**Monitoring CI/CD PHP Web Application with Grafana and Prometheus**

|  |  |
| --- | --- |
| **Version** | 1.0.0 |
| **Prepared by** | Rimah Houssameldine |
| **Audience** | ParkInnovation Team. |
| **Date** | 2024-3-3 |

**Table of Contents**

[Introduction: 2](#_Toc160390204)

[Technologies Used: 2](#_Toc160390205)

[Prerequisites 2](#_Toc160390206)

[Steps for Monitoring CI/CD PHP Web Application with Grafana and Prometheus 4](#_Toc160390207)

[Setting Up Prometheus 4](#_Toc160390208)

[Docker Engine Configuration 8](#_Toc160390209)

[Setting Up Grafana 9](#_Toc160390210)

[Docker metrics visualization 11](#_Toc160390211)

[Jenkins metrics visualization 11](#_Toc160390212)

[Windows Server metrics visualization 12](#_Toc160390213)

[Prometheus server metrics visualization 13](#_Toc160390214)

[Alerting Setup 14](#_Toc160390215)

[Best Practices 17](#_Toc160390216)

[Conclusion 17](#_Toc160390217)

# Introduction:

This documentation provides step-by-step instructions for setting up monitoring for a CI/CD PHP web application using Grafana and Prometheus. Monitoring your CI/CD pipeline is crucial for ensuring the reliability and performance of your application.

# Technologies Used:

* Grafana: A visualization and monitoring tool.
* Prometheus: A monitoring and alerting toolkit.
* Docker: For containerization.
* Jenkins: CI/CD automation server.
* Windows Server: Host for the PHP web application.

# Prerequisites

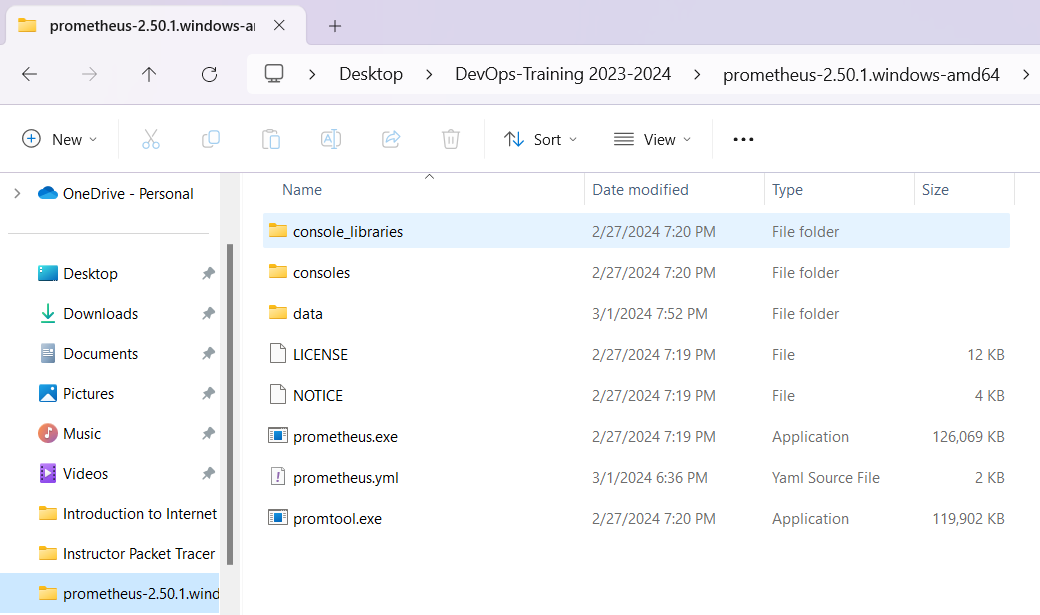
Before proceeding, ensure you have the following:

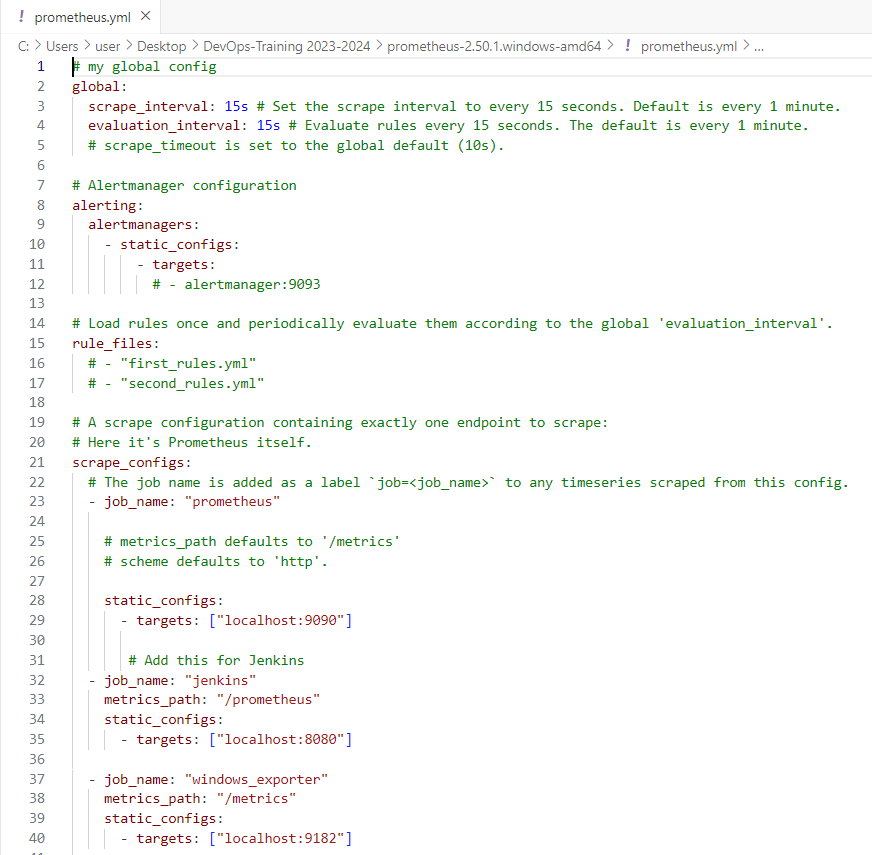
* Access to Grafana and Prometheus servers.
* Docker installed on the target hosts.
* Jenkins server up and running on localhost:8080.
* Prometheus server up and running on localhost:9090.
* Windows Server with appropriate metrics enabled.

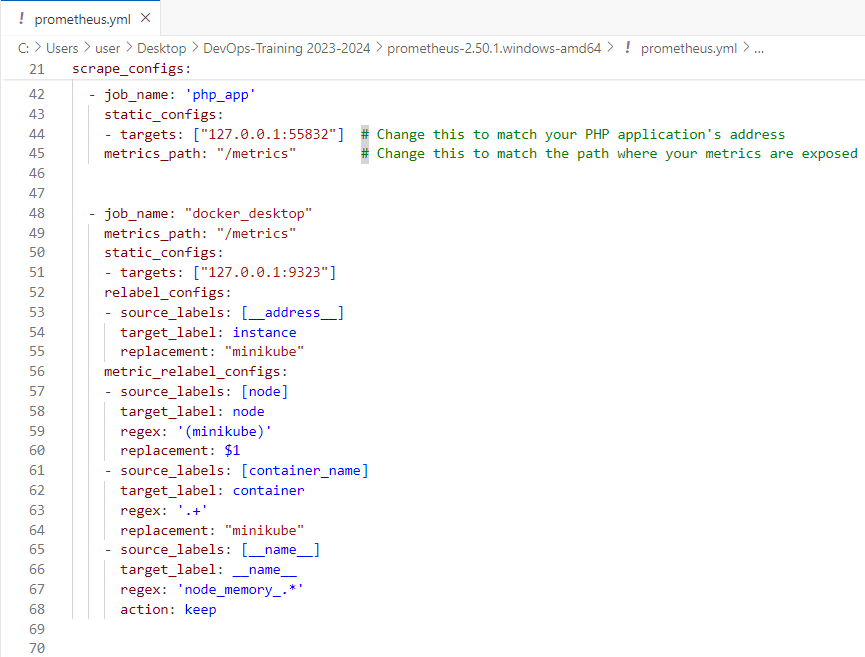
# Steps for Monitoring CI/CD PHP Web Application with Grafana and Prometheus

### Setting Up Prometheus

Configure Prometheus scrape jobs for Docker, Jenkins, Windows Server, and Prometheus itself.

