POC on Monitoring tools

(Grafana, Prometheus, Alert Manager, Blackbox exporter and Node Exporter)

Grafana

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources.

Prometheus

Prometheus is a free software application used for event monitoring and alerting. It records real-time metrics in a time series database built using a HTTP pull model, with flexible queries and real-time alerting.

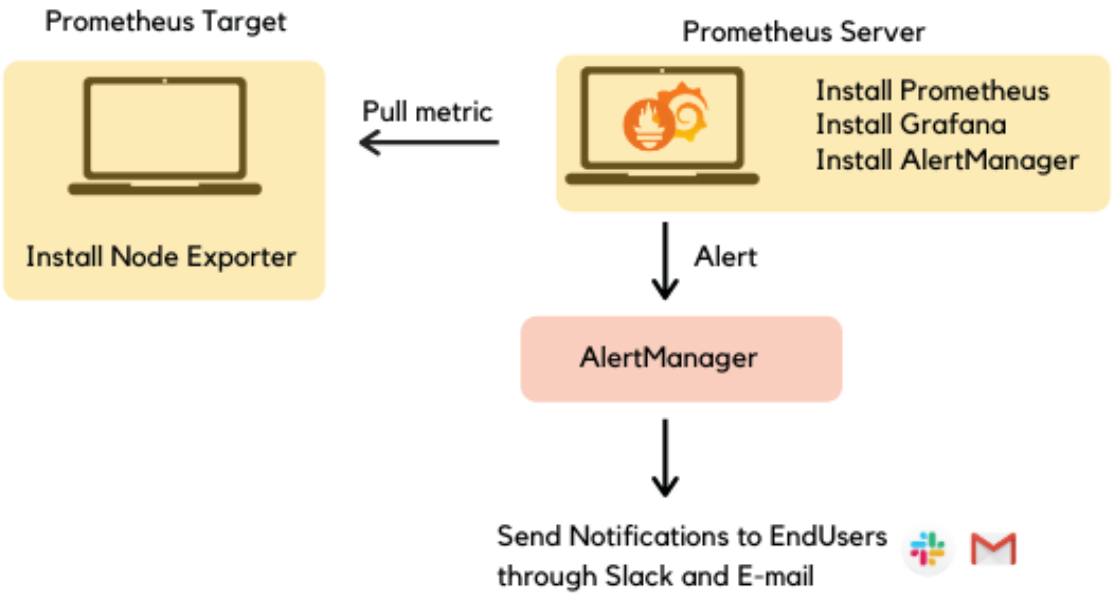
Prometheus Alert Manager

The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie. It also takes care of silencing and inhibition of alerts.

Node Exporter

Node Exporter is a Prometheus exporter for server level and OS level metrics with configurable metric collectors. It helps us in measuring various server resources such as RAM, disk space, and CPU utilization.

Working Diagram



Introduction

This POC includes the step by step procedure of installing different packages like prometheus, grafana, node exporter and alert manager to monitor system resources and generate alerts when a system metric crosses a specific threshold.

Installing Node Exporter

First, add a node_exporter user with no login permission and without home directory.
$ sudo adduser --no-create-home --disabled-login --shell /bin/false node_exporter

Next, download the node exporter binary file.
$ wget
https://github.com/prometheus/node_exporter/releases/download/v1.1.2/node_exporter-1.1.2.linux-386.tar.gz

Unzip the downloaded binary file using the following command.
$ tar -xvzf node_exporter-1.1.2.linux-386.tar.gz

Rename the extracted directory to node_exporter.
$ mv node_exporter-1.1.2.linux-386 node_exporter

Go to the node_exporter directory and copy the executable file to /usr/local/bin directory.
$ cd node_exporter
$ sudo cp node_exporter /usr/local/bin/

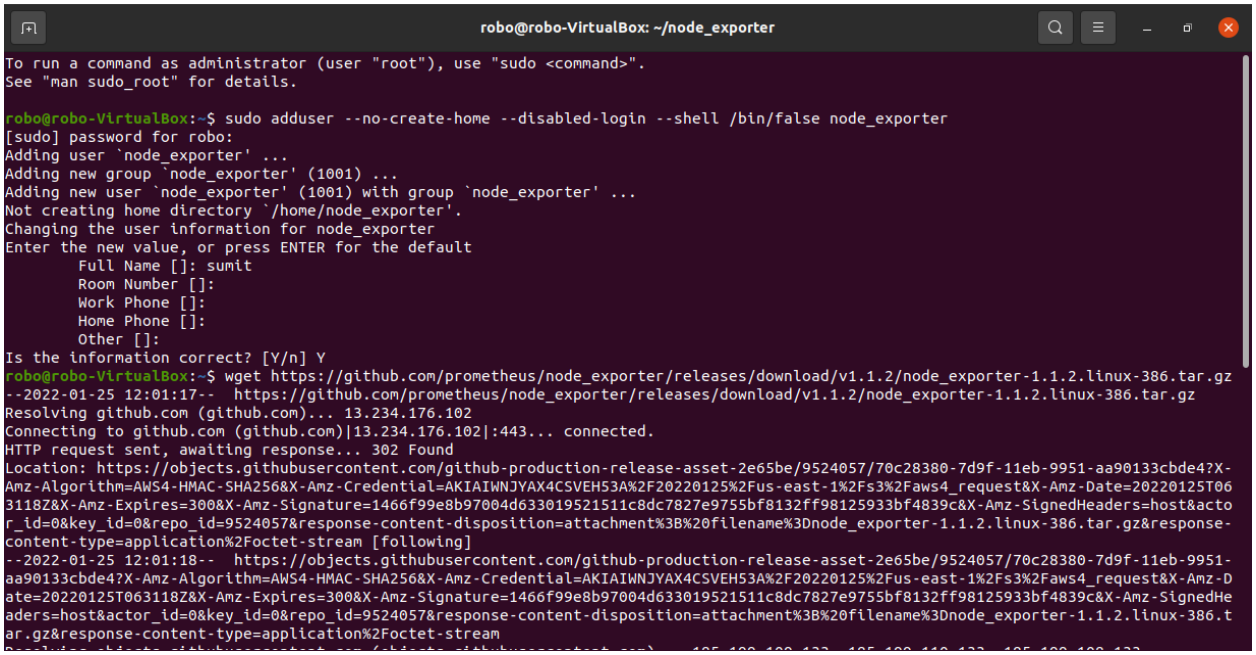
Change the ownership of the binary executable file.
$ sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter

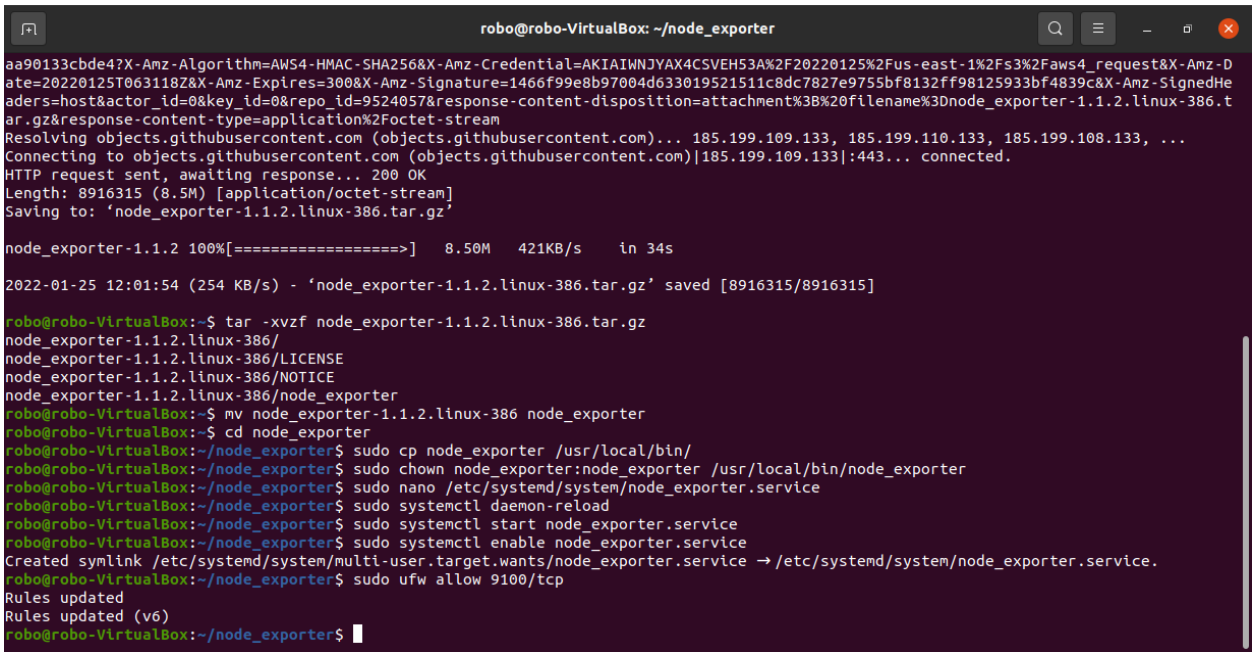
Create a service file for node exporter to run it is as a daemon and copy the following configuration there.
$ sudo nano /etc/systemd/system/node_exporter.service

```
[Unit]

Description=Node Exporter

[Service]

User=node_exporter

Group=node_exporter

Type=simple

ExecStart= /usr/local/bin/node_exporter

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target
```

Reload the daemon and start the node_exporter service.
$ sudo systemctl daemon-reload
$ sudo systemctl start node_exporter.service
$ sudo systemctl enable node_exporter.service

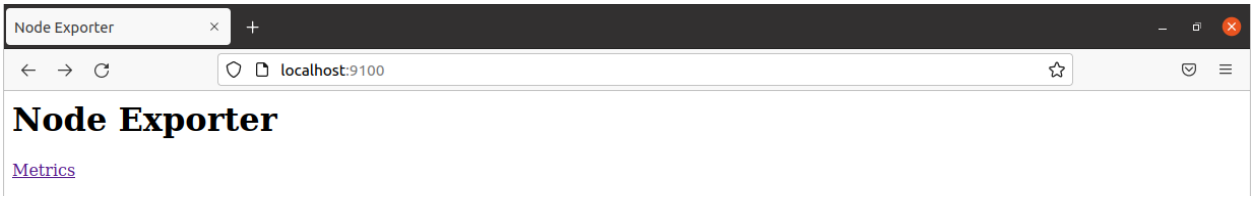
Note: By default Webdock Perfect Server stacks allows traffic from specific ports. You need to open the ports explicitly whenever needed.

