

# Linux Monitoring: Understand Logs, Linux monitor tools and learn ZABBIX

Check GitHub for helpful DevOps tools:

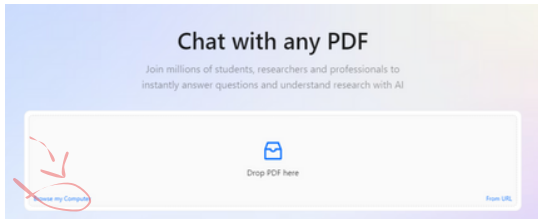
## Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



Ask Personal AI Document assistant to learn interactively (FASTER)!

- 1 Download PDF 1 <https://github.com/MichaelRobotics/DevOpsTools/blob/main/LinuxMonitoring.pdf>
- 2 Go to website 2 [Click there to go to ChatPdf website](#)
- 3 Browse file 3 
- 4 Chat with Document 4 Ask questions about document!

# Completly new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

 <https://www.hackthebox.com/>



## What is Linux Monitoring?

Linux monitoring is the process of tracking system performance, resource usage, and health metrics to ensure optimal operation and quickly detect issues. It involves monitoring components like CPU, memory, disk, network, and running processes.

## How to implement Linux Monitoring?

Implement Linux monitoring by using tools like top, htop, vmstat, or more comprehensive solutions like Zabbix and Nagios. These tools can be configured to collect, analyze, and alert on system metrics automatically.

# Linux Monitoring: Why and When

Monitoring is essential for maintaining system stability, detecting bottlenecks, and ensuring efficient resource usage, especially on servers running critical applications.

For example, monitoring CPU and memory usage can help identify when a service needs scaling to handle increased load.

## System Requirements

- 1 GB RAM (minimum, 2 GB recommended for better performance)
- 10 GB free storage (more may be required depending on the size of the DNS zone files and logs)
- Ubuntu 22.04

## Linux Monitoring: Main components & packages

- top / htop – Real-time system monitoring for CPU, memory, and process management.
- vmstat – Reports on system performance including memory, I/O, and CPU usage.
- iostat – Monitors disk I/O and performance statistics.
- sar – Collects, reports, and saves system activity data over time.
- netstat / ss – Displays network connections, routing tables, and interface statistics.
- Zabbix / Nagios – Comprehensive monitoring solutions for servers, services, and network devices.

# Linux Monitoring: Log files, services basic system info

## 1) Linux logs

Generally, log files of installed applications on Linux are stored in `/var/log`

- **`/var/log/message`** – Where whole system logs or current activity logs are available.
- **`/var/log/auth.log`** – Authentication logs.
- **`/var/log/kern.log`** – Kernel logs.
- **`/var/log/cron.log`** – Crond logs (cron job).
- **`/var/log/maillog`** – Mail server logs.
- **`/var/log/boot.log`** – System boot log.
- **`/var/log/mysqld.log`** – MySQL database server log file.
- **`/var/log/secure`** – Authentication log.
- **`/var/log/syslog`**: General system activity logs.
- **`/var/log/ufw.log`**: Logs for UFW (Uncomplicated Firewall) on Ubuntu and Debian-based systems.
- **`/var/log/dmesg`**: logging information about hardware detection, drivers, and system initialization

## 2) basic commands to measure system

Displays kernel version and system information.

```
uname -a
```

Shows Ubuntu version and release information.

```
lsb_release -a
```

Displays information about the hostname, operating system, and hardware.

```
hostnamectl
```

Shows available and used memory in human-readable format.

```
free -h
```

Displays disk space usage of all mounted filesystems.

```
df -h
```

Summarizes the size of a directory.

```
du -sh
```

Shows the status of all services (or a specific service)

```
systemctl status
```

### 3) Journalctl

journalctl is a command-line utility for querying and displaying logs generated by systemd services. It reads logs from the systemd journal, which collects data from various sources, including system boot messages, service logs, kernel logs, and application logs

Displays all logs collected by the systemd journal, starting from the oldest available entry. You can scroll through the logs using the arrow keys.

```
$ journalctl
```

