

Complete Project Report: Monitoring High CPU Utilization on a Linux Server using Prometheus and Grafana

**Integrating Grafana with a Linux Server for Real-Time Monitoring and
Visualization of High CPU Utilization**



1. Introduction

- In the modern era of cloud computing and high-performance computing environments, ensuring that server resources are efficiently utilized is essential. One of the most critical aspects of system performance is CPU utilization. High CPU usage, if undetected, can lead to system slowdowns, process failures, and eventually server crashes.
- This project focuses on building a real-time monitoring system for CPU usage using Prometheus and Grafana. Prometheus is a powerful metrics collection and storage tool, and Grafana is an open-source data visualization platform. Together, they provide a full monitoring and alerting solution for infrastructure observability.



2. Problem Statement

- Servers often experience unpredictable workloads due to variable user requests, application bugs, or background processes. Without proper monitoring, high CPU consumption can go unnoticed until the system performance is degraded or services crash.
- The key problem addressed in this project::::
 - > How can we monitor a Linux server's CPU usage in real-time and automatically alert system administrators when CPU usage becomes critically high?



3. Objective

- To implement a real-time monitoring solution for tracking CPU utilization of a Linux server.
- To visualize CPU usage metrics using interactive dashboards.
- To set up an automated alert system that notifies when CPU utilization crosses a predefined threshold (e.g., 80%).



