

# Open Enterprise Server 2023

## Monitoring Server Health with Prometheus and Grafana

July 2023

## **Legal Notices**

### **Copyright 2023 Open Text**

The only warranties for products and services of Open Text and its affiliates and licensors ("Open Text") are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

---

# Contents

<b>About This Guide</b>	<b>5</b>
<b>1 Overview</b>	<b>7</b>
1.1 Introducing Prometheus and Grafana	7
1.2 Additional Information	7
<b>2 Prometheus with Grafana Architecture</b>	<b>9</b>
2.1 Integrating OES 2023 or iPrint 5 Appliance with Prometheus and Grafana on OES 2023 Server	10
<b>3 Installation Overview of a Prometheus (Monitoring) Service</b>	<b>13</b>
3.1 Requirements	13
3.2 Installing Prometheus on OES 2023 or SLES 15 SP4	13
3.3 Installing Grafana on the Prometheus (Monitoring) Server	14
3.3.1 Configuring the Grafana dashboard	15
<b>4 How to Integrate Metric Endpoints (targets)</b>	<b>17</b>
4.1 Installing Node Exporter	17
4.2 Installing cAdvisor in Docker Environments	18
4.3 Installing Alertmanager on the Prometheus (Monitoring) Server	18
4.3.1 Configuring the Alertmanager notification system	19
<b>5 Troubleshooting</b>	<b>23</b>
5.1 Adding Prometheus Data Source Fails	23
5.2 Servers not Appearing on the Grafana Dashboard	23
5.3 Data not Appearing on the Grafana Dashboard	23
<b>6 Frequently Asked Questions</b>	<b>25</b>
6.1 How to View Logs for Prometheus, Alertmanager, Grafana, and Node Exporter?	25
6.2 Is it Possible to Install Prometheus with Grafana in a Clustered Environment?	25
6.3 Can Target Nodes in Prometheus and Grafana have their services discovered automatically?	26
6.4 Is it Possible to Monitor OES services like NSS, NCP, and eDir with Prometheus and Grafana?	26
6.5 Is Prometheus Implemented as a Pattern in OES	26
6.6 Is it possible to install and run Prometheus and Grafana on a non-OES machine?	26
<b>A Scripts</b>	<b>27</b>
A.1 cadvisor.sh	27
A.2 alert_manager.sh	28
A.3 node_exporter.sh	30
A.4 prometheus.sh	32



# About This Guide

This guide provides information about monitoring server health with Prometheus and Grafana.

- ♦ [Chapter 1, “Overview,” on page 7](#)
- ♦ [Chapter 2, “Prometheus with Grafana Architecture,” on page 9](#)
- ♦ [Chapter 3, “Installation Overview of a Prometheus \(Monitoring\) Service,” on page 13](#)
- ♦ [Chapter 4, “How to Integrate Metric Endpoints \(targets\),” on page 17](#)
- ♦ [Chapter 5, “Troubleshooting,” on page 23](#)
- ♦ [Chapter 6, “Frequently Asked Questions,” on page 25](#)
- ♦ [Appendix A, “Scripts,” on page 27](#)

## Audience

This guide is intended for network administrators.

## Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to [Micro Focus Documentation Feedback \(https://www.microfocus.com/documentation/open-enterprise-server/2023/\)](https://www.microfocus.com/documentation/open-enterprise-server/2023/) and enter your comments there.

## Documentation Updates

For the most recent version of this document, visit the [OES 2023 documentation website \(https://www.microfocus.com/documentation/open-enterprise-server/2023/\)](https://www.microfocus.com/documentation/open-enterprise-server/2023/).

## Additional Documentation

For information about other OES services and file systems, see the [OES 2023 documentation website \(https://www.microfocus.com/documentation/open-enterprise-server/2023/\)](https://www.microfocus.com/documentation/open-enterprise-server/2023/).



# 1 Overview

Ganglia is not supported from SUSE Linux Enterprise Server 15 Service Pack 4 and it has been replaced with Grafana. With this change, how can we monitor Open Enterprise Server (OES) and iPrint Appliance? Using Prometheus and Grafana is recommended approach.

- ♦ [Section 1.1, “Introducing Prometheus and Grafana,” on page 7](#)
- ♦ [Section 1.2, “Additional Information,” on page 7](#)

## 1.1 Introducing Prometheus and Grafana

Prometheus is an open-source, metrics-based monitoring system. It collects and stores its metrics as time-series data, i.e., metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

Numerical measures are referred to as metrics. Time series refers to data that documents changes over time. Depending on the application, users may want to measure different things. It could be the number of active connections or active queries for a database, or it could be the request timings for a web server.

Metrics collects data from services and hosts by sending HTTP requests on metrics endpoints. It then stores the results in a time-series database and makes it available for analysis and alerting.

Though Prometheus includes an expression browser that can be used for ad-hoc queries, the best tool available is Grafana. Grafana fully integrates with Prometheus and can produce a wide variety of dashboards.

Grafana is an open-source software (OSS) that enables you to query, visualize, alert on, and explore your metrics, logs, and traces wherever they are stored. Grafana OSS provides you with tools to turn your time-series data into insightful graphs and visualizations.

Prometheus and Grafana offer precompiled binaries for multiple platforms (such as various Linux distributions and Windows) along with Docker images.

This document provides detailed information on how to install Prometheus and Grafana on OES 2023 or on SLES15 SP4, monitor OES Servers, various appliances and create dashboards.

