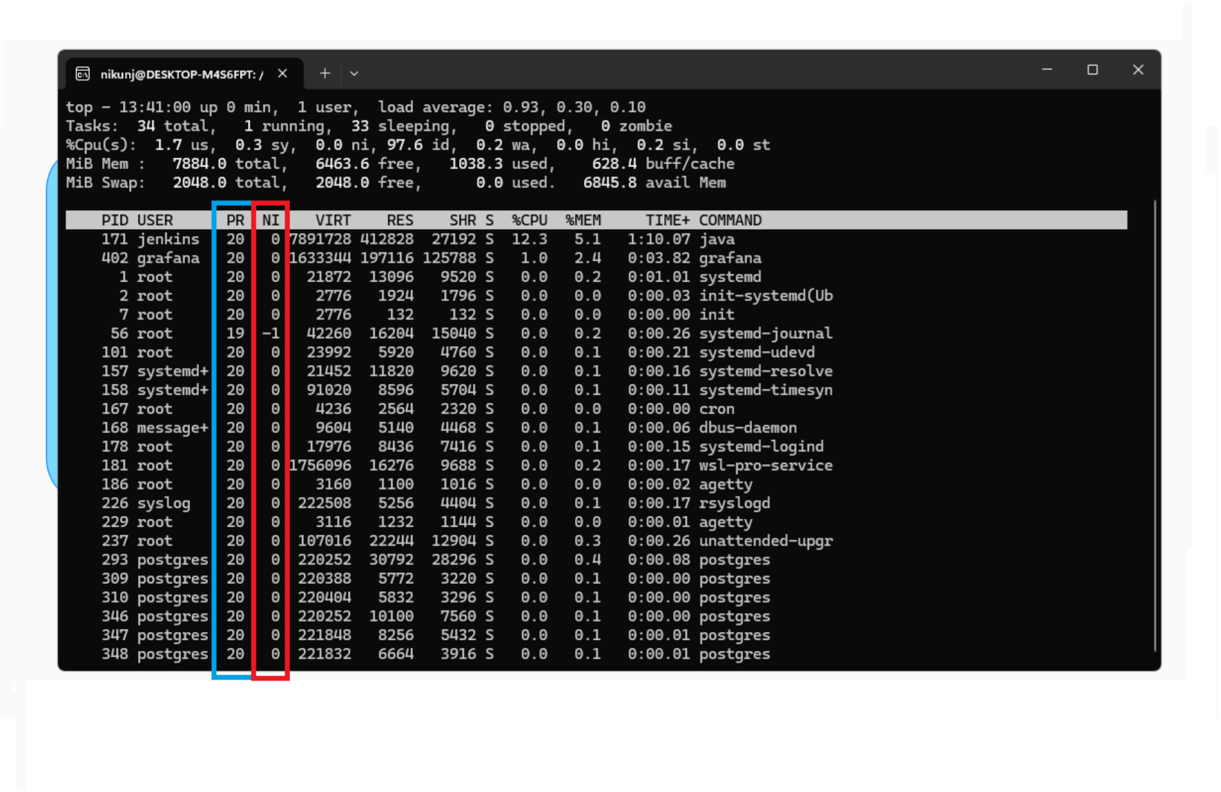
Session-6

Nice And Renice

- nice is used to start process with specific priority
- renice is used to change the priority of existing process

1. each process has a nice value ranging from -20 (highest Priority) to 19 (Lowest Priority)
2. by default new process starts with a nice value of 0.

