Prometheus + Grafana

Author – SUSHANT AHER

https://www.linkedin.com/in/sushant-aher-devops/

What is Prometheus?

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open-source project and maintained independently of any company. To emphasize this, and to clarify the project's governance structure, Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

Prometheus 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.

For more elaborate overviews of Prometheus, see the resources linked from the media section.

Features

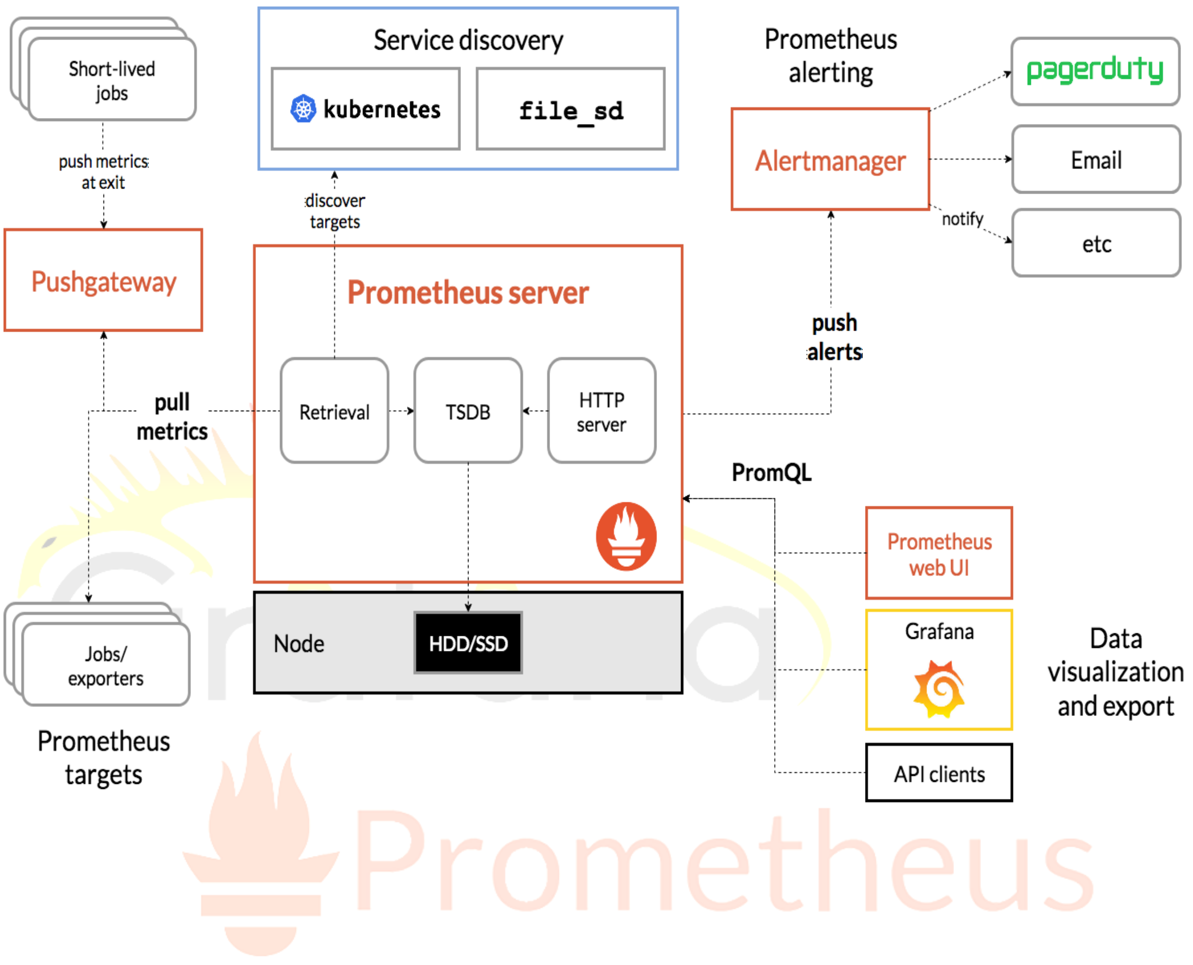
Prometheus's main features are:

- a multi-dimensional data model with time series data identified by metric name and key/value pairs
- PromQL, a flexible query language to leverage this dimensionality
- no reliance on distributed storage; single server nodes are autonomous
- time series collection happens via a pull model over HTTP
- pushing time series is supported via an intermediary gateway
- targets are discovered via service discovery or static configuration
- multiple modes of graphing and dashboarding support

What are metrics?

- Metrics are numerical measurements in layperson terms. The term time series refers to the recording of changes over time. What users want to measure differs from application to application.

Architecture –



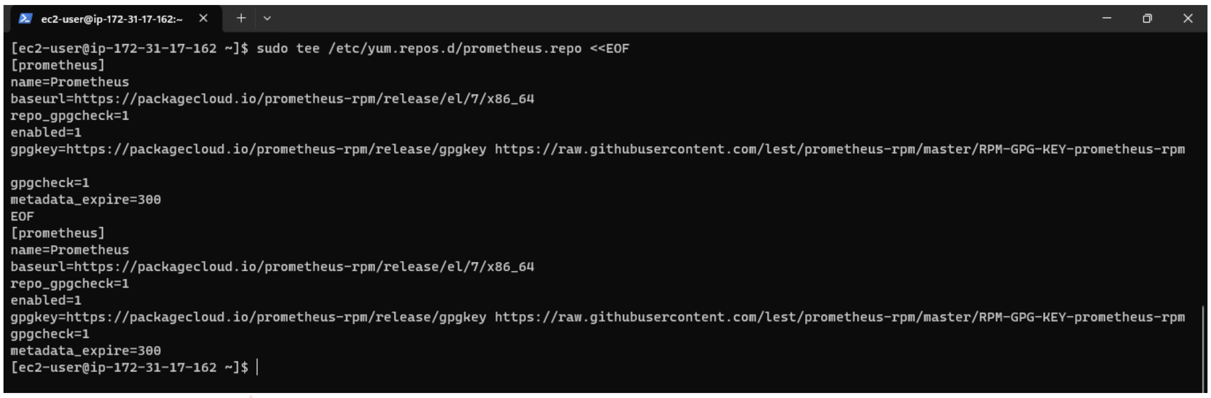
This diagram illustrates the architecture of Prometheus and some of its ecosystem components:

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.

