opentext™

OpenText Filr 2023

Monitoring Server Health with Prometheus and Grafana

January 2023

Legal Notices

Contents

About This Guide

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

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/filr/) and enter your comments there.

Documentation Updates

For the most recent version of this document, visit the Filr documentation website.

Additional Documentation

For information about other Filr services, see the Filr 2023 documentation website.

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 Filr Appliance? Using Prometheus and Grafana is recommended approach.

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.

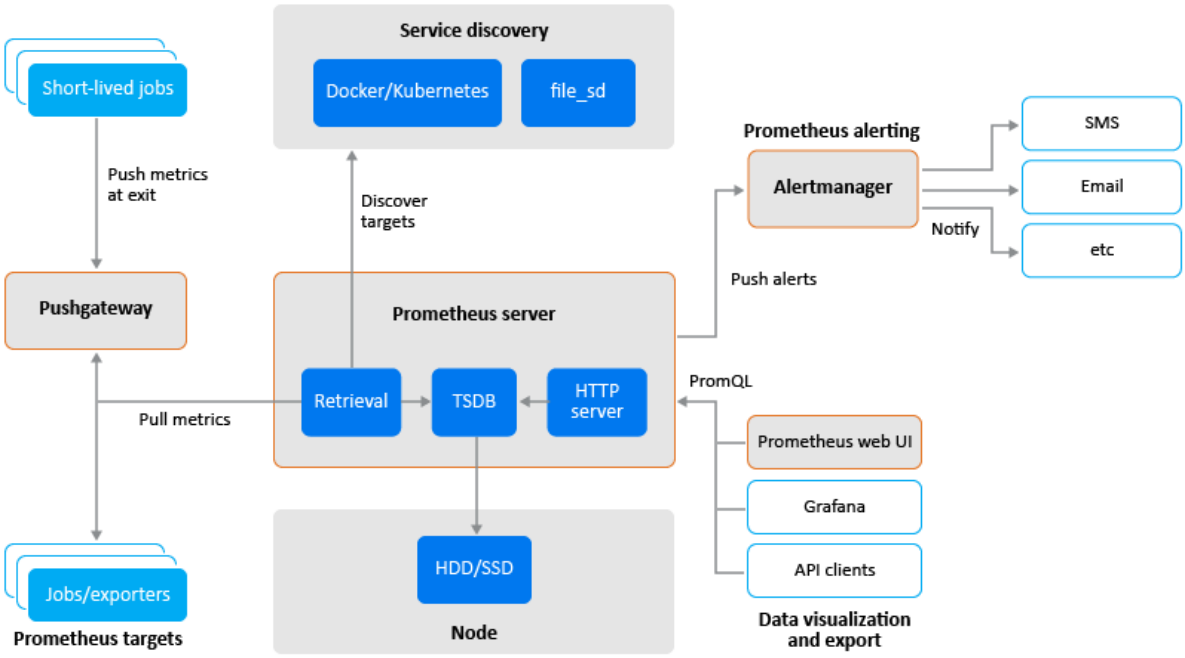
This document provides detailed information on how to install Prometheus and Grafana on Filr 24.1 or on SLES15 SP4, monitor Filr Servers, various appliances and create dashboards.

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/) and *Prometheus Documentation* (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/)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/) 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.

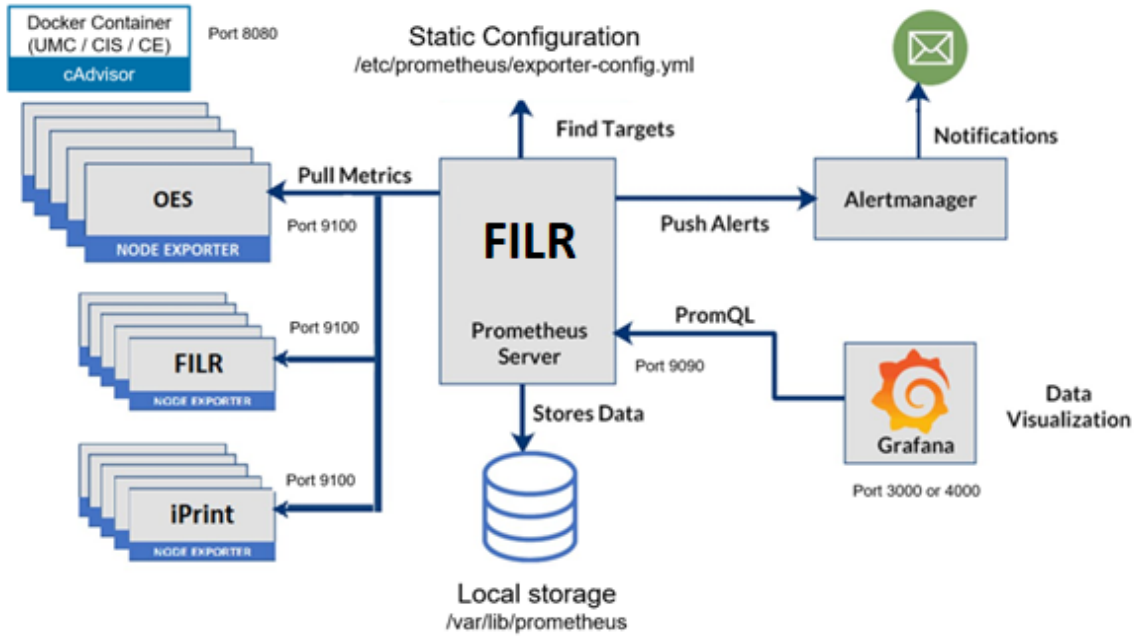
Each Prometheus server is standalone and does not rely on network storage or other external services.