Similar to tail -f, this will display the most recent logs and update in real-time as new entries are added. This is useful for monitoring logs as they are being written.

```
$ journalctl -f
```

Shows logs from the current boot session. To view logs from a previous boot session, use:

```
$ journalctl
```

#### View Logs for a Specific Service

```
$ journalctl -u <service-name>
```

## Filter Logs by Time

```
$ journalctl --since "2024-10-16 10:00:00"
```

or

```
$ journalctl --until "2024-10-16 18:00:00"
```

or

```
$ journalctl --since "2024-10-16" --until "2024-10-17"
```

or

```
$ journalctl --since "2024-10-16" --until "2024-10-17"
```

## View Kernel Logs

```
$ journalctl -k
```

## Show Logs for a Specific Executable or PID

```
$ journalctl /usr/bin/bash
```

or

```
$ journalctl _PID=1234
```

# Linux Monitoring: processes

## 4) processes commands and htop

Shows all running processes with detailed information (a for all users, u for user format, x for not tied to a terminal).

```
$ ps aux
```

Real-time, dynamic view of running processes. It shows the most resource-intensive processes at the top and updates automatically.

```
$ top
```

Interactive, enhanced version of top. It provides an easy-to-read, colorful interface with more functionality, such as scrolling and sorting.

```
$ htop
```

pgrep: Searches for processes by name.

```
$ pgrep firefox
```

Killing a Process:

```
$ kill <PID>
```



forcefully kill a process

```
$ kill -9 <PID>
```

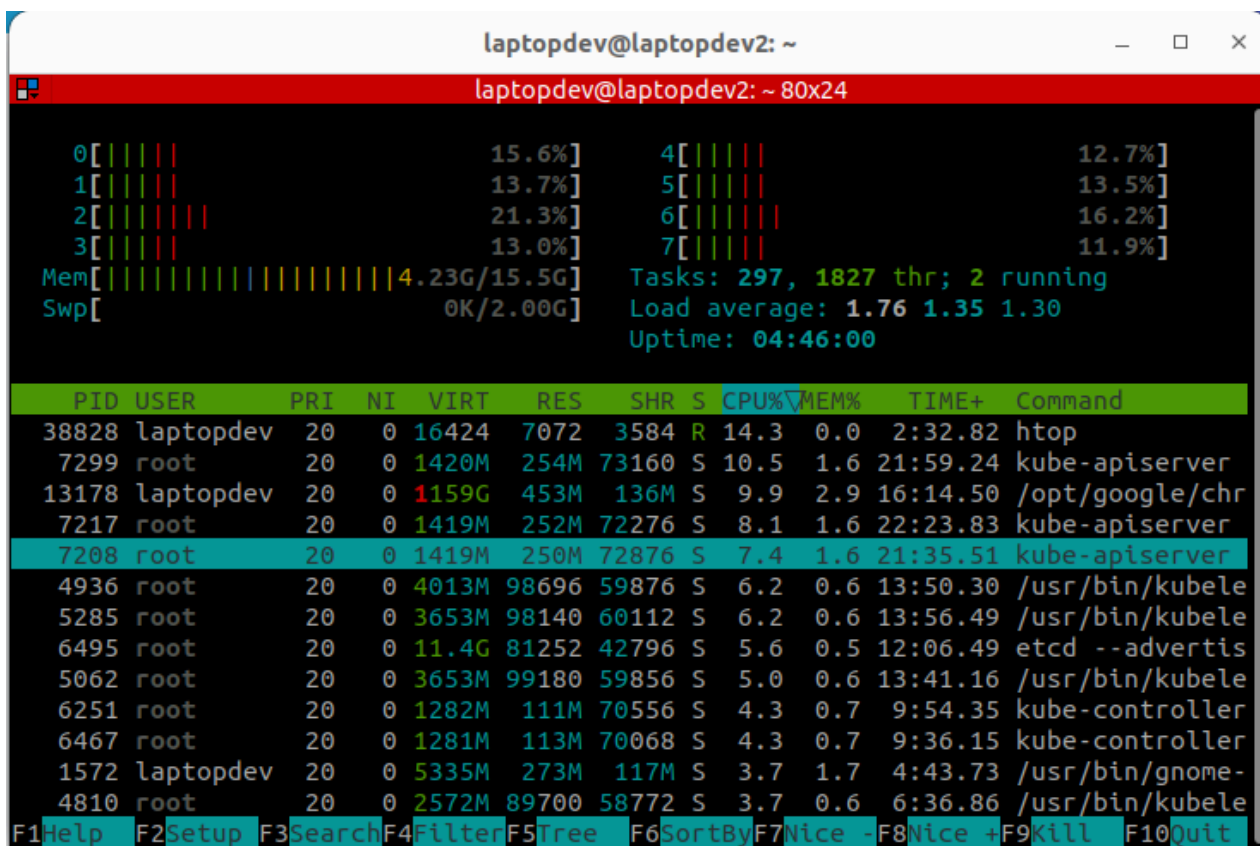
pkill: Kills a process by name.

```
$ pkill firefox
```