## 1.2 Additional Information

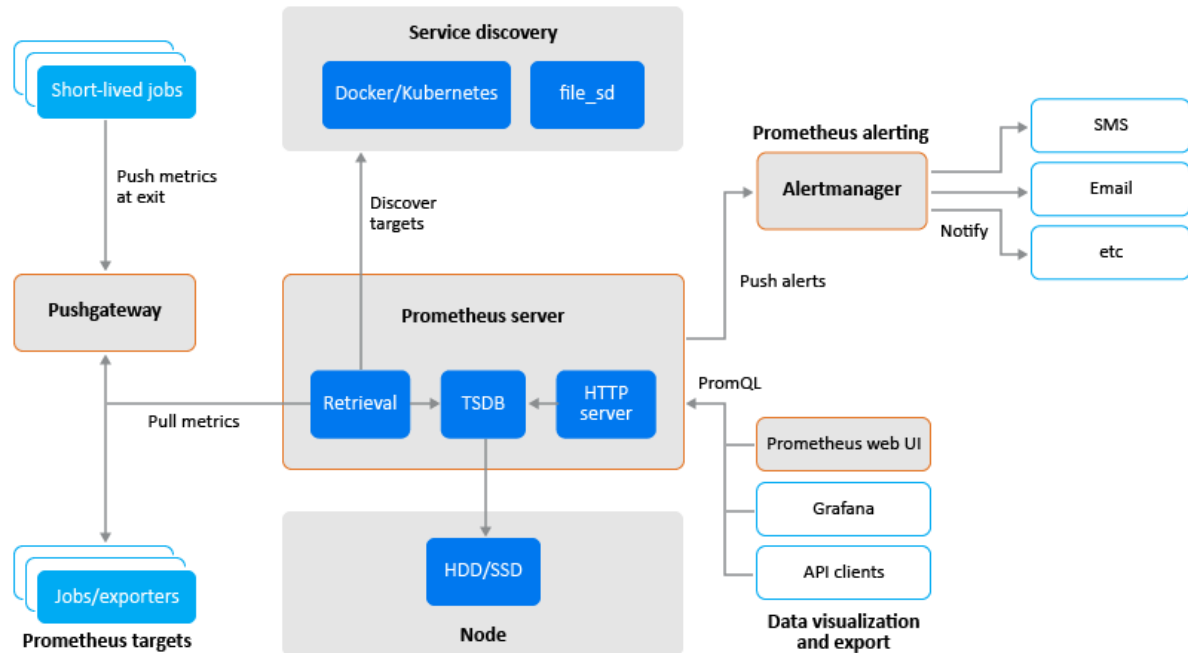
For detailed information about configuring and using Grafana and Prometheus to monitor your server and services, see the [Grafana Documentation \(https://grafana.com/\)](https://grafana.com/) and [Prometheus Documentation \(https://prometheus.io/docs\)](https://prometheus.io/docs).





# 2 Prometheus with Grafana Architecture

Figure 2-1 Prometheus with Grafana Architecture



The Prometheus ecosystem consists of multiple components, many of which are optional:

- ◆ the main [Prometheus server](#) which scrapes and stores time series data
- ◆ [client libraries](#) for instrumenting application code
- ◆ a [push gateway](#) for supporting short-lived jobs
- ◆ special-purpose [exporters](#) for services like HAProxy, StatsD, Graphite, etc.
- ◆ an [alertmanager](#) to handle alerts
- ◆ various support tools

Prometheus scrapes metrics from instrumented jobs, either directly or via an intermediary push gateway for short-lived jobs. It stores all scraped samples locally and runs rules over this data to either aggregate and record new time series from existing data or generate alerts. Grafana or other API consumers can be used to visualize the collected data.

Prometheus has standard exporters (like node exporter) available to export metrics. An exporter acts like a proxy between the application and Prometheus. It receives requests from the Prometheus server, collect data from the access and error logs of the application, transform it into the correct format, and then sends the data back to the Prometheus server. See, [Exporters and Integrations \(https://prometheus.io/docs/instrumenting/exporters/\)](https://prometheus.io/docs/instrumenting/exporters/) in Prometheus documentation.

The Prometheus Node Exporter exposes a wide variety of hardware- and kernel-related metrics. See, [Monitoring Linux Host Metrics with the Node Exporter \(https://prometheus.io/docs/guides/node-exporter/\)](https://prometheus.io/docs/guides/node-exporter/) in Prometheus documentation.

cAdvisor (container Advisor) is an open-source tool developed by Google to monitor containers. It can collect, aggregate, process, and export container-based metrics such as CPU and memory usage, filesystem, and network statistics. See, [cAdvisor \(https://prometheus.io/docs/guides/cadvisor/\)](https://prometheus.io/docs/guides/cadvisor/) in Prometheus documentation.

Prometheus collects data in the form of time series. The time series is built using a pull model at a specific polling frequency consisting of Prometheus server queries and exporters. Targets are discovered using service discovery or static configuration. Prometheus data is stored in the form of metrics, with each metric having a name that is used for referencing and querying it. Prometheus stores data locally on disk, which helps for fast data storage and fast querying but ability to store metrics in remote storage.

Prometheus has alert configuration options that allow you to specify a condition that must be met for a certain amount of time before triggering an alert. **Alertmanager** service receives notifications when the alerts are triggered.

