How To - Configure a Prometheus Monitoring Server with a Grafana Dashboard

Installing Prometheus Monitoring Server with a Grafana Dashboard:-

Prometheus :- is a flexible monitoring solution that is in development since 2012.
The software stores all its data in a time series database and offers a multi-dimensional data-model and a powerful query language to generate reports of the monitored resources.

Grafana :- is an open-source platform for data visualization, monitoring and analysis.Its allows users to create dashboards with panels, each representing specific metrics over a set time-frame.

There are five steps to use Prometheus with Grafana:

Preparing your Environment
Downloading and Installing Node Exporter
Downloading and Installing Prometheus
Configuring Prometheus
Downloading and Installing Grafana

Preparing your Environment

we use an instance running on Ubuntu Xenial (16.04).

1 . To run Prometheus safely on our server, we have to create a user for Prometheus and Node Exporter without the possibility to log in. To achieve this, we use the parameter `--no-create-home` which skips the creation of a home directory and disable the shell with `--shell /usr/sbin/nologin`.

```
sudo useradd --no-create-home --shell /usr/sbin/nologin prometheus
sudo useradd --no-create-home --shell /bin/false node_exporter
```

2 . Create the folders required to store the binaries of Prometheus and its configuration files:

```
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
```

3 . Set the ownership of these directories to our `prometheus` user, to make sure that Prometheus can access to these folders:

```
sudo chown prometheus:prometheus /etc/prometheus
sudo chown prometheus:prometheus /var/lib/prometheus
```

Downloading and Installing Node Exporter

As your Prometheus is only capable of collecting metrics, we want to extend its capabilities by adding **Node Exporter**, a tool that collects information about the system including CPU, disk, and memory usage and exposes them for scraping.

1 . Download the latest version of Node Exporter:

```
wget https://github.com/prometheus/node_exporter/releases/download/v0.16.0/node_exporter-0.16.0.linux-amd64.tar.gz
```

2 . Unpack the downloaded archive. This will create a directory `node_exporter-0.16.0.linux-amd64`, containing the executable, a readme and license file:

```
tar xvf node_exporter-0.16.0.linux-amd64.tar.gz
```

3 . Copy the binary file into the directory `/usr/local/bin` and set the ownership to the user you have created in step previously:

```
sudo cp node_exporter-0.16.0.linux-amd64/node_exporter /usr/local/bin
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

4 . Remove the leftover files of Node Exporter, as they are not needed any longer:

```
rm -rf node_exporter-0.16.0.linux-amd64.tar.gz node_exporter-0.16.0.linux-amd64
```

5 . To run Node Exporter automatically on each boot, a Systemd service file is required. Create the following file by opening it in Nano:

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

6 . Copy the following information in the service file, save it and exit Nano:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

7 . Collectors are used to gather information about the system. By default a set of collectors is activated. You can see the details about the set in the README-file. If you want to use a specific set of collectors, you can define them in the `ExecStart` section of the service. Collectors are enabled by providing a`--collector.<name>` flag. Collectors that are enabled by default can be disabled by providing a `--no-collector.<name>` flag.

8 . Reload Systemd to use the newly defined service:

```
sudo systemctl daemon-reload
```

9 . Run Node Exporter by typing the following command:

```
sudo systemctl start node_exporter
```

10 . Verify that the software has been started successfully:

```
sudo systemctl status node_exporter
```

You will see an output like this, showing you the status `active (running)` as well as the main PID of the application:

```
● node_exporter.service - Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; disabled; vendor preset: enabled)
   Active: active (running) since Mon 2018-06-25 11:47:06 UTC; 4s ago
 Main PID: 1719 (node_exporter)
   CGroup: /system.slice/node_exporter.service
           └─1719 /usr/local/bin/node_exporter
```

`11 . If everything is working, enable Node Exporter to be started on each boot of the server:`

```
sudo systemctl enable node_exporter
```

Downloading and Installing Prometheus

1 . Download and Unpack Prometheus latest release of Prometheus. As exemplified, the version is 2.2.1:

```
sudo apt-get update && apt-get upgrade
wget https://github.com/prometheus/prometheus/releases/download/v2.2.1/prometheus-2.2.1.linux-amd64.tar.gz
tar xfz prometheus-*.tar.gz
cd prometheus-*
```

- [ ] The following two binaries are in the directory:
  Prometheus - Prometheus main binary file
  promtool
- [ ] The following two folders (which contain the web interface, configuration files examples and the license) are in the directory:
  consoles
  console_libraries

2 . Copy the binary files into the `/usr/local/bin/`directory:

```
sudo cp ./prometheus /usr/local/bin/
sudo cp ./promtool /usr/local/bin/
```

3 . Set the ownership of these files to the `prometheus` user previously created:

```
sudo chown prometheus:prometheus /usr/local/bin/prometheus
sudo chown prometheus:prometheus /usr/local/bin/promtool
```

4 . Copy the `consoles` and `console_libraries` directories to `/etc/prometheus`:

```
sudo cp -r ./consoles /etc/prometheus
sudo cp -r ./console_libraries /etc/prometheus
```

5 . Set the ownership of the two folders, as well as of all files that they contain, to our `prometheus` user :

```
sudo chown -R prometheus:prometheus /etc/prometheus/consoles
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
```

6 . In our home folder, remove the source files that are not needed anymore:

```
cd .. && rm -rf prometheus-*
```

Configuring Prometheus

1 . Open the file `prometheus.yml` in a text editor:

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

```
global:
  scrape_interval:     15s
  evaluation_interval: 15s

rule_files:
  # - "first.rules"
  # - "second.rules"

scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
```

We add the following part below the configuration for scrapping Prometheus:

```
- job_name: 'node_exporter'
  scrape_interval: 5s
  static_configs:
    - targets: ['localhost:9100']
```

2 . Set the ownership of the file to our `Prometheus` user:

```
sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

Running Prometheus

1 . Start Prometheus directly from the command line with the following command, which executes the binary file as our `Prometheus` user:

```
sudo -u prometheus /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/ --web.console.templates=/etc/prometheus/consoles --web.console.libraries=/etc/prometheus/console_libraries
```

The server starts displaying multiple status messages and the information that the server has started:

```
level=info ts=2018-04-12T11:56:53.084000977Z caller=main.go:220 msg="Starting Prometheus" version="(version=2.2.1, branch=HEAD, revision=bc6058c81272a8d938c05e75607371284236aadc)"
level=info ts=2018-04-12T11:56:53.084463975Z caller=main.go:221 build_context="(go=go1.10, user=root@149e5b3f0829, date=20180314-14:15:45)"
level=info ts=2018-04-12T11:56:53.084632256Z caller=main.go:222 host_details="(Linux 4.4.127-mainline-rev1 #1 SMP Sun Apr 8 10:38:32 UTC 2018 x86_64 scw-041406 (none))"
level=info ts=2018-04-12T11:56:53.084797692Z caller=main.go:223 fd_limits="(soft=1024, hard=65536)"
level=info ts=2018-04-12T11:56:53.09190775Z caller=web.go:382 component=web msg="Start listening for connections" address=0.0.0.0:9090
level=info ts=2018-04-12T11:56:53.091908126Z caller=main.go:504 msg="Starting TSDB ..."
level=info ts=2018-04-12T11:56:53.102833743Z caller=main.go:514 msg="TSDB started"
level=info ts=2018-04-12T11:56:53.103343144Z caller=main.go:588 msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
level=info ts=2018-04-12T11:56:53.104047346Z caller=main.go:491 msg="Server is ready to receive web requests."
```

2 . Open your browser and type `http://IP.OF.YOUR.SERVER:9090` to access the Prometheus interface. If everything is working, we end the task by pressing on `CTRL + C` on our keyboard.
3 . The server is working now, but it cannot yet be launched automatically at boot. To achieve this, we have to create a new `systemd` configuration file that will tell your OS which services should it launch automatically during the boot process.

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

4 . Copy the following information in the file and save it, then exit the editor:

```
[Unit]
  Description=Prometheus Monitoring
  Wants=network-online.target
  After=network-online.target

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

[Install]
  WantedBy=multi-user.target
```

5 . To use the new service, reload `systemd`:

```
sudo systemctl daemon-reload
```

