Monitoring a Linux/windows server using Prometheus

Souvik Haldar Follow
Mar 25, 2019 · 7 min read

Why monitor?

Monitoring of a system is key to its smooth functioning. Going to the battlefield (production) without having proper monitoring setup done is like making your platform vulnerable, hence to obtain full control it becomes a must; as the popular say goes *"Failing to plan, is planning to fail"*. In this article, I'm going to show how you can monitor a system using Prometheus, node_exporter and the Grafana UI.

Difference between Pull and Push based monitoring architecture

Simply put, in push-based architecture each target node periodically sends metrics to a central collector. Examples of push architectures include sFlow, Ganglia, Graphite, collectd and StatsD. Whereas, in pull-based architecture the central collector periodically requests each of the target node to send metrics to it. Examples of pull architectures include SNMP, JMX, WMI, libvirt, Prometheus,etc.

Prometheus is primarily a pull-based system, however, it can act as a push-based system by using pushgateway

Installation and setup

At Kartbites, our HTTP server runs on Debian and the primary system for monitoring runs on Arch Linux, Hence I'll assume target machine runs on Debian (or Windows as a bonus) and monitor system runs on Arch Linux (or Ubuntu).

We will install Prometheus which will pull metrics from the target server, `node_exporter` which will make the target system's metrics available at an HTTP port for `prometheus` to pull; and `Grafana` is the UI for an amazing visualization.

1. Prometheus

The installation procedure is pretty simple, I'm going to show how to install on two platforms, Arch Linux and Ubuntu. For others, you can definitely follow the official docs

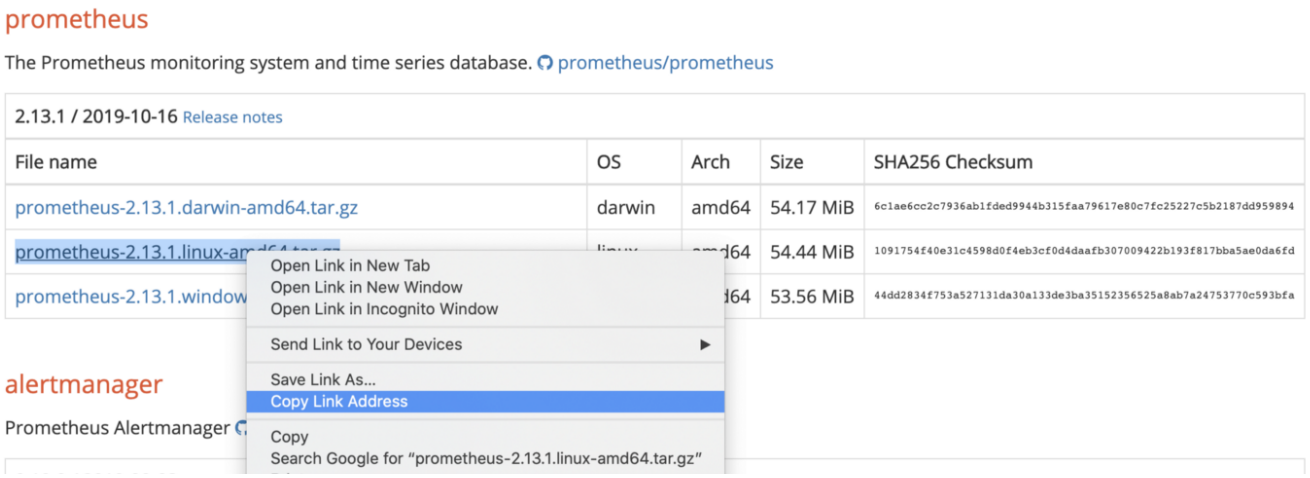
Arch Linux

Arch has a package for Prometheus, which is great because then you don't need to explicitly write unit file for the service.

1. `sudo pacman -S prometheus`
2. `sudo systemctl enable prometheus` (to create systemlink to the unit file in the systemd directory to that the systemd can always start it at boot)
3. `sudo systemctl start prometheus` to start the service right away.