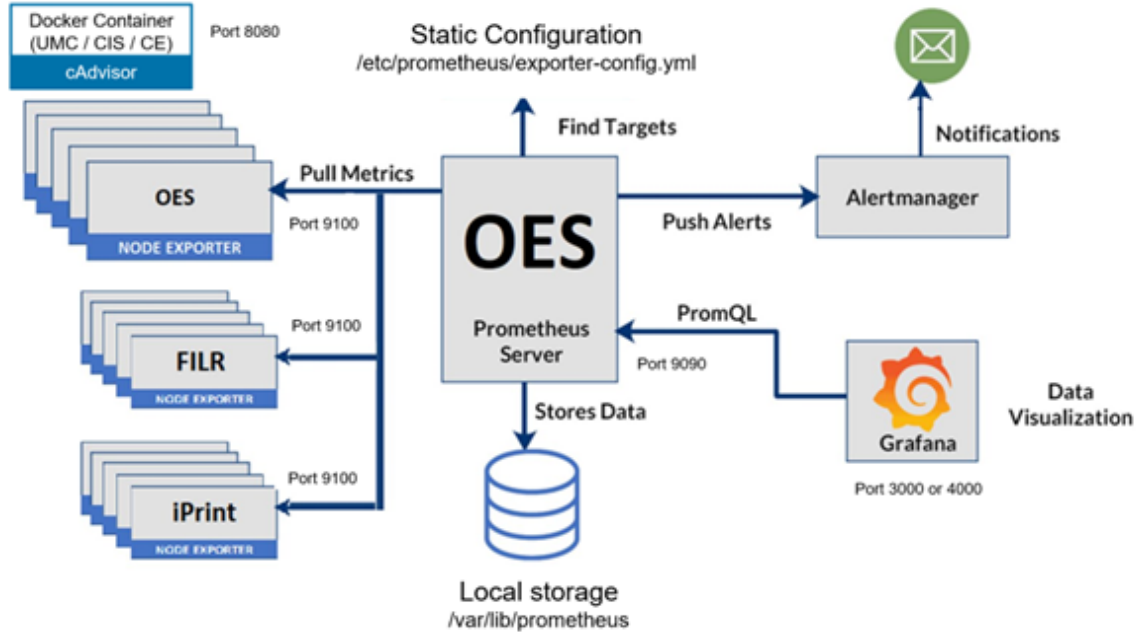
Each Prometheus server is standalone and does not rely on network storage or other external services.

- ♦ [Section 2.1, “Integrating OES 2023 or iPrint 5 Appliance with Prometheus and Grafana on OES 2023 Server,” on page 10](#)

## 2.1 Integrating OES 2023 or iPrint 5 Appliance with Prometheus and Grafana on OES 2023 Server

Install Prometheus on one OES 2023 or SLES 15 SP4 server which acts as the Prometheus server and install node exporter on the OES Server or iPrint Appliance to be monitored for collecting statistics like CPU, memory, and disk space utilization. Only when monitoring docker containers (CIS or UMC) cAdvisor is required.

Figure 2-2 Open Enterprise Server Integration



In the default Prometheus server configuration file, a job is added to scrape metrics from all the target servers. Prometheus can store pulled matrices in its default path, or it can use another volume to store the data.

**Alertmanager** and Grafana can co-exist on the same Prometheus server or on a standalone server. In the **alertmanager** configuration file, rules are added to trigger alerts based on specific conditions. Email is one such notification sent to the respective target when the condition is met.



# 3 Installation Overview of a Prometheus (Monitoring) Service

You must install Prometheus, Node Exporter, cAdvisor (CIS or UMC), and Grafana application on any OES or SLES server.

- ♦ [Section 3.1, “Requirements,” on page 13](#)
- ♦ [Section 3.2, “Installing Prometheus on OES 2023 or SLES 15 SP4,” on page 13](#)
- ♦ [Section 3.3, “Installing Grafana on the Prometheus \(Monitoring\) Server,” on page 14](#)

## 3.1 Requirements

- ♦ `prometheus-2.43.0.linux-amd64.tar.gz`
- ♦ `node_exporter-1.5.0.linux-amd64.tar.gz`
- ♦ Docker Image `google/cadvisor`
- ♦ `grafana-9.4.7-1.x86_64.rpm`
- ♦ `alertmanager-0.25.0.linux-amd64.tar.gz`
- ♦ `prometheus.sh`
- ♦ `node_exporter.sh`
- ♦ `cadvisor.sh`
- ♦ `alert_manager.sh`

## 3.2 Installing Prometheus on OES 2023 or SLES 15 SP4

Perform the following steps on an OES server:

- 1 Download and unzip the Prometheus file from Prometheus download site (<https://prometheus.io/download/>).

```
tar xfvz prometheus-2.43.0.linux-amd64.tar.gz
```

- 2 Copy the `prometheus.sh` script to the unzipped directory and run it. See, “[prometheus.sh](#)” on [page 32](#).

```
sh ./prometheus.sh
```

This script creates user, group, directories, and changes the ownership of the unzipped files and finally creates a service file to start Prometheus service.

- 3 Edit `/etc/prometheus/exporter-config.yml` file to add new metric endpoints (target).

---

**NOTE:** Use `static_configuration` to specify a list of targets and a common labels. It is the canonical way to specify static targets in a scrape configuration.

---

```

global:
  scrape_interval: 15s

scrape_configs:
  - job_name: Docker Servers
    static_configs:
      - targets: ['localhost:8080']
  - job_name: OES Servers
    static_configs:
      - targets: ['localhost:9100', 'oesnode01:9100', 'oesnode02:9100',
'oesnode03:9100', 'oesnode04.com']

  - job_name: 'alert-manager'
    static_configs:
      - targets: ['localhost:9093']

```

### 3.3 Installing Grafana on the Prometheus (Monitoring) Server

There are several ways to install Grafana, including as a Docker container. However, we will use the rpm-based setup because it is less complicated.

- 1 Download the Grafana RPM from the Grafana download site.

```
wget https://dl.grafana.com/oss/release/grafana-9.4.7-1.x86_64.rpm
```

- 2 Install Grafana.

```
rpm -i --nodeps grafana-9.4.7-1.x86_64.rpm
```

- 3 Modify the Grafana configuration file located at `/etc/grafana/grafana.ini` to set the default port.

```

##### Server #####;
[server]
# Protocol (http, https, h2, socket)
protocol=http

# The ip address to bind to, empty will bind to all interfaces
;http_addr =

# The http port to use
http_port = 4000

```

The edirapi docker image uses port 3000 by default, but if you install Grafana on an OES 2023 Server with UMC installed, it will instead utilize port 3000. In this instance, we switch the port to 4000 and this port must be added in the firewall allowed services.