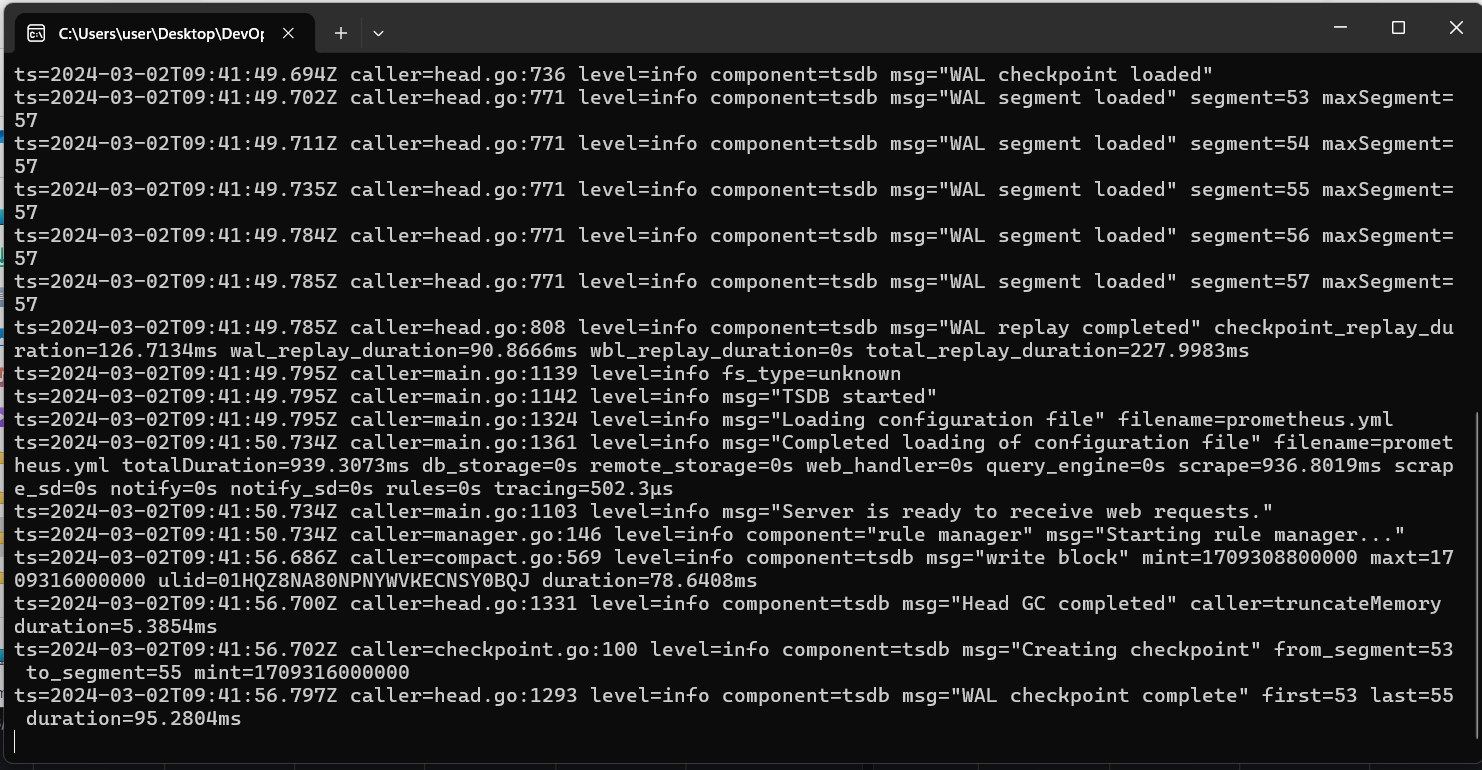


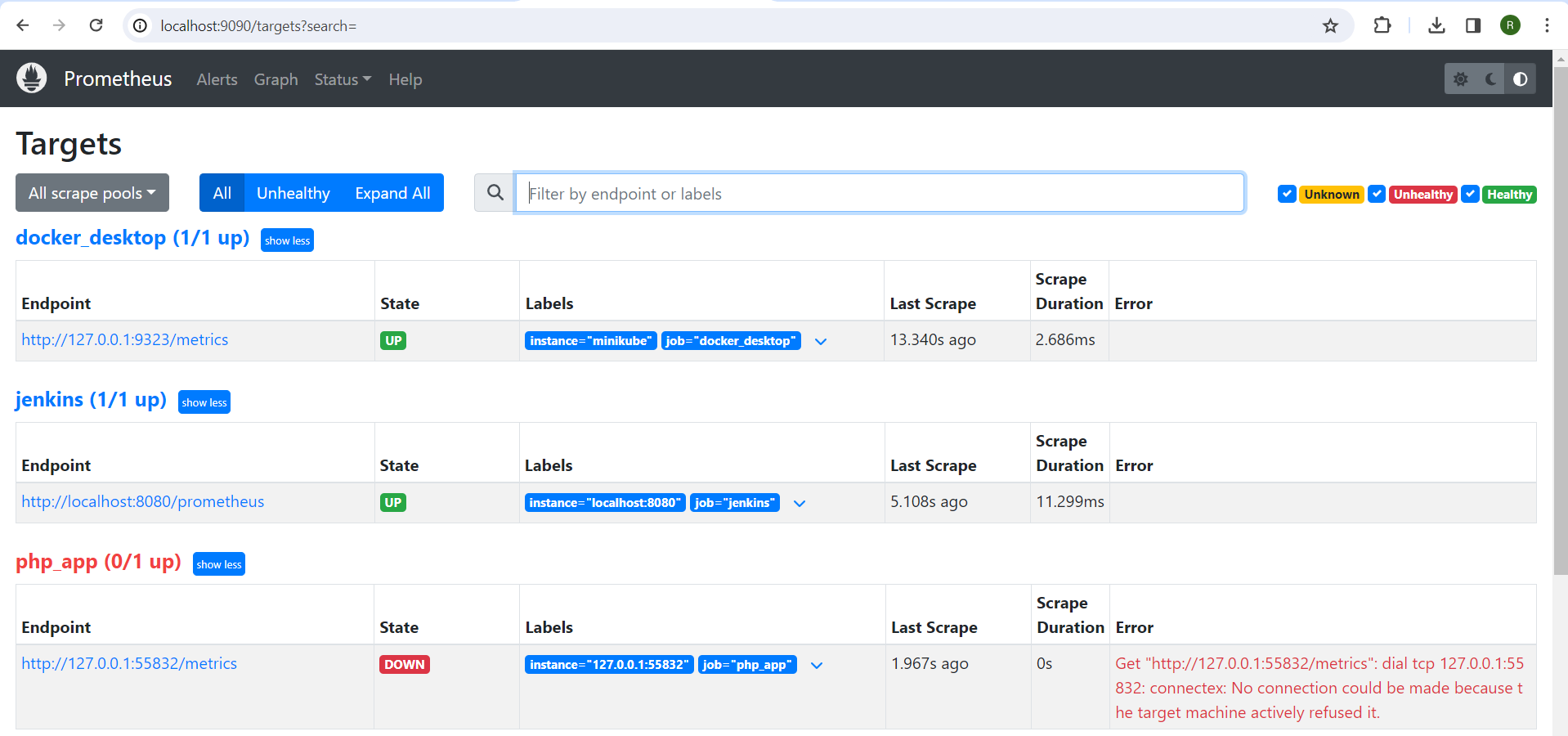




This YAML configuration file for Prometheus sets the global scrape and evaluation intervals to 15 seconds each, along with a default scrape timeout of 10 seconds. Alerting configurations, although present, are commented out, indicating no specific alert manager is configured. Rule files are also commented out, suggesting that no additional rule files are loaded. The file then defines scrape configurations for various job types including Prometheus itself, Jenkins, Windows Server, a PHP application, and Docker Desktop. Each job specifies the targets and metrics paths for scraping metrics, with additional configurations for modifying labels and filtering metrics as required.

Verify Prometheus configuration by accessing Prometheus web UI (<http://prometheus-server-ip:9090>).