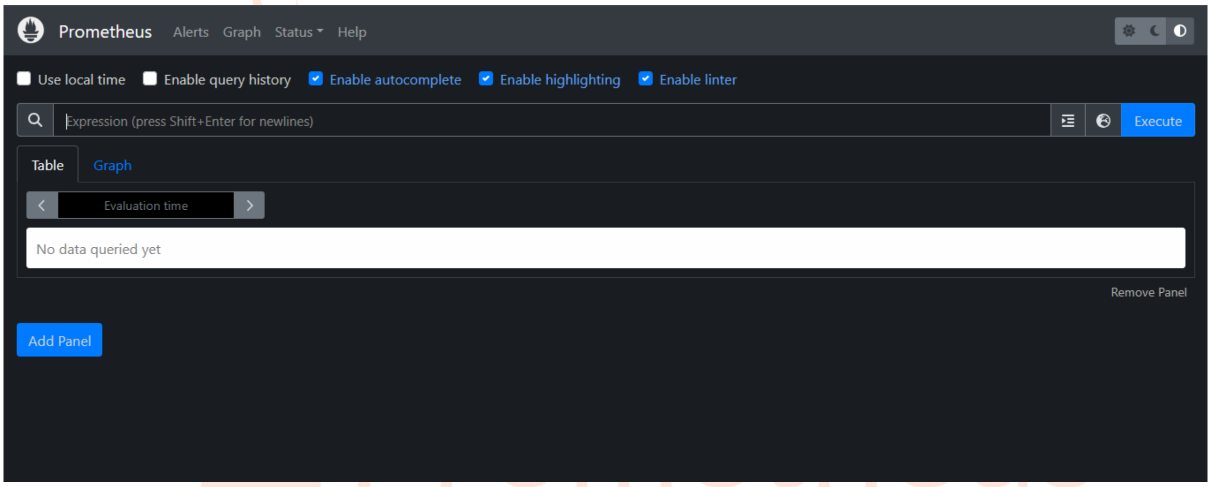
Installation

Githun repo –

https://github.com/SushantAher02/prome-grafana/blob/master/graphana%20installation.txt



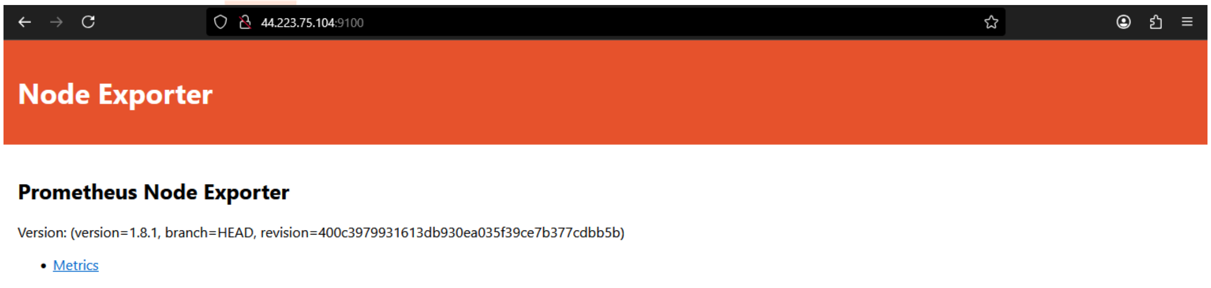
```
[ec2-user@ip-172-31-17-162 ~]$ sudo tee /etc/yum.repos.d/prometheus.repo <<EOF
[prometheus]
name=Prometheus
baseurl=https://packagecloud.io/prometheus-rpm/release/el/7/x86_64
repo_gpgcheck=1
enabled=1
gpgkey=https://packagecloud.io/prometheus-rpm/release/gpgkey https://raw.githubusercontent.com/lest/prometheus-rpm/master/RPM-GPG-KEY-prometheus-rpm

gpgcheck=1
metadata_expire=300
EOF
[prometheus]
name=Prometheus
baseurl=https://packagecloud.io/prometheus-rpm/release/el/7/x86_64
repo_gpgcheck=1
enabled=1
gpgkey=https://packagecloud.io/prometheus-rpm/release/gpgkey https://raw.githubusercontent.com/lest/prometheus-rpm/master/RPM-GPG-KEY-prometheus-rpm
gpgcheck=1
metadata_expire=300
[ec2-user@ip-172-31-17-162 ~]$
```



☐ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

Expression (press Shift+Enter for newlines)