- 4 Start the Grafana Service.

```
systemctl start grafana-server.service
```

### 3.3.1 Configuring the Grafana dashboard

#### Prerequisites:

Before configuring the Grafana dashboard for Node Exporter and cAdvisor ensure to integrate the Metric Endpoints. See, [Chapter 4, “How to Integrate Metric Endpoints \(targets\),”](#) on page 17.

- 1 Login to `http://<server_ip_of_grafana>:4000` with default user credentials (admin/admin).
- 2 Add the Prometheus data source to Grafana:
  - 2a Navigate to **Configuration > Datasource > Add datasource** in the Grafana sidebar.
  - 2b Select **Prometheus** as the type.
  - 2c Set the appropriate Prometheus server URL (for example, `http://localhost:9090/`).
  - 2d Adjust other data source settings as required (for example, choosing the right access method).
  - 2e Click **Save & Test** to save the new data source.

---

**NOTE:** In case of failure, ensure the service is up and running and that the correct port number has been added to the firewall.

---

- 3 Go to the [Grafana Community \(https://grafana.com/grafana/dashboards/\)](https://grafana.com/grafana/dashboards/) where you can find numerous ready-made dashboards that can be imported and utilized in your environment.

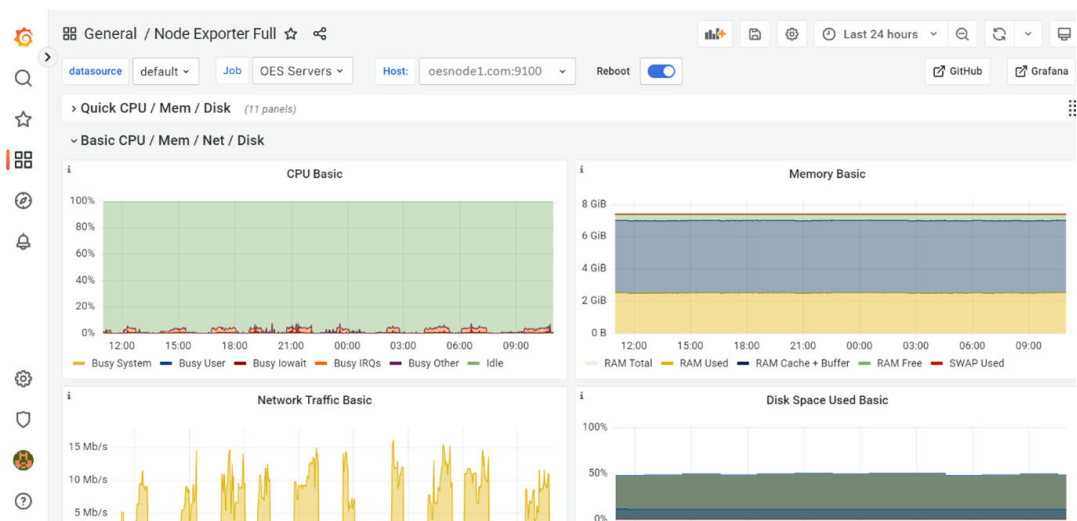
---

**NOTE:** Grafana dashboard provides different templates for cAdvisor and Node Exporter.

---

- 4 Select the required ready-made dashboards from and copy the ID of the dashboard.
- 5 Import the dashboard using the available template in Grafana labs.
  - 5a Navigate to **Home > Dashboard > +import** in the Grafana sidebar.
  - 5b In **Import via grafana.com** field, enter the copied ID of the ready-made dashboard, and click **Load**.

**Figure 3-1** Grafana Dashboard



- 5c From **Host** drop-down list, select the server for viewing the respective dashboard results.





# 4 How to Integrate Metric Endpoints (targets)

- ♦ Section 4.1, “Installing Node Exporter,” on page 17
- ♦ Section 4.2, “Installing cAdvisor in Docker Environments,” on page 18
- ♦ Section 4.3, “Installing Alertmanager on the Prometheus (Monitoring) Server,” on page 18

## 4.1 Installing Node Exporter

Node Exporters are deployed on the servers that need monitoring, such as OES Servers or iPrint Appliance.

- 1 Download and unzip the node exporter files from the [Prometheus website \(https://prometheus.io/download/\)](https://prometheus.io/download/).

```
tar xvfz node_exporter-1.5.0.linux-amd64.tar.gz
```

- 2 Copy `node_exporter.sh` script to unzipped directory and run the script. See, “`node_exporter.sh`” on page 30.

```
sh ./node_exporter.sh
```

On port 9190, node exporter receives the http requests from Prometheus.

- 3 After the installation of the node exporter on a target, update the static Prometheus server configuration and restart the Prometheus service.

- 3a On the Prometheus (Monitoring) server edit the Prometheus configuration file.

```
/etc/prometheus/exporter-config.yml.
```

- 3b Update the hostname or IP address of the node exporter in the targets section (highlighted in the example below).

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: Docker Servers
    static_configs:
      - targets: ['localhost:8080']
  - job_name: OES Servers
    static_configs:
      - targets: ['localhost:9100', 'oesnode01:9100', 'oesnode02:9100']
```

- 4 Restart the service after the configuration file is updated.

```
systemctl daemon-reload
systemctl restart prometheus.service
```

## 4.2 Installing cAdvisor in Docker Environments

cAdvisor monitors the dockerised endpoints information such as CPU, memory usage, filesystem, and network statistics.

Perform the following steps on the OES server (UMC or CIS):

- 1 Download the cAdvisor docker image from the docker hub.