- Section 2.1, “Integrating Filr Apliances with Prometheus and Grafana on Filr 24.1 Server,” on page 10

2.1 Integrating Filr Apliances with Prometheus and Grafana on Filr 24.1 Server

Install Prometheus on one Filr 24.1or SLES 15 SP4 server which acts as the Prometheus server and install node exporter on the Filr Appliance to be monitored for collecting statistics like CPU, memory, and disk space utilization.

Figure 2-2 *Filr 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 of a Prometheus (Monitoring) Service

You must install Prometheus, Node Exporter, and Grafana application on any Filr or SLES server.

3.1 Requirements

- `prometheus-2.43.0.linux-amd64.tar.gz`
- `node_exporter-1.5.0.linux-amd64.tar.gz`
- `grafana-9.4.7-1.x86_64.rpm`
- `prometheus.sh`
- `node_exporter.sh`

3.2 Installing Prometheus on Filr 24.1 or SLES 15 SP4

Perform the following steps on an Filr 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 25.

   ```
   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: Filr Servers
    static_configs:
      - targets: ['localhost:9100', 'filrnode01:9100', 'filrnode02:9100',
'filrnode03:9100', 'filrnode04.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 = 3000
   ```

   NOTE: To access with https, you can follow this steps mentioned here.

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 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>*:3000 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:9091/).
   - **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/) where you can find numerous ready-made dashboards that can be imported and utilized in your environment.

   NOTE: Grafana dashboard provides different templates for 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, we recommend 1860 and click **Load**.

   Figure 3-1 *Grafana Dashboard*

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

NOTE: Logs and dashboard data are getting stored in /var partition.

5d So, recommended to have enough storage size.

NOTE: Logs and dashboard data are being stored in /var partition. Hence,it is recommended to have enough storage size.

4 How to Integrate Metric Endpoints (targets)

4.1 Installing Node Exporter

Node Exporters are deployed on the servers that need monitoring, such as Filr Servers.

1 Download and unzip the node exporter files from the Prometheus website (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 23.

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

On port 9100, 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: Filr Servers
     static_configs:
        - targets:['localhost:9100', 'filrnode01:9100',
'filrnode02:9100']
```

4 Restart the service after the configuration file is updated.

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

5 Troubleshooting

5.1 Adding Prometheus Data Source Fails

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

- **Verify that Prometheus port (9091) is accepted by the firewall.**

  The default port 9090 is used for Fir Jetty Service, so we recomend to use port 9091.

  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 3-4 minutes to load the data when you create a dashboard for Node Exporter 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

6.1 How to View Logs for Prometheus, 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 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.

6.3 Is it Possible to Monitor Filr services like FAMT, Java, and Tomcat with Prometheus and Grafana?

Currently, Prometheus and Grafana can monitor the memory and CPU usage of Filr 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 Filr services such as FAMT, Java, and Tomcat .

6.4 Is it possible to install and run Prometheus and Grafana on a non-Filr machine?

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

A Scripts

A.1 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 "23.4" /etc/Novell-VA-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 firewalld.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.2 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: Filr 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 \
    --web.listen-address=:9091

[Install]
WantedBy=multi-user.target
EOF

echo "Prometheus service is created"

#Adding prometheus port to firewall
cat /etc/Novell-VA-release | grep "23.4"
if [ $? -eq 0 ]; then
   DEFAULT_ZONE=`firewall-cmd --get-default-zone`
   for port in 9091; do
           firewall-cmd --permanent --zone=$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="9091 /g' /etc/
sysconfig/firewalld
   systemctl restart firewalld.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
```