4. Tools and Technologies

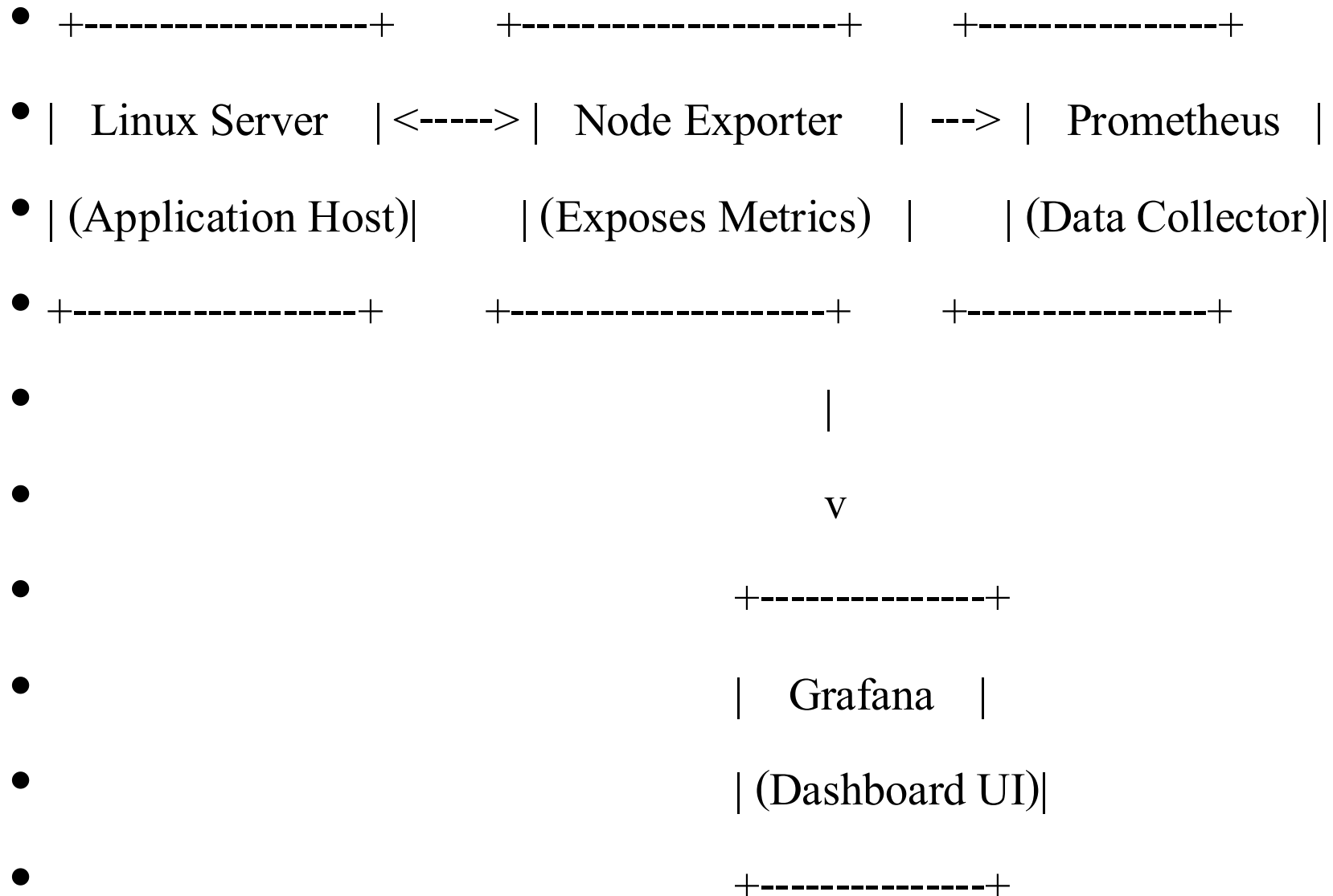
Tool/Technology

Purpose

- | | |
|---------------------------|---|
| • Linux Server (Ubuntu) | Base system where CPU usage is monitored |
| • Node Exporter | Collects system metrics (CPU, memory, disk, etc.) |
| • Prometheus | Pulls and stores metrics data in time series format |
| • Grafana | Displays metrics as graphs and dashboards |
| • AlertManager (optional) | Sends alerts when threshold is breached |
| • Web Browser | Used to access Grafana dashboards |




5. System Architecture





6. Implementation Steps

-  Step 1: Set Up Node Exporter
- Node Exporter exposes server metrics to Prometheus.
- `wget https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz`
- `tar -xvzf node_exporter-1.8.1.linux-amd64.tar.gz`
- `cd node_exporter-1.8.1.linux-amd64`
- `./node_exporter &`
- Node Exporter runs on port 9100 by default.
- Access metrics at: `http://<server-ip>:9100/metrics`



Step 2: Install and Configure Prometheus

- `wget https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz`
- `tar -xvf prometheus-2.52.0.linux-amd64.tar.gz`
- `cd prometheus-2.52.0.linux-amd64`

Modify the Prometheus config file:


- `scrape_configs:`
 - `- job_name: 'linux_cpu'`
- `static_configs:`
 - `- targets: ['localhost:9100']`
- Run Prometheus:
 - `./prometheus --config.file=prometheus.yml`
- Web UI: `http://localhost:9090`



Step 3: Install Grafana

- `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`
- `sudo systemctl start grafana-server`
- `sudo systemctl enable grafana-server`

- Grafana Web UI: `http://localhost:3000`
- Default login: admin / admin

-  Step 4: Connect Prometheus to Grafana
- Open Grafana → Settings → Data Sources
- 2. Add a new data source
- 3. Select Prometheus
- 4. Set URL to `http://localhost:9090`
- 5. Save and Test



Step 5: Create CPU Utilization Graph

- 1. Click “+” → Dashboard → Add new panel
- 2. In the query section, enter:

$$100 - (\text{avg by (instance)}(\text{rate}(\text{node_cpu_seconds_total}\{\text{mode}=\text{"idle"}\}[1\text{m}])) * 100)$$
- 3. Choose “Line Graph” for visualization
- 4. Set Title: “CPU Utilization (%)”
- 5. Define color thresholds:
 - Yellow for 70%
 - Red for 90%



Step 6: Set Up Alerts (Optional)

- Create alert rules for critical CPU usage:
- WHEN avg() OF query (A, 5m, now) IS ABOVE 80
- Set notification channels such as Email, Slack, or Webhooks.



7. Results

- After completing the setup:
- A real-time line graph is shown in Grafana indicating current CPU utilization.
- Prometheus successfully scrapes metrics from the Node Exporter every 15 seconds.
- If CPU usage exceeds the threshold (e.g., 80%), an alert is triggered.
- This setup ensures visibility and proactive management of system load.



8. Advantages

- Full-stack open-source monitoring solution.
- Real-time, customizable dashboards.
- Lightweight and efficient—no heavy agents required.
- Extendable to monitor other metrics (memory, disk, network).
- Scalable across multiple servers.



9. Real-World Applications

- Monitoring production web or application servers.
- Preventing service outages due to server overload.
- Auto-scaling cloud servers based on CPU load.
- DevOps observability and performance tuning.



10. Conclusion

- This project demonstrates how Prometheus and Grafana can be combined to build a powerful, real-time monitoring system for tracking CPU usage on a Linux server. Through this setup, server administrators gain visibility into system performance and can take immediate action when resource usage becomes critical. It provides an effective, open-source solution for maintaining server health and ensuring high availability.