No data queried yet

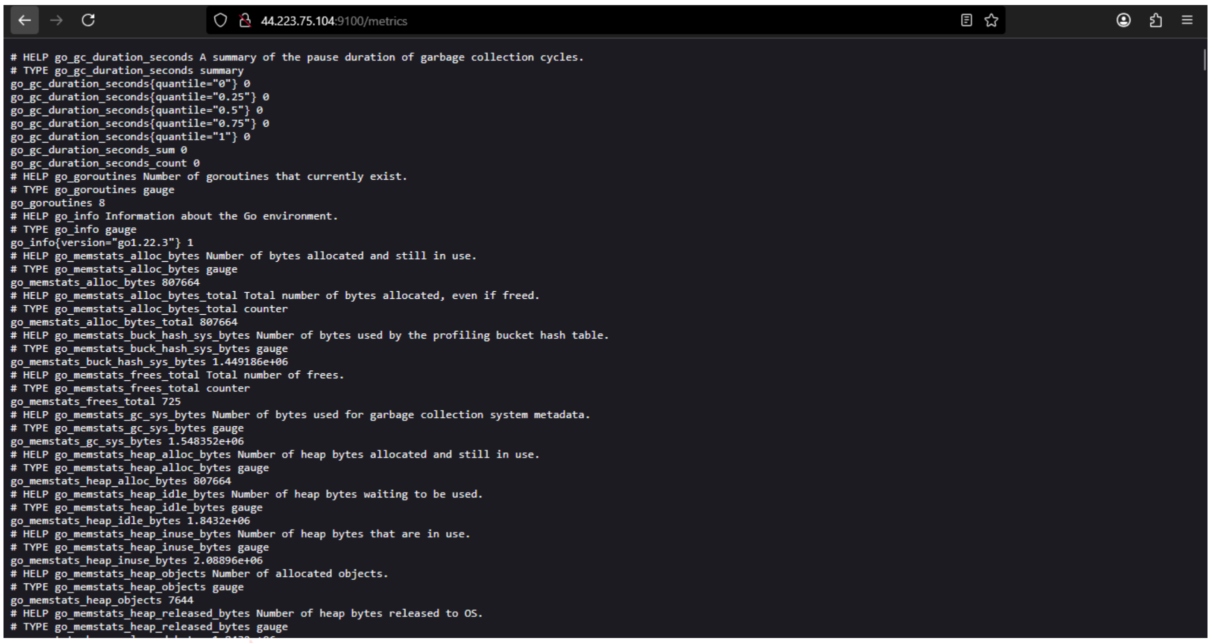


Node Exporter

Prometheus Node Exporter

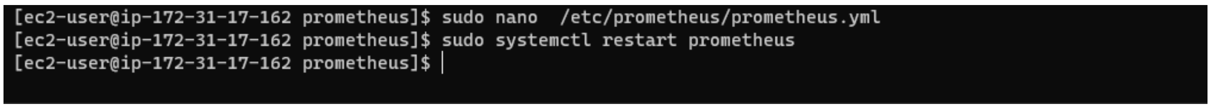
Version: (version=1.8.1, branch=HEAD, revision=400c3979931613db930ea035f39ce7b377cdbb5b)

- Metrics

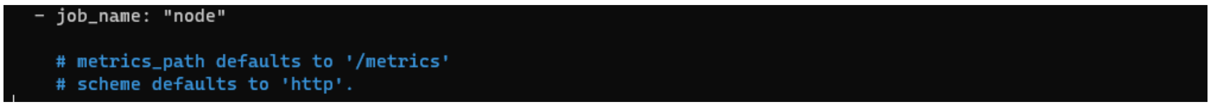

44.223.75.104:9100/metrics

```
# 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"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# 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="go1.22.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 807664
# 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 807664
# 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 1.449186e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 725
# 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 1.548352e+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 807664
# 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 1.8432e+06
# 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 2.08896e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 7644
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
```

