

Linux + Monitoring: A containerized ubuntu monitoring stack

This document intends to describe the process of configuration and provisioning of a Linux environment (Ubuntu) with a generic monitoring solution (Prometheus + Grafana using docker). The following items will describe each step with details using as reference the repository available at: <https://github.com/P-Saraiva/UbuntuMonitoring>.









Overview

This project sets up a comprehensive monitoring environment for an Ubuntu server using open-source tools: Prometheus, Node Exporter, Grafana, Loki, and Promtail. The goal is to provide real-time visibility into system metrics (CPU, RAM, disk, network) and user activity logs, with intuitive dashboards and log search capabilities.

You can clone the repository and follow the instructions at the Readme file to replicate the scenario at an Ubuntu server.

Ubuntu Provisioning

At first, using Proxmox, I provisioned a clean Ubuntu (22.04) virtual machine to serve as the foundation for the monitoring stack. I installed docker + docker-compose to install and manage monitoring services as containers.

 Memory	1.00 GiB/16.00 GiB
 Processors	16 (2 sockets, 8 cores) [host]
 BIOS	Default (SeaBIOS)
 Display	Default
 Machine	Default (i440fx)
 SCSI Controller	VirtIO SCSI single
 Hard Disk (scsi0)	local-lvm:vm-90413-disk-0,iothread=1,size=100G
 Network Device (net0)	virtio=BC:24:11:B3:86:D8,bridge=vmb1,firewall=1

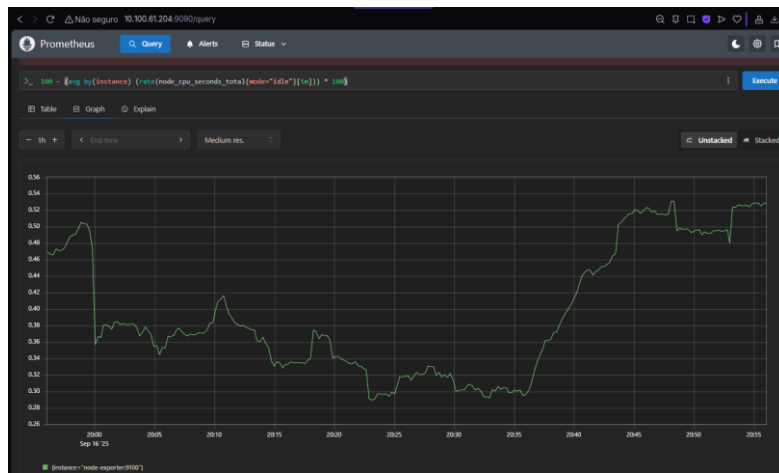
```
No LSB modules are available.  
Distributor ID: Ubuntu  
Description:    Ubuntu 22.04.5 LTS  
Release:        22.04  
Codename:       jammy
```

Prometheus Installation

As a first service, I installed Prometheus, an open-source monitoring and alerting toolkit. Prometheus is responsible for collecting and storing time-series data from various sources. Its primary objective in this setup is to gather system metrics from the Ubuntu machine for analysis and visualization using Node Exporter as it's target.

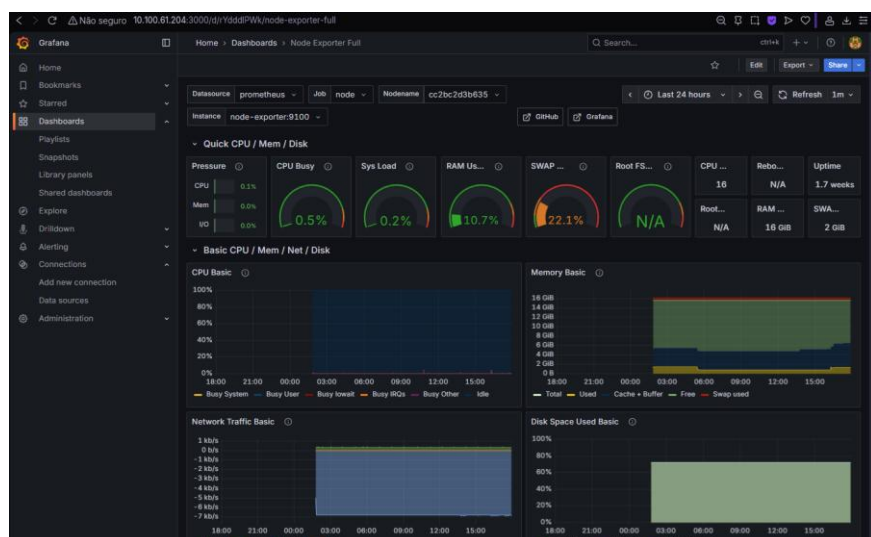
Node Exporter Deploy

To enable Prometheus to collect detailed system metrics, I deployed Node Exporter. Node Exporter is a agent that exposes hardware and OS metrics (CPU, memory, disk, and network statistics) in a format that Prometheus can scrape. With Node Exporter running, I had a functional prototype for basic system monitoring.



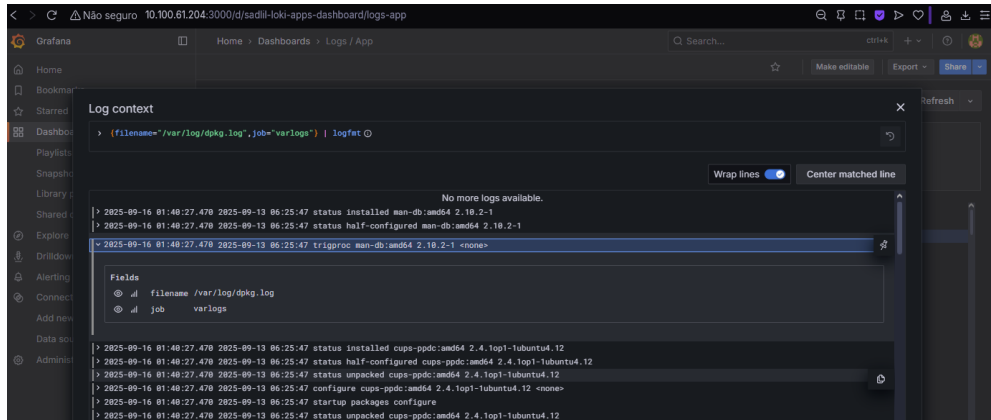
Grafana Integration

To enhance data visualization and make metric analysis more accessible, I integrated Grafana. Grafana is a powerful open-source analytics and monitoring platform that connects to Prometheus and provides customizable dashboards. With Grafana, I could use community-built templates and share intuitive visualizations of system health and performance.



Loki and Promtail for Logs

To extend monitoring to user activity and system logs, I added Loki and Promtail. Loki is a log aggregation system designed to work seamlessly with Grafana. Promtail is an agent that ships local log files to Loki. This combination allows for efficient log collection, storage, and querying within Grafana, providing a unified view of both metrics and logs.



How the stack monitors the machine

- **Metrics Collection:** Node Exporter exposes system metrics, which Prometheus scrapes at regular intervals. These metrics include CPU usage, memory consumption, disk space, and network activity.
- **Log Collection:** Promtail tails system log files (e.g., /var/log/*.log) and sends them to Loki, which stores and indexes the logs for efficient querying.
- **Visualization:** Grafana connects to both Prometheus and Loki, providing dashboards for real-time monitoring and historical analysis of both metrics and logs.
- **Orchestration:** .yaml config files and docker run services in containers binding volumes and ports, allowing integration with IaC, such as Ansible and/or Terraform.