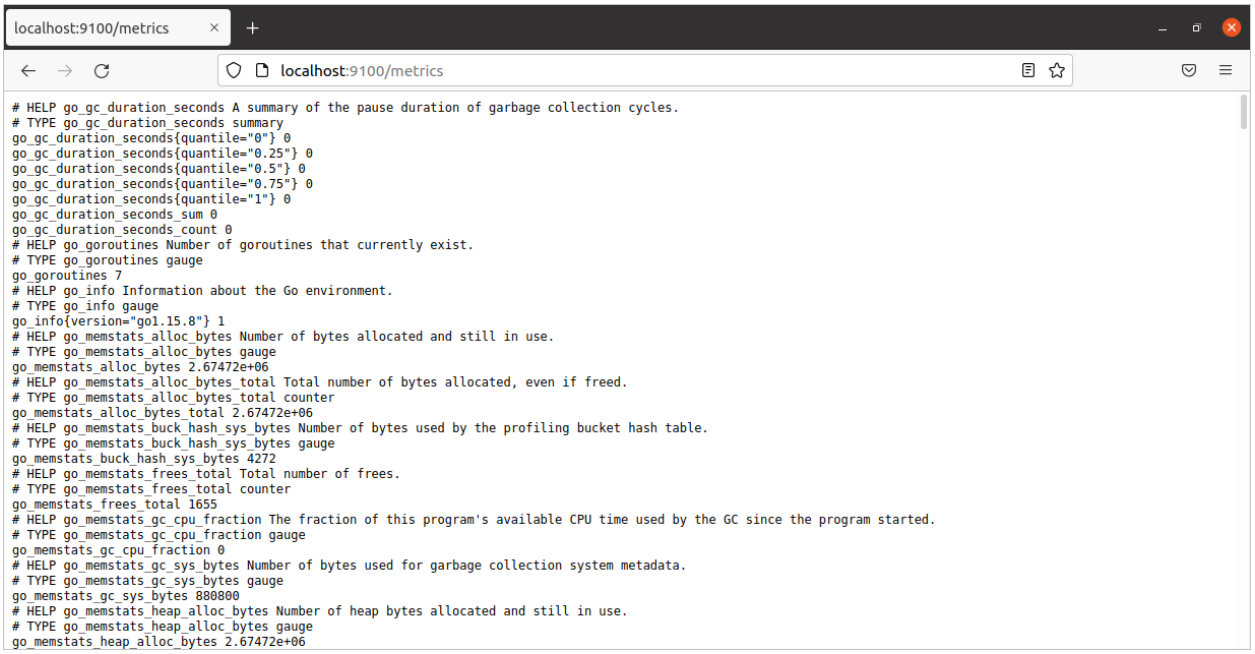




In case of a Webdock Perfect Server stack, open the port 9100 on the server.
$ sudo ufw allow 9100/tcp

Now check the port 9100 of the server, it will list the system metrics in raw form.





Installing Prometheus

Create a prometheus user with no login permission and without home directory.
$ sudo adduser --no-create-home --disabled-login --shell /bin/false prometheus

Download the Prometheus binary file.
$ wget
https://github.com/prometheus/prometheus/releases/download/v2.26.0/prometheus-2.26.0.linux-amd64.tar.gz

Unzip the downloaded binary file.
$ tar -xvzf prometheus-2.26.0.linux-amd64.tar.gz

Change the directory name to prometheus.
$ mv prometheus-2.26.0.linux-amd64 prometheus

Go to the prometheus directory.
$ cd prometheus

Create configuration and data directories for Prometheus.
$ sudo mkdir -p /etc/prometheus/{rules,rules.d,files_sd} /var/lib/prometheus

Copy the binary files to the /usr/local/bin directory.
$ sudo cp prometheus promtool /usr/local/bin

Copy the libraries and console file to the /etc/prometheus directory
$ sudo cp -r consoles/ console_libraries/ /etc/prometheus/

Copy the configuration file to the /etc/prometheus directory.
$ sudo cp prometheus.yml /etc/prometheus/

Change the ownership of the files and directories.
$ sudo chown -R prometheus:prometheus /etc/prometheus/ /var/lib/prometheus
/usr/local/bin/{promtool,prometheus}

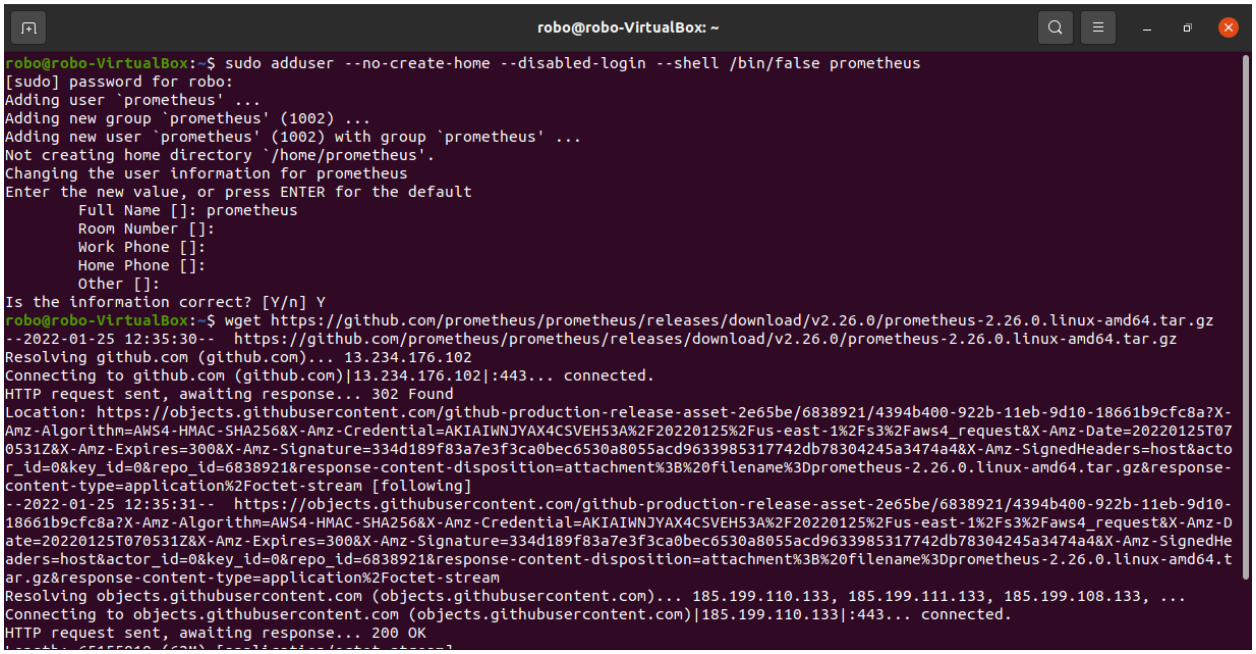
Create a service file to allow Prometheus to run in the background
$ sudo nano /etc/systemd/system/prometheus.service

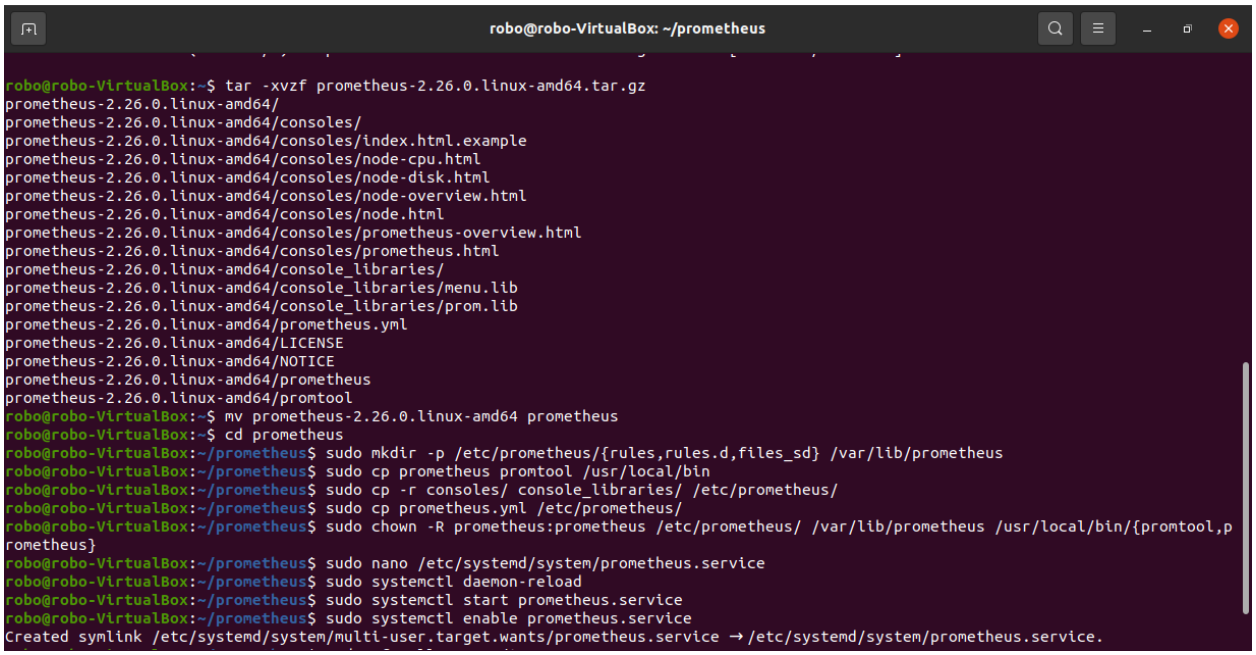
```
[Unit]
Description=Prometheus
[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart= /usr/local/bin/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/var/lib/prometheus \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries \
--web.listen-address=0.0.0.0:9090
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
```

Reload the daemon and start the service.

