# GRAFANA REPORTING AND ALERTING

**Micro Project Report Submitted by**

VEDHAVITHYA S

(22ITR115)

**Course Code & Name :** 22ITF02 - DEVOPS

**Programme :** B. TECH

**Branch :** INFORMATION TECHNOLOGY

# Kongu Engineering College, Perundurai May 2025

**BONAFIDE CERTIFICATE**

Certified that this micro project documentation of “**GRAFANA REPORTING AND ALERTING”** is the Bonafide work of **“VEDHAVITHYA S (22ITR115)”** who carried out the project under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of my other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Submitted on .

## SUPERVISOR HEAD OF THE DEPARTMENT

Mr. D. VIJAY ANAND

MCA ., ME

Dr. S. ANANDAMURUGAN

M.E., Ph.D.

## SUPERVISOR HEAD OF THE DEPARTMENT

Assistant Professor (SLG) Professor & HOD,

Department of IT, Department of IT,

Kongu Engineering College, Kongu Engineering College,

Perundurai Perundurai

# INTEGRATING PROMETHEUS WITH GRAFANA FOR MONITORING

**AIM**

The aim of this project is to build a real-time and efficient monitoring system using Prometheus and Grafana. It focuses on collecting system performance metrics such as CPU and memory usage from an Ubuntu virtual machine via Node Exporter. The collected metrics are visualized through Grafana dashboards to offer actionable insights. The monitoring tools are configured and managed using the terminal, with System ensuring continuous background operation. Configuration files are edited using Nano for quick and accessible setup management.

# ABSTRACT

Monitoring system performance is essential for ensuring stability and proactively addressing issues. This project showcases a terminal-based integration of Prometheus and Grafana to create an effective real-time monitoring solution. Prometheus is used to collect and store time-series data from an Ubuntu virtual machine, with Node Exporter providing detailed insights into system-level metrics such as CPU and memory usage. Grafana is configured to connect with Prometheus and visualize this data through interactive dashboards. All services—including Prometheus, Grafana. Configuration tasks are performed through YAML files and modified using the Nano text editor. The complete monitoring stack operates directly on the Ubuntu system without containerization, making it lightweight, portable, and easy to manage. This setup enables continuous analysis, smart alerting, and scheduled reporting, improving the efficiency of system monitoring and administration.

# SCOPE AND OBJECTIVES SCOPE:

* Focuses on implementing a real-time system monitoring setup using Prometheus, Node Exporter, and Grafana on an Ubuntu virtual machine.
* Covers the end-to-end process of collecting system performance metrics, configuring Prometheus, and visualizing data through Grafana dashboards.
* Enables real-time tracking of critical system metrics such as CPU usage, memory consumption, and disk space.
* Enhance understanding of how monitoring tools can be combined to improve system visibility and proactive alerting.
* Emphasizes continuous system health monitoring, data visualization, and ease of setup using lightweight, terminal-based tools and configuration methods.

# OBJECTIVES :

* To install and configure Prometheus for collecting essential system metrics like CPU, memory, and disk usage.
* To build user-friendly Grafana dashboards for visualizing collected data through graphs and charts.
* To run Prometheus and Node Exporter persistently using Systemd for background service management.
* To manage Prometheus configurations using YAML files for a structured and reusable setup.
* To perform configuration editing efficiently using the Nano text editor within the terminal environment.

# HARDWARE AND SOFTWARE REQUIREMENT

**Hardware Requirements:**

* + Internet connection to install tools
  + Computer or Laptop with minimum 4 GB RAM
  + Linux-based or Windows System with terminal access (WSL or native)

# Software Requirements:

* + **Prometheus –** For collecting and storing system metrics.
  + **Grafana –** For visualizing the metrics collected by Prometheus.
  + **Prometheus –** To collect, store, and serve time-series performance data
  + **Slack / Email (Optional) –** For receiving alert notifications
  + **Web Browser (Chrome/Firefox) –** To view and manage Grafana and

Prometheus UIs

* + **Terminal (Command Line Interface) –** For executing all setup,

configuration, and deployment steps

# PROJECT DESIGN

* The project establishes a real-time monitoring environment on an Ubuntu virtual machine using Prometheus and Grafana.
* Node Exporter is installed to expose system-level metrics such as CPU, memory, and disk usage.
* Prometheus is configured to scrape these metrics at regular intervals using settings defined in YAML configuration files.
* Grafana is connected to Prometheus as a data source and is used to create interactive dashboards for visualizing the collected metrics.
* All configurations are managed through the terminal, using Nano as a lightweight text editor for quick and direct file editing.
* The monitoring services—Prometheus and Node Exporter—are managed using System to ensure continuous operation in the background.
* The result is a lightweight, efficient, and extendable monitoring system that provides real-time visibility into system performance.