Ubuntu

1. Download `wget` `https://github.com/prometheus/prometheus/releases/download/v2.8.0/prometheus-2.8.0.linux-amd64.tar.gz` or if you want the latest version, get the download link of the latest version by right-clicking on the link and `copy link address` then pass it to `wget` . See:-



2. `tar -xzf prometheus-2.8.0.linux-amd64.tar.gz`

4. `cd prometheus-2.8.0.linux-amd64`

5. Move the `prometheus` binary executable to PATH `mv prometheus /usr/local/bin/`

6. Move `promtheus` configuration file i.e `promethues.yml` to `/etc/`. i.e `mv prometheus.yml /etc/`

Now set up the configuration of `prometheus` server-

Configure the Prometheus config file (`prometheus-2.8.0.linux-amd64/prometheus.yml`) with simple configurations:-

```
global:
scrape_interval: 15s
# By default, scrape targets every 15 seconds.
# Attach these labels to any time series or alerts when communicating with
# external systems (federation, remote storage, Alertmanager).
external_labels:
monitor: 'codelab-monitor'
# 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
- job_name: 'prometheus'
# Override the global default and scrape targets from this job every 5 seconds.
scrape_interval: 5s
static_configs:
- targets: ['localhost:9090']
```

prometheusConfig hosted with ❤ by **GitHub** **view raw**

Now write a unit file to run `prometheus` as a systemd service (There are many benefits of running a process as systemd service instead, like running in the background, auto-restart, logging using journalctl,etc) . For more info on systemd and unit files see here.

Create the unit file

```
sudo nano /etc/systemd/system/prometheus.service
```

Add the following contents:-

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
Type=simple
ExecStart=/usr/local/bin/prometheus --config.file=/etc/prometheus.yml
[Install]
WantedBy=multi-user.target
```

prometheusUnitFile hosted with ❤ by **GitHub** **view raw**

Now follow these steps to start and enable the service so that it can keep running in the background and would try to restart automatically in case of any failure.

1. `sudo systemctl daemon-reload`
2. `sudo systemctl start prometheus`
3. `sudo systemctl enable prometheus`

Now if you wish to check whether it is not or not execute `systemctl status prometheus`

Alternatively, you can use apt to install Prometheus as well.

Now you can visit `<ip-address>:9090` on the browser to see Prometheus running. Kudos you've successfully set up Prometheus on your monitoring server. (make sure port 9090 is open for HTTP requests)

2. node_exporter

The Prometheus Node Exporter exposes a wide variety of hardware- and kernel-related data, which prometheus can scrape metrics from. Typically `node_exporter` is installed on the target machine which you want to monitor and `prometheus` is installed on a server which is primarily used as the master to monitor the target servers. (the load can be horizontally distributed as well)

Here I will show you how you can install `node_exporter` on Debian and Windows server. Prometheus will keep pulling metrics from them and hence monitor.

Debian

The official docs will show to use the tarball (ref. — follow the above guide for installing Prometheus on ubuntu) but for convenience, we will use the official debian package.

1) `sudo apt-get install prometheus-node-exporter`

2) This by default enables and starts the node exporter service but you can cross check by `systemctl status node_exporter.service`

Windows

Unfortunately `node_exporter` is not well-supported on windows and hence we will use an alternative

1) Visit releases.

2) I would recommend to download and run the `.msi` as can setup most of the things for you. By default, the service will start running on port 9182 so make sure to open that port to Prometheus server.

. . .

Now, make this newly created exporter a target for Prometheus to pull. Add the following lines to Prometheus configuration file i.e `prometheus.yml` :-

For the above mentioned `debian` target server:-

```
scrape_configs:
- job_name: 'node'
  static_configs:
  - targets: ['<TARGET_IP>:9100']
```

nodeExporterPromConfig hosted with ❤ by **GitHub** **view raw**

Similar, for the above-mentioned `windows` target server but change the port address to `9182` since wmi-exporter runs on this port by default. These ports are the default values and can be changed according to need by making necessary changes to the configuration file of the exporters.

For the new target to get ready to be configured we need to restart the `prometheus` service so that it read the updated configuration i.e `prometheus.yml` . Run `sudo systemctl restart prometheus.service` . The downside to this `restart` is that this will cause downtime during the restart process. To get around this downtime follow this neat *trick*. The idea is to send a hang-up single to the `prometheus` service which will make it reload the configuration. For that first, we need to know the PID (process ID) of the process. Run `ps aux | grep prome*`

