comment-installer-prometheus-et-node-exporter-sur-debian-12

Prometheus is an open-source monitoring and alerting platform. Originally, Prometheus was created by Soundcloud in 2012. Since then, the Prometheus project adopted by some famous companies abe become a bigger project with very active developers and community. In 2016, the Prometheus project was graduated under the Cloud Native Computing Foundation (CNCF).

This guide will show you how to install Prometheus and Node Exporter on the Debian 12 server.

Prerequisites

To get started, ensure you have the following:

- A Debian 12 server.
- A non-root user with sudo administrator privileges.

Basic Concepts You Must Know

Basically, Prometheus collects data and metrics through HTTP endpoints from target servers and then stores all data as times series. In Prometheus, time-series data are identified by metric name and key/value pairs.

Prometheus provides flexibility through the Prometheus Query Language (PromQL). You can use PromQL to query the Prometheus time-series database.

On the target servers, you must install the 'exporter' application that exposes all data and metrics to Prometheus. 'Node Exporter' is a commonly used exporter to monitor Linux machines.

Node exporter exposes hardware and kernel-related matrics from Linux machines. It comes as a single binary file that will expose data end metrics to the Prometheus server.

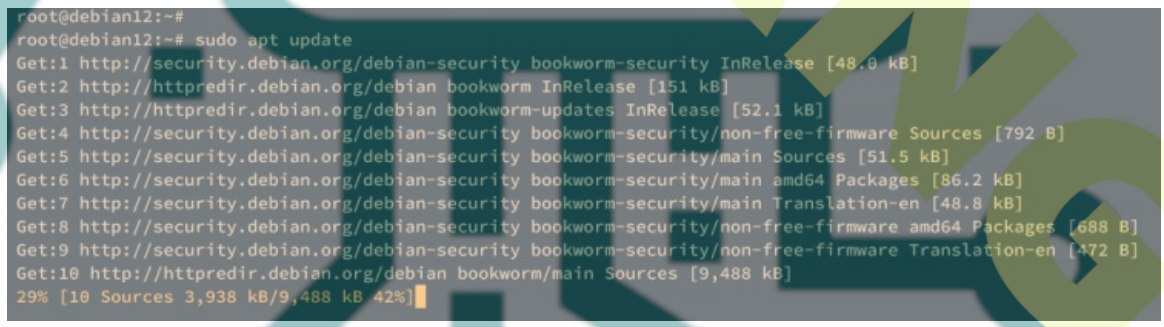
Installing Prometheus and Node Exporter

Prometheus is an open-source monitoring system for collecting systems metrics. Prometheus uses an exporter for collecting system metrics, such as Node Exporter which allows you to collect metrics for your system.

On the latest Debian version, both Prometheus and Node Exporter are available on the repository and can be installed using the APT package manager.

First, update and refresh your Debian package index by executing the following command.

```
sudo apt update
```



Now install Prometheus and Node Exporter via the apt command below. The Debian repository provides Prometheus **2.42.0** and Node Exporter **1.5.0**.

```
sudo apt install prometheus prometheus-node-exporter
```

