# Local DevOps Sandbox for Monitoring & Alerting (All-in-One VM) – Project Report

**Abstract**

The field of DevOps emphasizes automation, reliability, and continuous monitoring of systems. In this project, a local sandbox was created inside an Ubuntu virtual machine to simulate a real-world monitoring environment. Using Prometheus, Grafana, Node Exporter, and Alertmanager, the system provides metric collection, visualization, and alerting. The aim is to help learners and practitioners gain hands-on experience with monitoring stacks, without requiring cloud infrastructure or complex setups. This sandbox demonstrates how to monitor system health, configure dashboards, and set up alerts that notify administrators when thresholds are breached.

---

## Introduction

Monitoring is a critical part of any production-grade DevOps pipeline. It ensures systems remain healthy, performance is tracked, and issues are detected before they affect end users. Beginners often struggle to practice monitoring in real-world environments due to cost or complexity.

This project solves that by building a **self-contained sandbox environment** where all the monitoring tools run on a single virtual machine. By using Prometheus for data collection, Node Exporter for system metrics, Grafana for visualization, and Alertmanager for alerting, the sandbox mirrors how monitoring stacks work in enterprises. It provides a safe space to learn and experiment, making it an excellent foundation for students and new DevOps engineers.

---

## Tools Used

- **Ubuntu (VirtualBox VM):** Base operating system for the sandbox.
- **Prometheus:** Collects and stores time-series metrics from exporters.
- **Node Exporter:** Provides system-level metrics like CPU, memory, and disk usage.
- **Grafana:** Visualization platform to create interactive dashboards and alerts.
- **Alertmanager:** Integrates with Prometheus to manage and route alerts to notification channels (e.g., email).

---

## Steps Involved in Building the Project

1. **VM Setup:** A fresh Ubuntu VM was created in Oracle VirtualBox.
2. **Node Exporter Installation:** Downloaded the tarball, extracted the binary, moved it to /usr/local/bin/, created a systemd service, and started it.
3. **Prometheus Installation:** Installed via tarball, moved binaries (prometheus, promtool), created prometheus.yml configuration, and set up a dedicated systemd service to manage it.

4. **Grafana Installation:** Installed using the official APT repository. Enabled and started the Grafana server, then accessed the web UI on port 3000.
5. **Alertmanager Installation:** Installed from tarball, configured for Prometheus alerts, and managed through systemd service.
6. **Integration:**
   - Configured Prometheus to scrape Node Exporter metrics.
   - Added Prometheus as a data source in Grafana.
   - Created dashboards to visualize CPU, memory, and disk usage.
   - Imported JSON-based dashboards for Node Exporter.
7. **Alerts:**
   - Created a CPU usage alert rule in Grafana.
   - Integrated with email notifications.
   - Verified alerts through Grafana's alerting interface.
8. **Service Management:** Used systemctl commands to start, stop, enable, disable, and check status of services (Prometheus, Grafana, Node Exporter, Alertmanager).
9. **Deliverables:** Uploaded configuration files (prometheus.yml), dashboards (.json), alert rules, and screenshots to GitHub for reproducibility.

---

**Conclusion**

This project successfully demonstrates the setup of a **complete DevOps monitoring sandbox** within a single VM. By combining Prometheus, Grafana, Node Exporter, and Alertmanager, it provides a practical environment to learn monitoring concepts such as metrics collection, visualization, and alerting.

Key learnings include:

- Setting up services using systemd for reliability.
- Configuring Prometheus to scrape exporters.
- Building and importing Grafana dashboards.
- Writing and testing alert rules with email notifications.

The sandbox is extendable, meaning more exporters (e.g., MySQL, Blackbox, Docker) can be added, and it can even be integrated with Kubernetes or cloud services in the future. Overall, the project achieves its objective of providing a **hands-on, cost-free, and scalable learning platform** for monitoring in DevOps.