Docker Observability Stack Documentation

By: Ritvik Indupuri

Date: 11/7/2025

Overview

This documentation outlines a lightweight observability stack deployed with Docker, designed to provide real-time system telemetry for the **Purdue BoilerExams** environment.

The monitoring stack uses **Prometheus, Grafana, Node Exporter, and Telegraf** to collect, store, and visualize system metrics.

This setup observes both the host system (via Node Exporter) and container-level metrics (via Telegraf's Prometheus output) and displays them inside custom Grafana dashboards.

Why We Built This System

BoilerExams requires a dependably performing backend, and monitoring resource consumption is critical for stability—especially under heavy student load.
This observability stack helps us:

- Track CPU, memory, disk, and network utilization
- Detect early warning signals (spikes, saturation, anomalies)
- Monitor both the local Linux server and Docker containers
- Provide real-time dashboards for troubleshooting
- Build operational awareness similar to industry-grade DevOps setups

---

System Architecture

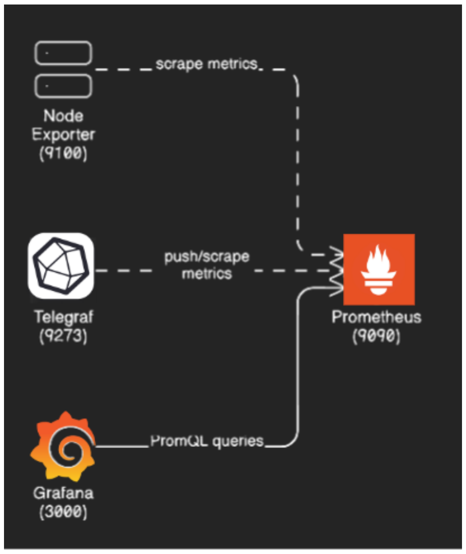

Figure 1: Observability stack architecture

This architecture illustrates the metrics collection and visualization pipeline implemented in the observability stack. **Node Exporter** runs on the host system and exposes low-level hardware and OS metrics (CPU, RAM, disk, filesystem, network) on port **9100**. At the same time, **Telegraf** runs as a containerized metrics agent on port **9273**, gathering richer system data such as process statistics, disk I/O, network throughput, and container-aware metrics.

Both Node Exporter and Telegraf expose their metrics over HTTP endpoints in Prometheus format. **Prometheus**, running on port **9090**, periodically scrapes these endpoints at a configured interval (5 seconds in this deployment). After Prometheus stores the scraped time-series data, **Grafana** (port **3000**) queries Prometheus using PromQL to generate live dashboard visualizations.

Container Documentation (High-Level)

---

1. Prometheus

Purpose:
Central metrics collector. Scrapes metrics from Node Exporter and Telegraf on a fixed interval and stores them as time-series.

Exposed Port:

- **9090/tcp** — Prometheus web UI and querying endpoint.

Interactions:

- Scrapes **Node Exporter** at 10.0.0.1:9100
- Scrapes **Telegraf** at telegraf:9273 (Docker network)
- Grafana uses Prometheus as its **only datasource**

Config Notes:

- scrape_interval: 5s
- Static scrape jobs (no service discovery)
- Config stored in prometheus/prometheus.yml

---

2. Grafana

Purpose:
Visualization layer for CPU, memory, disk I/O, disk usage, and system statistics from Prometheus.

Exposed Port:

- **3000/tcp** — Grafana dashboard UI.

Interactions:

- Uses **Prometheus** as the datasource
- Your dashboard contains:
    - CPU Usage % (by subtracting idle rate)
    - Memory Usage % (mem_used / mem_total * 100)
    - Disk I/O panels from Telegraf diskio plugin
    - Disk usage gauge (node filesystem metrics)
    - Multi-panel layout for monitoring

Dashboard Notes:

- All panels built manually using PromQL
- Gauges + time-series used for CPU, memory
- Disk Space gauge created for Node Exporter filesystem usage
- Everything based strictly on Telegraf + Node Exporter data shown in screenshots

---

3. Node Exporter (node_exporter_firewall)

Purpose:
Exposes Linux host metrics: CPU, memory, filesystem, disk usage, and network statistics.

Exposed Port:

- **9100/tcp** — Node Exporter /metrics endpoint.

Interactions:

- Prometheus scrapes this endpoint directly
- Only host providing disk usage information
- Used for the "Disk Space (Filesystem Usage)" gauge panel

Notes:

- Running on **10.0.0.1** (your local network)
- Provides raw kernel stats (e.g., node_filesystem_free, node_filesystem_avail)
- Only real filesystems used in your dashboard (loop/overlay not shown)

---

4. Telegraf

Purpose:
Container-level metrics collector. Provides CPU, memory, disk I/O metrics in Prometheus format.

Exposed Port:

- **9273/tcp** — /metrics output from Telegraf Prometheus client.

Interactions:

- Prometheus scrapes Telegraf for:
    - cpu_time_*
    - mem_used, mem_total
    - diskio_read_bytes, diskio_write_bytes
- Grafana panels use these for:
    - CPU usage %
    - Memory %
    - Disk read/write rates
    - Container-level time-series charts

Notes:

- Telegraf config located at /etc/telegraf/telegraf.conf
- ignore_fs enabled by default — required to avoid errors
- Plugins automatically detected (no custom plugins added)
- Metrics confirmed working via http://10.0.1.15:9273/metric

Ports & Endpoints Summary

Service	Port	Endpoint	Purpose
Prometheus	**9090**	http://10.0.1.15:9090	Query UI + metrics store
Grafana	**3000**	http://10.0.1.15:3000	Dashboards
Node Exporter	**9100**	http://10.0.0.1:9100/metrics	Host metrics

Telegraf	**9273**	http://telegraf:9273/metrics	Container metrics