Setup Monitoring Tools using Grafana, Prometheus, and Node-exporter

Monitoring :

Regular collection of information and data to measure progress of projects and activities so we can track performance and resources utilisation over time.

Key Features :

- Quick problem detection.
- We can solve problems easily.

- Regular activity.
- Manage resources effectively.

- Help organisations learn from mistakes.
- Help organisations stay organised.

- Network management.
- Maintaining network issues.

- Justify network upgrade.
- Easy to manage troubleshooting.

Grafana :

Grafana open-source software enables you to query, visualise, alert on, and explore your metrics, logs, and traces wherever they are stored.

- Query
- Visualise
- Alert

Linux Distribution:

- OS Name : Ubuntu 20.04.6 LTS

System Configuration

- RAM : 5.6 GiB
- CPU : 12
- STORAGE : 512.1 GB

Prerequisites tools:

- Prometheus
- Node-exporter

Installation Process:

Step-1 Create a Grafana container:

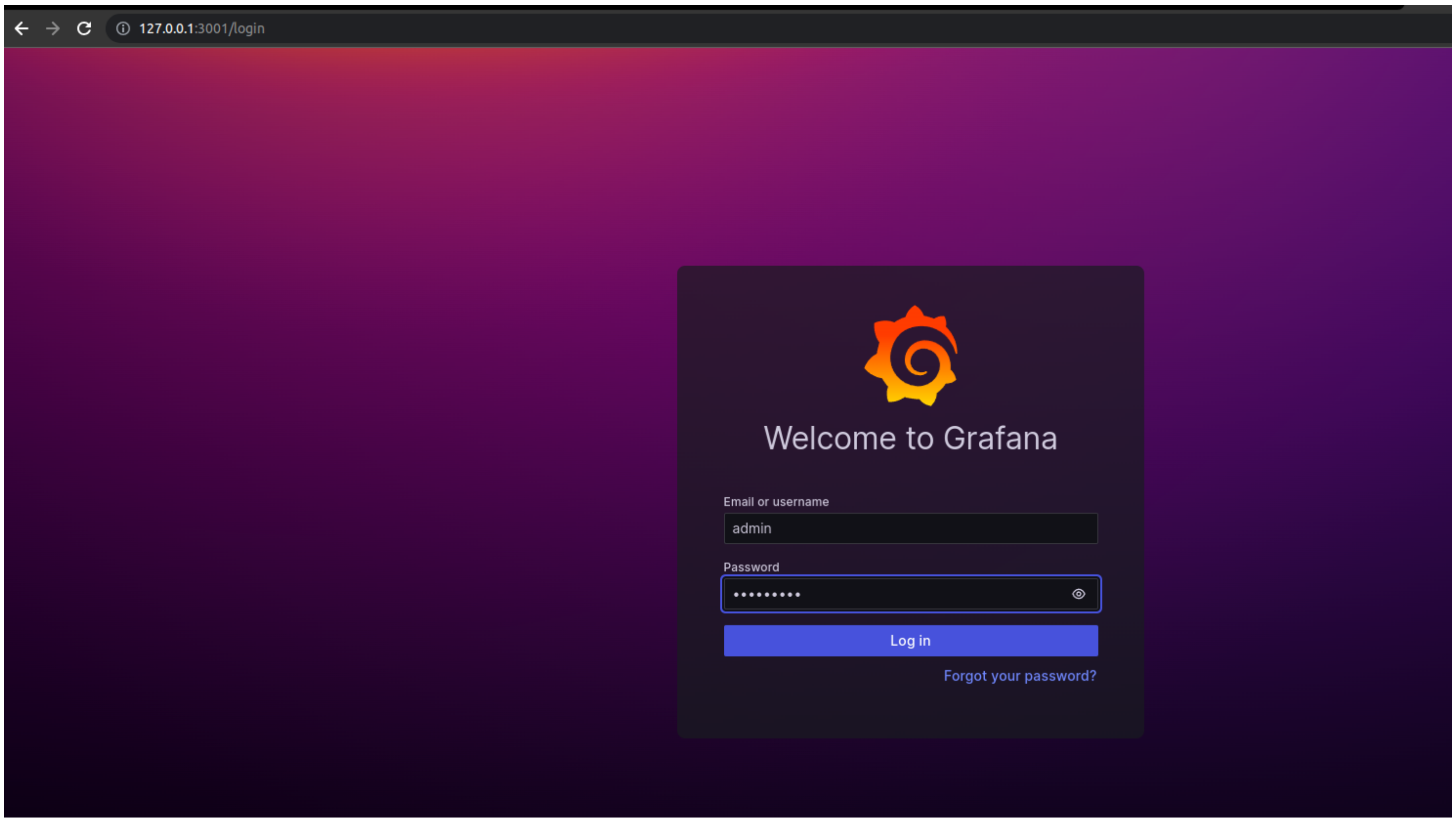
To run the latest stable version of Grafana, run the following command:

```
podman run -d -p 3001:3000 --name=grafana docker.io/grafana/grafana-enterprise
```