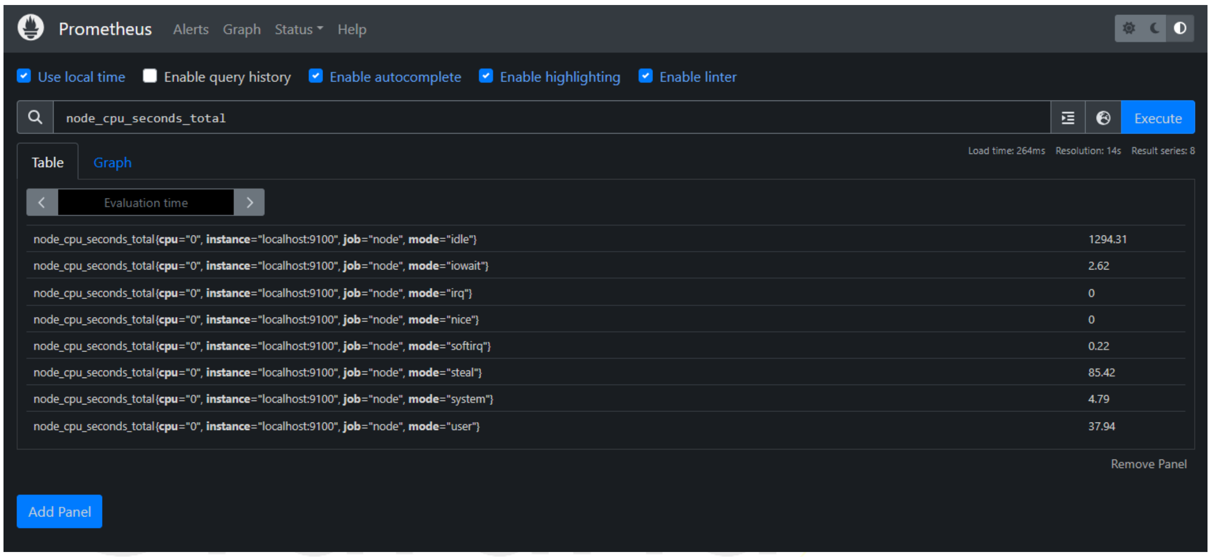
Checking CPU usage –



```
[ec2-user@ip-172-31-17-162 prometheus]$ sudo nano  /etc/prometheus/prometheus.yml
[ec2-user@ip-172-31-17-162 prometheus]$ sudo systemctl restart prometheus
[ec2-user@ip-172-31-17-162 prometheus]$ 
```



```
  - job_name: "node"

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

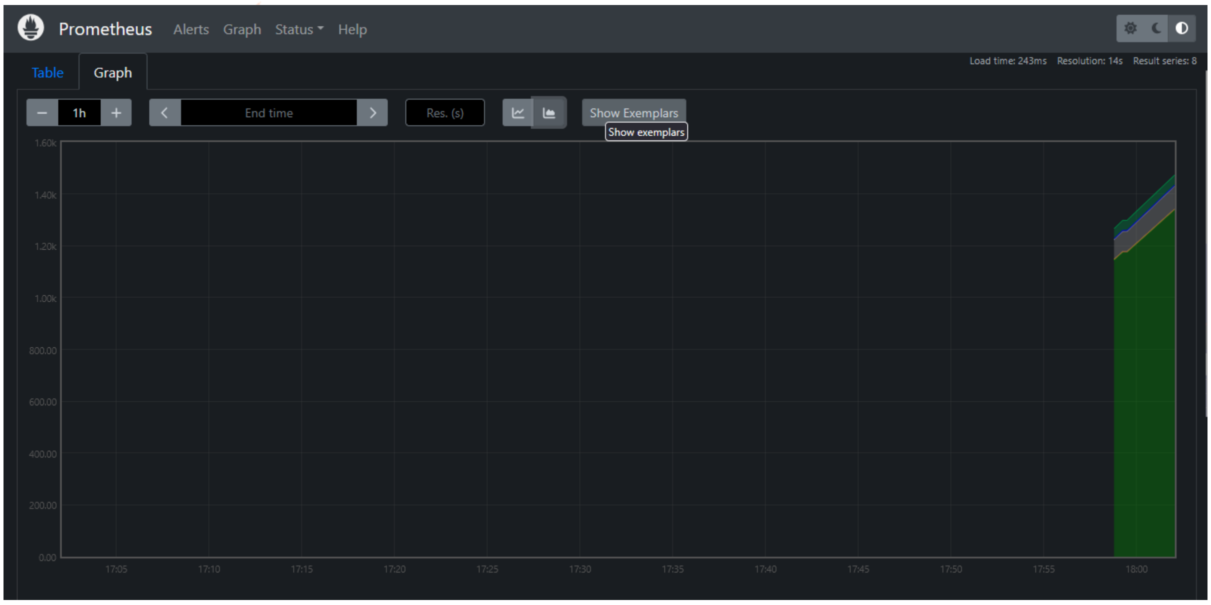


☑ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

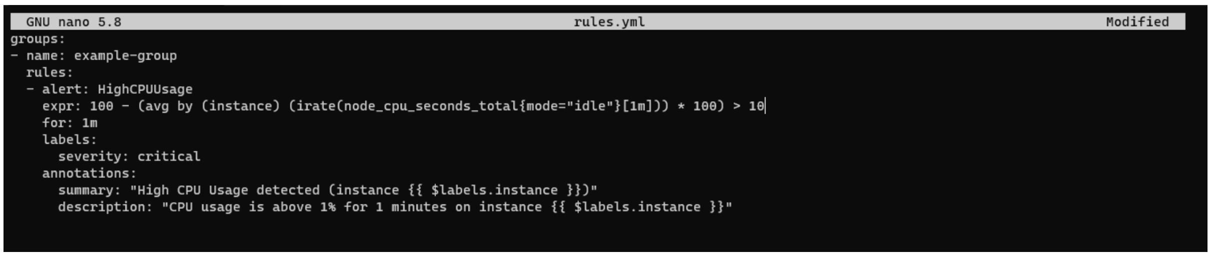
```
node_cpu_seconds_total
```

Load time: 264ms Resolution: 14s Result series: 8

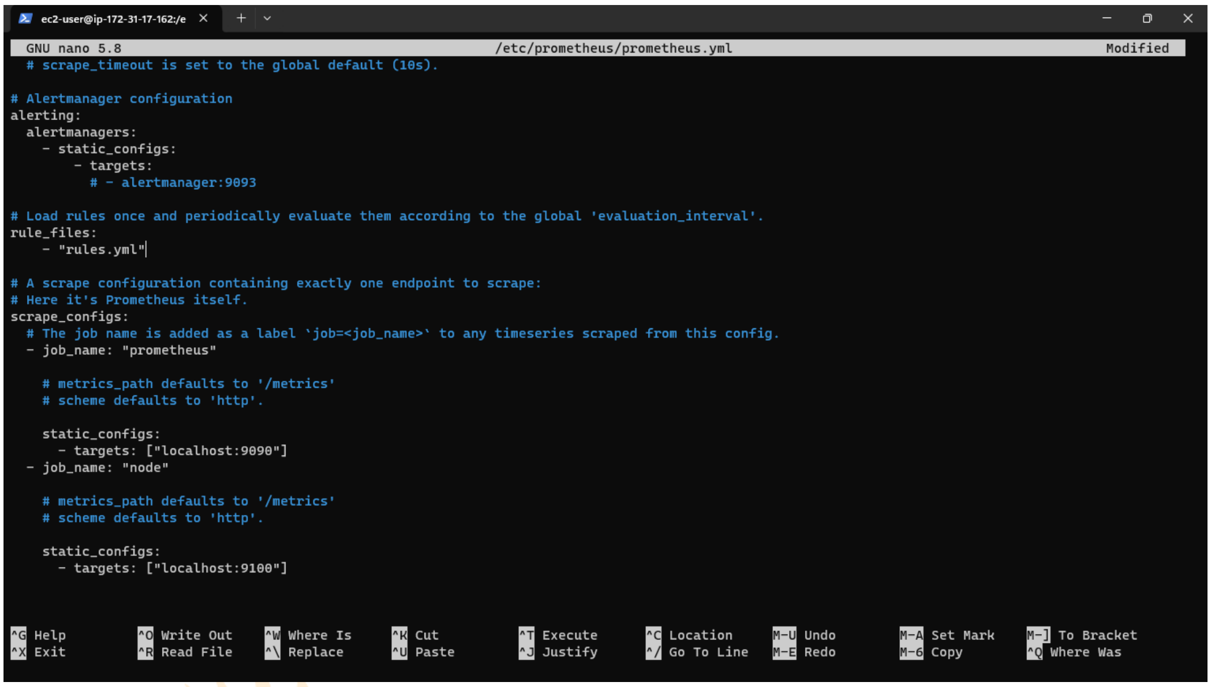
```
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="idle"}      1294.31
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="iowait"}    2.62
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="irq"}       0
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="nice"}      0
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="softirq"}   0.22
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="steal"}     85.42
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="system"}    4.79
node_cpu_seconds_total{cpu="0", instance="localhost:9100", job="node", mode="user"}      37.94
```



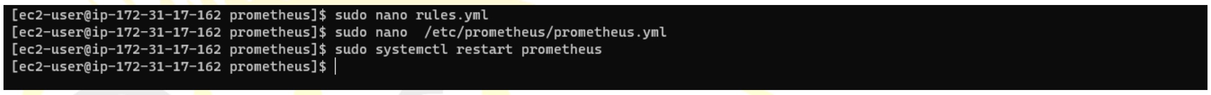
Load time: 243ms Resolution: 14s Result series: 8



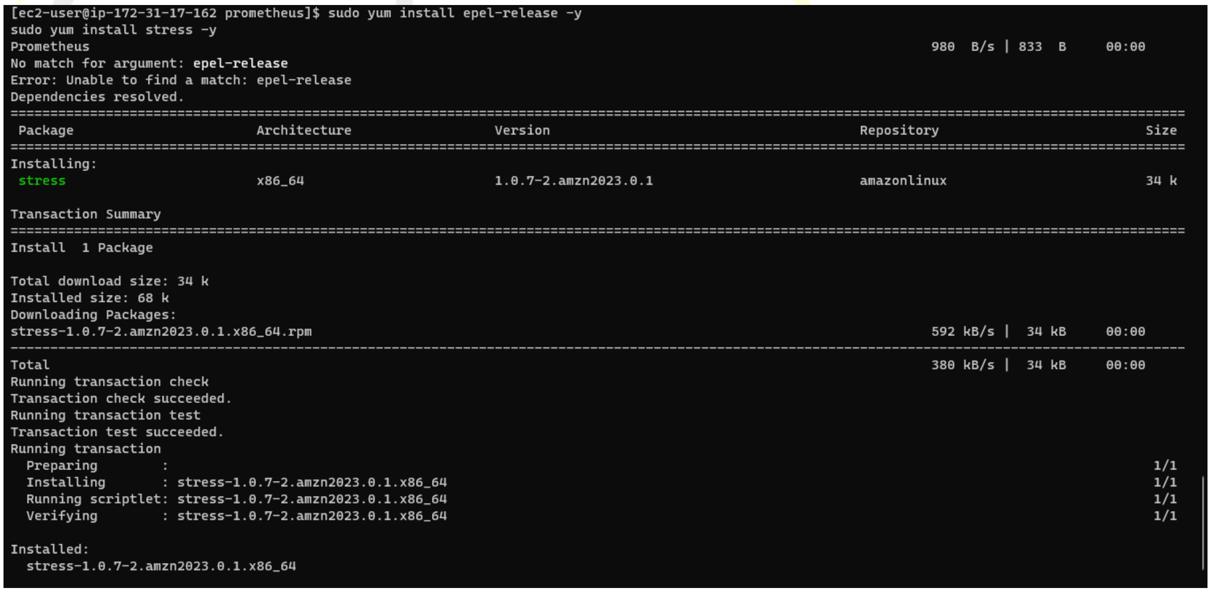
```
  GNU nano 5.8                              rules.yml                                   Modified
groups:
- name: example-group
  rules:
  - alert: HighCPUUsage
    expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[1m])) * 100) > 10
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "High CPU Usage detected (instance {{ $labels.instance }})"
      description: "CPU usage is above 1% for 1 minutes on instance {{ $labels.instance }}"
```



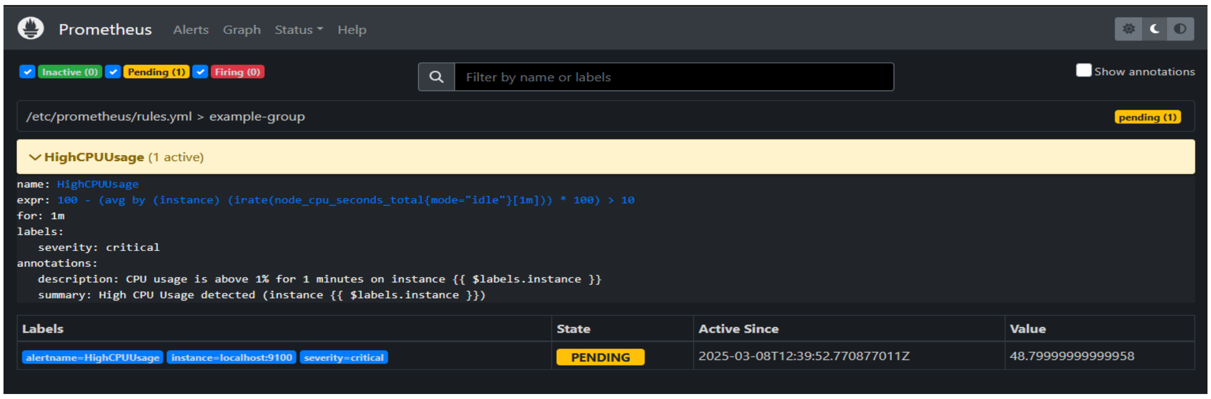
```
  GNU nano 5.8                           /etc/prometheus/prometheus.yml                                    Modified
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
    - "rules.yml"