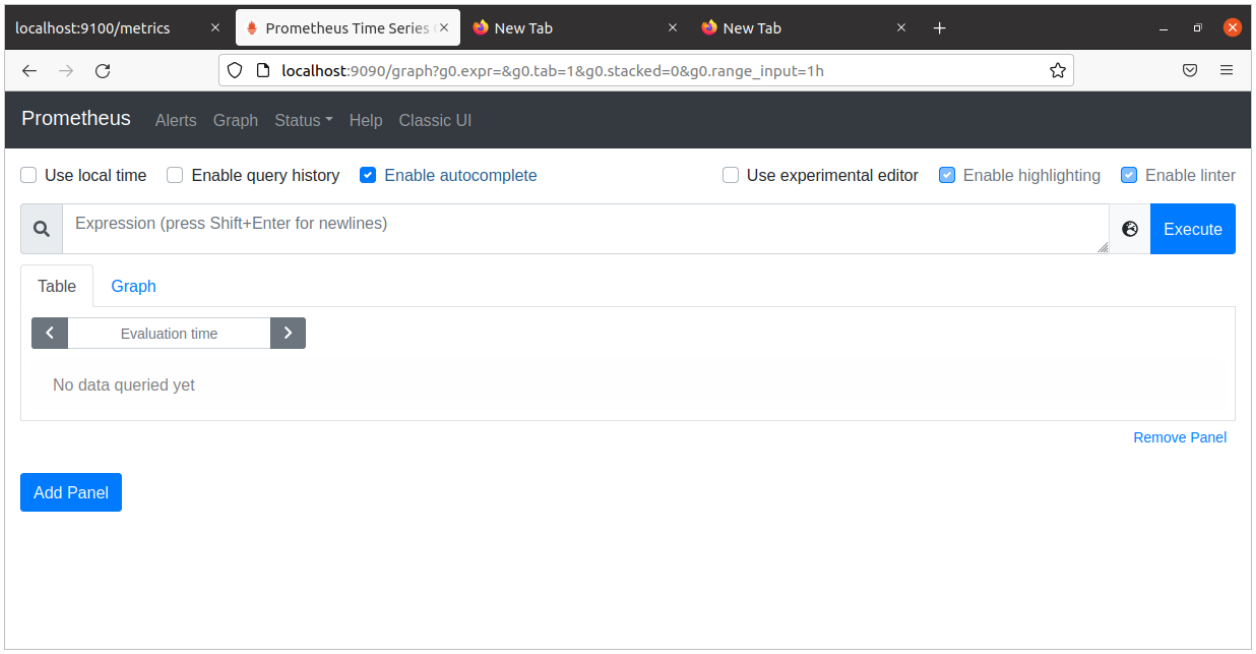
$ sudo systemctl daemon-reload
$ sudo systemctl start prometheus.service
$ sudo systemctl enable prometheus.service

In case of Webdock Perfect Server stack, open the port 9090.
$ sudo ufw allow 9090/tcp





Visit port 9090 of the server in a browser (e.g. http://mysite.com:9090) and it will show the following web page.



Installing Grafana

Update the system.
$ sudo apt-get update -y

Add gpg key for Grafana.
$ sudo apt-get install gnupg2 curl -y
$ curl https://packages.grafana.com/gpg.key | $ sudo apt-key add -

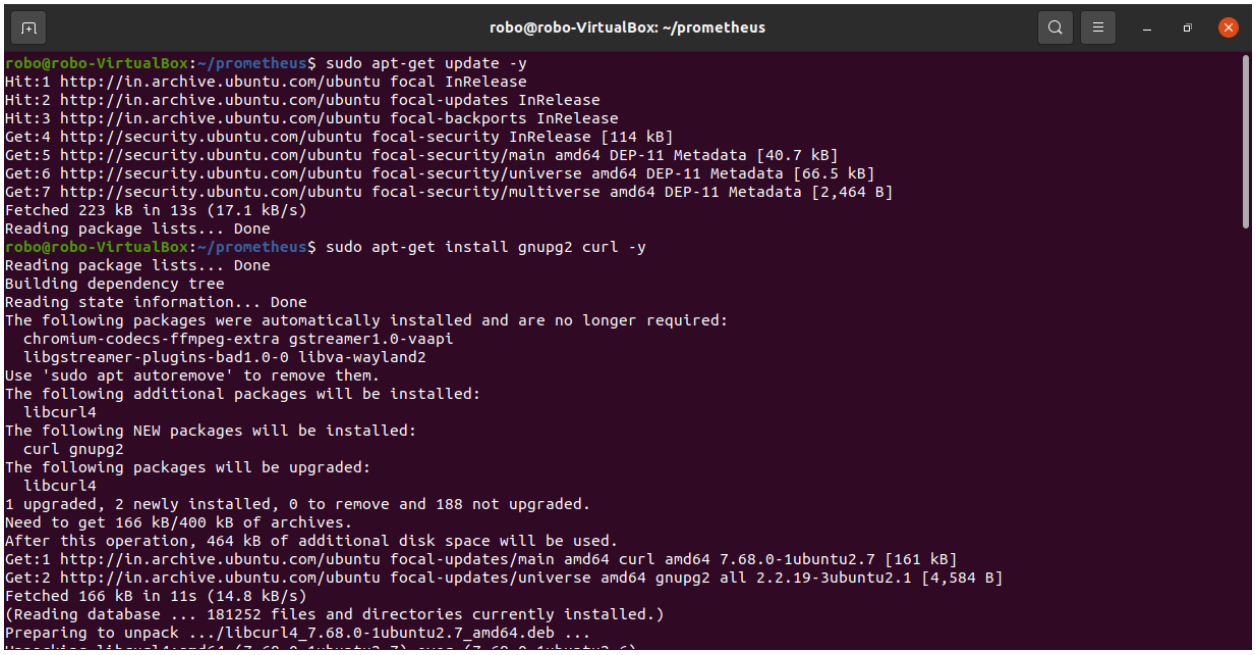
Add apt-repository for Grafana.
$ sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"

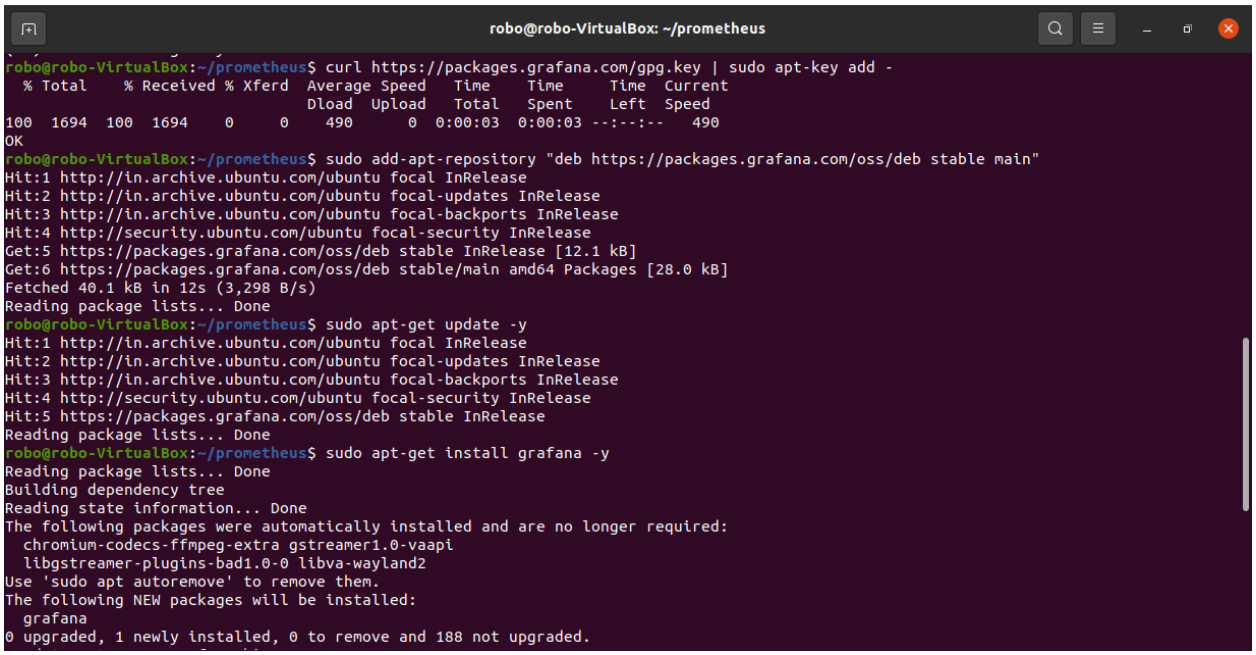
Update the apt-repositories on ubuntu.
$ sudo apt-get update -y

Install Grafana using the following command.
$ sudo apt-get install grafana -y

Reload the daemon and start the Grafana server.
$ sudo systemctl daemon-reload
$ sudo systemctl start grafana-server.service
$ sudo systemctl enable grafana-server.service