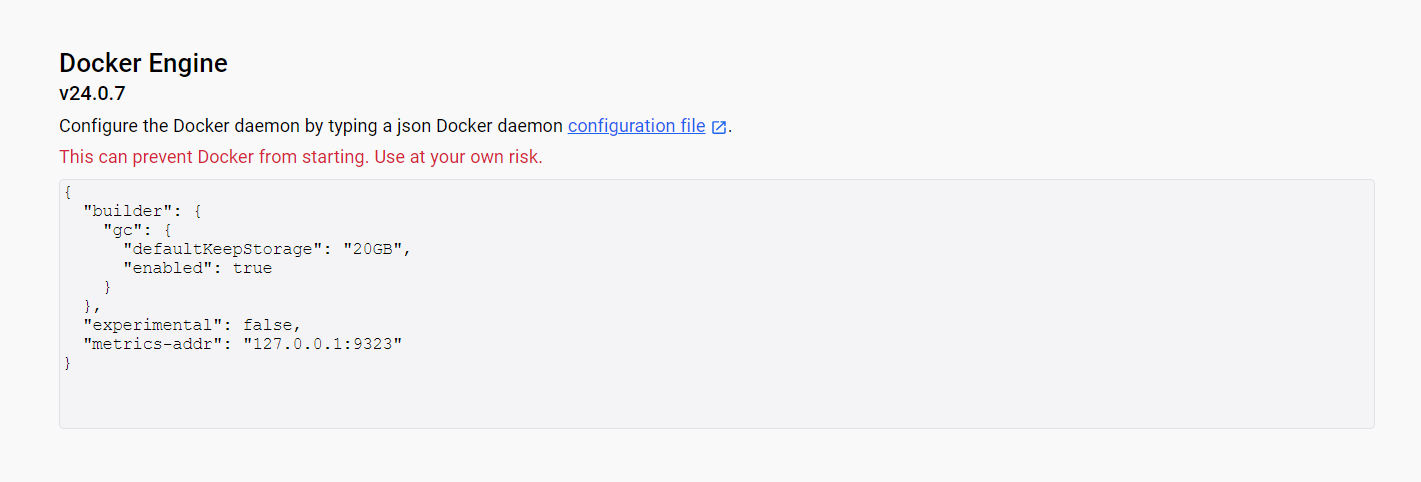




A screenshot of a chat

Description automatically generated

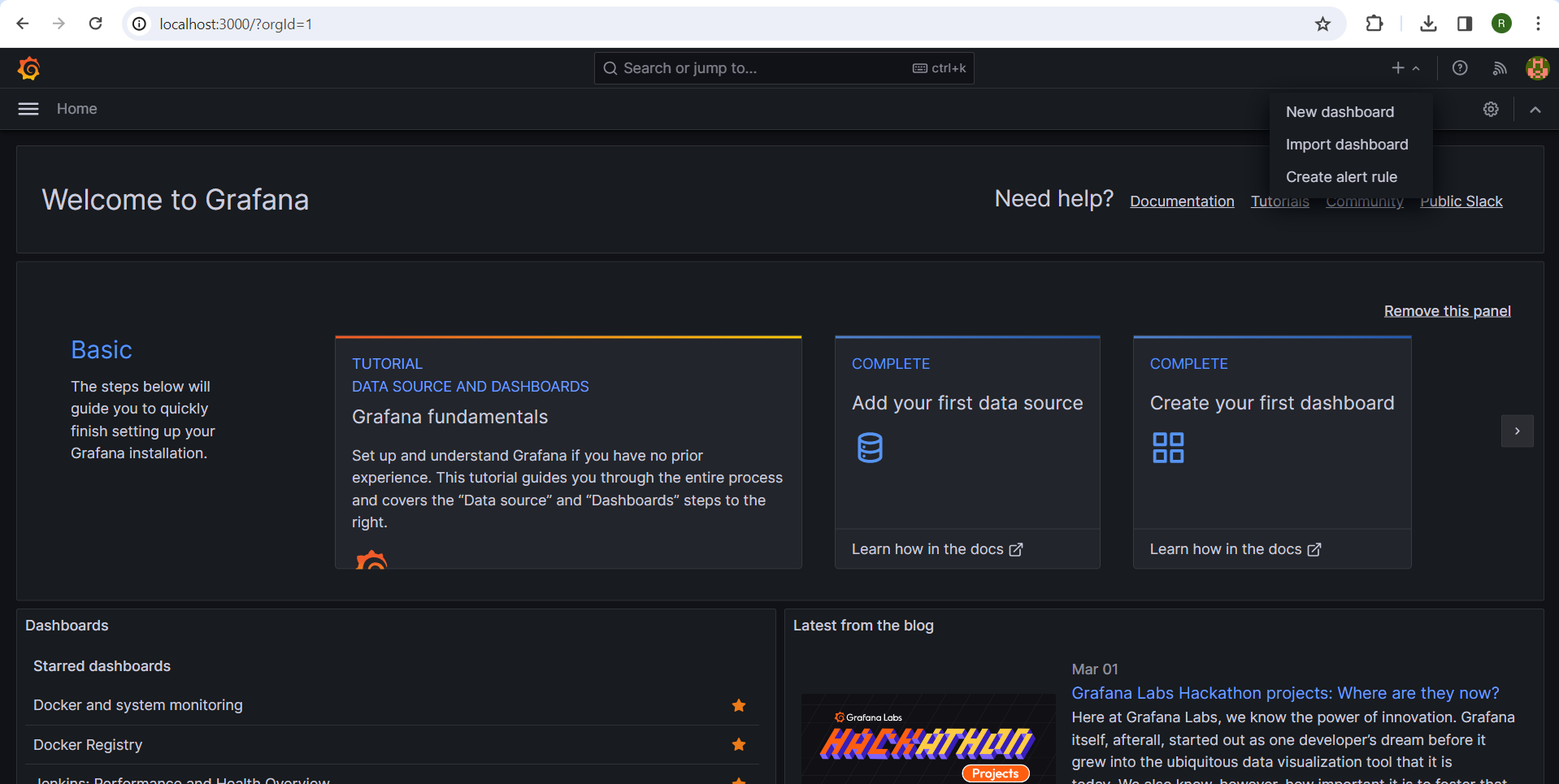
### Docker Engine Configuration



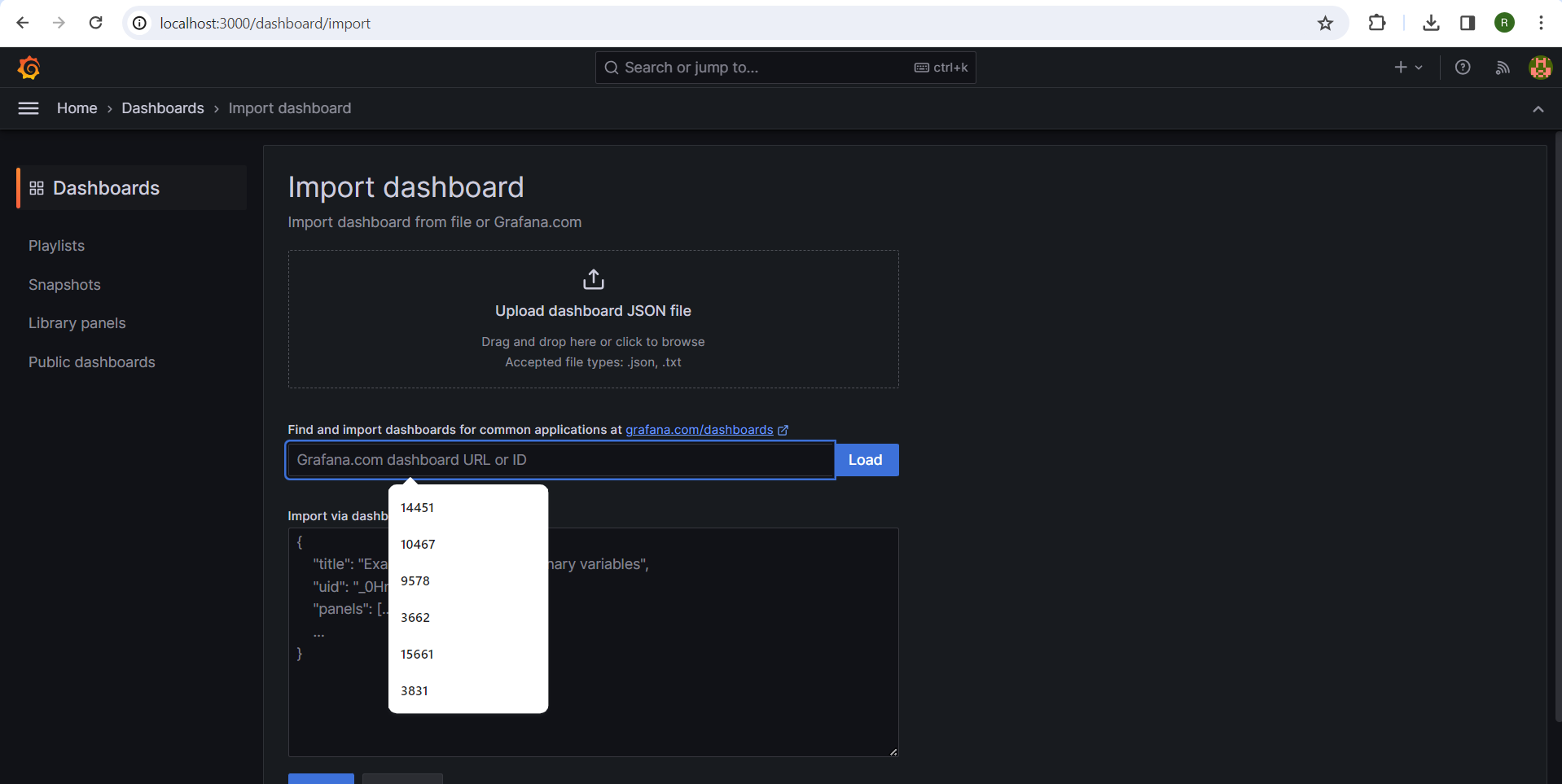
This JSON snippet configures settings for a container runtime environment, likely Docker. Within the "builder" section, it defines parameters related to garbage collection, including the default amount of storage to retain ("20GB") and whether garbage collection is enabled ("true"). The "experimental" parameter is set to "false," indicating that experimental features are disabled. Lastly, the "metrics-add" parameter specifies the address and port ("127.0.0.1:9323") where metrics are exposed, facilitating monitoring of the container runtime environment. Overall, this configuration snippet enables efficient resource management, ensures stability by disabling experimental features, and provides metrics for monitoring purposes.