introduction to Nice and Renice



```
top - 13:41:00 up 0 min,  1 user,  load average: 0.93, 0.30, 0.10
Tasks:  34 total,   1 running,  33 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.7 us,  0.3 sy,  0.0 ni, 97.6 id,  0.2 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   7884.0 total,   6463.6 free,   1038.3 used,    628.4 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   6845.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    171 jenkins   20   0 7891728 412828  27192 S  12.3   5.1   1:10.07 java
    402 grafana   20   0 1633344 197116 125788 S   1.0   2.4   0:03.82 grafana
      1 root      20   0   21872  13096   9520 S   0.0   0.2   0:01.01 systemd
      2 root      20   0    2776   1924   1796 S   0.0   0.0   0:00.03 init-systemd(Ub
      7 root      20   0    2776    132    132 S   0.0   0.0   0:00.00 init
     56 root      19  -1   42260  16204  15040 S   0.0   0.2   0:00.26 systemd-journal
    101 root      20   0   23992   5920   4760 S   0.0   0.1   0:00.21 systemd-udevd
    157 systemd+  20   0   21452  11820   9620 S   0.0   0.1   0:00.16 systemd-resolve
    158 systemd+  20   0   91020   8596   5704 S   0.0   0.1   0:00.11 systemd-timesyn
    167 root      20   0    4236   2564   2320 S   0.0   0.0   0:00.00 cron
    168 message+  20   0    9604   5140   4468 S   0.0   0.1   0:00.06 dbus-daemon
    178 root      20   0   17976   8436   7416 S   0.0   0.1   0:00.15 systemd-logind
    181 root      20   0 1756096  16276   9688 S   0.0   0.2   0:00.17 wsl-pro-service
    186 root      20   0    3160   1100   1016 S   0.0   0.0   0:00.02 agetty
    226 syslog    20   0  222508   5256   4404 S   0.0   0.1   0:00.17 rsyslogd
    229 root      20   0    3116   1232   1144 S   0.0   0.0   0:00.01 agetty
    237 root      20   0  107016  22244  12904 S   0.0   0.3   0:00.26 unattended-upgr
    293 postgres  20   0  220252  30792  28296 S   0.0   0.4   0:00.08 postgres
    309 postgres  20   0  220388   5772   3220 S   0.0   0.1   0:00.00 postgres
    310 postgres  20   0  220404   5832   3296 S   0.0   0.1   0:00.00 postgres
    346 postgres  20   0  220252  10100   7560 S   0.0   0.1   0:00.00 postgres
    347 postgres  20   0  221848   8256   5432 S   0.0   0.1   0:00.01 postgres
    348 postgres  20   0  221832   6664   3916 S   0.0   0.1   0:00.01 postgres
```

- higher the value means lower priority(the process is "nicer" to others)
- Lower nice value means Higher the priority (requires root access for negative values)
- you can start a particular process by using priority as

```
nice -n 10 myscript.sh
```

```
renice 10 -p <PID>
```

Create The Script "myscript.sh"

```
#!/bin/bash
echo "Starting Script with priority $(nice)"

sleep 100 # simulate a process running
echo "Script Completed."
```

start the Script:

```
bash myscript.sh
```

change the nice value

```
renice 10 -p <PID>
```

to get the PID

```
htop
```

search for the myscript name using F3

Example:2

```
#! /bin/bash

echo "Starting a Background CPU-Intensive process with nice and renice value 
10...."
nice -n 10 bash -c 'for i in {1..100}; do echo "Nice Process running ....$i"; 
sleep 1; done' &
PID=$!

echo "Process started with PID: $PID"
sleep 3 # let it run for a few seconds
# sleep 10

echo "Now changing the priority of PID $PID using renice to -5 (higher 
priority)..."
sudo renice -n -5 -p $PID
# sudo renice 5 -p $PID
```

```
echo "Use 'top -p $PID or 'htop' to observer the priority in real time."
```

CPU Bottlenecks

- High CPU Usage (nearly 100%)
- Slow Application Response time
- High Load Average (uptime,top,htop to monitor This)

How to Deal with It?

- Use **ps** to list the CPU consuming process

```
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -10 # 10 is a numbers of
processes creating CPU bottlenecks
```

- also you can check using systats

```
sudo apt install mpstat
mpstat -P All 1
```

Solution: Optimise the code, add caching or Scale the server HORIZONTALLY by Adding More and More Servers

MEMORY BOTTLENECKS

- System slowdown Frequently
- High RAM Usage (You can check this in free -m in CMD)
- Out of Memory (OOM error)
- How to Deal With IT?

```
free -h # used to check free memory usage



ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -10 # find the process
consuming the most of the memories
```

I/O Bottlenecks (DISK & NETWORK)

- slow file read /write operation
- High Disk Utilization
- High Network latency or dropped packets

- How to Identify?

```
#monitor disk I/O

iostat

iostat -dx

# to find the input and output i/o consuming process
sudo iotop
```

A database query performing full table scan instead of using indexes can cause excessive disk I/O
Solution: Optimise the queries, add indexes or caching

INTRODUCTION TO SAR

(SYSTEM ACTIVITY REPORT)