# ADVANTAGES

* + Fully automated monitoring and alerting
  + Real-time issue detection and notification
  + Easy-to-understand visual dashboards
  + Scheduled reporting without manual effort
  + Scalable and containerized for quick deployment

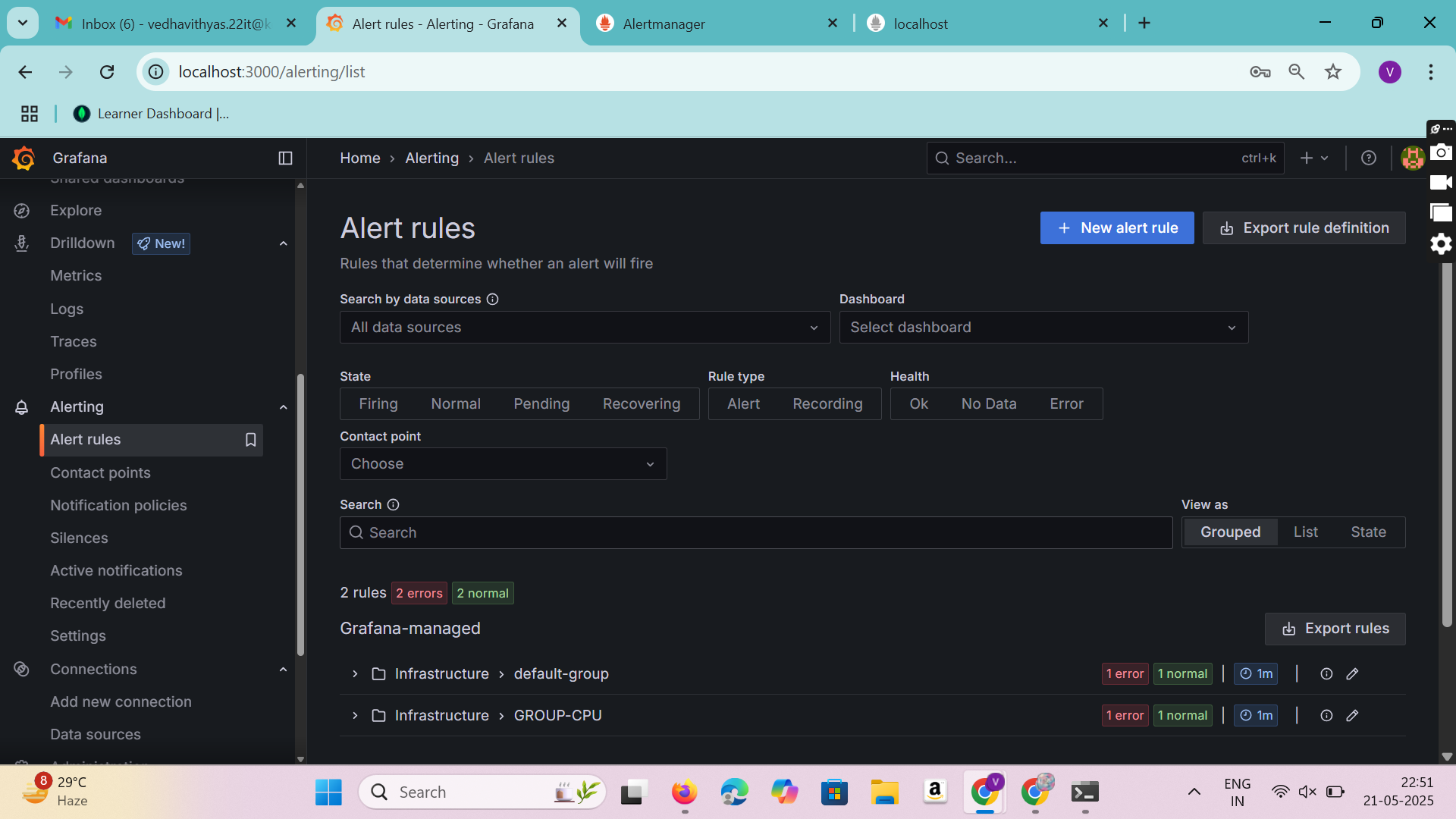
# PROPOSED SYSTEM

* Utilizes Prometheus to collect and store time-series metrics from an Ubuntu virtual machine.
* Implements Node Exporter to expose essential system-level metrics such as CPU usage, memory consumption, and disk utilization.
* Uses YAML configuration files to define Prometheus scraping targets, allowing for a clear, structured, and repeatable setup.
* Runs Prometheus and Node Exporter as background services using Systemd, ensuring continuous system monitoring without manual intervention.
* Integrate Grafana as the visualization layer to create customizable dashboards that offer clear insights into system performance.