Where:

run = run directly from the command line
d = run in the background
p = assign the port number, which in this case is 3001
name = assign a name to the container, for example, grafana

docker.io/grafana/grafana-enterprise: This is the name of the image we want to run inside the container. It's called 'grafana-enterprise,' and it's stored in a special place on the internet called 'Docker Hub.'



Prometheus :

Prometheus is an open source monitoring solution written in Go that collects metrics data and stores that data in a time series database.
Grafana allows to visualise the data stored in prometheus.

Step-2 Create Prometheus container on Podman:

Create directory :-

```
mkdir prometheus
```

Create file:-

```
vim prometheus.yml
```

prometheus.yml is a configuration file of prometheus.

(push all the data in your prometheus.yml file which has been given below)

```
global:
  scrape_interval: 5s
  external_labels:
        monitor: 'node'

scrape_configs:
  - job_name: 'prometheus'
        static_configs:
        - targets: ['192.168.1.57:9090']

  - job_name: 'node-exporter'
        static_configs:
        - targets: ['192.168.1.57:9100']
```

```
podman run -d --name prometheus -p 9090:9090 -v /home/pinki/prometheus/prometheus.yml:/etc/prometheus
```

podman run: This part of the command instructs podman to run a container.

d: This flag stands for "detached" mode. It runs the container in the background

--name prometheus: This flag assigns a name to the container. In this case, the container is named "prometheus."

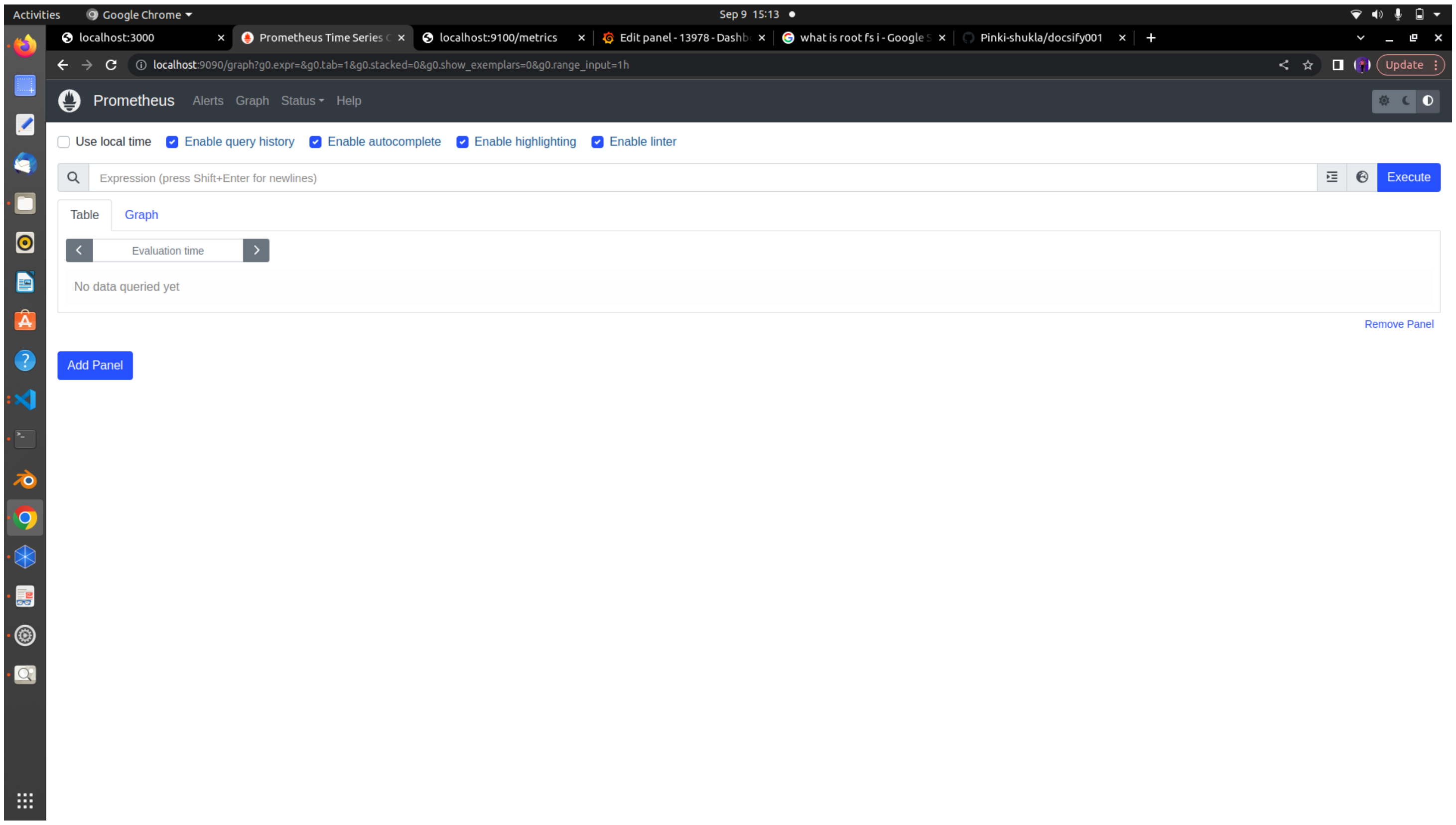
-p 9090:9090: This flag specifies port mapping. It tells podman to map port 9090 on the host to port 9090 inside the container.