display processes in a tree format

```
$ pstree
```

Lets decompose htop command:



The top bar of htop provides a quick overview of the system's overall performance. It includes:

### 1.1 CPU Usage Bars

- CPU(n): Displays usage per CPU core, showing the percentage of CPU time used by different types of processes:
  - Green: User processes (normal programs)
  - Red: System processes (kernel tasks)
  - Blue: Low-priority processes (nice values)
  - Orange/Yellow: Virtualized processes (IO wait)
  - Gray: Steal time (CPU cycles taken by other virtual machines)

### 1.2 Memory Usage Bar

- Mem: Shows the current memory (RAM) usage. The bar indicates:
  - Green: Used memory
  - Blue: Buffers (memory used for temporary data)
  - Yellow: Cached (recently used data stored for faster access)

### 1.3 Swap Usage Bar

- Swp: Displays the swap usage, indicating how much swap space is used (swap is disk space used when RAM is full):
  - Green: Used swap space
  - Gray: Free swap space

### 1.4 Load Average

- Load average: Displays the system's load average over the last 1, 5, and 15 minutes. It indicates how many processes are actively running or waiting for CPU time.
  - A load average of 1.0 per core means the CPU is fully utilized. If it exceeds the number of CPU cores, the system might be overloaded.

## 1.5 Uptime

- Uptime: Shows how long the system has been running since the last reboot.

## 2. Process List (Middle Section)

This is the main part of htop, displaying all the active processes on the system. Each process is represented by a row with the following columns:

### 2.1 PID

- PID: The Process ID. A unique identifier for each running process.

### 2.2 USER

- USER: The name of the user who started the process.

### 2.3 PRI

- PRI: The priority of the process. Lower numbers mean higher priority.

### 2.4 NI

- NI: The "nice" value, which affects the process priority. Processes with a lower nice value are given more CPU time.

### 2.5 VIRT

- VIRT: The total amount of virtual memory used by the process (including all code, data, and shared libraries).

### 2.6 RES

- RES: The resident memory, which is the actual physical RAM currently being used by the process.

### 2.7 SHR

- SHR: The amount of shared memory used by the process (shared with other processes).

## 2.8 S (State)

- S: The current state of the process:
  - D: Uninterruptible Sleep (waiting for I/O)
  - R: Running (actively using CPU)
  - S: Sleeping (idle, but can be woken up)
  - T: Stopped (paused)
  - Z: Zombie (process has finished, but is still in the process table)

## 2.9 %CPU

- %CPU: The percentage of CPU time the process is using.

## 2.10 %MEM

- %MEM: The percentage of RAM the process is using.

## 2.11 TIME+

- TIME+: The total CPU time used by the process since it started.

## 2.12 COMMAND

- COMMAND: The name of the command or program being executed by the process. By default, it shows the full command line, but this can be configured.