```
docker pull google/cadvisor
```

- 2 Run the script to configure cAdvisor service. See, “[cadvisor.sh](#)” on page 27.

```
cadvisor.sh
```

cAdvisor metrics is available on port 8080 to view individual server metrics.

- 3 After the installation of the cAdvisor on a target, update the static Prometheus server configuration and restart the Prometheus service.

- 3a On the Prometheus (Monitoring) server edit the Prometheus configuration file.

```
/etc/prometheus/exporter-config.yml.
```

- 3b Update the IP address and port details of the cAdvisor in the targets section (highlighted in the example below).

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: Docker Servers
static_configs:
  - targets: ['localhost:8080'. '<cadvisor installed
server>:8080']
  - job_name: OES Servers
    static_configs:
      - targets:
        ['localhost:9100', 'oesnode01:9100', 'oesnode02:9100']
```

- 4 Restart the service after the configuration file is updated.

```
systemctl daemon-reload
systemctl restart prometheus.service
```

## 4.3 Installing Alertmanager on the Prometheus (Monitoring) Server

The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of routing them to the correct receiver through an integration such as email. See, [Alertmanager \(https://prometheus.io/docs/alerting/latest/alertmanager/\)](https://prometheus.io/docs/alerting/latest/alertmanager/) in Prometheus documentation.

Perform the following steps on an OES server where Prometheus is installed:

- 1 Download and unzip the Alertmanager files from the [Prometheus website \(https://prometheus.io/download/\)](https://prometheus.io/download/).

```
tar xvfz alertmanager-0.25.0.linux-amd64.tar.gz
```

- 2 Copy `alert_manager.sh` script to unzipped directory and run the script. See, [“alert\\_manager.sh” on page 28](#).

```
sh ./alert_manager.sh
```

- 3 After the installation of the Alertmanager on a target, update the static Prometheus server configuration and restart the Prometheus service.

- 3a On the Prometheus (Monitoring) server edit the Prometheus configuration file.

```
/etc/prometheus/exporter-config.yml.
```

- 3b Update the hostname or IP address of the Alertmanager in the targets section (highlighted in the example below).

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: Docker Servers
    static_configs:
      - targets: ['localhost:8080']
  - job_name: OES Servers
    static_configs:
      - targets: ['localhost:9100', 'oesnode01:9100',
'oesnode02:9100', 'oesnode03:9100', 'oesnode04.com']
  - job_name: 'alert-manager'
    static_configs:
      - targets: ['localhost:9093']
```

- 4 Restart the service after the configuration file is updated.

```
systemctl daemon-reload
systemctl restart prometheus.service
```

### 4.3.1 Configuring the Alertmanager notification system

Perform the following steps to configure Alertmanager notification system:

- 1 Enter the SMTP server information in the `/etc/alertmanager/alertmanager.yml` file.
- 2 Change the example information according to your requirements.

```

route:
  group_by: [Alertname]
  group_interval: 30s
  repeat_interval: 30s
  # Send all notifications to me.
  receiver: email-me
receivers:
- name: email-me
  email_configs:
  - send_resolved: true
    to: admin@email.com
    from: demo@email.com
    smarthost: smtp.email.com:587
    auth_username: demo@email.com
    auth_identity: demo@email.com
    auth_password: <enter_the_password>

```

### 3 Create a rule file named `prometheus_rules.yml` in the `/etc/prometheus` directory.

The example that follows will alert you if any node is unavailable for more than a minute or if there is less than 10% of its disk space left.

```

groups:
- name: custom_rules
  rules:
  - record: node_memory_MemFree_percent
    expr: 100 - (100 * node_memory_MemFree_bytes /
node_memory_MemTotal_bytes)

  - record: node_filesystem_free_percent
    expr: 100 * node_filesystem_free_bytes{mountpoint="/" } /
node_filesystem_size_bytes{mountpoint="/" }
- name: alert_rules
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "Instance [{{ $labels.instance }}] down"
      description: "[{{ $labels.instance }}] of job [{{ $labels.job
}}] has been down for more than 1 minute."
  - alert: DiskSpaceFree10Percent
    expr: node_filesystem_free_percent <= 10
    labels:
      severity: warning
    annotations:
      summary: "Instance [{{ $labels.instance }}] has 10% or less
Free disk space"
      description: "[{{ $labels.instance }}] has only {{ $value }}%
or less free."

```

For more information about Alertmanager, see Configuration and Alerting Rules in Prometheus documentation site.

- 4** Edit the configuration file (`/etc/prometheus/exporter-config.yml`) to include the rule global file and alerting configuration for the notification system (highlighted in the example below).

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: Docker Servers
    static_configs:
      - targets: ['localhost:8080']
  - job_name: OES Servers
    static_configs:
      - targets: ['localhost:9100', 'oesnode01:9100', 'oesnode02:9100',
'oesnode03:9100', 'oesnode04.com']

  - job_name: 'alert-manager'
    static_configs:
      - targets: ['localhost:9093']

rule_files:
  - "prometheus_rules.yml"

alerting:
  alertmanagers:
    - static_configs:
      - targets:
          # alertmanager:9093
          - localhost:9093
```

- 5** Restart the service after the configuration file is updated.

```
systemctl daemon-reload
systemctl restart prometheus.service
systemctl restart alertmanager.service
```



# 5 Troubleshooting

- ♦ [Section 5.1, “Adding Prometheus Data Source Fails,” on page 23](#)
- ♦ [Section 5.2, “Servers not Appearing on the Grafana Dashboard,” on page 23](#)
- ♦ [Section 5.3, “Data not Appearing on the Grafana Dashboard,” on page 23](#)

## 5.1 Adding Prometheus Data Source Fails

If adding Prometheus data source is failing, ensure to check the following two scenarios:

- ♦ **Verify that Prometheus's default port (9090) is accepted by the firewall.**

If the port is blocked, it can prevent Grafana from establishing a connection to Prometheus. Consult your system's documentation or network administrator for instructions on how to open ports in the firewall.