Type Y to proceed with the installation.

```
root@debian12:~#
root@debian12:~# sudo apt install prometheus prometheus-node-exporter
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fonts-glyphicons-halflings freeipmi-common ipmitool javascript-common jq libfreei
  libjs-bootstrap4 libjs-d3 libjs-eonasdan-bootstrap-datetimepicker libjs-jquery li
  libjs-mustache libjs-popper.js libjs-rickshaw libjs-sizzle libncurses6 libnvme1 l
  libsnmp-base libsnmp40 libtime-duration-perl libtimedate-perl moreutils node-jque
  smartmontools uuid-runtime
Suggested packages:
  freeipmi-tools apache2 | lighttpd | httpd lm-sensors snmp-mibs-downloader gsmartc
The following NEW packages will be installed:
  fonts-glyphicons-halflings freeipmi-common ipmitool javascript-common jq libfreei
  libjs-bootstrap4 libjs-d3 libjs-eonasdan-bootstrap-datetimepicker libjs-jquery li
  libjs-mustache libjs-popper.js libjs-rickshaw libjs-sizzle libncurses6 libnvme1 l
  libsnmp-base libsnmp40 libtime-duration-perl libtimedate-perl moreutils node-jque
  prometheus-node-exporter-collectors smartmontools uuid-runtime
0 upgraded, 40 newly installed, 0 to remove and 55 not upgraded.
Need to get 36.1 MB of archives.
After this operation, 372 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

After installation is finished, both Prometheus and Node Exporter will be running as a systemd service and enabled automatically. The prometheus will have the service prometheus, and the Node Exporter will have the service prometheus-node-exporter.

Verify the prometheus service using the following command.

```
sudo systemctl is-enabled prometheus
sudo systemctl status prometheus
```

The displayed output below reveals that the prometheus service is running and enabled. And by default, it is running on port **9090**.

```
root@debian12:~#
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled prometheus
enabled
root@debian12:~# sudo systemctl status prometheus
● prometheus.service - Monitoring system and time series database
     Loaded: loaded (/lib/systemd/system/prometheus.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://prometheus.io/docs/introduction/overview/
             man:prometheus(1)
   Main PID: 3727 (prometheus)
      Tasks: 9 (limit: 4642)
     Memory: 24.3M
        CPU: 551ms
     CGroup: /system.slice/prometheus.service
             └─3727 /usr/bin/prometheus
```

Now verify the prometheus-node-exporter service by executing the following command.

```
sudo systemctl is-enabled prometheus-node-exporter
sudo systemctl status prometheus-node-exporter
```

The displayed output will be similar, which reveals the prometheus-node-exporter service is running and enabled. The default port for Node Exporter is **9100**.

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled prometheus-node-exporter
enabled
root@debian12:~# sudo systemctl status prometheus-node-exporter
● prometheus-node-exporter.service - Prometheus exporter for machine metrics
     Loaded: loaded (/lib/systemd/system/prometheus-node-exporter.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://github.com/prometheus/node_exporter
   Main PID: 2887 (prometheus-node)
      Tasks: 6 (limit: 4642)
     Memory: 17.1M
        CPU: 1.006s
     CGroup: /system.slice/prometheus-node-exporter.service
             └─2887 /usr/bin/prometheus-node-exporter
```

Next, run the prometheus command below to verify the Prometheus version.

```
prometheus --version
```

Then, check the binary path of promtool and its version like this. The promtool is a command line for managing the Prometheus monitoring system.

```
which promtool
promtool --version
```

Based on the displayed output, Prometheus and promtool **2.42.0** are installed.

```
root@debian12:~#
root@debian12:~# prometheus --version
prometheus, version 2.42.0+ds (branch: debian/sid, revision: 2.42.0+ds-5+b5)
  build user:       team+pkg-go@tracker.debian.org
  build date:       20230518-08:49:35
  go version:       go1.19.8
  platform:         linux/amd64
root@debian12:~#
root@debian12:~# which promtool
/usr/bin/promtool
root@debian12:~#
root@debian12:~# promtool --version
promtool, version 2.42.0+ds (branch: debian/sid, revision: 2.42.0+ds-5+b5)
  build user:       team+pkg-go@tracker.debian.org
  build date:       20230518-08:49:35
  go version:       go1.19.8
  platform:         linux/amd64
root@debian12:~#
```

Lastly, verify the Node Exporter binary path prometheus-node-exporter and its version using the following command.

```
which prometheus-node-exporter
prometheus-node-exporter --version
```

The displayed output below confirms that Node Exporter **1.5** is installed.

```
root@debian12:~#
root@debian12:~# which prometheus-node-exporter
/usr/bin/prometheus-node-exporter
root@debian12:~#
root@debian12:~# prometheus-node-exporter --version
node_exporter, version 1.5.0 (branch: debian/sid, revision: 1.5.0-1+b6)
  build user:       team+pkg-go@tracker.debian.org
  build date:       20230409-10:11:09
  go version:       go1.19.8
  platform:         linux/amd64
root@debian12:~#
```

Configuring Prometheus

After installing Prometheus and Node Exporter, the next step you will configure Prometheus by editing the default configuration file *prometheus.yml* which is located in the prometheus configuration directory */etc/prometheus*.

