:~$ sudo systemctl status node_exporter.service
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service;
     Active: active (running) since Mon 2025-04-21 22:51:38 PDT;
 Invocation: f52bdc15db1f4982847e3e03f67f3fb4
   Main PID: 9014 (node_exporter)
      Tasks: 5 (limit: 22916)
     Memory: 7.9M (peak: 8M)
        CPU: 109ms
     CGroup: /system.slice/node_exporter.service
             └─9014 /opt/node_exporter/node_exporter
```

### Automated Recovery:

•The playbook detected the inactive status and restarted the service automatically.
•Log files in /var/log/recovery.log confirmed:

- The service was checked.
- The inactive status was identified.
- The restart action was performed immediately.

### Service Restoration:

- A follow-up systemctl status command showed the service as **active (running)**.
- The system successfully resumed normal operation, restoring metrics to Grafana.

This demonstration validates a core functionality of the project: **automated detection and remediation of critical services**, reinforcing the reliability and autonomy of the infrastructure management pipeline.

## Summary and Conclusions

This senior design project explored the integration of Kubernetes (K8s) with automated infrastructure management tools (Prometheus, Grafana, and Ansible), enhanced by planned AI-driven automation. The system architecture was successfully designed and partially implemented, demonstrating the feasibility of combining monitoring and automation tools within a DevOps framework.

### Key achievements include:

- Successful establishment of a virtualized environment using VMware Workstation.
- Installation of essential software components (Kubernetes, Prometheus, Grafana, Ansible) on CentOS-based virtual machines.
- Functional virtual network allowing inter-VM communication.

Due to time limitations, the complete Kubernetes networking setup and subsequent full-system validation were not finalized. Nonetheless, the project's initial phases illustrate significant potential in streamlining Kubernetes operations, improving resource management, and increasing system resilience through automated, intelligent interventions.

Future work will focus on completing the Kubernetes cluster configuration, fully implementing the ETL pipeline, and integrating machine learning algorithms to enable proactive, data-driven system management.

## Future Work

To build upon the current progress and fully realize the project's goals, future phases will include:

### Complete Kubernetes Networking Setup:

- Finalize network configuration between master and worker nodes.
- Validate cluster communication and functionality.

### ETL Pipeline Implementation:

- Develop and deploy Python scripts to extract metrics from Prometheus.
- Establish a robust data transformation and loading process for AI integration.

### AI Model Development and Integration:

- Train and test machine learning models to predict system anomalies and resource needs.
- Implement real-time decision-making capabilities to trigger Ansible playbooks proactively.

### Full-System Validation:

- Perform comprehensive testing of the integrated system under realistic conditions.
- Monitor effectiveness of AI-driven automation in maintaining Kubernetes cluster health.

### Expansion of Monitoring Capabilities:

- Incorporate additional critical metrics such as disk I/O, network latency, evicted pods, and custom application health metrics.
- Enhance predictive accuracy and remediation effectiveness.

Completion of these future tasks will advance the system from a foundational prototype to a robust, intelligent, and automated Kubernetes management solution.

## References

1. **"Kubernetes Documentation."** Kubernetes.io. *Online*. Available: https://kubernetes.io/docs/ (accessed Apr. 16, 2025).
2. **"Prometheus Documentation."** Prometheus.io. *Online*. Available: https://prometheus.io/docs/ (accessed Apr. 16, 2025).
3. **"Ansible Documentation."** Ansible. *Online*. Available: https://docs.ansible.com/ (accessed Apr. 16, 2025).