```
[souvik@quiche ~]$ ps aux | grep prome*

prometh+ 29115 0.1 1.5 1142212 126020 ? Ssl Mar26 1:32
/usr/bin/prometheus — config.file=/etc/prometheus/prometheus.yml —
storage.tsdb.path=/var/lib/prometheus/data

souvik 33410 0.0 0.4 891840 39956 pts/1 S+ 03:12 0:00 journalctl -u
prometheus -f

souvik 33445 0.0 0.0 6268 2316 pts/2 S+ 03:18 0:00 grep prome*

[souvik@quiche ~]$
```

So now we know that the PID of the prometheus service is `29915` .

Now we need the hang-up signal to it. Run `sudo kill -s HUP 29115`

This will make Prometheus reload the configuration as evident in the below logs:-

```
Mar 27 03:20:36 quiche prometheus[29115]: level=info ts=2019-03-
27T10:20:36.618684382Z caller=main.go:724 msg="Loading configuration
file" filename=/etc/prometheus/prometheus.yml

Mar 27 03:20:36 quiche prometheus[29115]: level=info ts=2019-03-
27T10:20:36.619961054Z caller=main.go:751 msg="Completed loading of
configuration file" filename=/etc/prometheus/prometheus.yml
```

Congratulations

Now visit the `Status --> Targets` on the Prometheus address in the browser and your target server will appear there. *Now you can query for basic metrics and see the corresponding graph on the dashboard*. But Grafana makes it cooler. Let's grab it now.

3. Grafana

Now it's time to give our our monitoring solution a beautiful UI and for that what can be a better option than Grafana! Let's dive quickly into installing it.

Debian

1) `wget https://dl.grafana.com/oss/release/grafana_6.0.2_amd64.deb`

2) `sudo dpkg -i grafana_6.0.2_amd64.deb`

3) `sudo systemctl start grafana-server.service`

4) `sudo systemctl enable grafana-server.service`

Or, you can use the package manager (in this case APT) to install the same. For that, I would recommend you to follow this link

However while installing via `dpkg` or `apt` I faced an issue that some of it's dependencies coulld not be installed. If you are also facing this issue run the following command-

```
sudo apt --fix-broken install
```

Arch Linux

On arch, you can use the package manager as well for the installation.

```
sudo pacman -S grafana
```

Now that installation is successful, you need to enable and start the grafana `systemd` service. For that you need to execute the following commands:-