Open the default Prometheus configuration */etc/prometheus/prometheus.yml* using the following nano editor command.

```
sudo nano /etc/prometheus/prometheus.yml
```

Within the **scrape_configs** section, add a new job **prometheus** with the target endpoint **192.168.10.15:9090**, which is the Prometheus server itself.

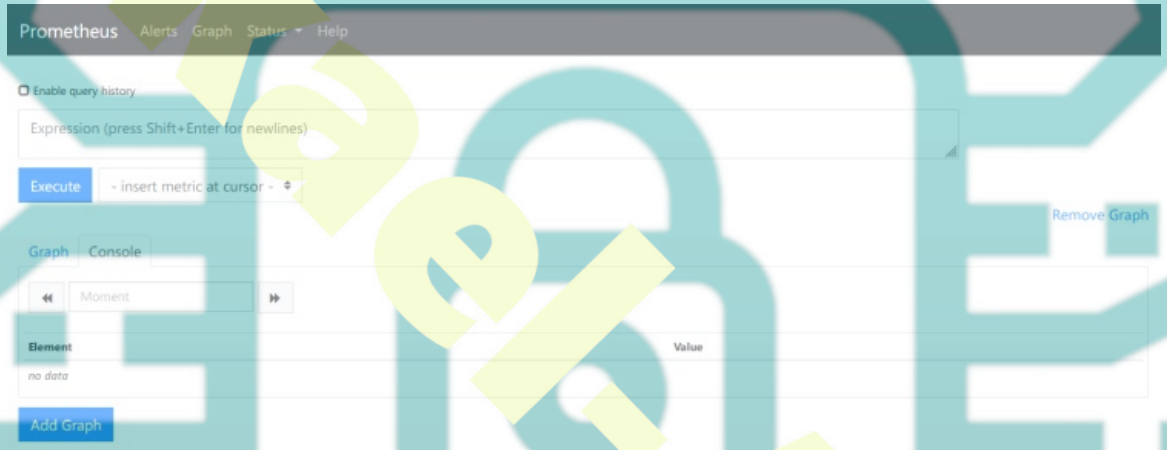
```
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["192.168.10.15:9090"]
```

Save the file and exit the editor when you're finished.

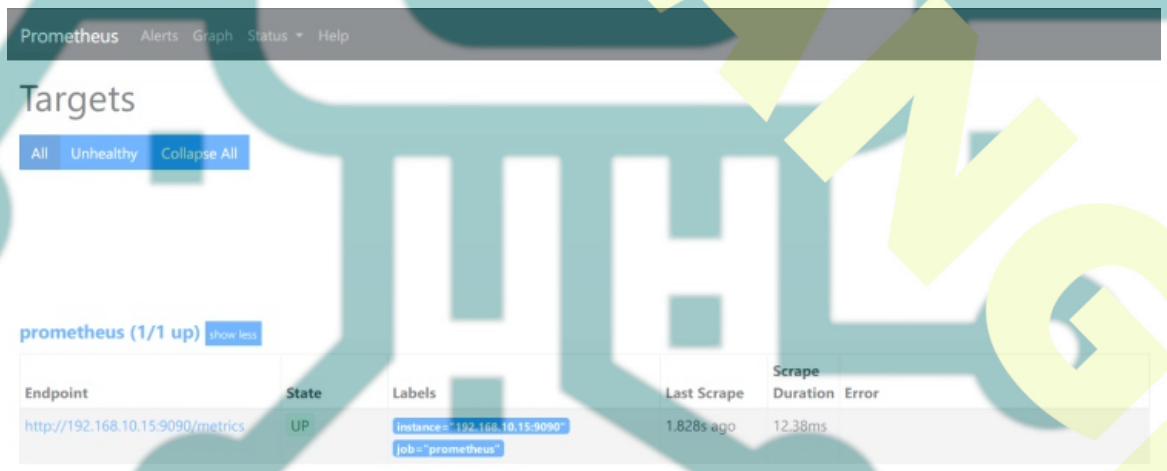
Now run the systemctl command below to restart the prometheus service and apply the changes you've made.

```
sudo systemctl restart prometheus
```