- ♦ **Verify that the Prometheus service is running on the specified server.**

Check the status of the Prometheus service to ensure the Prometheus service is up and running

```
journalctl -u prometheus.service
```

This command displays the logs related to the Prometheus service. Check for any errors or issues that may indicate why the service is not running correctly and fix them.

## 5.2 Servers not Appearing on the Grafana Dashboard

If the servers are not appearing in the host drop-down on the Grafana dashboard, ensure to check the following two scenarios:

- ♦ **Verify that the scrape targets (servers) are correctly configured in Prometheus**

Check the `scrape_configs` section in the Prometheus configuration file to ensure that the correct endpoints, ports, and protocols are specified for each target. Ensure that the scrape targets are actively exposing metrics in the expected format.

- ♦ **Validate metrics availability**

Verify that the targets for the scrape are actually exposing metrics for the Prometheus scraper. You can use tools like `curl` or Prometheus' expression browser to directly query the scrape target's metrics endpoint and verify if the expected metrics are returned.

## 5.3 Data not Appearing on the Grafana Dashboard

Grafana dashboard takes approximately 2-3 minutes to load the data when you create a dashboard for Node Exporter or cAdvisor because the scrape interval varies for the services.

You may notice this behavior, only when adding the dashboard for the first time.





# 6 Frequently Asked Questions

- ♦ [Section 6.1, “How to View Logs for Prometheus, Alertmanager, Grafana, and Node Exporter?”](#) on page 25
- ♦ [Section 6.2, “Is it Possible to Install Prometheus with Grafana in a Clustered Environment?”](#) on page 25
- ♦ [Section 6.3, “Can Target Nodes in Prometheus and Grafana have their services discovered automatically?”](#) on page 26
- ♦ [Section 6.4, “Is it Possible to Monitor OES services like NSS, NCP, and eDir with Prometheus and Grafana?”](#) on page 26
- ♦ [Section 6.5, “Is Prometheus Implemented as a Pattern in OES,”](#) on page 26
- ♦ [Section 6.6, “Is it possible to install and run Prometheus and Grafana on a non-OES machine?”](#) on page 26

## 6.1 How to View Logs for Prometheus, Alertmanager, Grafana, and Node Exporter?

You can view logs to debug any errors or warnings by using the following command:

```
journalctl -u <service_name>
```

## 6.2 Is it Possible to Install Prometheus with Grafana in a Clustered Environment?

In general, neither the Prometheus nor Grafana projects natively support running Prometheus and Grafana in a clustered environment. However, you can use other approaches and techniques, including load balancing, container orchestration systems, and other clustering techniques to achieve high availability and scalability for both tools.

For the most recent information and best practices regarding clustered installations of Prometheus and Grafana, it is recommended to consult the official documentation and community resources.

## 6.3 Can Target Nodes in Prometheus and Grafana have their services discovered automatically?

Yes, it is possible to automate the service discovery of target nodes in Prometheus and Grafana. With static configuration, the target nodes and the accompanying metrics endpoints are manually entered into a configuration file.

However, Prometheus provides additional service discovery methods that can be automated. These methods include file-based service discovery, DNS-based service discovery, and integration with popular orchestration systems like Kubernetes. These automated service discovery methods allow Prometheus to dynamically discover and monitor new nodes as they are added or removed from the environment.

## 6.4 Is it Possible to Monitor OES services like NSS, NCP, and eDir with Prometheus and Grafana?

Currently, Prometheus and Grafana can monitor the memory and CPU usage of OES systemd services using the Node Exporter.

Though it can provide metrics related to system-level resource usage, however, it does not have built-in support for directly monitoring specific OES services such as NSS, NCP, and eDirectory.

## 6.5 Is Prometheus Implemented as a Pattern in OES

No, Prometheus is not implemented as a pattern in OES. To use Prometheus with OES, you must download the Prometheus binaries from the official Prometheus website and configure it independently.

## 6.6 Is it possible to install and run Prometheus and Grafana on a non-OES machine?

Yes, you can install both Prometheus and Grafana and run on any Linux or SLES (SUSE Linux Enterprise Server) machines and they are not limited to only operating on OES.

# A Scripts

- [Section A.1, “cadvisor.sh,” on page 27](#)
- [Section A.2, “alert\\_manager.sh,” on page 28](#)
- [Section A.3, “node\\_exporter.sh,” on page 30](#)
- [Section A.4, “prometheus.sh,” on page 32](#)

## A.1 cadvisor.sh

```
#!/bin/sh

#Creating node_exporter service file
sudo touch /etc/systemd/system/cadvisor.service
cat <<EOF >/etc/systemd/system/cadvisor.service
[Unit]
Description=cadvisor
Wants=network-online.target
After=network-online.target

[Service]
ExecStart= /usr/bin/docker run --rm --name cadvisor -p 8080:8080 -v "/:/
rootfs:ro" -v "/var/run:/var/run:rw" -v "/sys:/sys:ro" -v "/var/lib/docker/
:/var/lib/docker:ro" google/cadvisor:latest

[Install]
WantedBy=multi-user.target

EOF

echo "cAdvisor service is created"

#Adding grafana port to firewall
cat /etc/novell-release | grep "VERSION = 2023"
if [ $? -eq 0 ]; then
    DEFAULT_ZONE=`firewall-cmd --get-default-zone`
    for port in 8080; do
        firewall-cmd --permanent --zone=$DEFAULT_ZONE --add-
port=${port}/tcp > /dev/null 2>&1
    done
    firewall-cmd --add-masquerade --permanent --zone=$DEFAULT_ZONE > /dev/
null 2>&1
    firewall-cmd --reload > /dev/null 2>&1
else
    sed -i 's/FW_SERVICES_EXT_TCP="/FW_SERVICES_EXT_TCP="8080 /g' /etc/
sysconfig/SuSEfirewall2
```

```

    systemctl restart SuSEfirewall2.service
fi

sleep 5
#After creating and modifying service file need to restart daemom
sudo systemctl daemon-reload
sudo systemctl start cadvisor.service
sleep 5

systemctl is-active --quiet cadvisor.service
if [ $? -eq 0 ]; then
    echo cadvisor service is running
else
    echo cadvisor service is not running
fi

exit