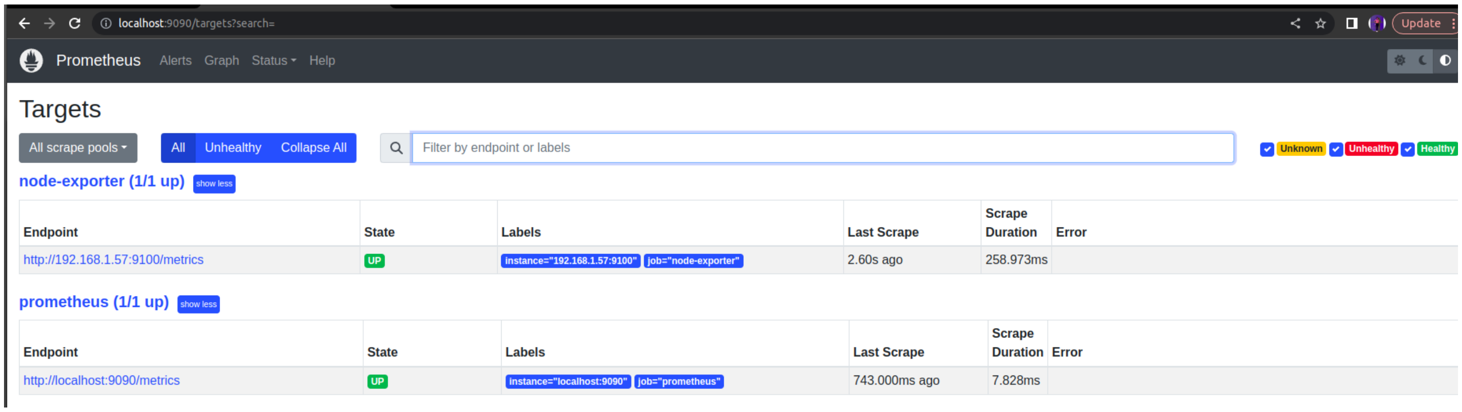
-v /home/amit/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml: This flag specifies a volume mount. It connects a directory or file on your host system to a location inside the container.

In this case, it's mounting the file /home/pinki/prometheus/prometheus.yml from your host into the container at /etc/prometheus/prometheus.yml

- **" /home/pinki/prometheus/prometheus.yml":** This is the path to the Prometheus configuration file on your host. It's being shared with the container.
- **"/etc/prometheus/prometheus.yml":** This is where Prometheus expects its configuration file to be inside the container.

docker.io/prom/prometheus: This is the name of the Docker image you want to run as a container. It specifies the image's repository and name. In this case, you are running the "prometheus" image from the "prom" repository on Docker Hub





localhost:9090/targets?search=

Targets

node-exporter (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.1.57:9100/metrics	UP	instance="192.168.1.57:9100" job="node-exporter"	2.60s ago	258.973ms	

prometheus (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	743.000ms ago	7.828ms	

Node-exporter:

Node Exporters collect data from the system.