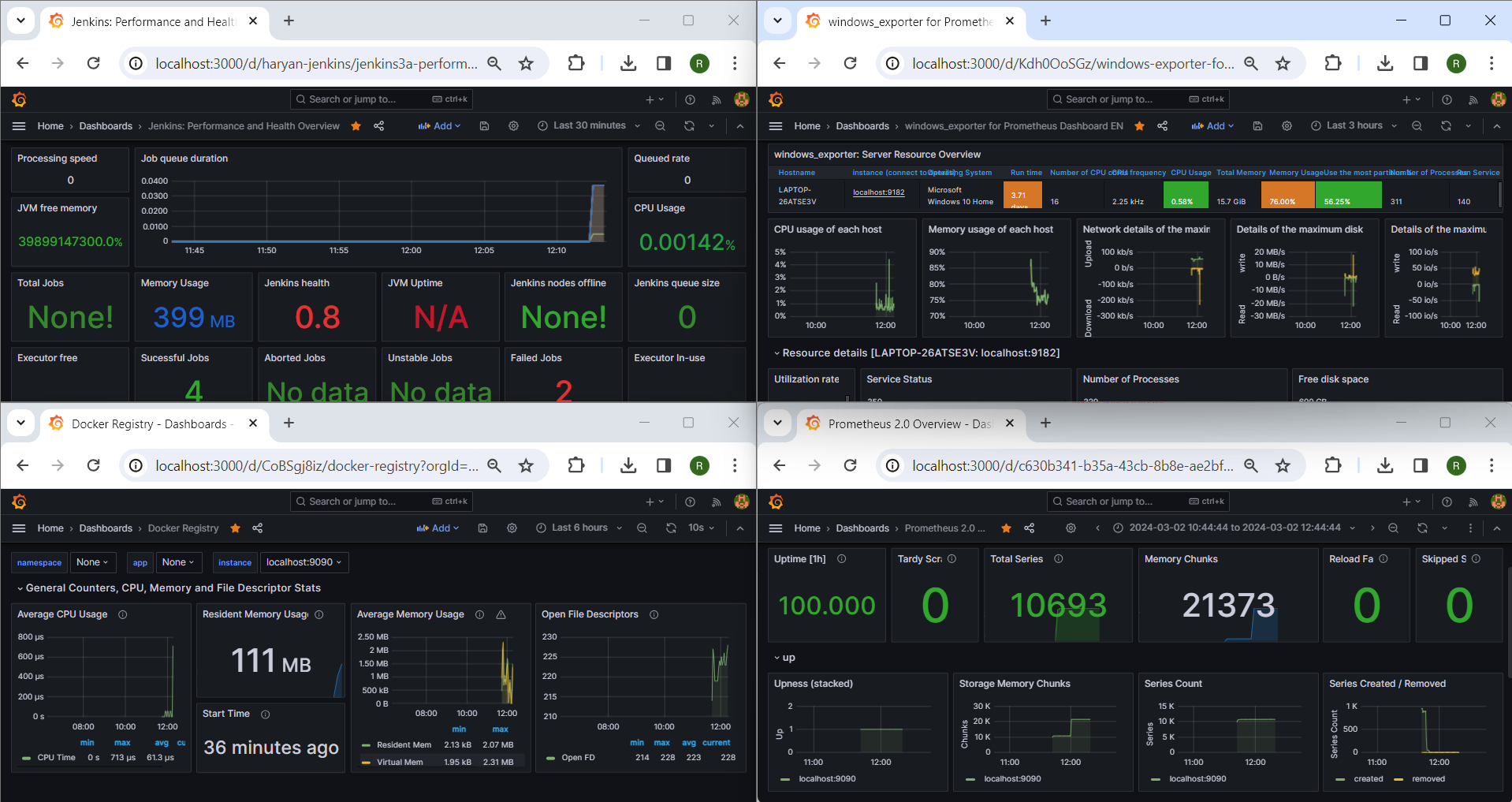
### Setting Up Grafana

Verify Grafana configuration by accessing Grafana web UI (<http://grafana-server-ip:3000>).

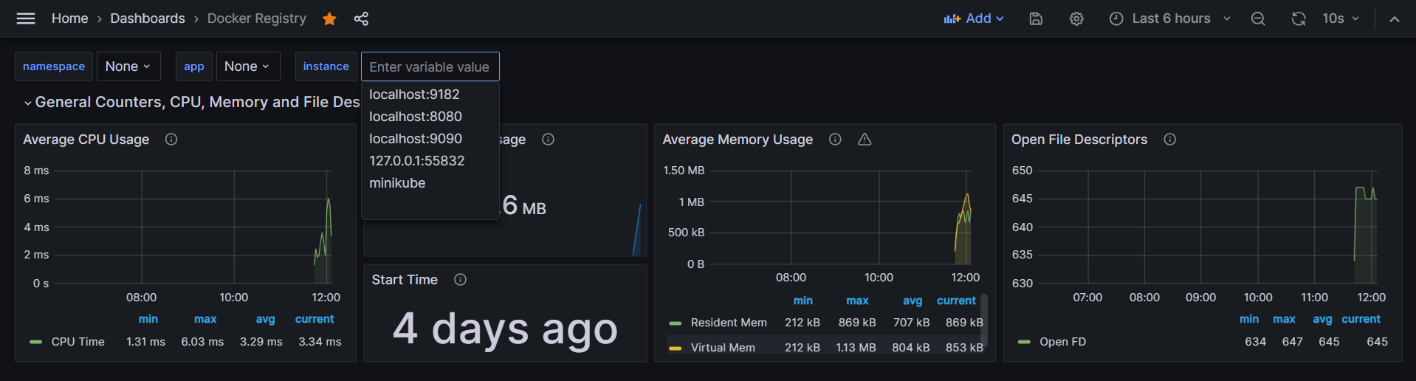


Create Grafana dashboard for Docker metrics visualization, Jenkins metrics visualization, Windows Server metrics visualization, and Prometheus server metrics visualization.