# 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: ["localhost:9090"]
  - job_name: "node"

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

    static_configs:
      - targets: ["localhost:9100"]

^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark    M-] To Bracket
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy        ^Q Where Was
```



```
[ec2-user@ip-172-31-17-162 prometheus]$ sudo nano rules.yml
[ec2-user@ip-172-31-17-162 prometheus]$ sudo nano  /etc/prometheus/prometheus.yml
[ec2-user@ip-172-31-17-162 prometheus]$ sudo systemctl restart prometheus
[ec2-user@ip-172-31-17-162 prometheus]$
```



```
[ec2-user@ip-172-31-17-162 prometheus]$ sudo yum install epel-release -y
sudo yum install stress -y
Prometheus                                                                                     980  B/s | 833  B     00:00
No match for argument: epel-release
Error: Unable to find a match: epel-release
Dependencies resolved.
===========================================================================================================================
 Package                        Architecture                 Version                                Repository         Size
===========================================================================================================================
Installing:
 stress                         x86_64                       1.0.7-2.amzn2023.0.1                   amazonlinux        34 k

Transaction Summary
===========================================================================================================================
Install  1 Package

Total download size: 34 k
Installed size: 68 k
Downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm                                                         592 kB/s |  34 kB     00:00
---------------------------------------------------------------------------------------------------------------------------
Total                                                                                          380 kB/s |  34 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                    1/1
  Installing       : stress-1.0.7-2.amzn2023.0.1.x86_64                                                                 1/1
  Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64                                                                 1/1
  Verifying        : stress-1.0.7-2.amzn2023.0.1.x86_64                                                                 1/1