Now launch your web browser and visit your Prometheus installation, such as http://192.168.10.15:9090. If everything is working correctly, you should see the Prometheus dashboard in the following command.



Click on the menu **Status** > **Targets** to get the list of endpoints of the target monitoring system. You should see the endpoint prometheus with the status Up.



Lastly, you can also verify the Prometheus metrics by visiting the path URL */metrics*, such as http://192.168.10.15:9090/metrics. You should see similar metrics data generated by Prometheus.

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.4827e-05
go_gc_duration_seconds{quantile="0.25"} 0.000103186
go_gc_duration_seconds{quantile="0.5"} 0.000214654
go_gc_duration_seconds{quantile="0.75"} 0.000616367
go_gc_duration_seconds{quantile="1"} 0.000853543
go_gc_duration_seconds_sum 0.003714829
go_gc_duration_seconds_count 12
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 40
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.19.8"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.7384768e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 9.430704e+07
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.474815e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 546892
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 9.91544e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 2.7384768e+07
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.929216e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 3.018752e+07
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 217651
```

Adding Node Exporter to Prometheus

Now that you've configured Prometheus, the next step is to add Node Exporter to your Prometheus installation. The Node Exporter will gather metrics from your server. To achieve that, you must complete the following:

- Checking Node Exporter Status: This will ensure that Node Exporter is running before going further.
- Adding New Job to Prometheus: This will show you how to add a target monitoring system to Prometheus.

Now let's begin.

Checking Node Exporter Status

Before adding Node Exporter to Prometheus, you must ensure that Node Exporter is running without any errors. This can be achieved by checking the prometheus-node-exporter service status, checking the port 9100 that is used by Node Exporter, and accessing the Node Exporter metrics from your browser.

Check the *prometheus-node-exporter* service status by executing the following command.

```
sudo systemctl status prometheus-node-exporter
```

If running, the *prometheus-node-exporter* service status should be like this:

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled prometheus-node-exporter
enabled
root@debian12:~# sudo systemctl status prometheus-node-exporter
● prometheus-node-exporter.service - Prometheus exporter for machine metrics
     Loaded: loaded (/lib/systemd/system/prometheus-node-exporter.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: https://github.com/prometheus/node_exporter
   Main PID: 2887 (prometheus-node)
      Tasks: 6 (limit: 4642)
     Memory: 17.1M
        CPU: 1.006s
     CGroup: /system.slice/prometheus-node-exporter.service
             └─2887 /usr/bin/prometheus-node-exporter
```

Now run the command below to ensure port **9100** is in the **LISTEN** state, which the Node Exporter uses.

```
ss -tulpn | grep 9100
```

The following output confirms that Node Exporter uses port 9100.

```
root@debian12:~#
root@debian12:~# ss -tulpn | grep 9100
tcp   LISTEN 0      4096               *:9100            *:*    users:(("prometheus-node",pid=2887,fd=3))
root@debian12:~#
root@debian12:~#
```