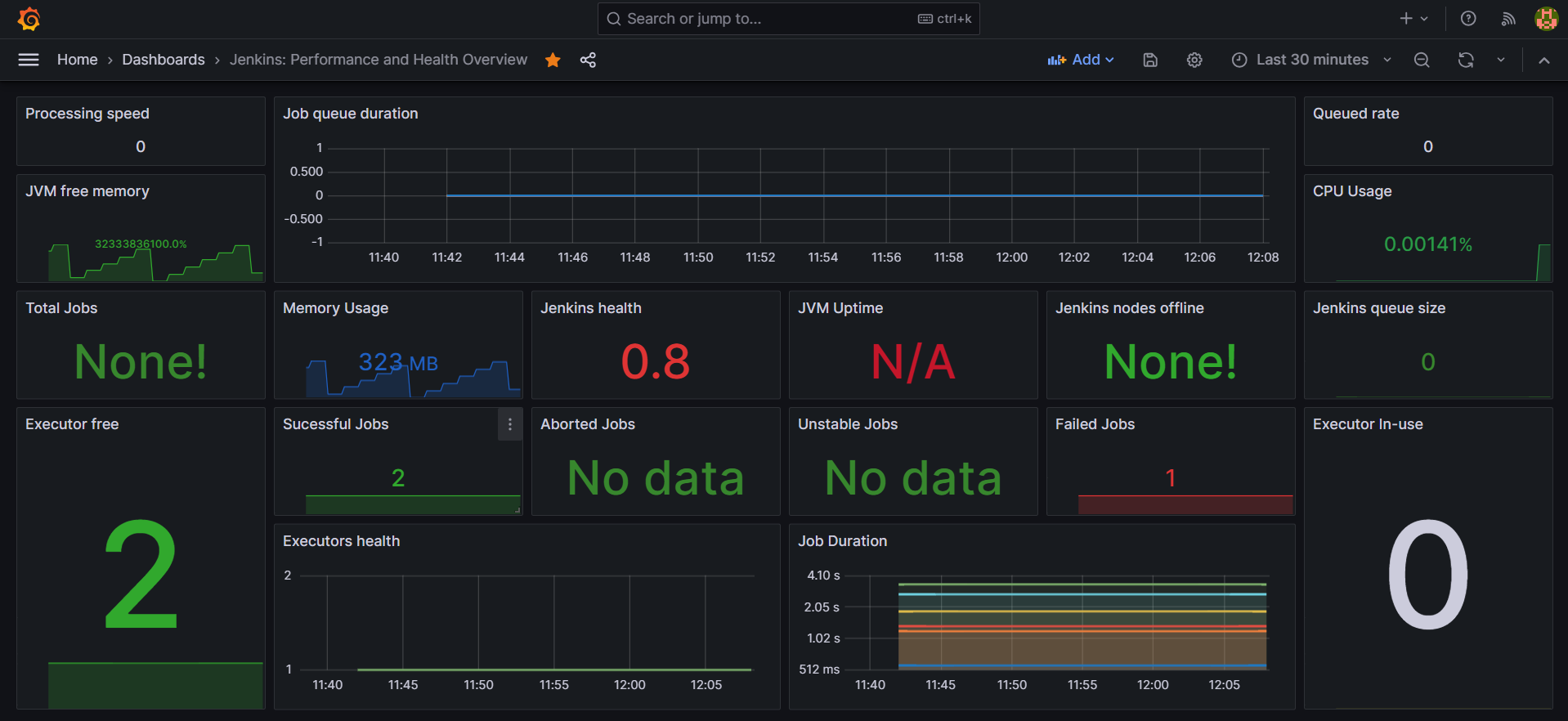




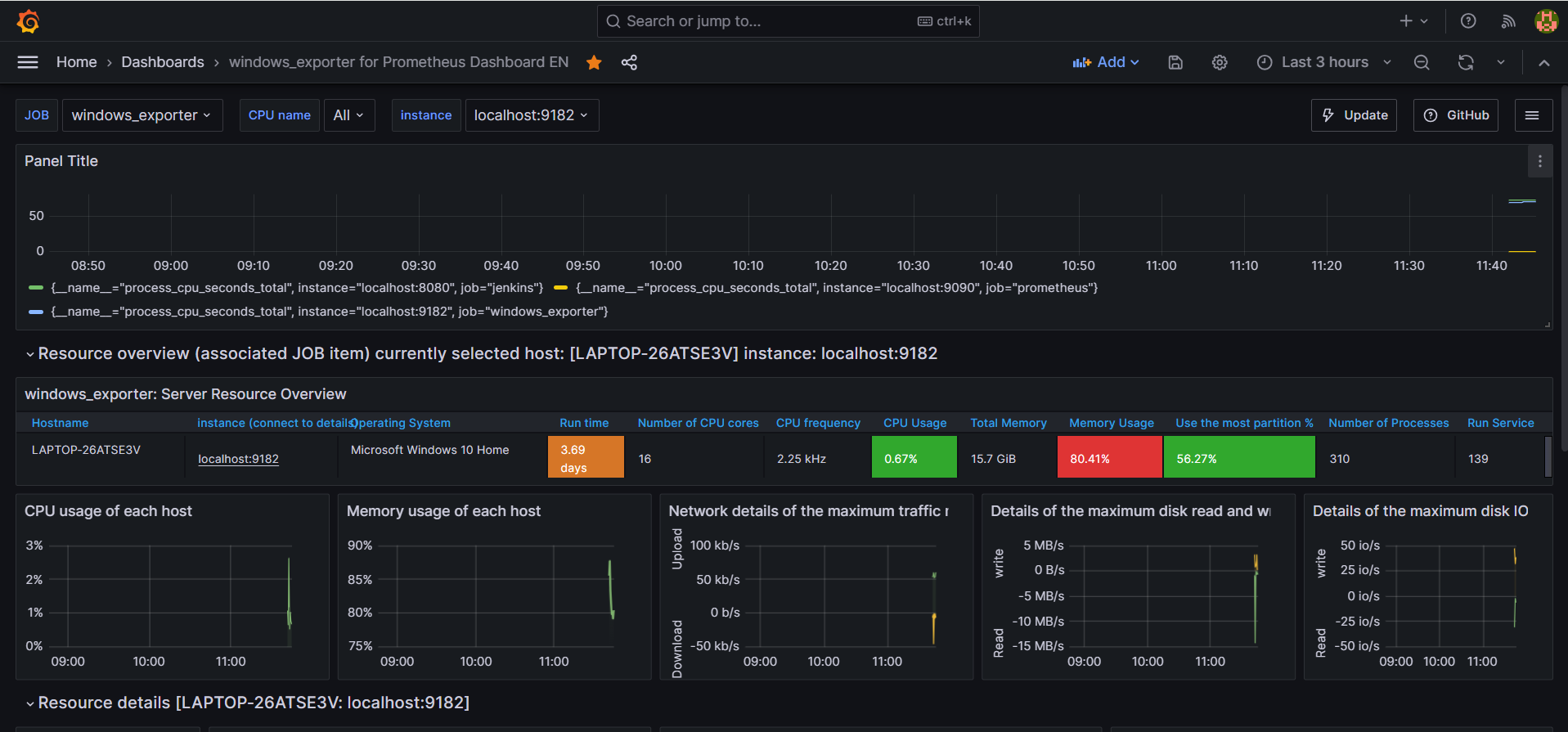
### Docker metrics visualization

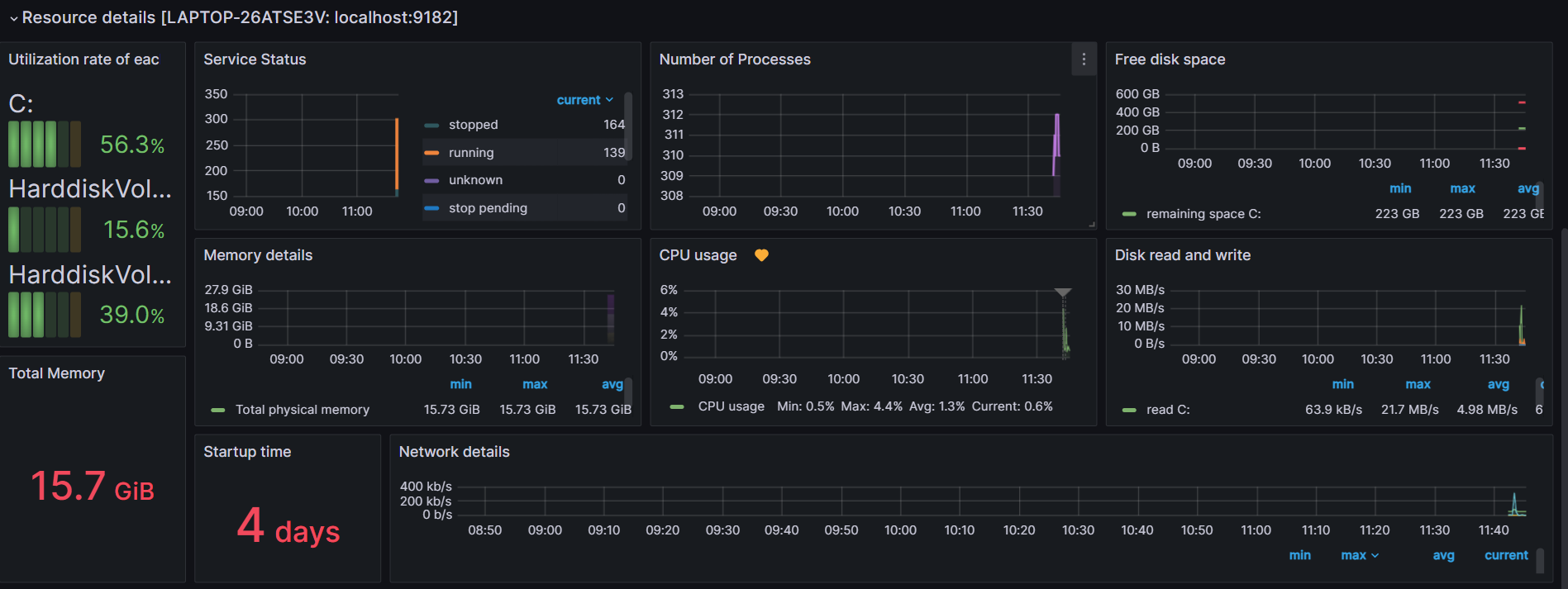


### Jenkins metrics visualization