Installed:
  stress-1.0.7-2.amzn2023.0.1.x86_64
```



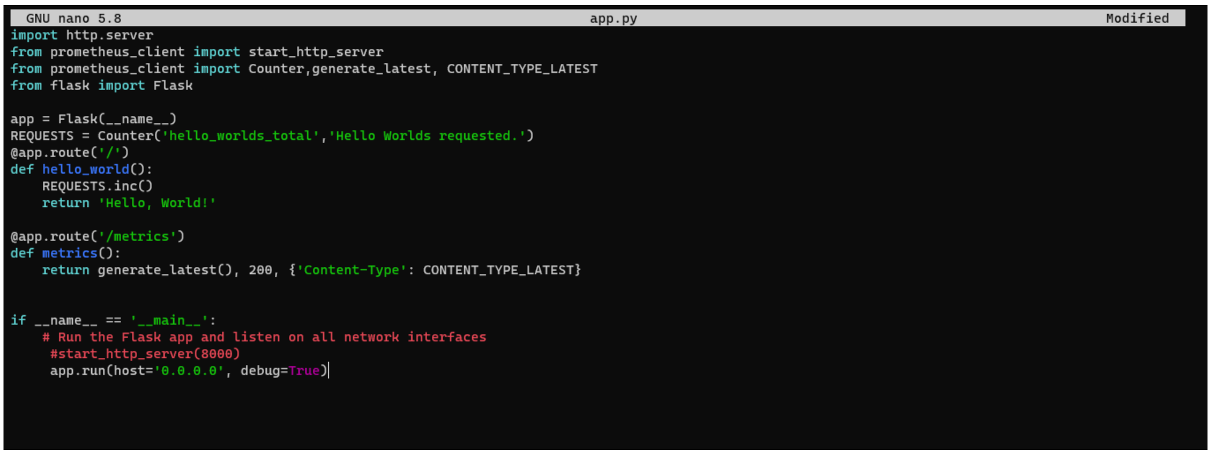
/etc/prometheus/rules.yml > example-group   pending (1)

HighCPUUsage (1 active)

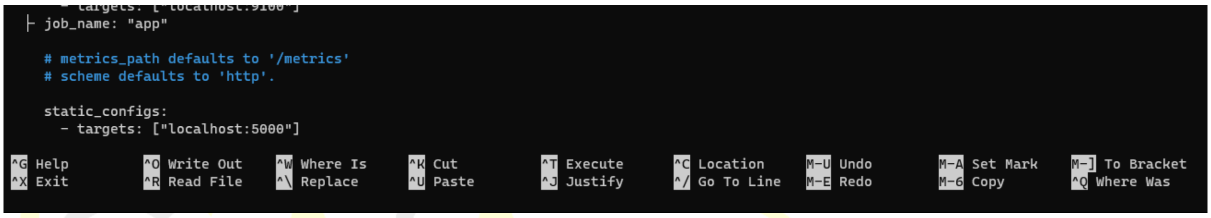
```
name: HighCPUUsage
expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[1m])) * 100) > 10
for: 1m
labels:
   severity: critical
annotations:
   description: CPU usage is above 1% for 1 minutes on instance {{ $labels.instance }}
   summary: High CPU Usage detected (instance {{ $labels.instance }})
```

Labels	State	Active Since	Value
alertname=HighCPUUsage instance=localhost:9100 severity=critical	PENDING	2025-03-08T12:39:52.770877011Z	48.79999999999958

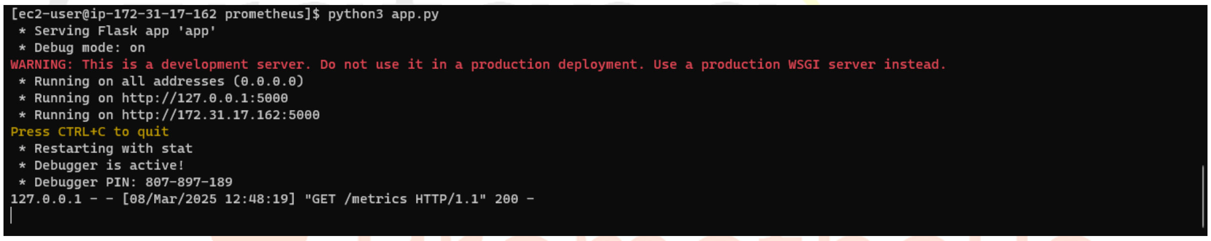
Application usage checking –



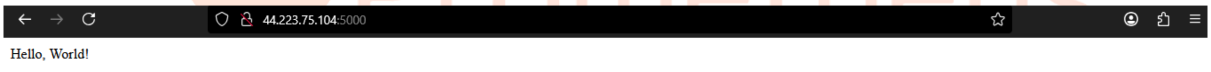
```
  GNU nano 5.8                                app.py                                          Modified
import http.server
from prometheus_client import start_http_server
from prometheus_client import Counter,generate_latest, CONTENT_TYPE_LATEST
from flask import Flask

app = Flask(__name__)
REQUESTS = Counter('hello_worlds_total','Hello Worlds requested.')
@app.route('/')
def hello_world():
    REQUESTS.inc()
    return 'Hello, World!'

@app.route('/metrics')
def metrics():
    return generate_latest(), 200, {'Content-Type': CONTENT_TYPE_LATEST}


if __name__ == '__main__':
    # Run the Flask app and listen on all network interfaces
     #start_http_server(8000)
     app.run(host='0.0.0.0', debug=True)
```



```
  ├ job_name: "app"

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

    static_configs:
      - targets: ["localhost:5000"]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark    M-] To Bracket
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy        ^Q Where Was
```

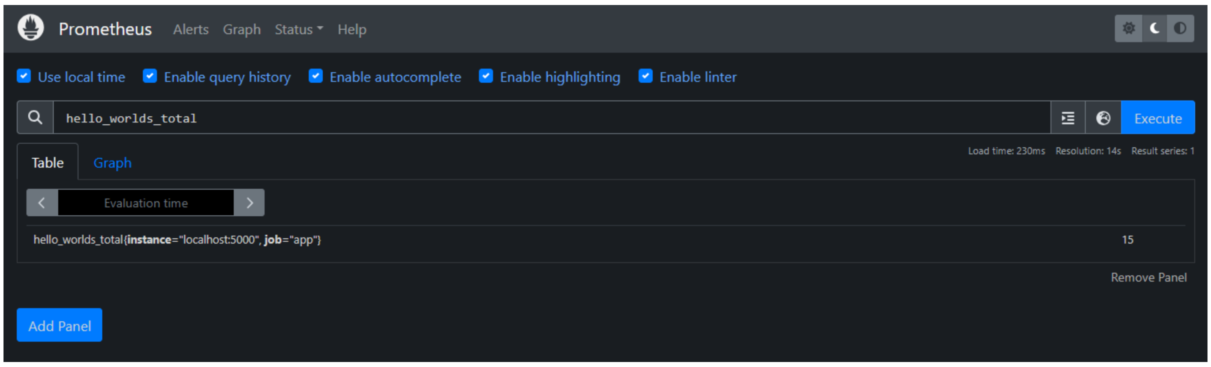