Step-3 Create Node exporter container on podman

```
podman run -d --name=node-exporter -p 9100:9100 -v"/:/host:ro,rslave" quay.io/prometheus/node-exporter:
```

podman run: This part of the command instructs podman to run a container.

-d: This flag stands for "detached" mode.

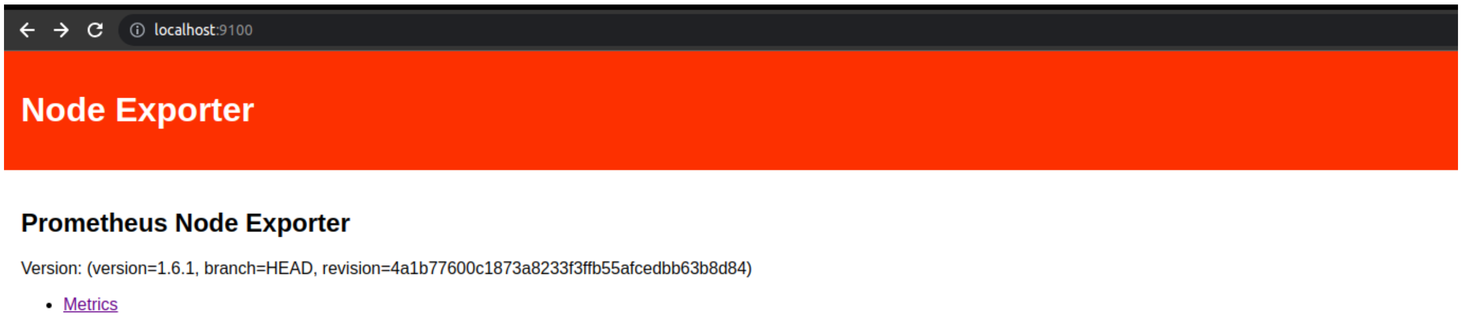
--name=node-exporter: This flag assigns a name to the container. In this case, the container is named "node-exporter."

-p 9100:9100: This flag specifies port mapping. it allows you to access the service inside the container via port 9100 on your host.

"/:/host:ro,rslave": This part specifies the volume configuration. It tells podman to mount the root directory of your host (represented by **"/"**) to the **"/host"** directory inside the container. The **"ro"** option stands for "read-only," which means the container can read the files on the host but can't modify them. The rslave option is related to mount propagation, allowing mounted file systems to be shared among containers.

quay.io/prometheus/node-exporter:latest: In this case, you are running the "node-exporter" image from the "prometheus" repository on Quay.io. The ":latest" tag indicates that you want to use the latest version of this image.

--path.rootfs=/host: This is an additional command passed to the container. It specifies the root file system path as "/host" inside the container. This can be important for some containerized applications to correctly access system resources.