* Enables real-time monitoring with a lightweight and efficient setup, which can be extended to support multi-node or cloud-based environments if required.

# IMPLEMENTATION OF THE PROJECT

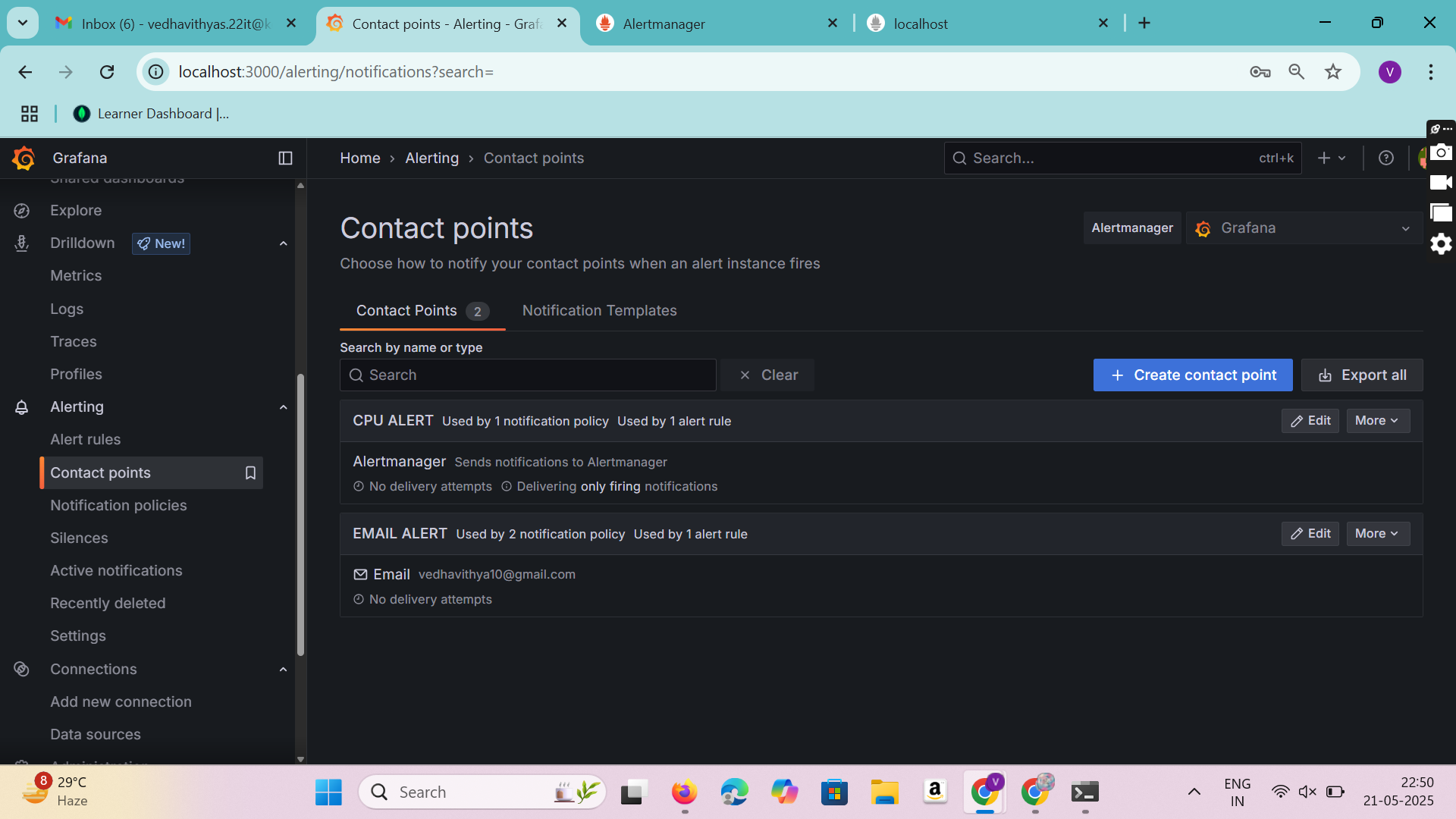
The project was implemented by setting up an Ubuntu virtual machine to serve as the monitoring environment. Prometheus was installed and configured using YAML files to define scraping intervals and target endpoints. To ensure both Prometheus and Node Exporter run continuously, System service files were created for automatic background execution. Grafana was then installed and connected to Prometheus as a data source. Custom dashboards were built in Grafana to visualize the collected metrics in real time. Configuration files were edited directly in the terminal using Nano, enabling quick and efficient setup adjustments. The final implementation resulted in a lightweight, reliable, and real-time monitoring system tailored for system performance tracking.