Lastly, open your web browser and visit the Node Exporter metrics URL, such as http://192.168.10.15:9100/metrics. You should see the generated metrics by Node Exporter like the following:

```
# HELP apt_autoremove_pending Apt packages pending autoremoval.
# TYPE apt_autoremove_pending gauge
apt_autoremove_pending 0
# HELP apt_upgrades_held Apt packages pending updates but held back.
# TYPE apt_upgrades_held gauge
apt_upgrades_held{arch="",origin=""} 0
# HELP apt_upgrades_pending Apt packages pending updates by origin.
# TYPE apt_upgrades_pending gauge
apt_upgrades_pending{arch="all",origin="Debian:bookworm-security/stable-security"} 2
apt_upgrades_pending{arch="all",origin="Debian:bookworm/stable"} 5
apt_upgrades_pending{arch="amd64",origin="Debian:bookworm-security/stable-security"} 10
apt_upgrades_pending{arch="amd64",origin="Debian:bookworm-security/stable-security,Debian:bookworm/stable"} 4
apt_upgrades_pending{arch="amd64",origin="Debian:bookworm/stable"} 34
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.8482e-05
go_gc_duration_seconds{quantile="0.25"} 4.7539e-05
go_gc_duration_seconds{quantile="0.5"} 5.3363e-05
go_gc_duration_seconds{quantile="0.75"} 8.0271e-05
go_gc_duration_seconds{quantile="1"} 0.000529001
go_gc_duration_seconds_sum 0.058958629
go_gc_duration_seconds_count 734
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.19.8"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.839112e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.365062656e+09
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.614513e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 3.2462254e+07
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 9.386944e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
```

Adding New Job to Prometheus

Now that the Node Exporter is running, you're ready to add Node Exporter to the Prometheus.

Open the Prometheus configuration */etc/prometheus/prometheus.yml* using the following nano editor command.

```
sudo nano /etc/prometheus/prometheus.yml
```

Within the **scrape_configs** section, add a new job **prometheus-node-exporter** with the endpoint of the Node Exporter metrics like the following.

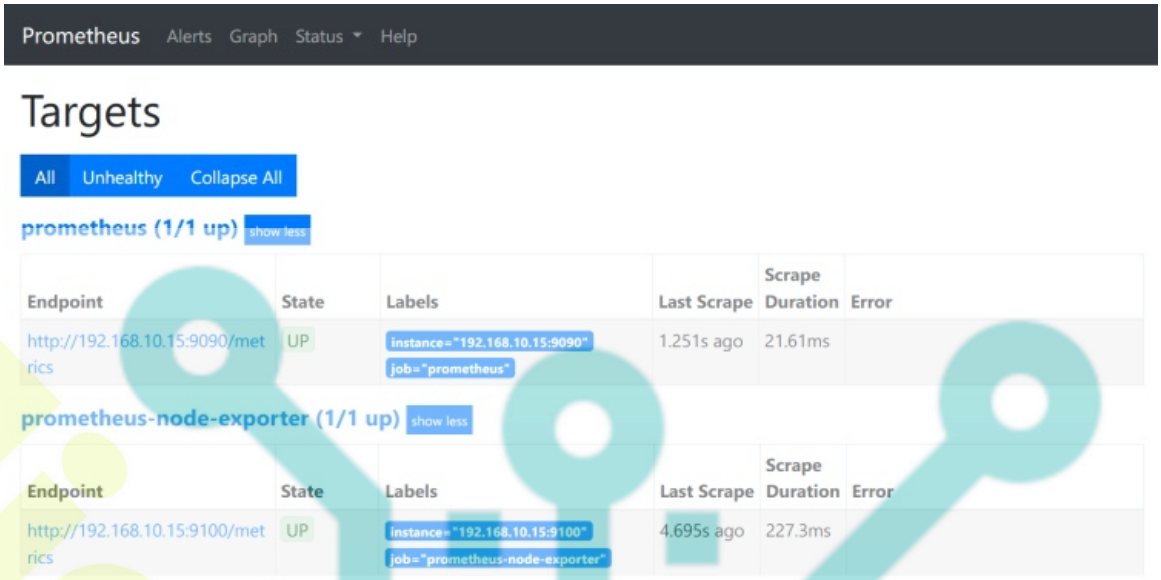
```
  - job_name: 'prometheus-node-exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['192.168.10.15:9100']
```

Save the file and exit the editor when finished.