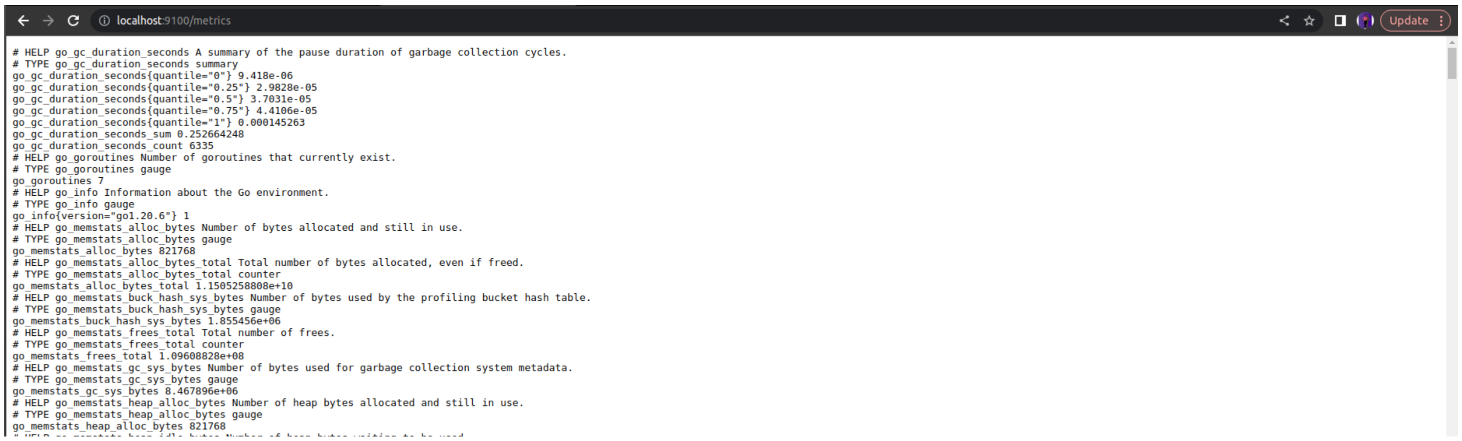


Node Exporter

Prometheus Node Exporter

Version: (version=1.6.1, branch=HEAD, revision=4a1b77600c1873a8233f3ffb55afcedbb63b8d84)

- Metrics

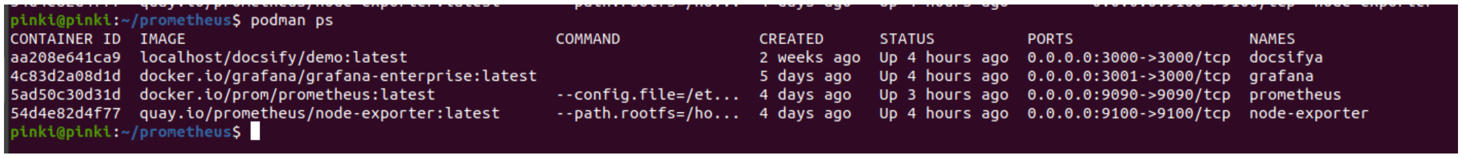


```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 9.418e-06
go_gc_duration_seconds{quantile="0.25"} 2.9828e-05
go_gc_duration_seconds{quantile="0.5"} 3.7031e-05
go_gc_duration_seconds{quantile="0.75"} 4.4106e-05
go_gc_duration_seconds{quantile="1"} 0.000145263
go_gc_duration_seconds_sum 0.252664248
go_gc_duration_seconds_count 6335
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.20.6"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 821768
# 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 1.1505258808e+10
# 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 1.855456e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 1.09608828e+08
# 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 8.467896e+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 821768
```

Now you can see our all containers are ready grafana,prometheus and node-exporter