# OUTPUT

****

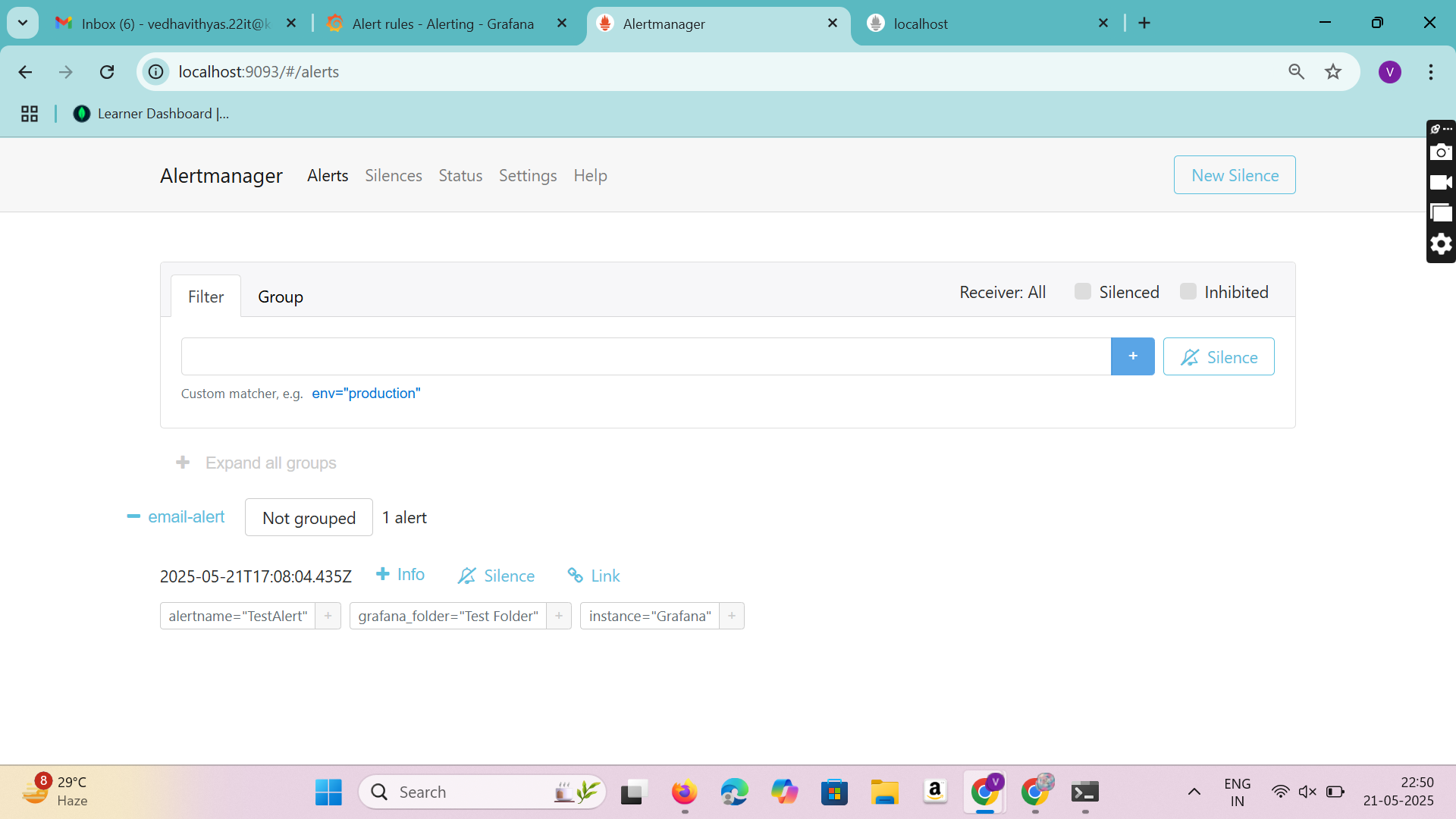
**Figure:1**

As shown in Figure 1, the Alert Rules section are created in Grafana, where alert conditions are defined to monitor system metrics. In this case, alert rules have been created to trigger when CPU usage exceeds 80%. These alerts are part of the Infrastructure group and are evaluated every minute. When the condition is met, Grafana sends notifications through two contact points: Alertmanager and Email. This setup ensures that both automated systems and users are notified immediately, enabling quick response to high CPU usage.