```
sudo systemctl enable grafana
```

```
sudo systemctl start grafana
```

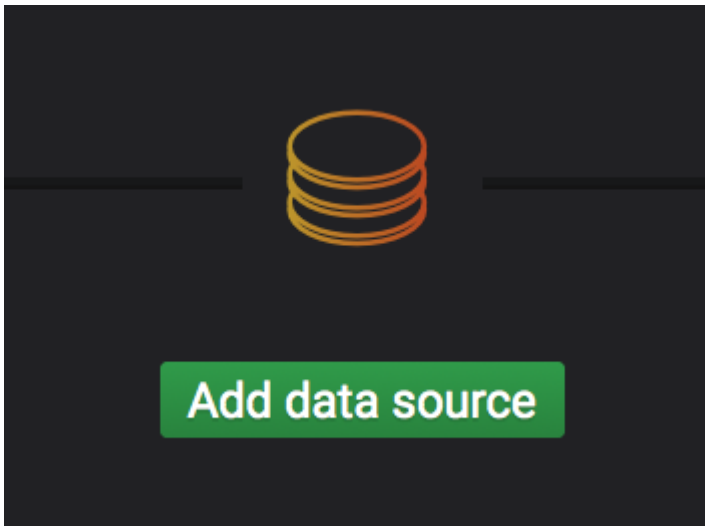
Now open port 3000 in server's firewall policy because by default Grafana listens on port 3000. Now hit port 3000 and you can see Grafana running. The default username and password is `admin` and `admin` respectively.

Congratulations, your pretty dashboard is now setup correctly!

Now we need to configure Grafana to set `prometheus` as a data source.

1) Visit `<montoring-server-ip>:3000` .

2) Click on `Add data source`



3) Select `prometheus`

4) Let the defaults be. Check if the address the alright.

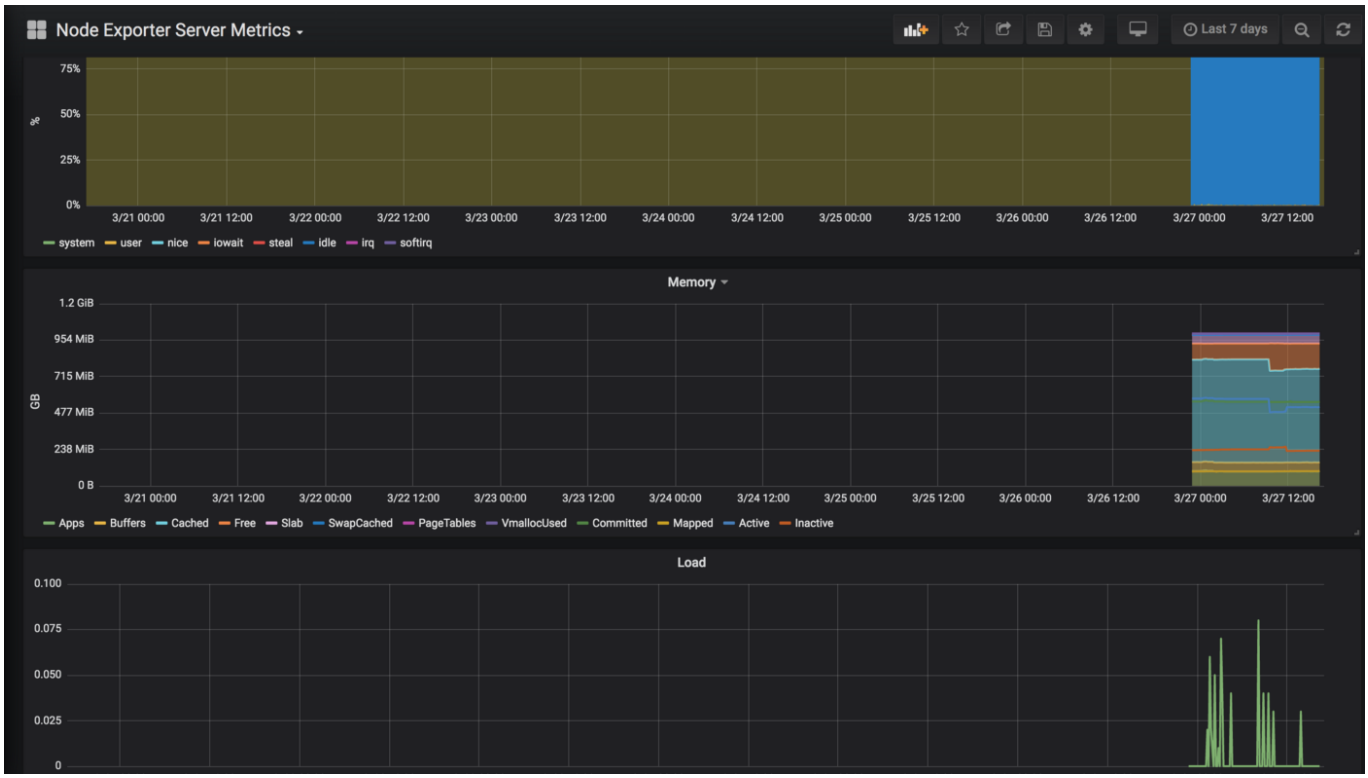
5) Import a pre-built dashboard but clicking on `+` icon.

6) Import `1860` and `405` as Dashboard ID. (personal choice, you can import any)

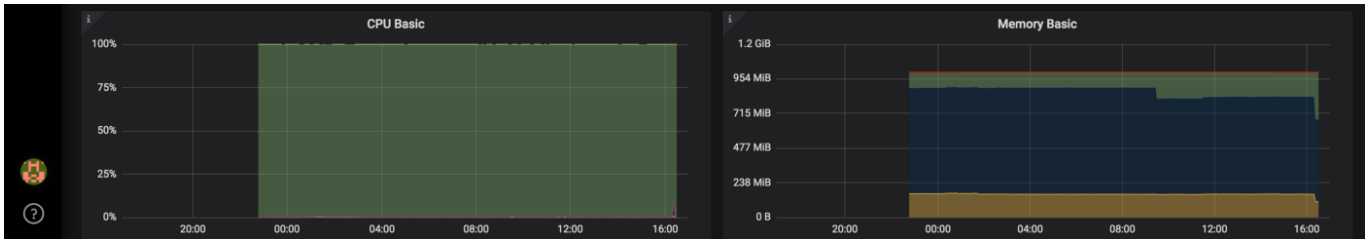
Demo

Voila, you have the pretty dashboard ready!