```
[ec2-user@ip-172-31-17-162 prometheus]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.17.162:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 807-897-189
127.0.0.1 - - [08/Mar/2025 12:48:19] "GET /metrics HTTP/1.1" 200 -
```



44.223.75.104:5000

Hello, World!

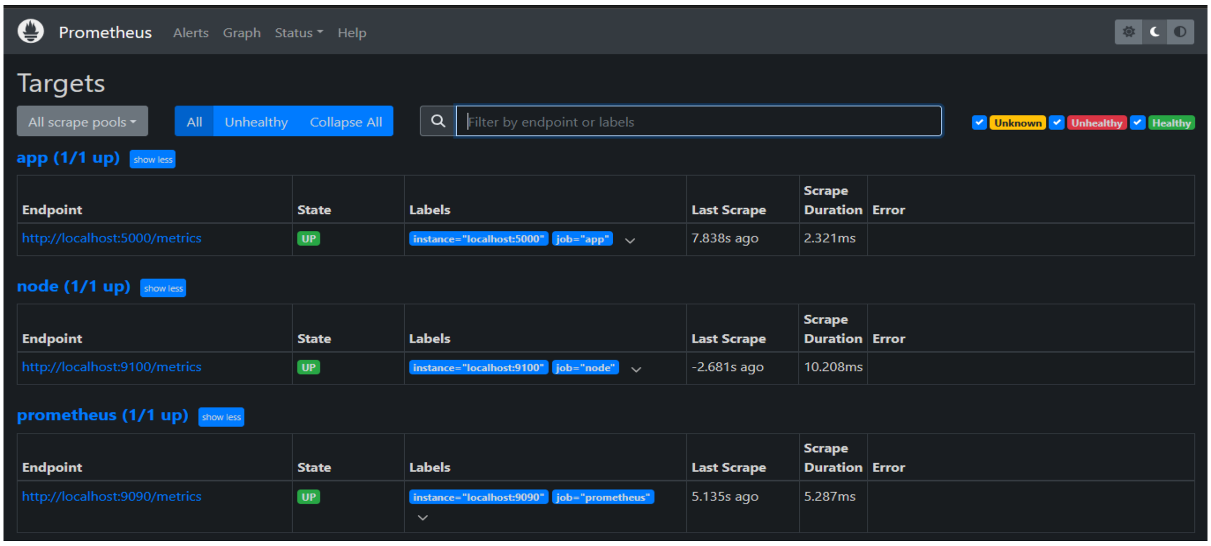


[✓] Use local time [✓] Enable query history [✓] Enable autocomplete [✓] Enable highlighting [✓] Enable linter