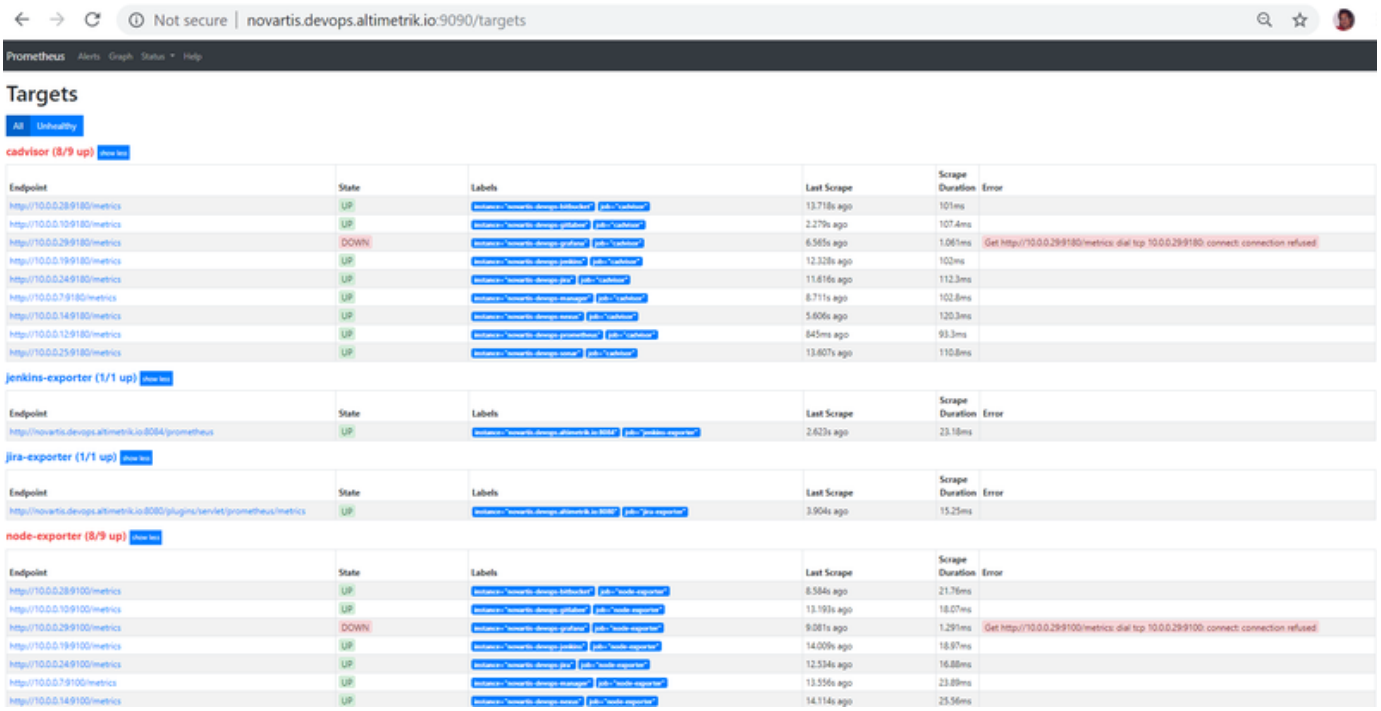
```
sudo systemctl enable prometheus
```

6 . Start Prometheus:

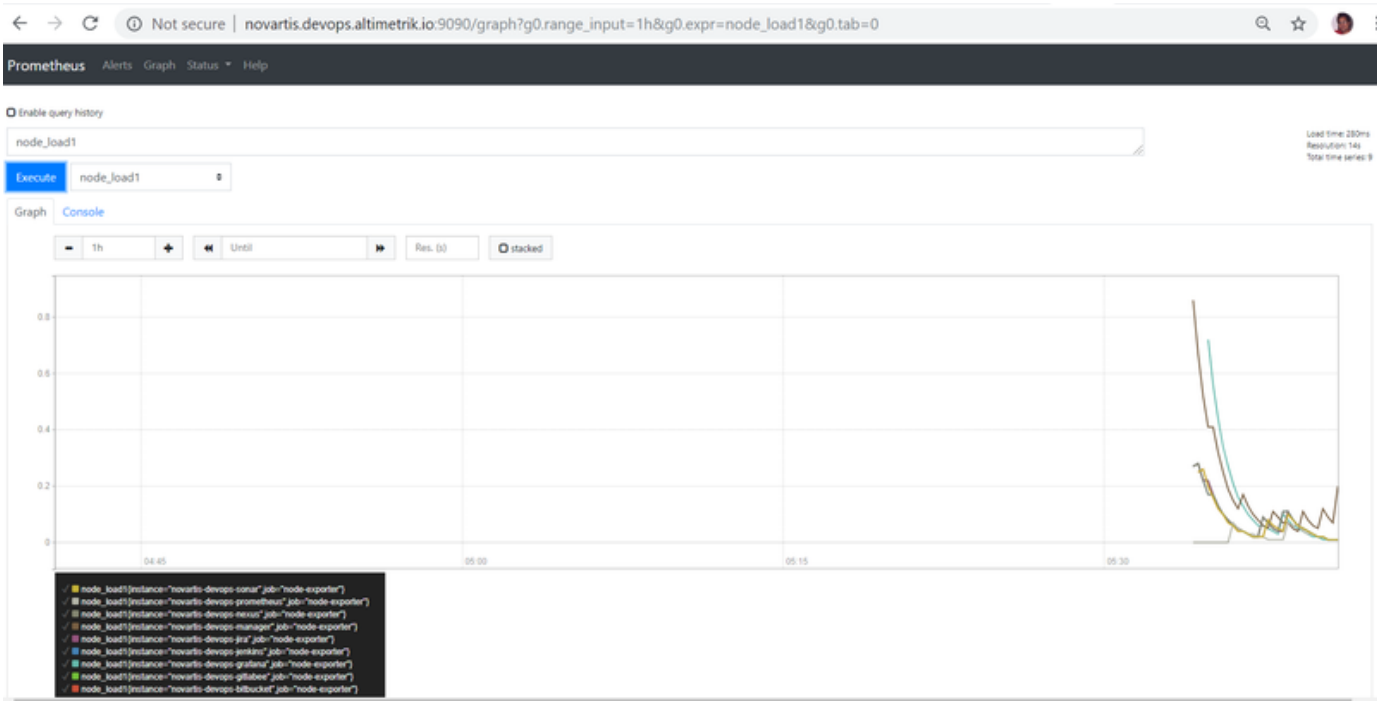
```
sudo systemctl start prometheus
```

Prometheus Web Interface

Prometheus provides a basic web server running on `http://novartis.devops.altimetrik.io:9090/`



`Moreover, do some queries in the data that has been collected.`



The interface is very lightweight, and the Prometheus team recommend to use a tool like Grafana if you want to do more than testing and debugging the installation.

Installing Grafana

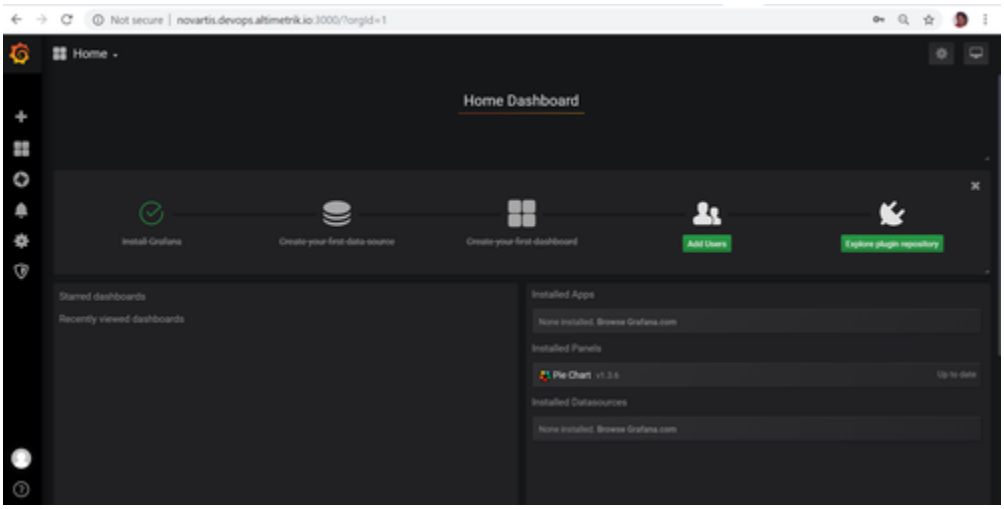
1 . Install Grafana on our instance which queries our Prometheus server.

```
wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_5.0.4_amd64.deb
sudo apt-get install -y adduser libfontconfig
sudo dpkg -i grafana_5.0.4_amd64.deb
```

2 . Enable the automatic start of Grafana by `systemd`:

```
sudo systemctl daemon-reload && sudo systemctl enable grafana-server && sudo systemctl start grafana-server
```