visit `<monitor-server-ip>:3000`

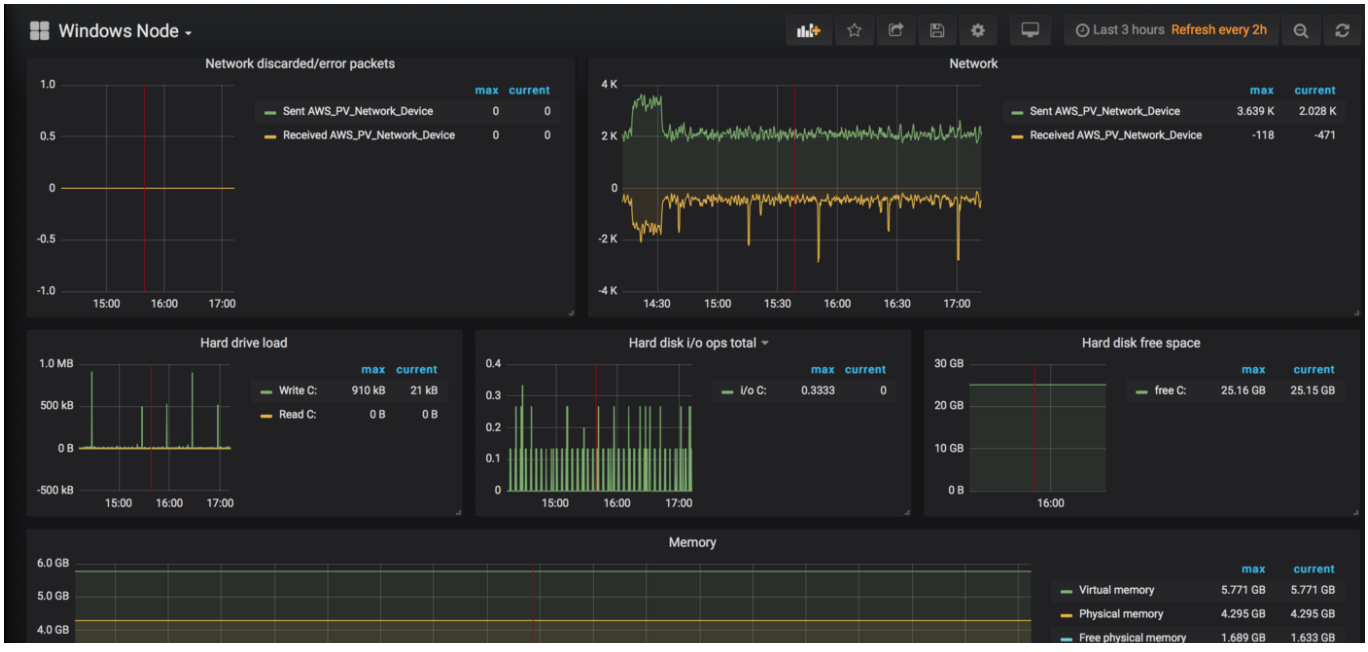


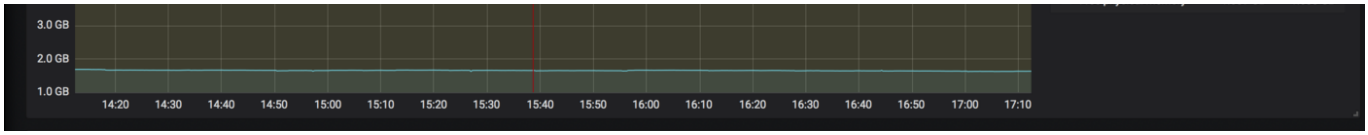
405



1860

Some more





windows node

Conclusion

So that's a brief walk-through of the setup of the monitoring system for a Linux and windows server. Now you can customize further and set up alerts for different scenarios using `Alert Manager`. You can go through this cool video tutorial if you are more of a video person. Thanks and I hope I was able to at least get you started with the popular and amazing monitoring tool **Prometheus**.

. . .

Originally published at http://souvikhaldar.info/programming/prometheus/*on March 25, 2019. Republished here on behalf of Kartbites.*

. . .

Kartbites is a community-powered street food app where any place you eat automatically becomes a place for others to discover. Learn more at http://kartbites.com