- it is command line tool that collects , reports and saves system performance data
- it is the part of sysstat package
- it provides insights of CPU Usage Memory Utilization , Network Activity

How to Install

```
sudo apt update && sudo apt install sysstat -y
```

Enable and Start Data Collection

```
sudo systemctl enable systat



sudo systemctl start sysstat
```

lets start collecting the data

```
sar -u 5 5
```

here -u : CPU Usage Report
5 5 : collects the data every 5 seconds fro 5 iteration

%user	-> CPU time spent on user process
%system	-> CPU time spent on System or Kernel Process

%IO Wait -> Time Waiting for I/O operations
%idle -> Available CPU Time

Lets start collecting the data based on Swap Usage

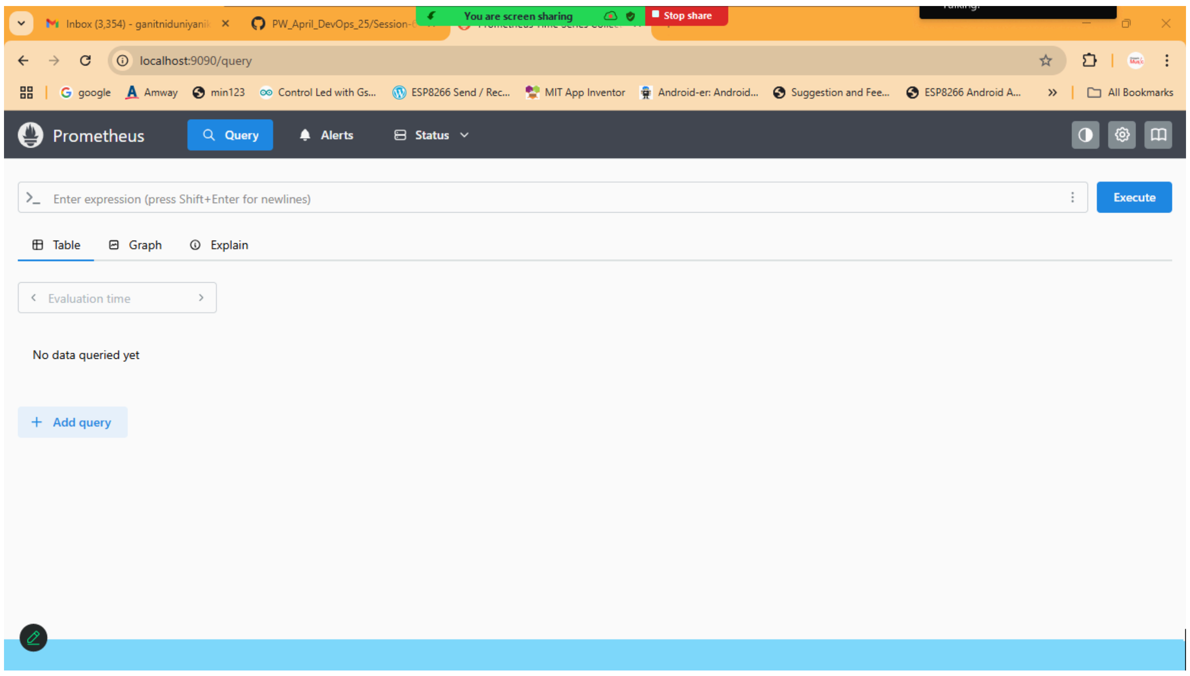
```
sar -s 5 5
```

lets start collecting the data based on Disk Memory

```
sar -d 5 5
```

Prometheus

- download prometheus from : https://prometheus.io/download/
- extract it to the folder
- double click on prometheus.exe file and run it
- allow the permission | turn off the Windows Defender for sometime
- start the prometheus
- goto > browser> localhost:9090
-



Prometheus Popular Queries

{__name__=~".+"} --> this will show all metrics currently scraped by prometheus

- process_cpu_seconds_total --> shows total cpu seconds consumed by the Prometheus process
- go_goroutines --> numbers of goroutines running in prometheus process
- http_requests_total{job="prometheus"}
- sum(http_requests_total)
- rate(prometheus_http_requests_total[5m]) --> get the per-second rate of increase

How to run This Queries?

1. open prometheus> localhost:9090/query
2. click Graph Tab
3. Enter Query in the Input Box
4. press Enter and Execute
5. You Can View the result in Graph or Table