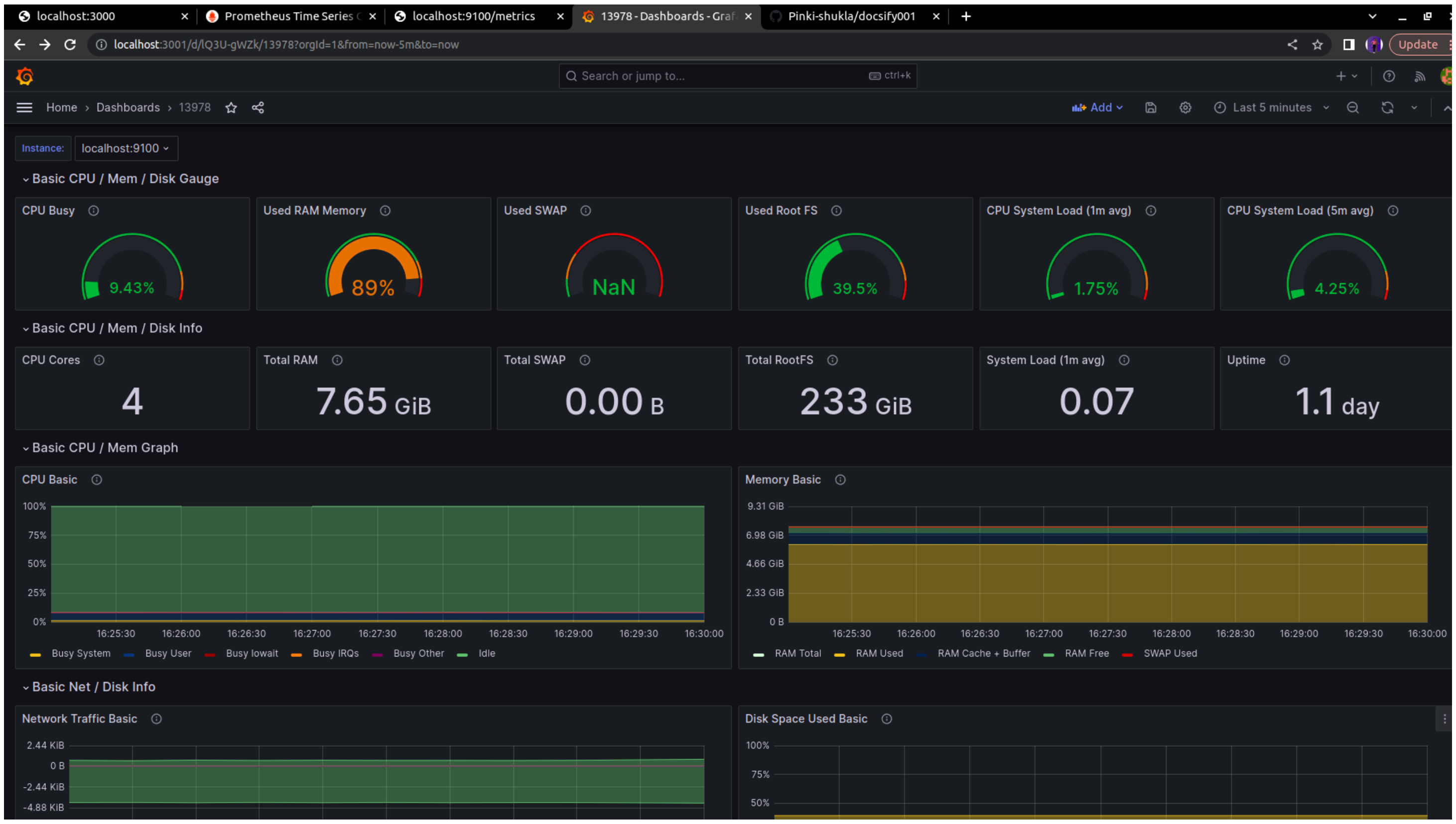
```
podman ps
```

The **podman ps** command is used to list the currently running containers on your system. It provides information about the containers that are actively running and includes details such as the container ID, names, status, and other relevant information.



```
pinki@pinki:~/prometheus$ podman ps
CONTAINER ID  IMAGE                                     COMMAND               CREATED      STATUS          PORTS                   NAMES
aa208e641ca9  localhost/docsify/demo:latest                                   2 weeks ago  Up 4 hours ago  0.0.0.0:3000->3000/tcp  docsifya
4c83d2a08d1d  docker.io/grafana/grafana-enterprise:latest                     5 days ago   Up 4 hours ago  0.0.0.0:3001->3000/tcp  grafana
5ad50c30d31d  docker.io/prom/prometheus:latest          --config.file=/et...  4 days ago   Up 3 hours ago  0.0.0.0:9090->9090/tcp  prometheus
54d4e82d4f77  quay.io/prometheus/node-exporter:latest   --path.rootfs=/ho...  4 days ago   Up 4 hours ago  0.0.0.0:9100->9100/tcp  node-exporter
pinki@pinki:~/prometheus$
```

Grafana setup has been ready


localhost:3000
Prometheus Time Series
localhost:9100/metrics
13978 - Dashboards - Graf
Pinki-shukla/docsify001
localhost:3001/d/lQ3U-gWZk/13978?orgId=1&from=now-5m&to=now
Update
Search or jump to...
ctrl+k
Home › Dashboards › 13978
Add
Last 5 minutes
Instance: localhost:9100
Basic CPU / Mem / Disk Gauge
CPU Busy
9.43%
Used RAM Memory
89%
Used SWAP
NaN
Used Root FS
39.5%
CPU System Load (1m avg)
1.75%
CPU System Load (5m avg)
4.25%
Basic CPU / Mem / Disk Info
CPU Cores
4
Total RAM
7.65 GiB
Total SWAP
0.00 B
Total RootFS
233 GiB
System Load (1m avg)
0.07
Uptime
1.1 day
Basic CPU / Mem Graph
CPU Basic
100%
75%
50%
25%
0%
16:25:30
16:26:00
16:26:30
16:27:00
16:27:30
16:28:00
16:28:30
16:29:00
16:29:30
16:30:00
Busy System
Busy User
Busy Iowait
Busy IRQs
Busy Other
Idle
Memory Basic
9.31 GiB
6.98 GiB
4.66 GiB
2.33 GiB
0 B
RAM Total
RAM Used
RAM Cache + Buffer
RAM Free
SWAP Used
Basic Net / Disk Info
Network Traffic Basic
2.44 KiB
0 B
-2.44 KiB
-4.88 KiB
Disk Space Used Basic
100%
75%
50%