```

## A.2 alert\_manager.sh

```

#!/bin/sh

#Creating the alertmanager directory to keep configuration files

DIR="/etc/alertmanager"
if [ -d "$DIR" ]; then
    ### Take action if $DIR exists ###
    echo "Directory already exists"
else
    ### Control will jump here if $DIR does NOT exists ###
    sudo mkdir /etc/alertmanager
    echo "${DIR} Created"
fi

DIR="/usr/local/bin/alertmanager"
if [ -d "$DIR" ]; then
    ### Take action if $DIR exists ###
    echo "Directory already exists"
else
    ### Control will jump here if $DIR does NOT exists ###
    sudo mkdir /usr/local/bin/alertmanager
    echo "${DIR} Created"
fi

#copy prometheus binaries to the following location and assign the
perimission
sudo cp alertmanager /usr/local/bin/alertmanager
sudo cp amtool /usr/local/bin/alertmanager
echo "Files are copied to target location"

#Creating alertmanager configuration file
sudo touch /etc/alertmanager/alertmanager.yml
cat <<EOF >/etc/alertmanager/alertmanager.yml
route:

```

```

    group_by: [Alertname]
    group_interval: 30s
    repeat_interval: 30s
    # Send all notifications to me.
    receiver: email-me
receivers:
- name: email-me
  email_configs:
  - send_resolved: true
    to: test@example.com
    from: alertmanager@example.com
    smarthost: smtp.gmail.com:587
    auth_username: "alertmanager@example.com"
    auth_identity: "alertmanager@example.com"
    auth_password: "*****"

EOF
echo "Configuration file is created"

#Creating the alertmanager service file
sudo touch /etc/systemd/system/alertmanager.service

cat <<EOF >/etc/systemd/system/alertmanager.service
[Unit]
Description=Prometheus Alert Manager Service
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/alertmanager/alertmanager --config.file=/etc/
alertmanager/alertmanager.yml
[Install]
WantedBy=multi-user.target
EOF

echo "Alert manager service is created"

#Adding alertmanager port to firewall
cat /etc/novell-release | grep "VERSION = 2023"
if [ $? -eq 0 ]; then
    DEFAULT_ZONE=`firewall-cmd --get-default-zone`
    for port in 9093; do
        firewall-cmd --permanent --zone=$DEFAULT_ZONE --add-
port=${port}/tcp > /dev/null 2>&1
    done
    firewall-cmd --add-masquerade --permanent --zone=$DEFAULT_ZONE > /dev/
null 2>&1
    firewall-cmd --reload > /dev/null 2>&1

else
    sed -i 's/FW_SERVICES_EXT_TCP="/FW_SERVICES_EXT_TCP="9093 /g' /etc/
sysconfig/SuSEfirewall2
    systemctl restart SuSEfirewall2.service
fi

```

```

sleep 5

#After creating and modifying service file need to restart daemom
sudo systemctl daemon-reload
sudo systemctl start alertmanager
sleep 5

#Checking the status of alertmanager service
systemctl is-active --quiet alertmanager.service
if [ $? -eq 0 ]; then
    echo Alert Manager service is running
else
    echo Alert Manager service is not running
fi

exit

```

## A.3 node\_exporter.sh

```

#!/bin/bash

# Checking for prometheus user
USERID="$1"

if grep -q "^prometheus:" /etc/passwd; then
    echo "User $USERID exists in /etc/passwd"
else
    echo "User $USERID does not exist in /etc/passwd"
    sudo useradd --no-create-home --shell /bin/false prometheus
    echo "User $USERID created"
fi

# Checking for prometheus group
if grep -q "^prometheus:" /etc/group; then
    echo "Group $USERID exists in /etc/group"
else
    echo "Group $USERID does not exist in /etc/group"
    sudo groupadd prometheus
    echo "Group $USERID created"
fi

# Creating the node_exporter directory to keep binary files
DIR="/etc/nodeexporter"

if [[ -d "$DIR" ]]; then
    echo "Directory already exists: $DIR"
else
    sudo mkdir "$DIR"
    echo "Directory created: $DIR"
fi

# Copying binary to /etc/nodeexporter
if sudo cp node_exporter "$DIR"; then

```

```

    echo "Copy of node exporter binary successful"
else
    echo "Copy of node exporter binary unsuccessful"
fi

# Creating node_exporter service file
SERVICE_FILE="/etc/systemd/system/node_exporter.service"

sudo tee "$SERVICE_FILE" > /dev/null <<EOF
[Unit]
Description=node_exporter
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/etc/nodeexporter/node_exporter

[Install]
WantedBy=multi-user.target
EOF

echo "node_exporter service file created: $SERVICE_FILE"

# Adding prometheus port to firewall
if grep -q "VERSION = 2023" /etc/novell-release; then
    DEFAULT_ZONE=$(firewall-cmd --get-default-zone)
    for port in 9100; do
        sudo firewall-cmd --permanent --zone="$DEFAULT_ZONE" --add-
port="{port}/tcp" > /dev/null
    done
    sudo firewall-cmd --add-masquerade --permanent --zone="$DEFAULT_ZONE" >
/dev/null
    sudo firewall-cmd --reload > /dev/null
else
    sudo sed -i 's/FW_SERVICES_EXT_TCP="/FW_SERVICES_EXT_TCP="9100 /g' /etc/
sysconfig/SuSEfirewall2
    sudo systemctl restart SuSEfirewall2.service
fi