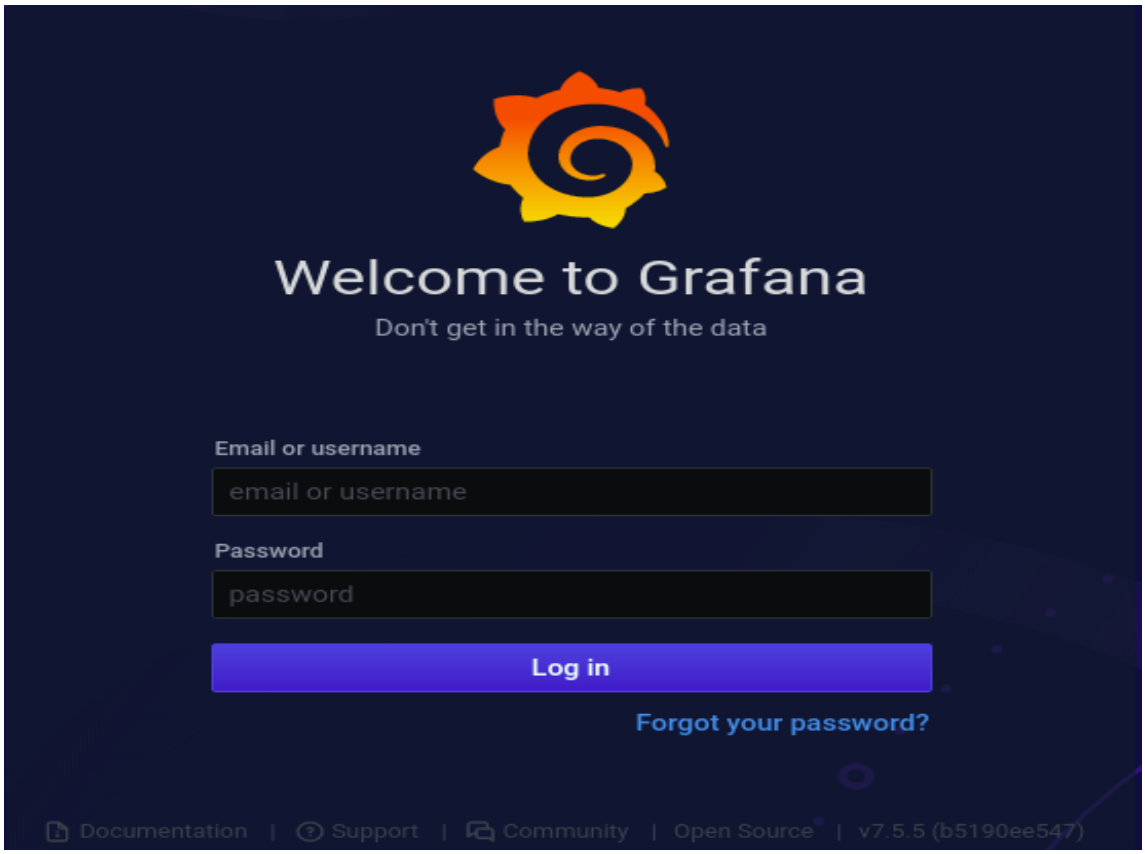
In case of a Webdock Perfect Server stack, open the port 3000.
$ sudo ufw allow 3000/tcp





```
robo@robo-VirtualBox: ~/prometheus
31% [1 grafana 28.5 MB/72.7 MB 39%]                                                8,088 B/s 1h 3
Get:1 https://packages.grafana.com/oss/deb stable/main amd64 grafana amd64 8.3.4 [72.7 MB]
73% [1 grafana 66.3 MB/72.7 MB 91%]                                                2,726 B/s 39
Get:1 https://packages.grafana.com/oss/deb stable/main amd64 grafana amd64 8.3.4 [72.7 MB]
Fetched 6,456 kB in 1h 25min 7s (1,264 B/s)
Selecting previously unselected package grafana.
(Reading database ... 181265 files and directories currently installed.)
Preparing to unpack .../grafana_8.3.4_amd64.deb ...
Unpacking grafana (8.3.4) ...
Setting up grafana (8.3.4) ...
Adding system user `grafana' (UID 127) ...
Adding new user `grafana' (UID 127) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to configure grafana to start automatically using systemd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
 sudo /bin/systemctl start grafana-server
Processing triggers for systemd (245.4-4ubuntu3.11) ...
robo@robo-VirtualBox:~/prometheus$ sudo systemctl daemon-reload
[sudo] password for robo:
robo@robo-VirtualBox:~/prometheus$ sudo systemctl start grafana-service
Failed to start grafana-service.service: Unit grafana-service.service not found.
robo@robo-VirtualBox:~/prometheus$ sudo systemctl start grafana-server.service
robo@robo-VirtualBox:~/prometheus$ sudo systemctl enable grafana-server.serviceSynchronizing state of grafana-server.service with SysV
 service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /lib/systemd/system/grafana-server.service.
robo@robo-VirtualBox:~/prometheus$ sudo ufw allow 3000/tcp
Rules updated
Rules updated (v6)
robo@robo-VirtualBox:~/prometheus$
```

Visit port 3000 of the server in a browser (e.g. http://mysite.com:3000) of the server and it will display a login page.



Adding Targets to Prometheus

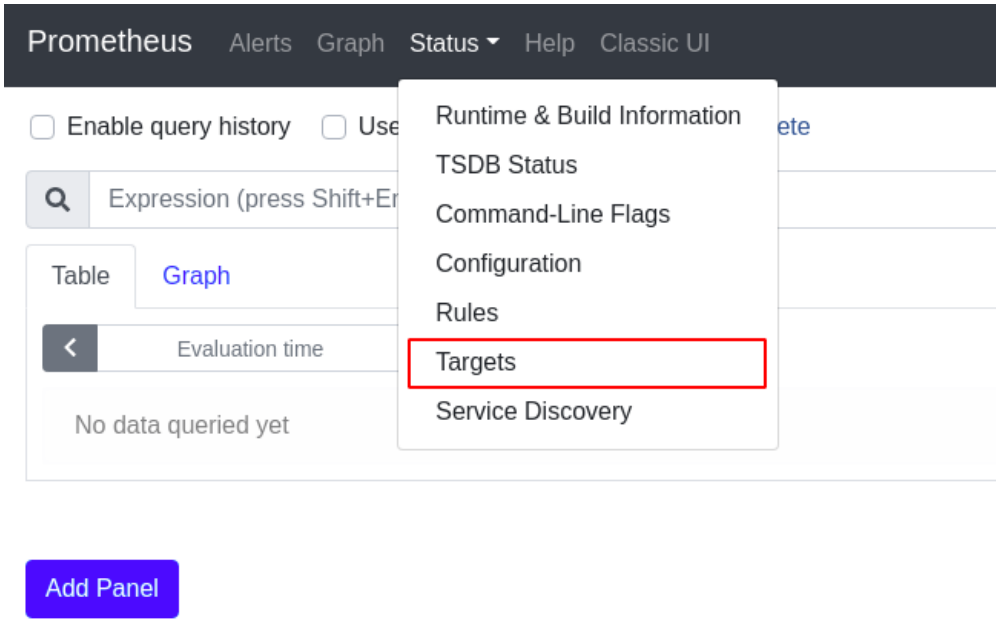
Open the prometheus.yml file.
$ sudo nano /etc/prometheus/prometheus.yml

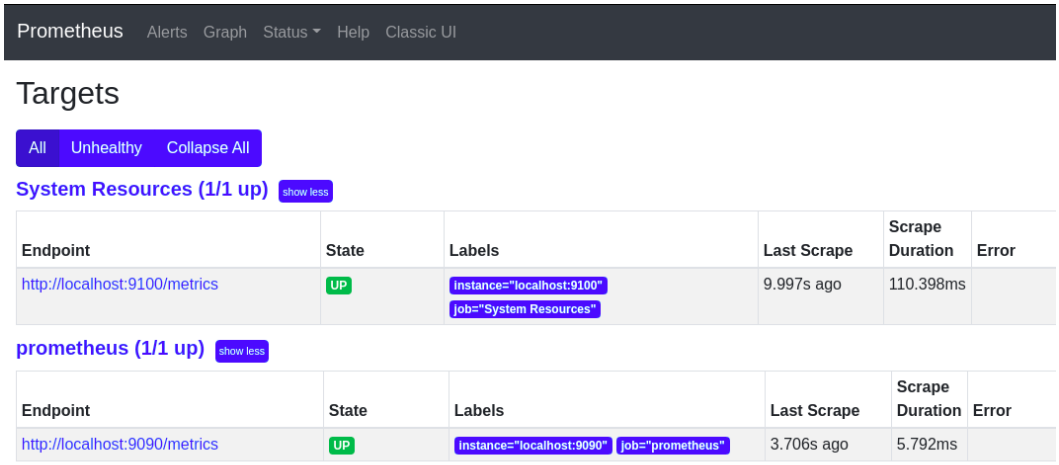
Add the following configuration in the file.