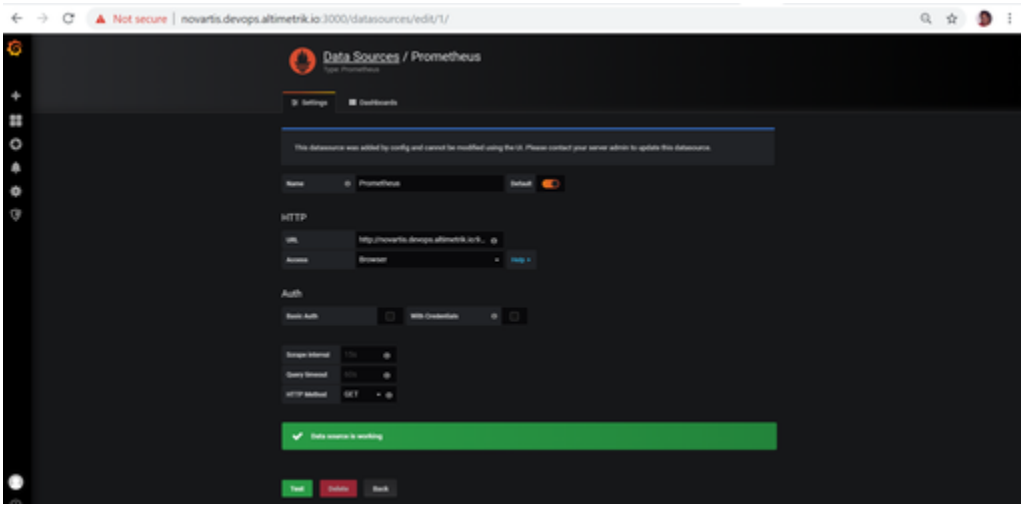
Grafana is running now, and we can connect to it at `http://novartis.devops.altimetrik.io:3000/`



Now you have to create a Prometheus data source:

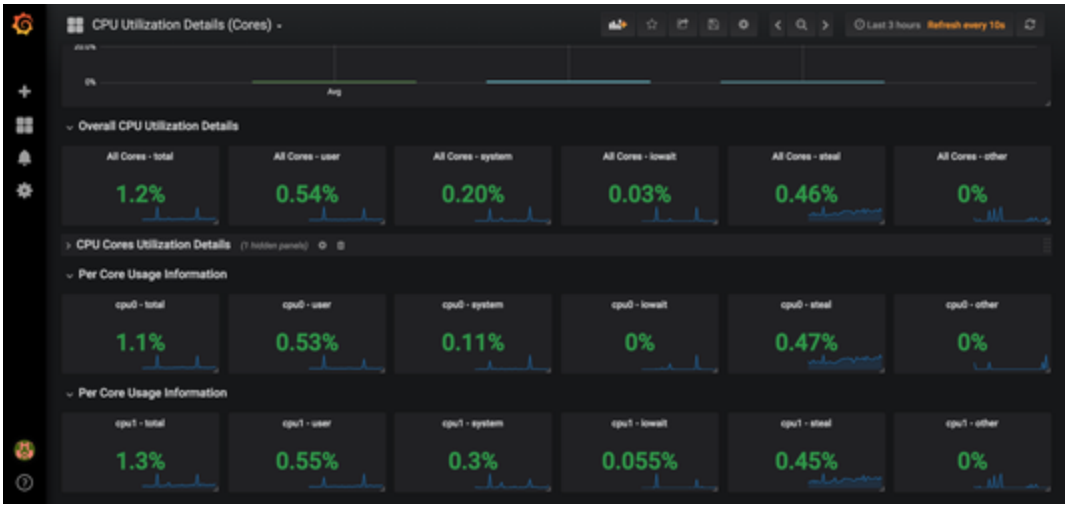
- Click on the Grafana logo to open the sidebar.
- Click on “Data Sources” in the sidebar.
- Choose “Add New”.
- Select “Prometheus” as the data source.
- Set the Prometheus server URL (in our case: http://novartis.devops.altimetrik.io:9090)
- Click “Add” to test the connection and to save the new data source.

Your settings should look like this:



You are now ready to create your first dashboard from the information collected by Prometheus. You can also import some dashboards from a collection of shared dashboards

Here is an example of a Dashboard that uses the CPU usage of our node and presents it in Grafana:


CPU Utilization Details (Cores)
Last 3 hours
Refresh every 10s
0%
Avg
Overall CPU Utilization Details
All Cores - total
1.2%
All Cores - user
0.54%
All Cores - system
0.20%
All Cores - iowait
0.03%
All Cores - steal
0.46%
All Cores - other
0%
CPU Cores Utilization Details
(1 hidden panels)
Per Core Usage Information
cpu0 - total
1.1%
cpu0 - user
0.53%
cpu0 - system
0.11%
cpu0 - iowait
0%
cpu0 - steal
0.47%
cpu0 - other
0%
Per Core Usage Information
cpu1 - total
1.3%
cpu1 - user
0.55%
cpu1 - system
0.3%
cpu1 - iowait
0.055%
cpu1 - steal
0.45%
cpu1 - other
0%