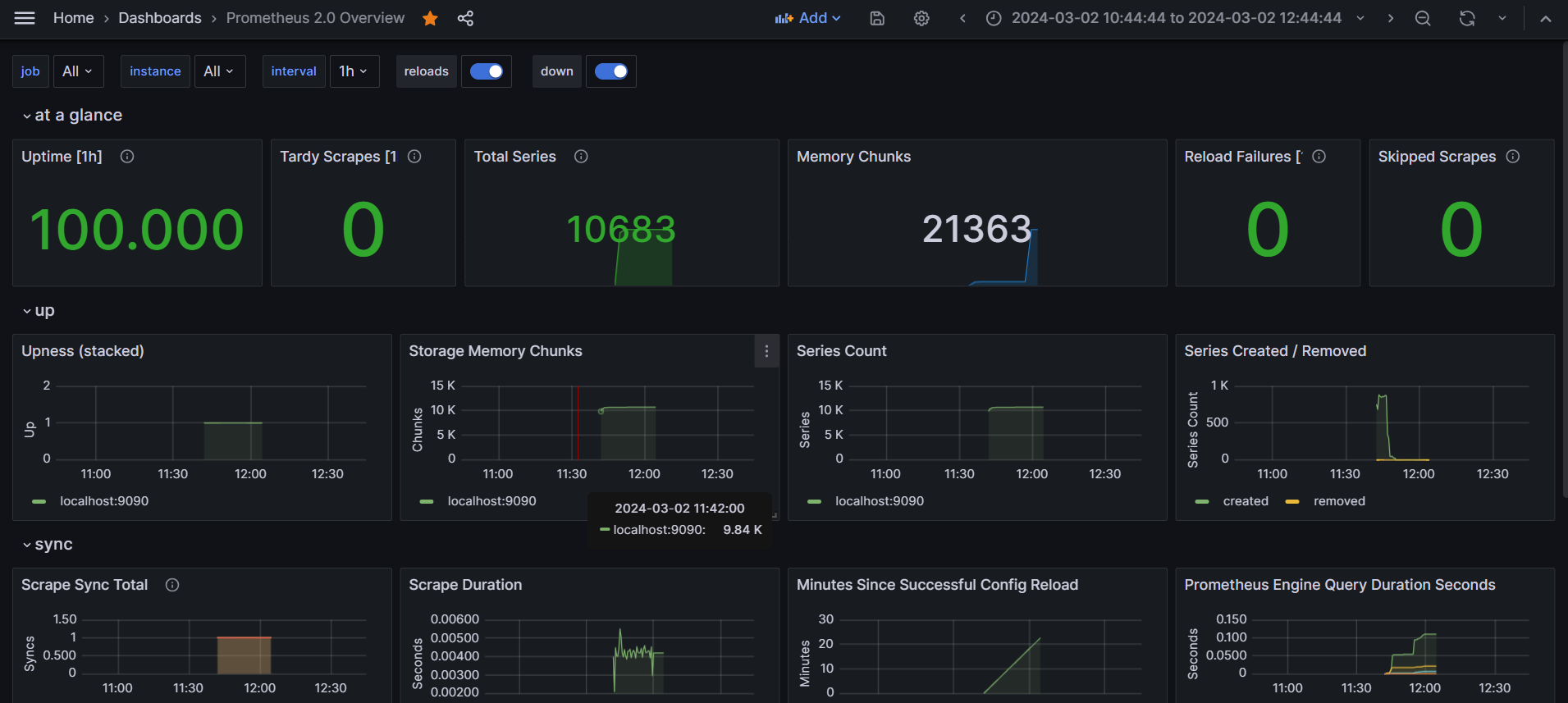


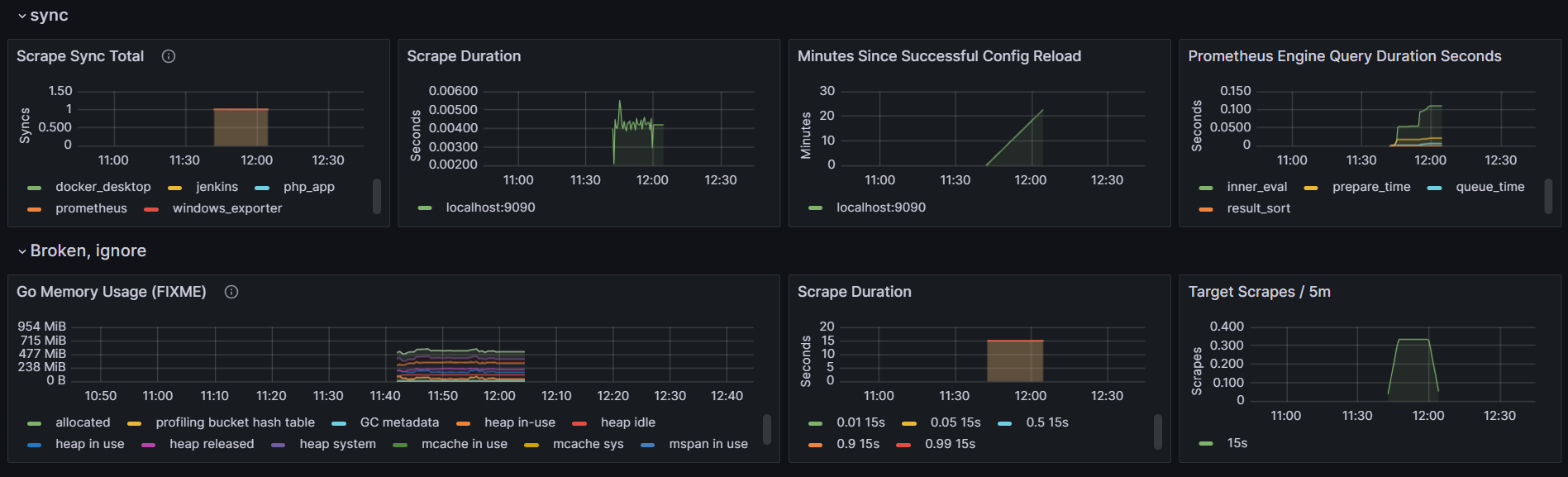
### Windows Server metrics visualization





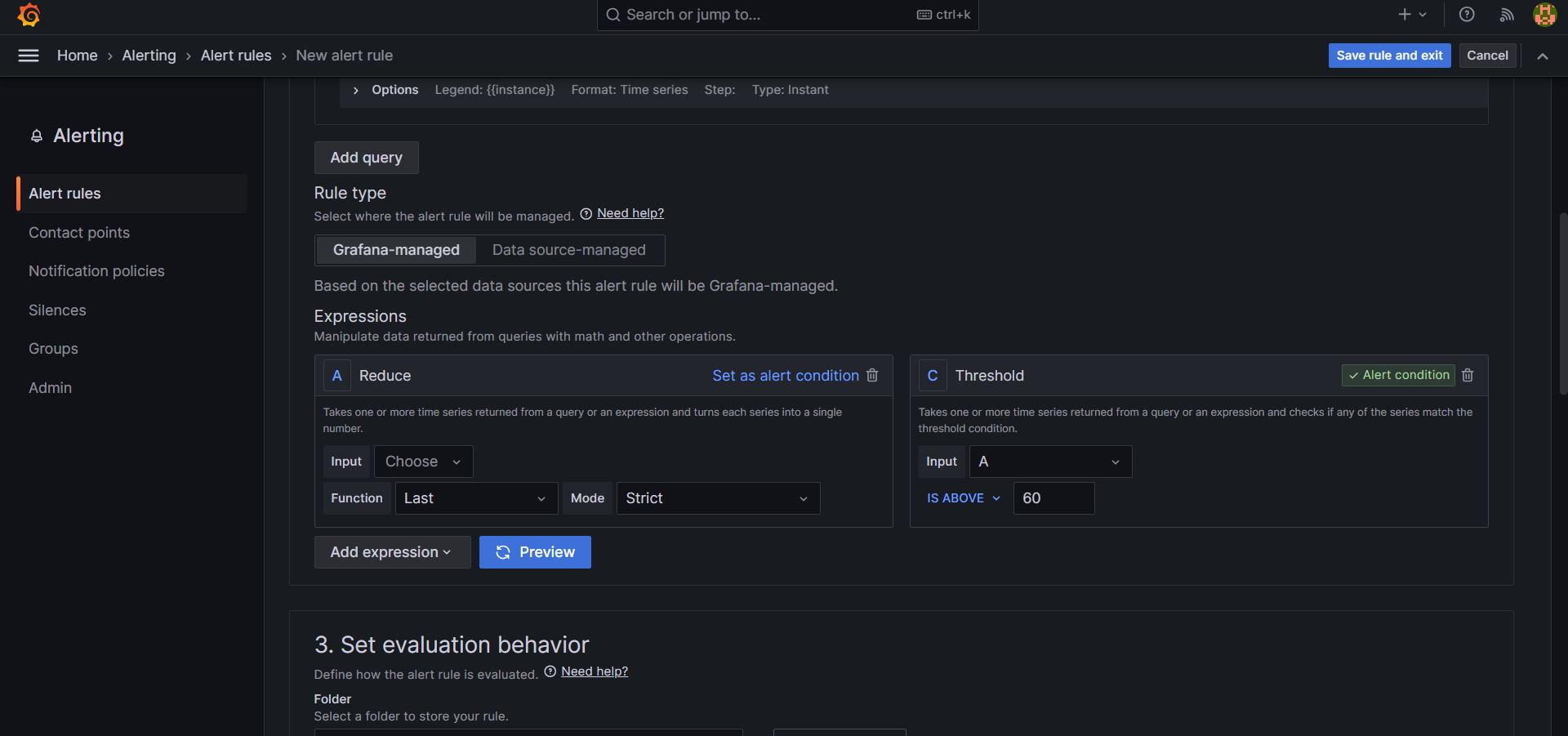
### Prometheus server metrics visualization