```
hello_worlds_total
```

Load time: 230ms Resolution: 14s Result series: 1

```
hello_worlds_total{instance="localhost:5000", job="app"}    15
```



Targets

app (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:5000/metrics	UP	instance="localhost:5000" job="app"	7.838s ago	2.321ms	

node (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node"	-2.681s ago	10.208ms	

prometheus (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	5.135s ago	5.287ms	

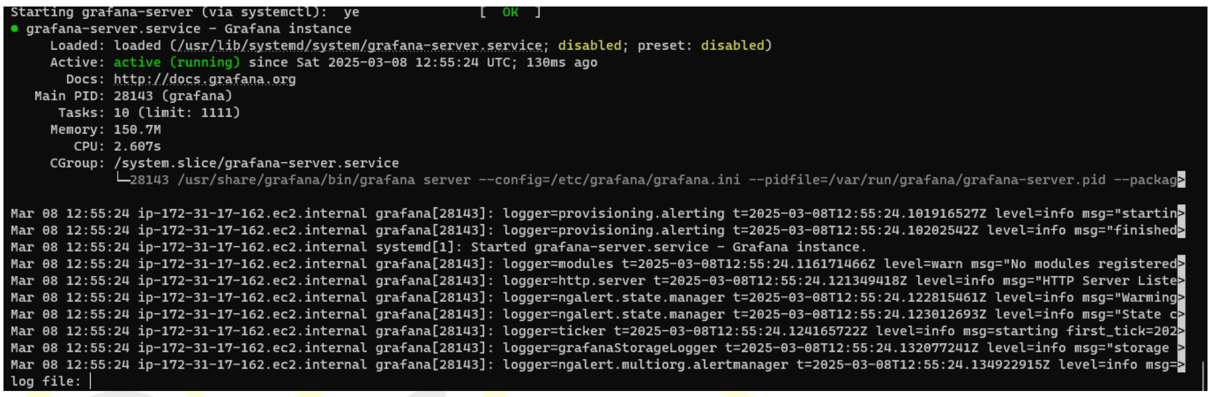
Grafana –

Grafana installation

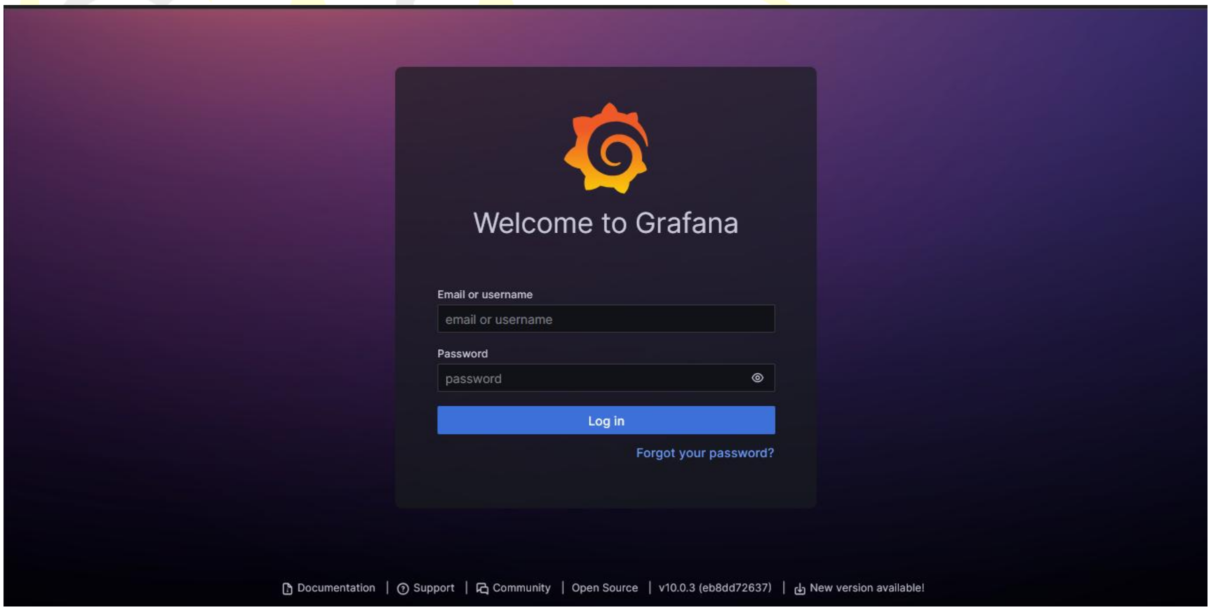
sudo yum install -y https://dl.grafana.com/oss/release/grafana-10.0.3-1.x86_64.rpm

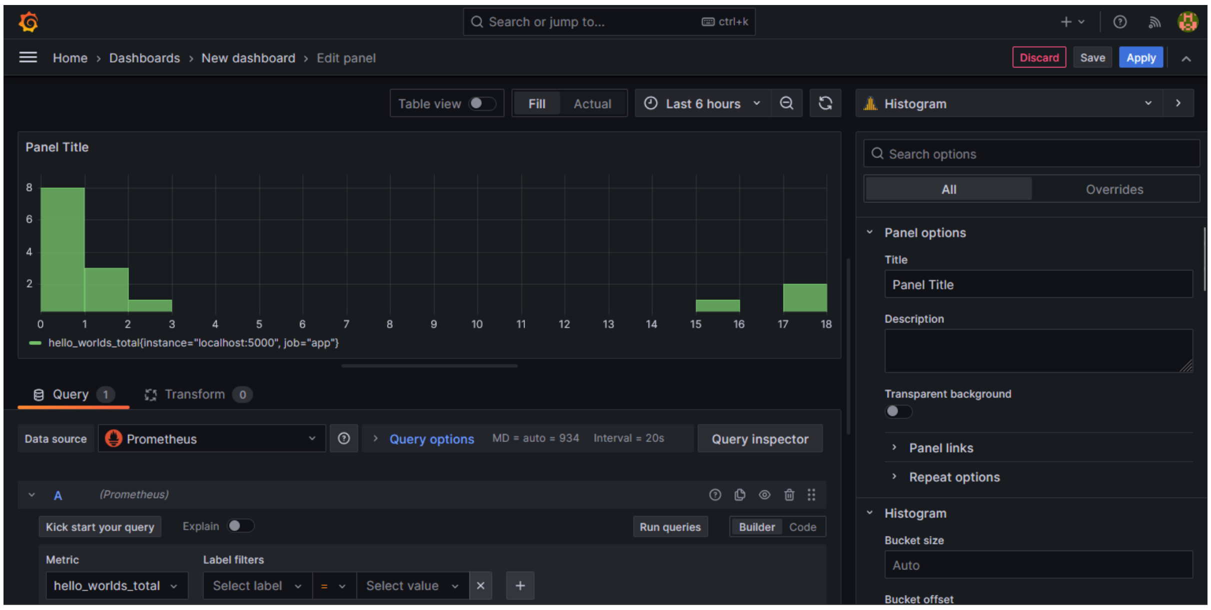
sudo service grafana-server start

sudo service grafana-server status



```
Starting grafana-server (via systemctl):  ye                 [  OK  ]
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; preset: disabled)
     Active: active (running) since Sat 2025-03-08 12:55:24 UTC; 130ms ago
       Docs: http://docs.grafana.org
   Main PID: 28143 (grafana)
      Tasks: 10 (limit: 1111)
     Memory: 150.7M
        CPU: 2.607s
     CGroup: /system.slice/grafana-server.service
             └─28143 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/var/run/grafana/grafana-server.pid --packag>

Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=provisioning.alerting t=2025-03-08T12:55:24.101916527Z level=info msg="startin>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=provisioning.alerting t=2025-03-08T12:55:24.10202542Z level=info msg="finished>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal systemd[1]: Started grafana-server.service - Grafana instance.
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=modules t=2025-03-08T12:55:24.116171466Z level=warn msg="No modules registered>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=http.server t=2025-03-08T12:55:24.121349418Z level=info msg="HTTP Server Liste>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=ngalert.state.manager t=2025-03-08T12:55:24.122815461Z level=info msg="Warming>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=ngalert.state.manager t=2025-03-08T12:55:24.123012693Z level=info msg="State c>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=ticker t=2025-03-08T12:55:24.124165722Z level=info msg=starting first_tick=202>
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=grafanaStorageLogger t=2025-03-08T12:55:24.132077241Z level=info msg="storage >
Mar 08 12:55:24 ip-172-31-17-162.ec2.internal grafana[28143]: logger=ngalert.multiorg.alertmanager t=2025-03-08T12:55:24.134922915Z level=info msg=>
log file: 
```


Welcome to Grafana
Email or username
email or username
Password
password
Log in
Forgot your password?
Documentation | Support | Community | Open Source | v10.0.3 (eb8dd72637) | New version available!



Panel Title

hello_worlds_total{instance="localhost:5000", job="app"}

Data source: Prometheus

Query options MD = auto = 934 Interval = 20s

A (Prometheus)

Metric: hello_worlds_total

Label filters: Select label = Select value

Panel options

Title: Panel Title

Description

Transparent background

Histogram

Bucket size: Auto

Bucket offset