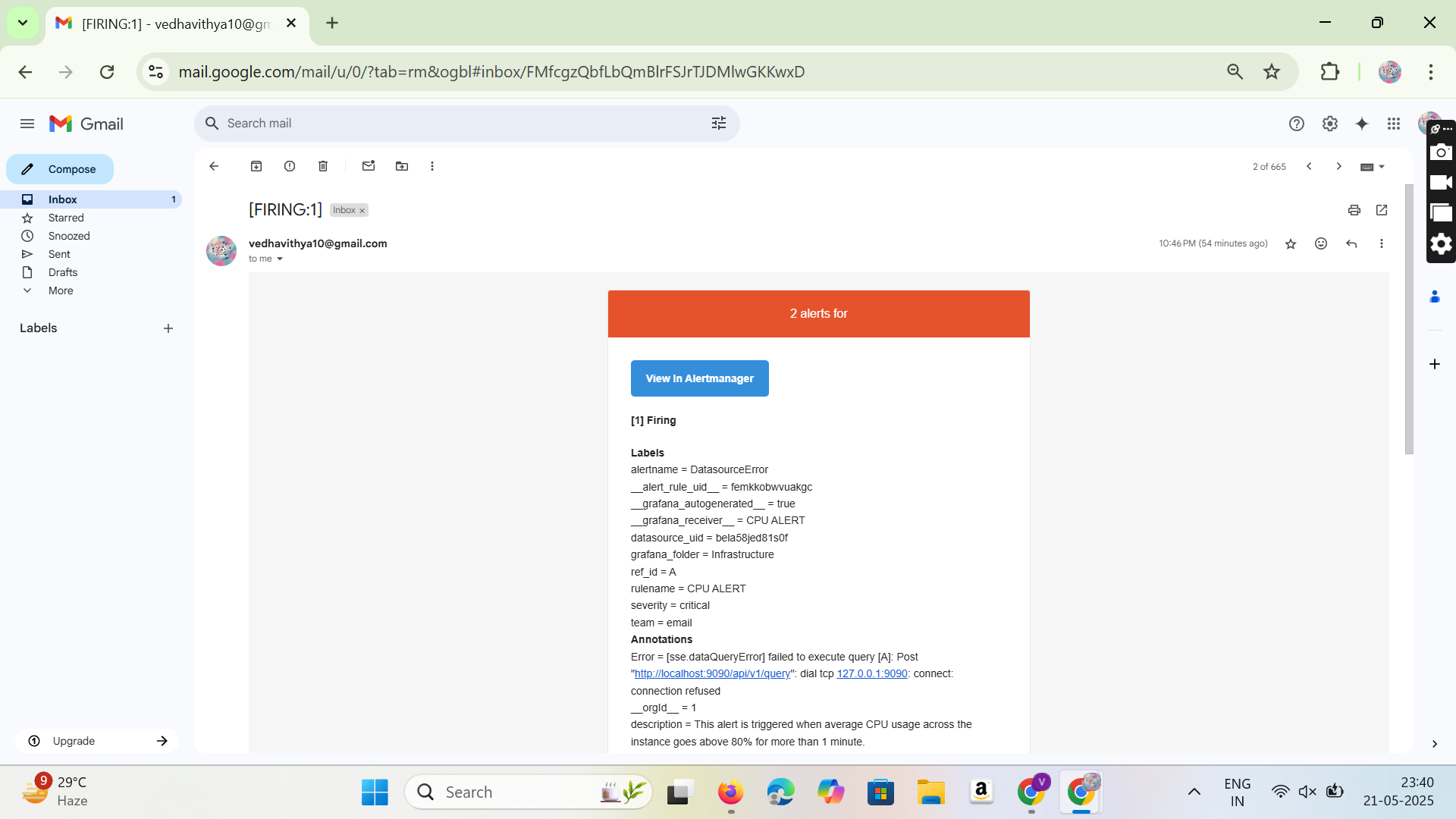
**FIGURE : 2**

As shown in Figure **2**, create the **Contact Points** section in the Grafana dashboard under the Alerting menu, where users can configure how and where alerts are sent when triggered. In this setup, two contact points have been created: **CPU ALERT**, which routes alerts to Alertmanager, and **EMAIL ALERT**, which sends notifications to the email address vedhavithya10@gmail.com. Each contact point is associated with one or more alert rules and notification policies. This interface allows users to easily manage, edit, and test notification endpoints to ensure timely alert delivery through preferred channels.



**Figure : 3**

As shown in Figure **3**, the **Alertmanager** interface, where alerts sent from Grafana are received and managed. An alert named **"Test Alert"** has been triggered and routed through the contact point labeled **email-alert**. This alert is not grouped and contains metadata such as the Grafana folder and instance. The Alertmanager UI provides options to view alert details, create silences, or follow links for further investigation. This confirms that the alert rule configured in Grafana is functioning correctly and successfully forwarding alerts to Alertmanager.



**Figure : 4**

As shown in Figure **4**, there is a **Gmail inbox** where an alert email from **Alertmanager** has been successfully received. Email indicating that one alert is currently active. The alert is labeled **CPU ALERT**, and the alert message includes detailed metadata such as description: “This alert is triggered when average CPU usage across the instance goes above 80% for more than 1 minute. “This confirms that the Grafana alert rule is correctly integrated with Alert manager and email notifications are being triggered and delivered successfully.

**COMMANDS**

# 1. Install Prometheus, Grafana, and Node Exporter Manually

# Install Prometheus on Ubuntu

sudo apt update

sudo apt install prometheus

# Start Prometheus

sudo systemctl start prometheus

sudo systemctl enable prometheus

# Install Grafana on Ubuntu

sudo apt-get install -y software-properties-common

sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"

sudo apt-get update

sudo apt-get install grafana

# Start Grafana

sudo systemctl start grafana-server

sudo systemctl enable grafana-server

# Download and install Node Exporter

tar -xvf node\_exporter-\*.\*.\*.tar.gz