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
scrape_configs:
  - job_name: 'System Resources'
    scrape_interval: 10s
    static_configs:
    - targets: ['localhost:9100']
  - job_name: 'Prometheus'
    scrape_interval: 10s
    static_configs:
    - targets: ['localhost:9090']
```

Restart the prometheus service.
$ sudo systemctl restart prometheus

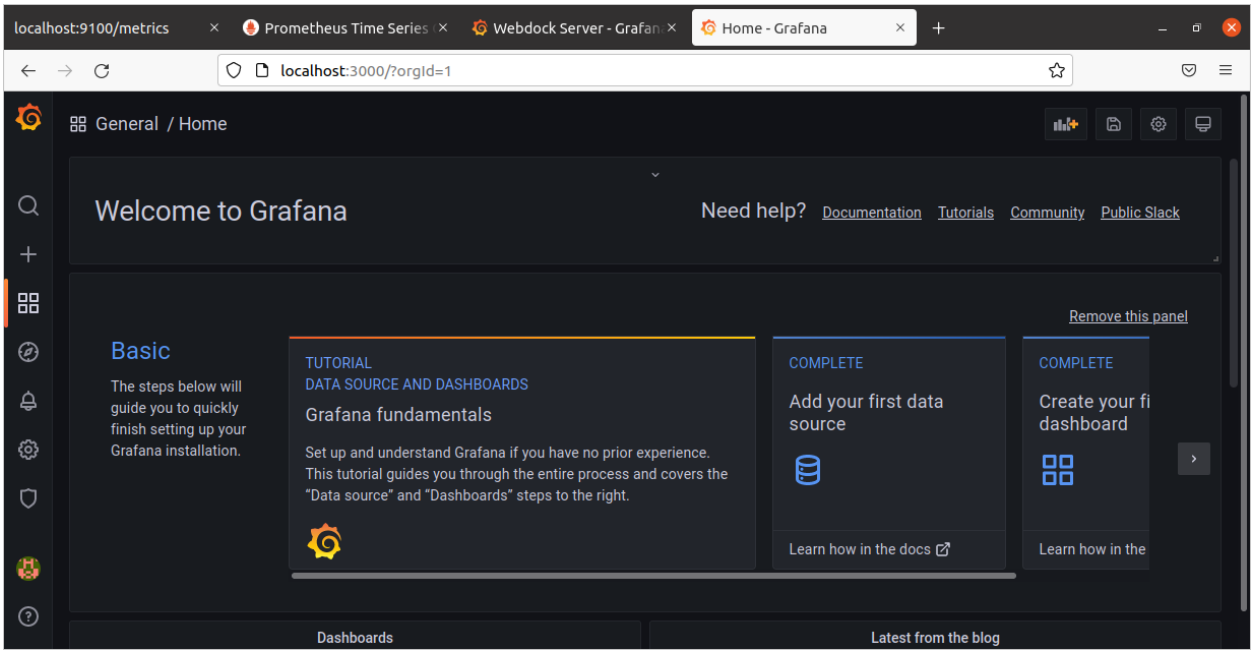
Visit port 9090 of the server in a browser (e.g. http://mysite.com:9090 and list all the targets.

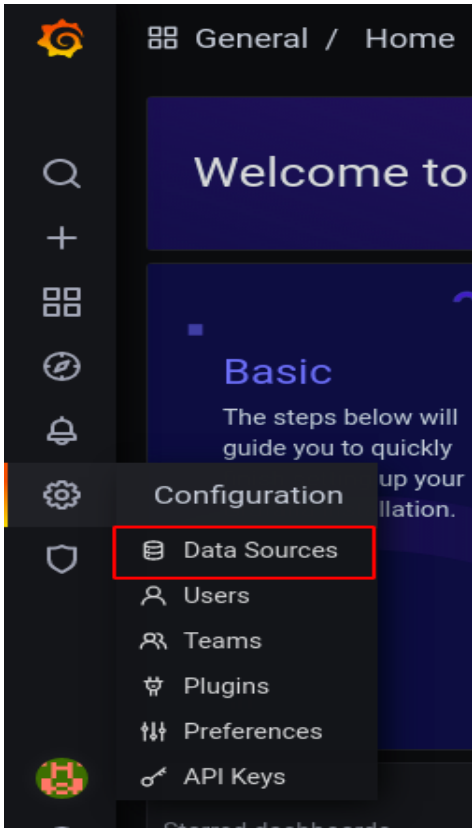




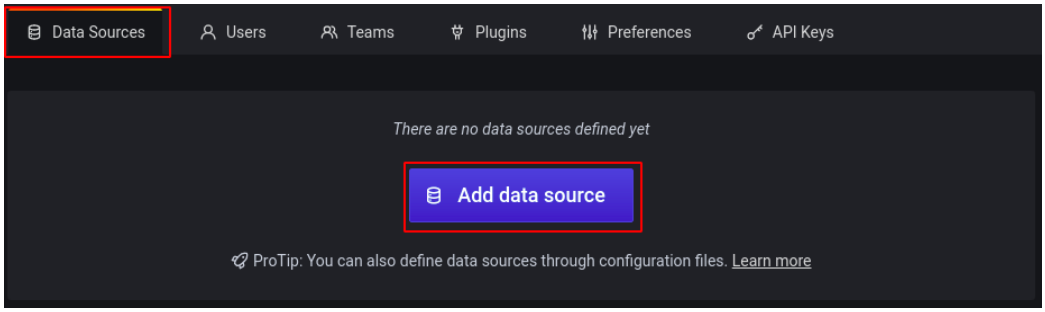
Adding Data Source and Dashboard to Grafana

Log into the Grafana Dashboard using the default username and password (i.e admin). Hover over the settings icon on the left side menu and then click on ‘Data Sources’.

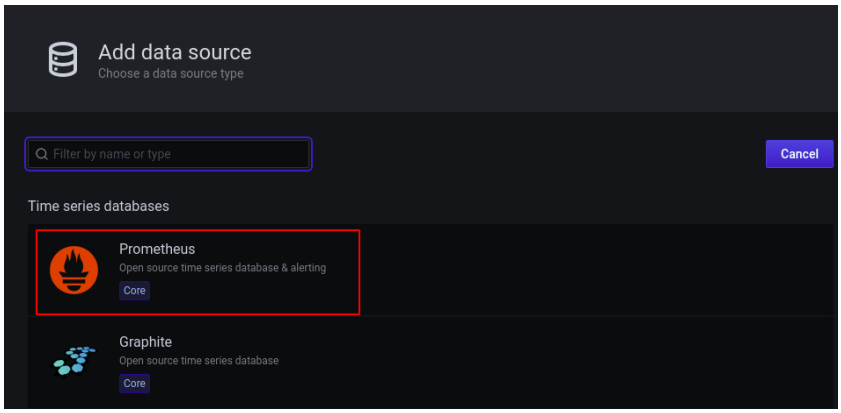




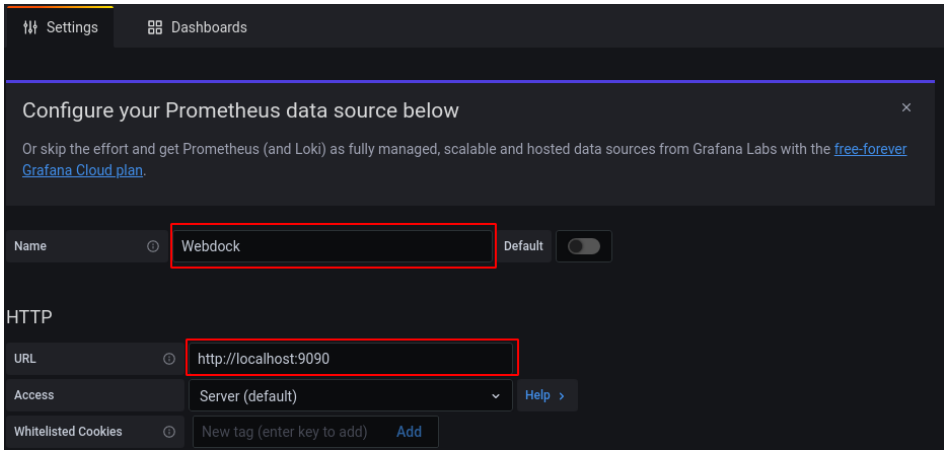
It will open a configuration page. Select the ‘Data Sources’ tab and click on ‘Add data source’.



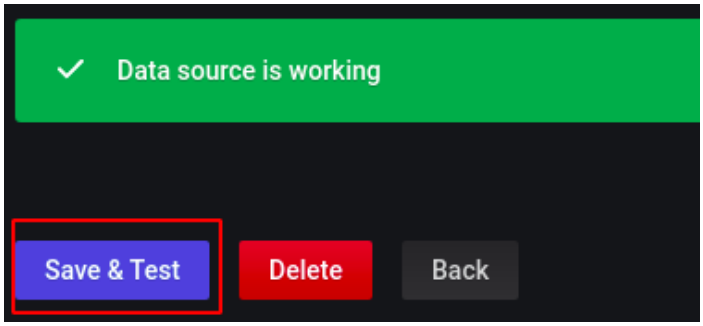
On the ‘Add data source’ page select the Prometheus as data source.



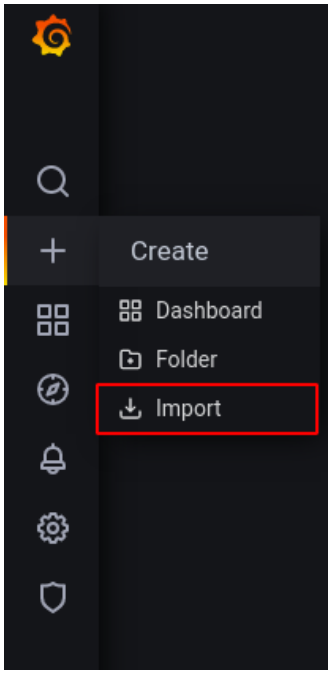
Provide the name and the url of the Prometheus data source.



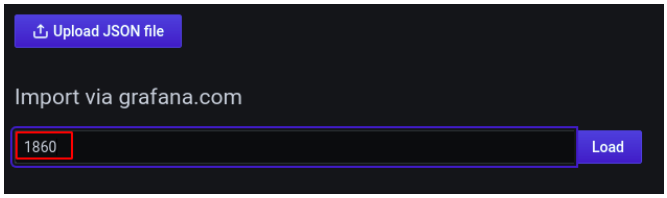
Click on the ‘save & test’ button at the end of the page and it will show the status of the Data Source.



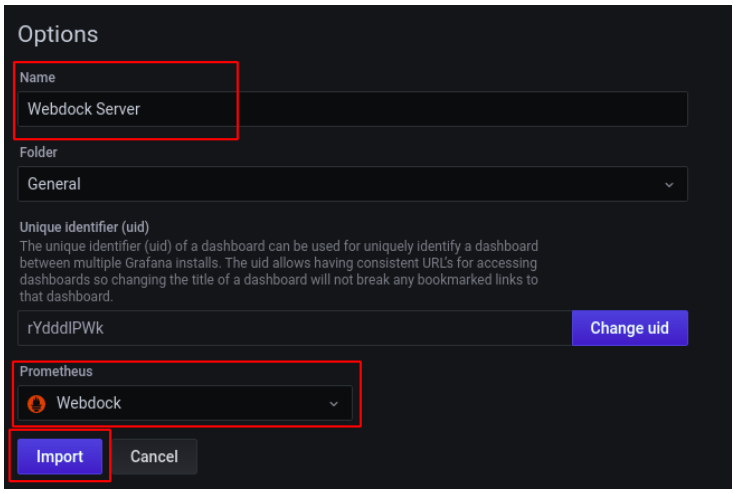
In order to import Grafana dashboard, hover over the ‘Add’ symbol and click on ‘import’.



Enter the Grafana dashboard id and click on ‘load’.

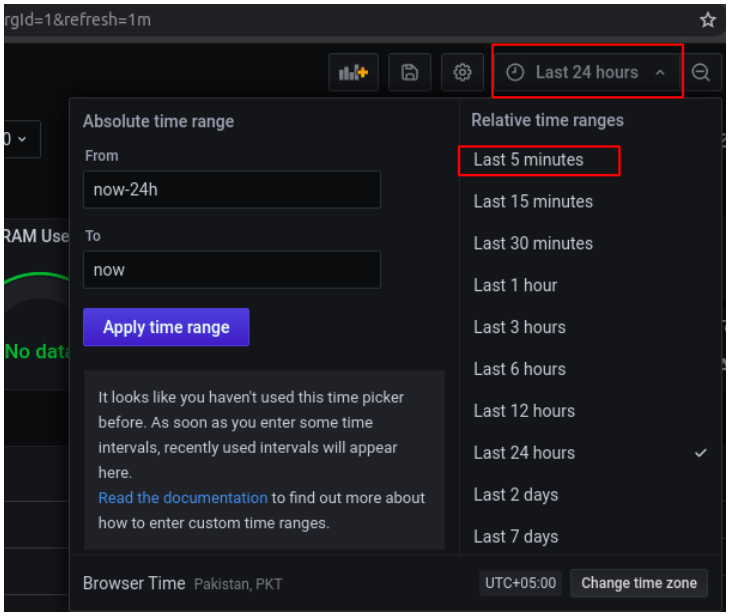


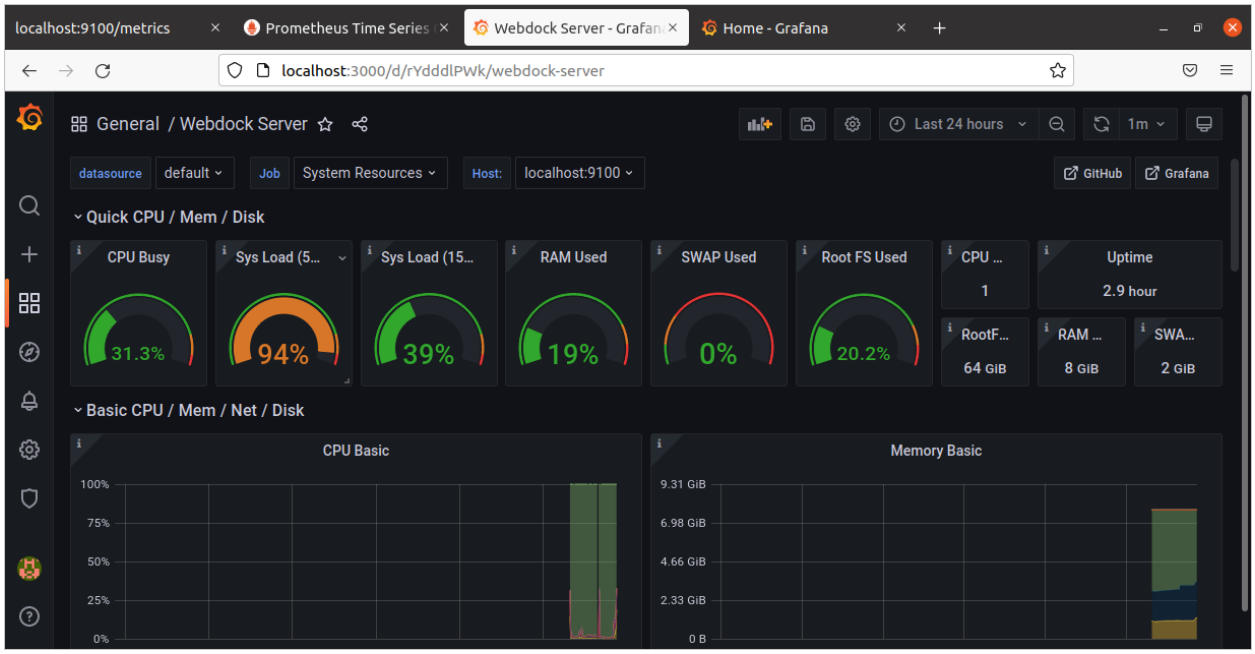
After loading the dashboard, it will ask for the dashboard name and the Prometheus data source. Click on import after providing the required details.

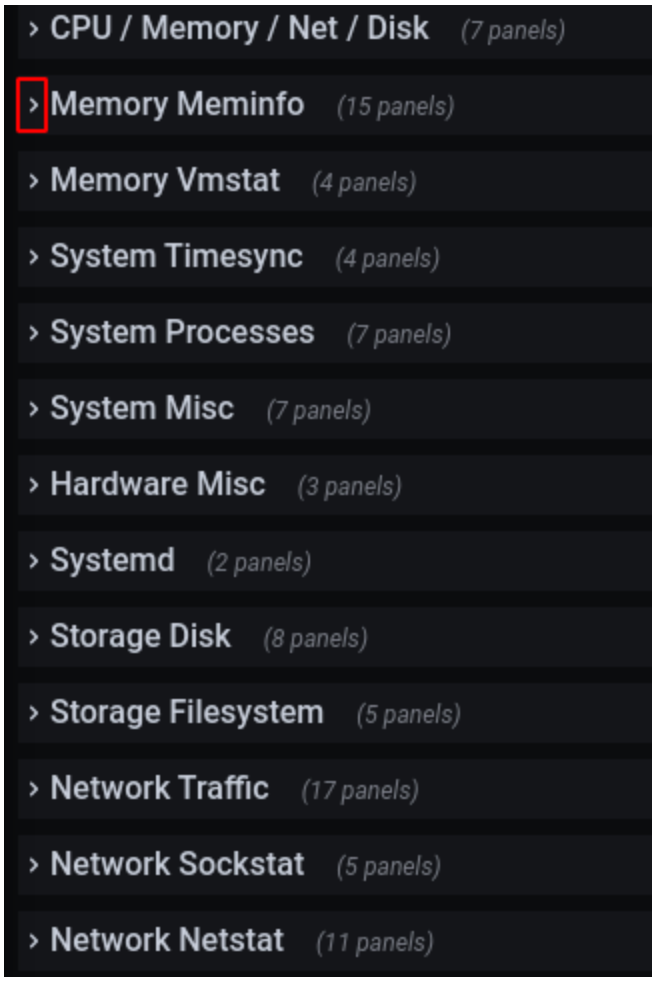


Monitoring System Resources using Grafana

After importing the dashboard, it will display a monitoring dashboard. Click on the time range selector on the top right corner of the dashboard and select a time range.




localhost:9100/metrics