Next, execute the following systemctl command to restart the prometheus service and apply the changes.

```
sudo systemctl restart prometheus
```

Lastly, back to the Prometheus dashboard, then click **Status** > **Targets** menu. If everything goes well, you should see the Node Exporter on the target endpoint.



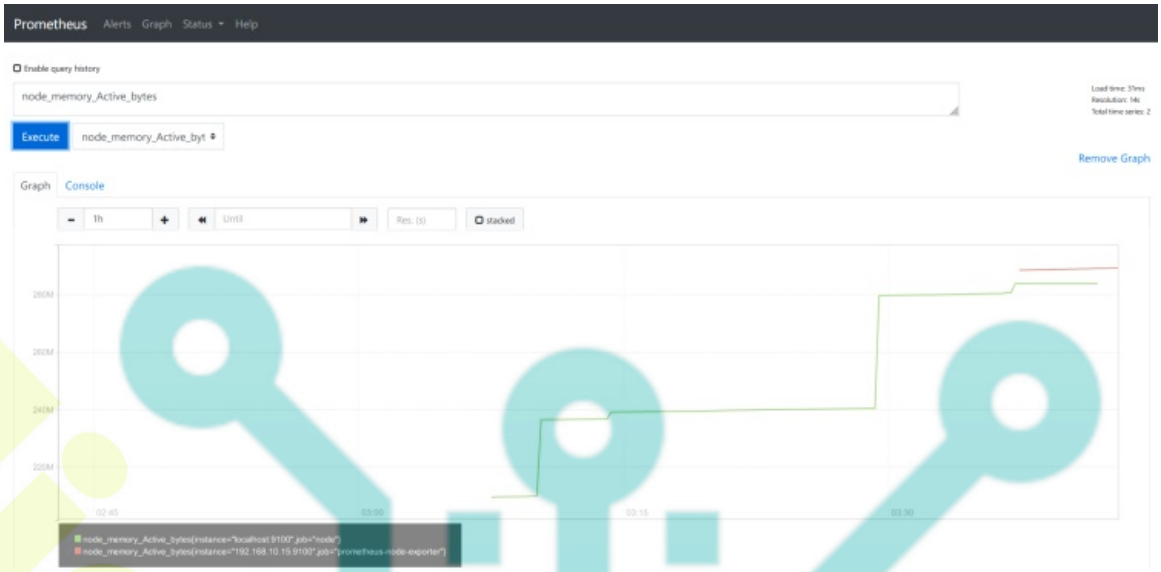
Basic Usage of Prometheus Dashboard

In this section, you will learn the basic query of Prometheus and Node Exporter, which can be executed from the Prometheus dashboard.

Type the query *node_os_info* and click the **Execute** button. The *node_os_info* is a query language provided by Node Exporter for checking operating system details.

On the **Console** section, you should see detailed host informations like the following:

Next, type another query such as *node_memory_Active_bytes* to check active memory on the target server. In the **Graph** section, you should see similar results like the following:



Furthermore, you can also use PromQL (Prometheus Query Language) to get specific data, such as *node_memory_Active_bytes[5]* that will show you data for the last 5 minutes.

Conclusion

To wrap up, you've completed the installation of Prometheus and Node Exporter on the Debian 12 server step-by-step. You've installed Prometheus and Node Exporter via APT from the official Debian repository. You've also learned the basic usage of the Prometheus dashboard for testing queries. You can now install additional components such as Alert Manager and Grafana to your monitoring system.