sudo mv node\_exporter-\*/node\_exporter /usr/local/bin/

# Run Node Exporter

nohup /usr/local/bin/node\_exporter &

# 2. Access Prometheus, Grafana, and Node Exporter

# Prometheus: Open http://localhost:9090

# Grafana: Open http://localhost:3000 (Login: admin / admin)

# Node Exporter: Open http://localhost:9100/metrics

# 3. Configure Prometheus to Scrape Node Exporter

# Edit Prometheus config

sudo nano /etc/prometheus/prometheus.yml

# Add Node Exporterscraping config

# scrape\_configs:

# - job\_name: 'node\_exporter'

# static\_configs:

# - targets: ['localhost:9100']

# Restart Prometheus

sudo systemctl restart prometheus

# 4. Set Up Grafana Dashboard

# Log in to Grafana (http://localhost:3000) with admin/admin

# Add Prometheus as a data source: Go to Configuration → Data Sources → Add

Data Source → Prometheus

# Set URL as http://localhost:9090 and save

# CONCLUSION

The integration of Prometheus and Grafana in this project successfully enabled a real-time system monitoring and alerting solution. Prometheus collected time-series performance metrics from an Ubuntu virtual machine using Node Exporter, while Grafana provided interactive dashboards for effective visualization. The use of Systemd ensured that monitoring services ran continuously in the background without manual intervention. Configurations were handled using YAML files and edited with Nano, resulting in a lightweight, reliable, and maintainable setup. This implementation enhanced system visibility, enabled proactive issue detection, and improved the overall efficiency of infrastructure monitoring.

**FUTURE WORK**

To enhance the monitoring setup, support should be added for monitoring multiple systems and remote nodes, enabling a more comprehensive view of infrastructure health. Integration with advanced notification tools such as Microsoft Teams or Opsgenie can improve incident response and team collaboration. Implementing email-based alerts, which were not configured in the current version, would ensure timely notifications. Additionally, incorporating anomaly detection through machine learning models can enable smarter and more proactive alerting. Extending the setup to cloud environments would provide scalable and centralized monitoring capabilities. Finally, configuring role-based access control in Grafana will ensure secure and controlled data sharing among users.

**GITHUB LINK**

* https://github.com/vedhavithyaseenivasan