Prometheus Time Series
Webdock Server - Grafan
Home - Grafana
localhost:3000/d/rYdddlPWk/webdock-server
General / Webdock Server
Last 24 hours
1m
datasource
default
Job
System Resources
Host:
localhost:9100
GitHub
Grafana
Quick CPU / Mem / Disk
CPU Busy
31.3%
Sys Load (5...
94%
Sys Load (15...
39%
RAM Used
19%
SWAP Used
0%
Root FS Used
20.2%
CPU ...
1
Uptime
2.9 hour
RootF...
64 GiB
RAM ...
8 GiB
SWA...
2 GiB
Basic CPU / Mem / Net / Disk
CPU Basic
100%
75%
50%
25%
0%
Memory Basic
9.31 GiB
6.98 GiB
4.66 GiB
2.33 GiB
0 B


CPU / Memory / Net / Disk (7 panels)
Memory Meminfo (15 panels)
Memory Vmstat (4 panels)
System Timesync (4 panels)
System Processes (7 panels)
System Misc (7 panels)
Hardware Misc (3 panels)
Systemd (2 panels)
Storage Disk (8 panels)
Storage Filesystem (5 panels)
Network Traffic (17 panels)
Network Sockstat (5 panels)
Network Netstat (11 panels)

Installing Prometheus Alert Manager

Create an alert_manager user without a home directory and no login permissions.
$ sudo adduser --no-create-home --disabled-login --shell /bin/false alert_manager

Download the alert manager binary file.
$ wget https://github.com/prometheus/alertmanager/releases/download/v0.22.0-rc.2/alertmanager-0.22.0-rc.2.linux-amd64.tar.gz

Unzip the downloaded binary file.
$ tar -xvzf alertmanager-0.22.0-rc.2.linux-amd64.tar.gz

Rename the extracted directory.
$ mv alertmanager-0.22.0-rc.2.linux-amd64 alertmanager

Go to the alertmanager directory.
$ cd alertmanager

Create a data directory for alertmanager.
$ sudo mkdir /var/lib/alertmanager

Copy the amtool and alertmanager executable files to /usr/local/bin directory.
$ sudo cp amtool alertmanager /usr/local/bin/

Copy the alertmanager.yml file to the /etc/prometheus/ directory.
$ sudo cp alertmanager.yml /etc/prometheus/

Change the ownership of the executable files and data directory.
$ sudo chown -R alert_manager:alert_manager /usr/local/bin/{amtool,alertmanager} /var/lib/alertmanager

Create a service file to run the alertmanager in the background.
$ sudo nano /etc/systemd/system/alertmanager.service