```

```

sleep 5

# Restarting node_exporter service
sudo systemctl daemon-reload
sudo systemctl start node_exporter.service
sleep 5

if sudo systemctl is-active --quiet node_exporter.service; then
    echo "Node exporter service is running"
else
    echo "Node exporter service is not running"
fi

exit

```

## A.4 prometheus.sh

```

#!/bin/bash

# Define constants
PROMETHEUS_USER="prometheus"
PROMETHEUS_GROUP="prometheus"
PROMETHEUS_BIN_PATH="/usr/local/bin"
PROMETHEUS_CONF_DIR="/etc/prometheus"
PROMETHEUS_DATA_DIR="/var/lib/prometheus"
PROMETHEUS_CONFIG_FILE="$PROMETHEUS_CONF_DIR/exporter-config.yml"
PROMETHEUS_SERVICE_FILE="/etc/systemd/system/prometheus.service"

# Check if user exists and create if not
if ! getent passwd $PROMETHEUS_USER > /dev/null 2>&1; then
    sudo useradd --no-create-home --shell /bin/false $PROMETHEUS_USER
    echo "User $PROMETHEUS_USER created"
else
    echo "User $PROMETHEUS_USER already exists"
fi

# Check if group exists and create if not
if ! getent group $PROMETHEUS_GROUP > /dev/null 2>&1; then
    sudo groupadd $PROMETHEUS_GROUP
    echo "Group $PROMETHEUS_GROUP created"
else
    echo "Group $PROMETHEUS_GROUP already exists"
fi

# Create required directories and assign ownership
for DIR in $PROMETHEUS_CONF_DIR $PROMETHEUS_DATA_DIR; do
    if [ -d "$DIR" ]; then
        echo "Directory $DIR already exists"
    else
        sudo mkdir -p "$DIR"
        echo "Directory $DIR created"
    fi
    sudo chown -R $PROMETHEUS_USER:$PROMETHEUS_GROUP "$DIR"
    echo "Ownership set for $DIR"

```



```

done

# Copy Prometheus binaries to the specified location and assign ownership
for FILE in prometheus promtool; do
    sudo cp "$FILE" "$PROMETHEUS_BIN_PATH/"
    sudo chown $PROMETHEUS_USER:$PROMETHEUS_GROUP "$PROMETHEUS_BIN_PATH/
$FILE"
done
echo "Files copied to $PROMETHEUS_BIN_PATH"

# Copy console and console library files to the Prometheus configuration
directory and assign ownership
for DIR in consoles console_libraries; do
    sudo cp -r "$DIR" "$PROMETHEUS_CONF_DIR/"
    sudo chown -R $PROMETHEUS_USER:$PROMETHEUS_GROUP "$PROMETHEUS_CONF_DIR/
$DIR"
done
echo "Consoles and console libraries copied to $PROMETHEUS_CONF_DIR"

# Create Prometheus configuration file and assign ownership
cat <<EOF > "$PROMETHEUS_CONFIG_FILE"
global:
    scrape_interval: 15s

scrape_configs:
  - job_name: Docker Servers
    static_configs:
      - targets: ['localhost:8080']
  - job_name: OES Servers
    static_configs:
      - targets: ['localhost:9100']
EOF
sudo chown $PROMETHEUS_USER:$PROMETHEUS_GROUP "$PROMETHEUS_CONFIG_FILE"
echo "Configuration file created at $PROMETHEUS_CONFIG_FILE"

# Create Prometheus service file
cat <<EOF > "$PROMETHEUS_SERVICE_FILE"
[Unit]
Description=Prometheus
Wants=network-online.target ndsd.service
After=network-online.target ndsd.service

[Service]
User=$PROMETHEUS_USER
Group=$PROMETHEUS_GROUP
Type=simple
ExecStart=/usr/local/bin/prometheus \
    --config.file /etc/prometheus/exporter-config.yml \
    --storage.tsdb.path /var/lib/prometheus/ \
    --web.console.templates=/etc/prometheus/consoles \
    --web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
EOF

```

```

echo "Prometheus service is created"

#Adding prometheus port to firewall
cat /etc/novell-release | grep "VERSION = 2023"
if [ $? -eq 0 ]; then
    DEFAULT_ZONE=`firewall-cmd --get-default-zone`
    for port in 9090; do
        firewall-cmd --permanent --zone=$DEFAULT_ZONE --add-
port=${port}/tcp > /dev/null 2>&1
    done
    firewall-cmd --add-masquerade --permanent --zone=$DEFAULT_ZONE > /dev/
null 2>&1
    firewall-cmd --reload > /dev/null 2>&1

else
    sed -i 's/FW_SERVICES_EXT_TCP="/FW_SERVICES_EXT_TCP="9090 /g' /etc/
sysconfig/SuSEfirewall2
    systemctl restart SuSEfirewall2.service
fi

sleep 5

#After creating and modifying service file need to restart daemom
sudo systemctl daemon-reload
sudo systemctl start prometheus
sleep 5

#Checking the status of prometheus service
systemctl is-active --quiet prometheus.service
if [ $? -eq 0 ]; then
    echo Pormetheus service is running
else
    echo Pormetheus service is not running
fi

exit

```