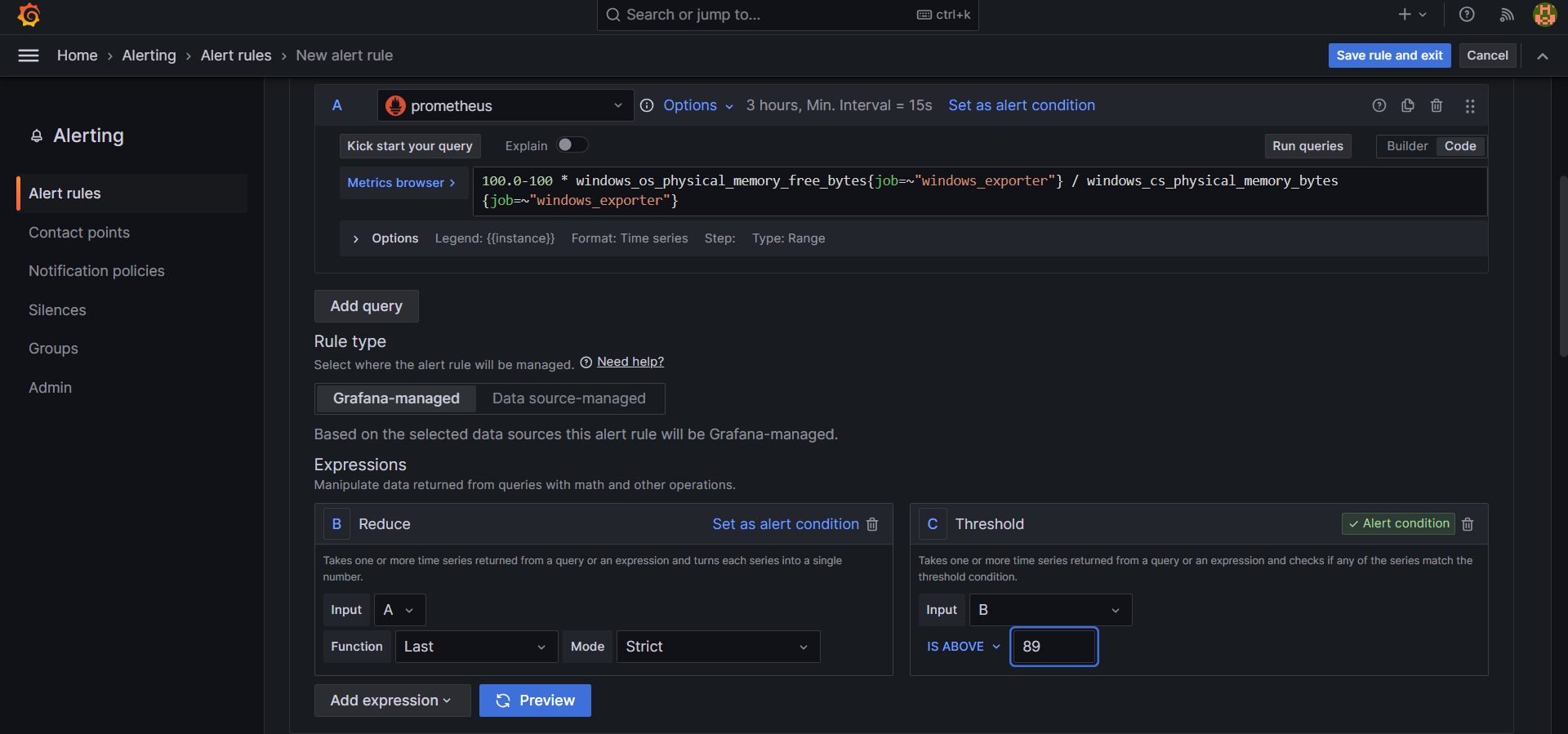




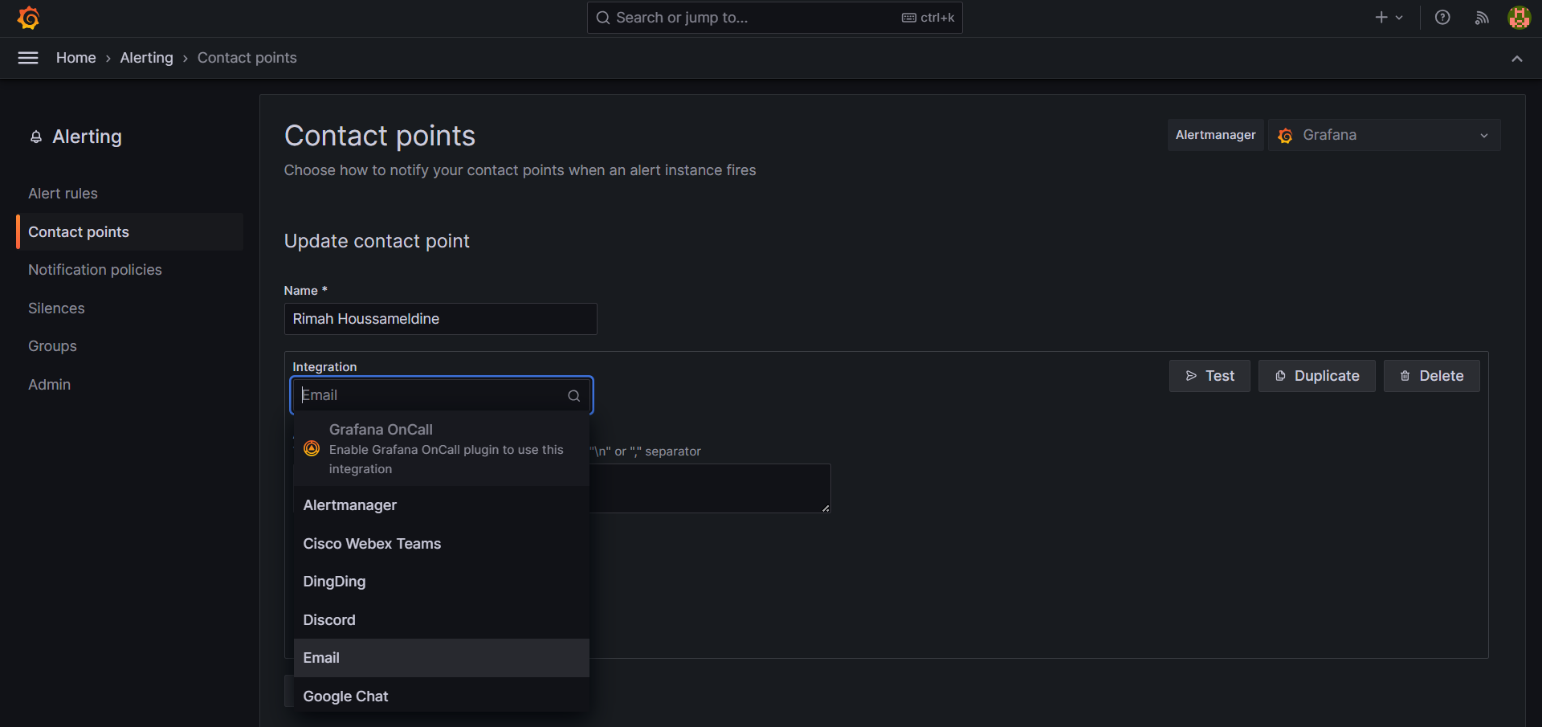
### Alerting Setup

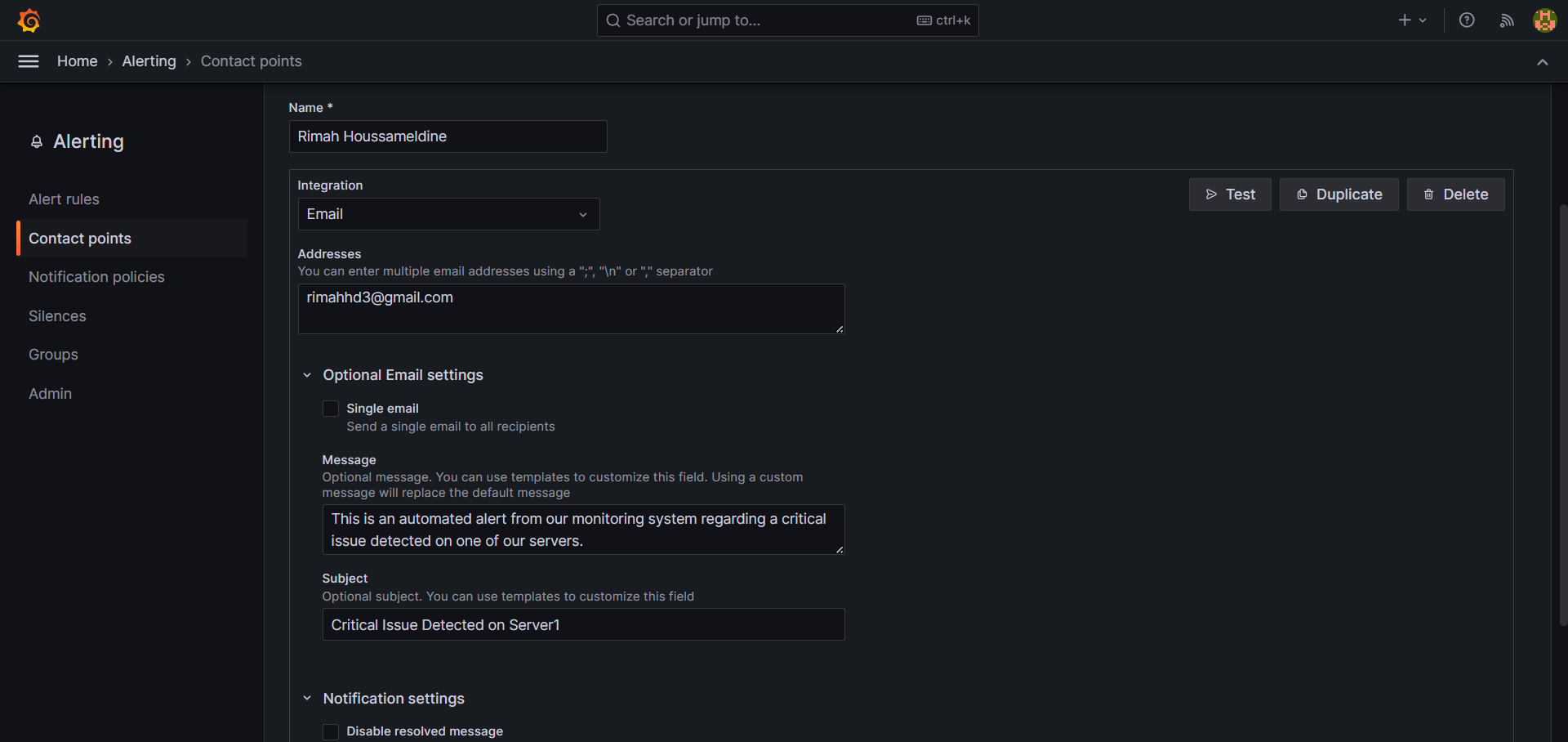
Set up alerting rules in Prometheus for critical metrics.



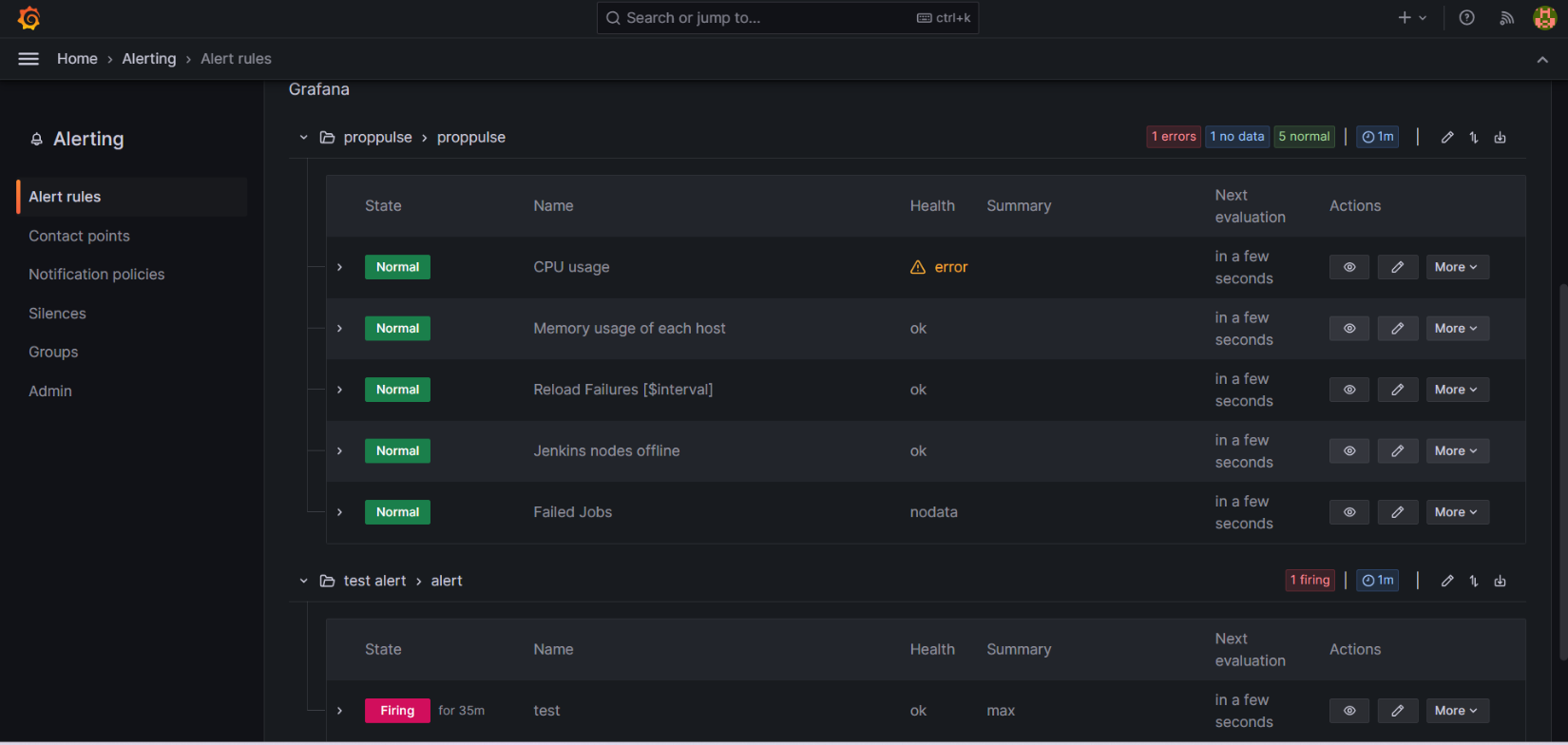


Configure alert notification channels in Grafana (email, Slack, etc.).





Test alerting setup to ensure proper functioning.



# Best Practices

Regularly review and optimize dashboards and alerting rules.

Ensure monitoring covers key performance indicators (KPIs) for your application.

Implement automated monitoring checks as part of your CI/CD pipeline.

# Conclusion

Monitoring your CI/CD PHP web application with Grafana and Prometheus enhances visibility into your deployment pipeline, enabling proactive issue detection and resolution. Follow the steps outlined in this documentation to establish a robust monitoring infrastructure for your application.