```
[Unit]

Description=Alert Manager

[Service]

User=alert_manager

Group=alert_manager

Type=simple

ExecStart= /usr/local/bin/alertmanager \

--config.file=/etc/prometheus/alertmanager.yml \

--storage.path=/var/lib/alertmanager/ \
```

--cluster.listen-address=127.0.0.1:9094

Restart=on-failure

RestartSec=5

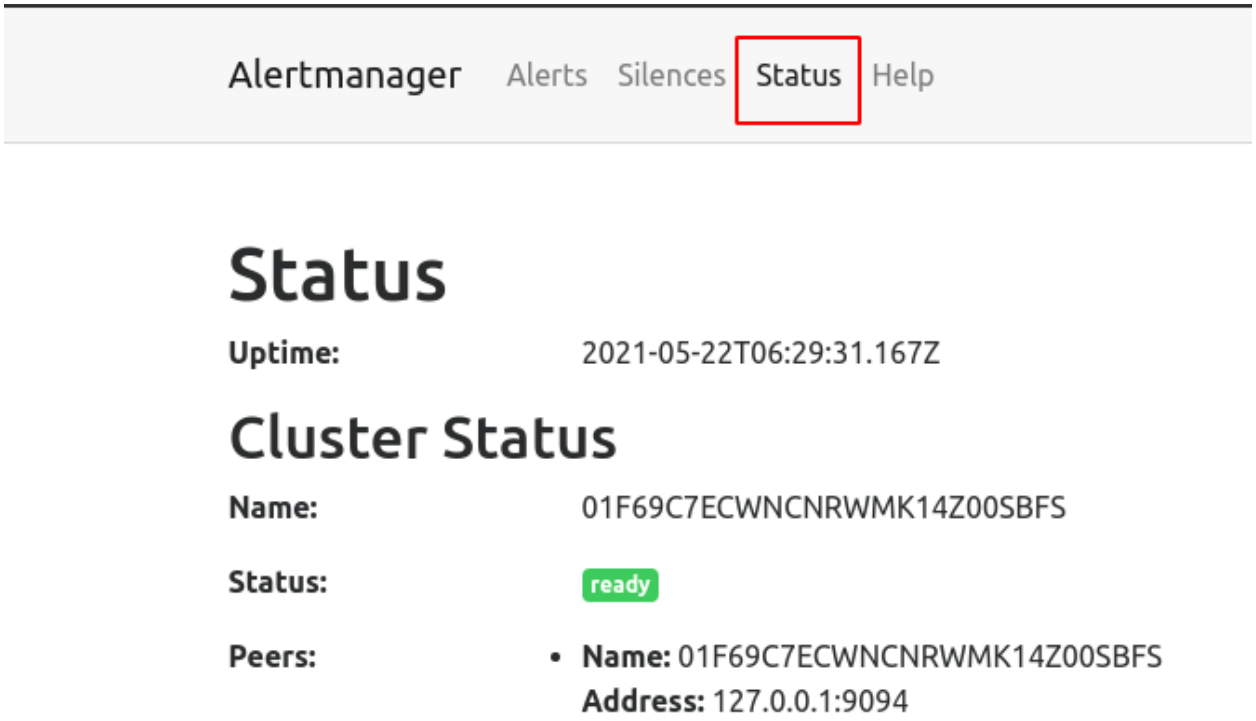
[Install]

WantedBy=multi-user.target

Reload the daemon and start the service.
$ sudo systemctl daemon-reload
$ sudo systemctl start alertmanager.service
$ sudo systemctl enable alertmanager.service

In case of a Webdock Perfect Server stack, open port 9094 and 9093.
$ sudo ufw allow 9094/tcp
$ sudo ufw allow 9093/tcp

Visit port 9093 of the server in a browser (e.g. http://mysite.com:9093) and check the alert manager cluster status.



Configuring Alert Manager

Open the prometheus.yml file and append the following configuration.

$ sudo nano /etc/prometheus/prometheus.yml

```
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - 'localhost:9093'
rule_files:
  - "rules.yml"
```

Create a file named ‘rules.yml’ at /etc/prometheus directory and add the alert rules there.

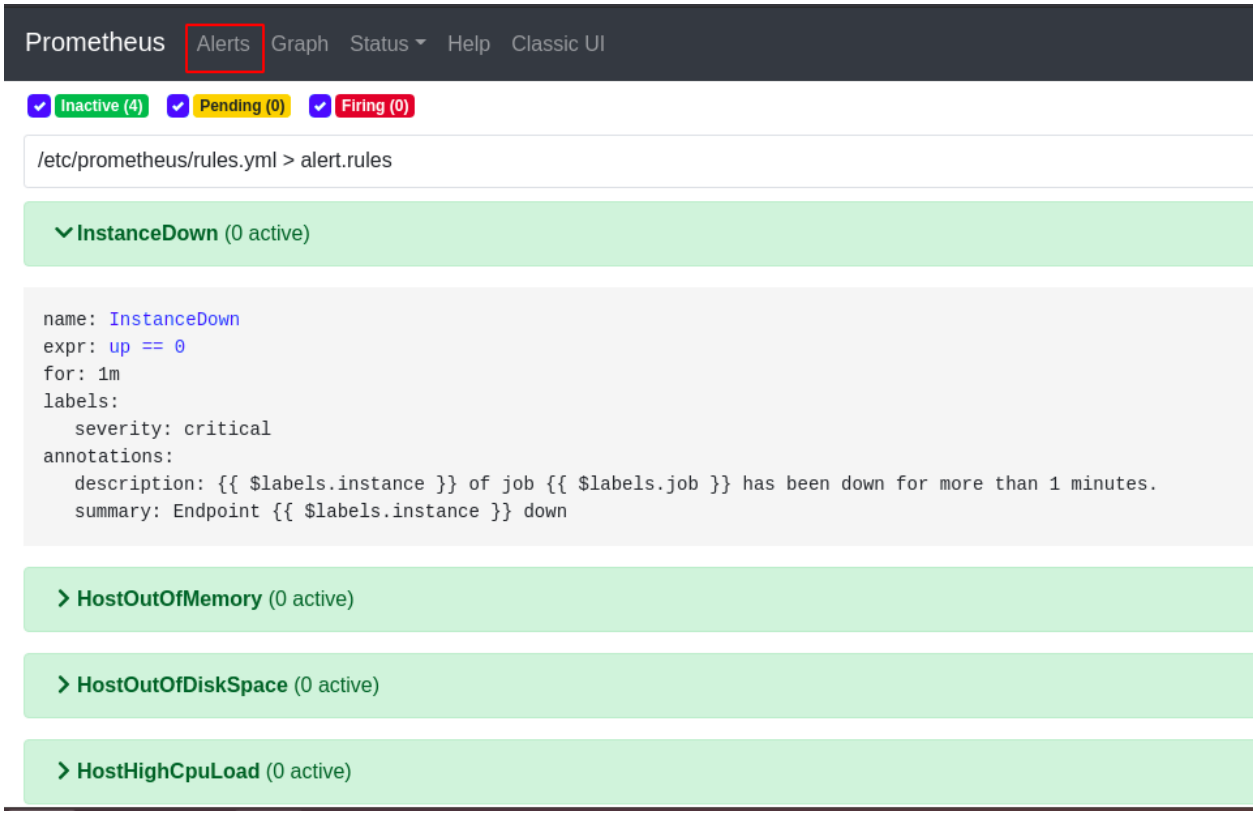
$ sudo nano /etc/prometheus/rules.yml

```
groups:
- name: alert.rules
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
    labels:
      severity: "critical"
    annotations:
      summary: "Endpoint {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 1 minutes."
  - alert: HostOutOfMemory
    expr: node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 25
    for: 5m
    labels:
      severity: warning
    annotations:
      summary: "Host out of memory (instance {{ $labels.instance }})"
      description: "Node memory is filling up (< 25% left)\n VALUE = {{ $value }}\n LABELS: {{ $labels }}"
  - alert: HostOutOfDiskSpace
    expr: (node_filesystem_avail_bytes{mountpoint="/"} * 100) / node_filesystem_size_bytes{mountpoint="/"} < 50
    for: 10s
    labels:
      severity: warning
    annotations:
```

```
    summary: "Host out of disk space (instance {{ $labels.instance }})"
    description: "Disk is almost full (< 50% left)\n VALUE = {{ $value }}\n LABELS: {{ $labels }}"
  - alert: HostHighCpuLoad
    expr: (sum by (instance) (irate(node_cpu_seconds_total{job="System Resources",mode="idle"}[5m]))) > 80
    for: 5m
    labels:
      severity: warning
    annotations:
      summary: "Host high CPU load (instance {{ $labels.instance }})"
      description: "CPU load is > 80%\n VALUE = {{ $value }}\n LABELS: {{ $labels }}"
```

Restart the Prometheus service to apply these changes.
$ sudo systemctl restart prometheus.service

Visit port 9090 of the server in a browser (e.g. http://mysite.com:9090 and click on the ‘Alert’ button and it will show all the alerts configured in the ‘rules.yml’ file.



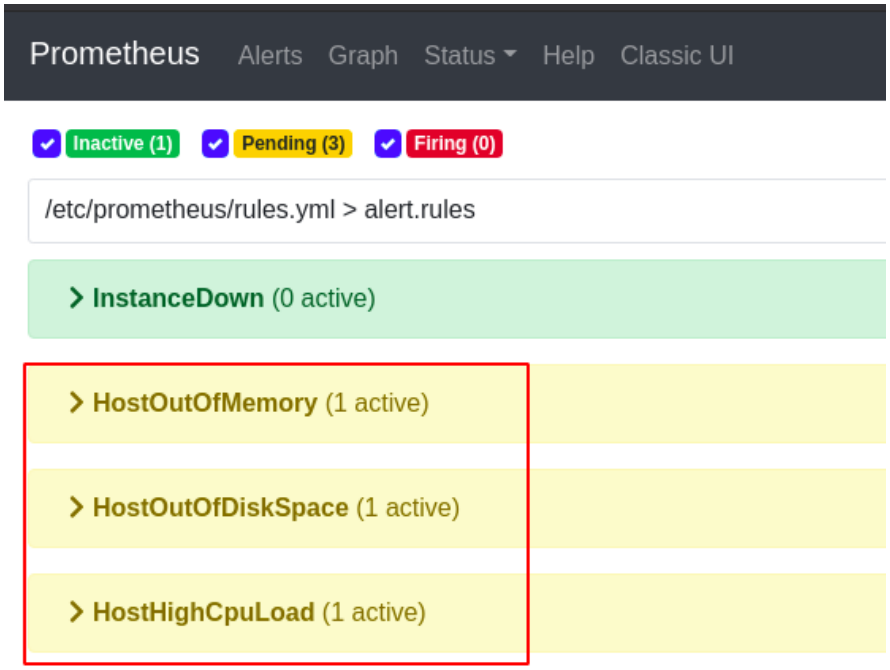
Configuring Email Notification

Open the alertmanager.yml file and add the email configuration.
$ sudo nano /etc/prometheus/alertmanager.yml

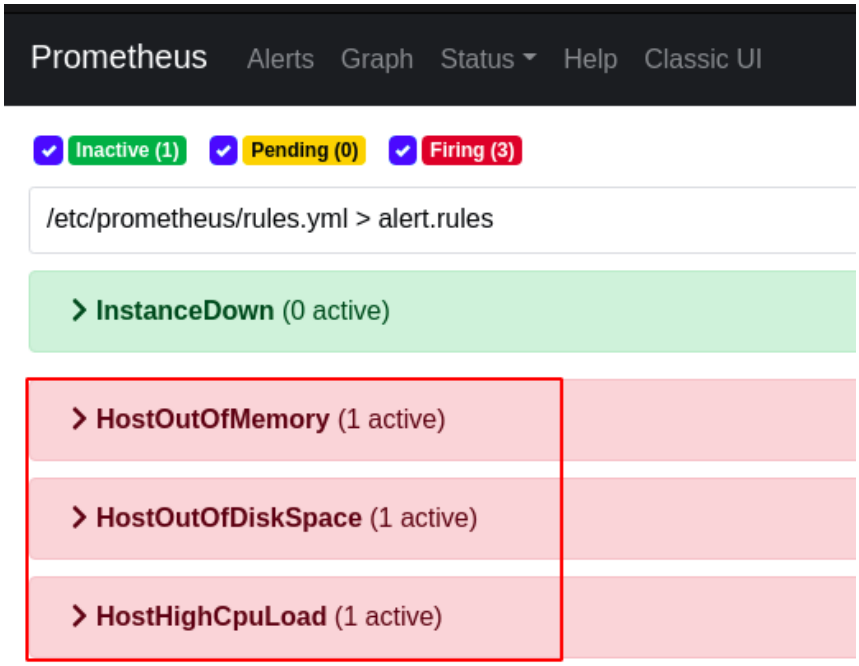
```
global:
  resolve_timeout: 1m
route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
  receiver: 'email_notification'
receivers:
- name: 'email_notification'
  email_configs:
  - to : "to"
    from: "from"
    smarthost: smtp.gmail.com:587
    auth_username: "email"
    auth_identity: "email"
    auth_password: "password"
    send_resolved: true
inhibit_rules:
  - source_match:
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'dev', 'instance']
```

Restart the alert manager and prometheus services.
$ sudo systemctl restart prometheus.service
$ sudo systemctl restart alertmanager.service

Now whenever system resources cross a specific threshold defined in rules.yml file, the status of alerts will change from ‘inactive’ to ‘pending’.



And after some time, the status will be in ‘Firing’ state and an email notification will be sent.



Installing Blackbox for URL Monitoring

Download the alert manager binary file.
$ wget
https://github.com/prometheus/blackbox_exporter/releases/download/v0.19.0/blackbox_exporter-0.19.0.linux-amd64.tar.gz

Unzip the downloaded binary file.
$ tar -xvzf blackbox_exporter-0.19.0.linux-amd64.tar.gz

Go to the alertmanager directory.
$ cd blackbox_exporter-0.19.0.linux-amd64

Create a config file
$ sudo nano monitor_website.yml

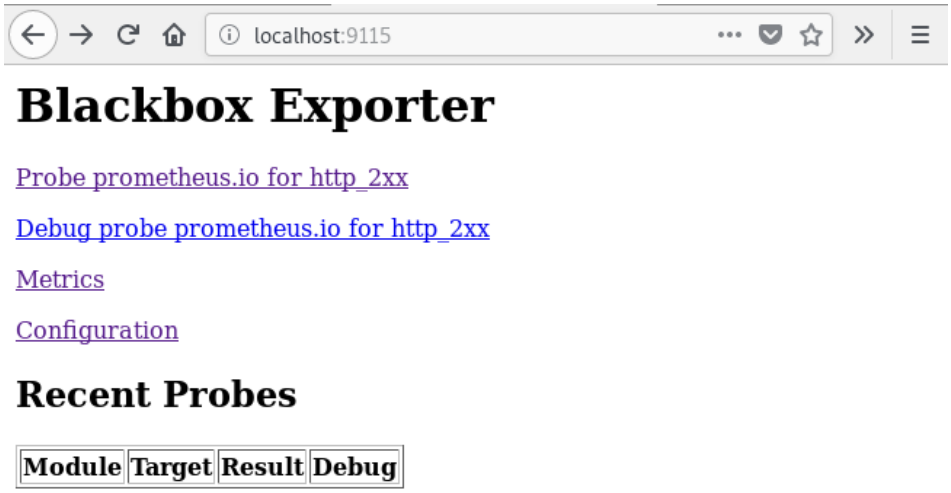
```
modules:
  http_2xx:
    prober: http
    timeout: 5s
    http:
      valid_http_versions: ["HTTP/1.1", "HTTP/2.0"]
      valid_status_codes: []  # Defaults to 2xx
      method: GET
```

Create a service file for the Blackbox exporter
$ sudo nano /etc/systemd/system/blackbox.service

```
[Unit]
Description=Blackbox Exporter Service
Wants=network-online.target
After=network-online.target
[Service]
Type=simple
User=root
ExecStart=/home/ubuntu/blackbox_exporter-0.19.0.linux-amd64/blackbox_exporter \
  --config.file=/home/ubuntu/blackbox_exporter-0.19.0.linux-amd64/monitor_website.yml \
  --web.listen-address=":9115"
Restart=always
[Install]
WantedBy=multi-user.target
```

Start blackbox service
$ sudo systemctl start blackbox.service

Open localhost:9115 and check the output as:



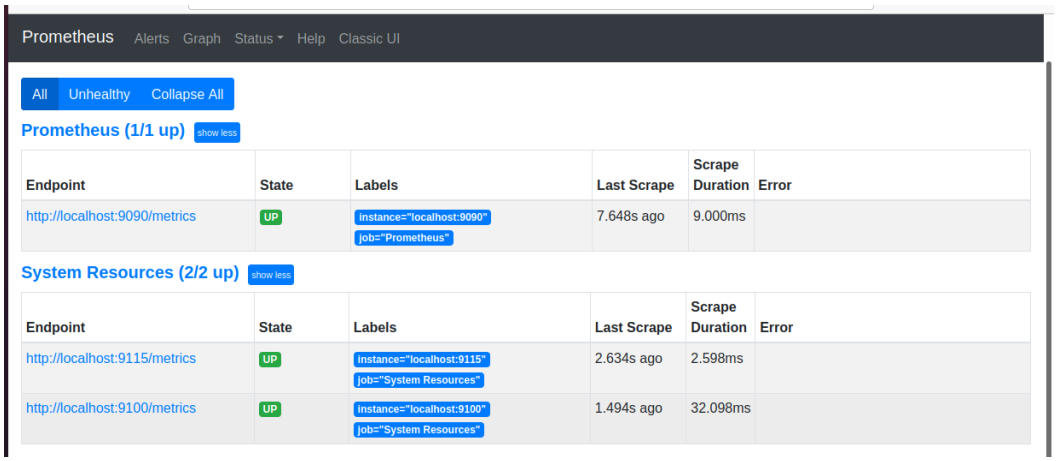
Update Prometheus config file
$ sudo nano /etc/prometheus/prometheus.yml

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


scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets: ['localhost:9090', 'localhost:9115']


  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_prometheus]
    static_configs:
      - targets:
        - https://127.0.0.1:80
        - https://127.0.0.1:81
    relabel_configs:
```

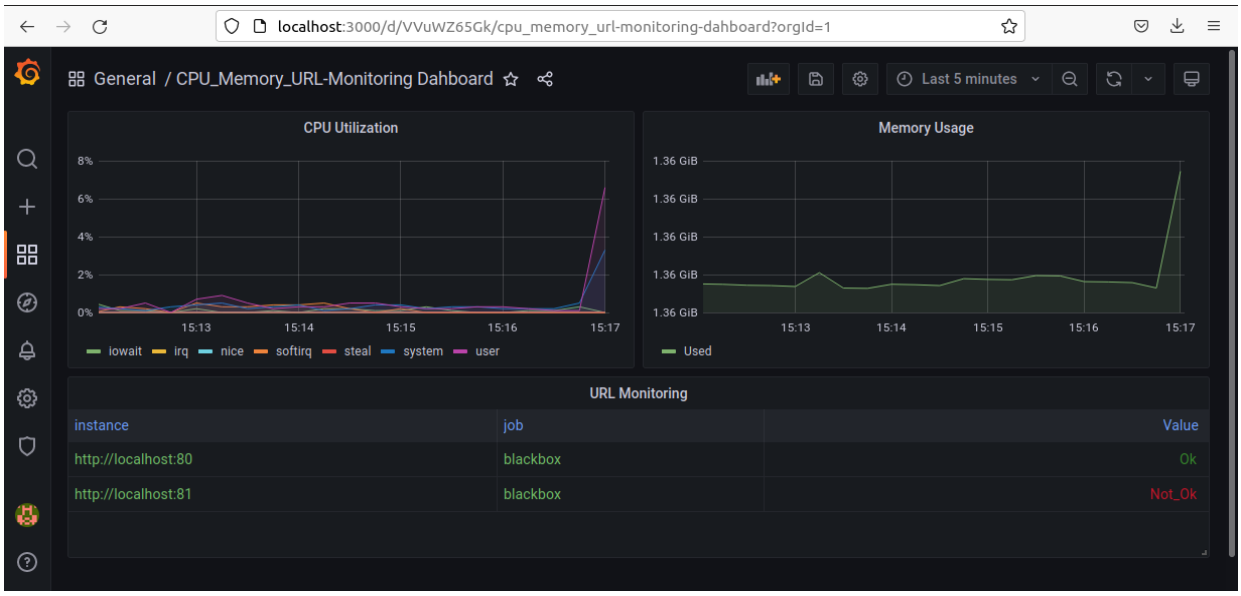
```
- source_labels: [__address__]
  target_label: __param_target
- source_labels: [__param_target]
  target_label: instance
- target_label: __address__
  replacement: 127.0.0.1:9115  # The blackbox exporter's real hostname:port.
```

Restart Prometheus service
$ sudo systemctl daemon-reload
$ sudo systemctl restart prometheus

Open localhost:9090/targets and check the new target as 9115.



Goto Grafana and configure new panel as URL monitoring with query as:
Probe_http_status_code



Conclusion

Node exporter, Prometheus and Grafana work together to display all the system metrics on a single dashboard. Prometheus alert manager is a tool used to alert when a specific system metric crosses a threshold specified in rules.yml file. Blackbox exporter is used to collect website related metrics to monitor them.