## 5) kernel /proc etc

The /proc directory in Linux is a virtual filesystem that offers real-time information about the system and its processes. Unlike regular files, the contents of /proc are dynamically generated by the kernel, reflecting the current state of the system. It serves as a key resource for monitoring system performance and managing processes

View details about the CPUs/cores in the system.

```
/proc/cpuinfo
```

Detailed information about system memory usage.

```
/proc/meminfo
```

View command-line arguments → how process started

```
cat /proc/[PID]/cmdline
```

Check process limits:

```
cat /proc/[PID]/limits
```

memory usage of process:

```
cat /proc/[PID]/status
```

Checking I/O Activity

```
cat /proc/diskstats
```

# Linux Monitoring: More Linux tools to measure

show load average 1 minute ago 5 and 15 minute ago

```
$ uptime
```

last 10 system messages

```
$ dmesg | tail
```

command that displays system performance statistics, refreshing every second. It provides information about:

```
$ vmstat 1
```

Key vmstat columns to check:

- r: Processes running on the CPU or waiting. Values above the CPU count indicate saturation.
- free: Free memory in KB. High values mean sufficient memory.
- si, so: Swap-ins and swap-outs. Non-zero values indicate low memory.
- us, sy, id, wa, st: CPU time breakdown: user, system, idle, I/O wait, and stolen (by other guests or drivers).

This command prints CPU time breakdowns per CPU, which can be used to check for an imbalance. A single hot CPU can be evidence of a single-threaded application.

```
$ mpstat -P ALL 1
```

Pidstat is a little like top's per-process summary, but prints a rolling summary instead of clearing the screen

```
$ pidstat 1
```

iostat: This tool helps analyze disk workload and performance:

- r/s, w/s, kB/s, kB/s: Delivered reads/writes and data per second. High values indicate a heavy workload.
- await: Average I/O time (ms), including queue and service time. High values suggest saturation or device issues.
- avgqu-sz: Average queued requests. Values >1 may indicate saturation, though parallel processing can help.
- %util: Device busy time per second. Above 60% may affect performance; close to 100% indicates saturation.

```
$ iostat -xz 1
```

The free -m command shows system memory usage in megabytes:

- total: Physical RAM and swap space available.
- used: Memory and swap currently in use (programs, kernel).
- free: Completely unused memory and swap space.
- shared: Memory shared between processes (e.g., tmpfs).
- buff/cache: Memory for I/O buffers and file caches.
- available: Estimated memory available for new applications, considering free and reclaimable cache.

```
$ free -m
```

Use `sar -n DEV 1` to check network interface throughput: rxkB/s and txkB/s, as a measure of workload, and also to check if any limit has been reached. In the above example, eth0 receive is reaching 22 Mbytes/s, which is 176 Mbits/sec (well under, say, a 1 Gbit/sec limit).

```
$ sar -n DEV 1
```

Use `sar -n TCP,ETCP 1` to check key TCP metrics:

- active/s: Locally-initiated TCP connections per second (e.g., `connect()`).
- passive/s: Remotely-initiated TCP connections per second (e.g., `accept()`).
- retrans/s: TCP retransmissions per second.

Active and passive counts indicate server load: outbound (active) and inbound (passive) connections. High retransmits suggest network issues or server overload.

```
$ sar -n TCP,ETCP 1
```

The `perf` command is useful for diagnosing short-lived CPU-intensive processes:

```
$ perf record -F 99 -a -g --sleep 10
```

Check report

```
$ perf report -n --stdio
```

This command traces the last system calls of the lab0031 process, helping identify if it has entered an infinite loop or is stuck.

```
$ strace -tp `pgrep lab0031 2>&1 | head -100
```



# Linux Monitoring: More commands to measure system performance

## 1) Zabbix

Zabbix is an Open Source, high-level enterprise software designed to monitor and keep track of networks, servers, and applications in real-time. Built in a server-client model, Zabbix can collect different types of data that are used to create historical graphics and output performance or load trends of the monitored targets.

The server has the ability to check standard networking services (HTTP, FTP, SMTP, IMAP, etc) without the need to install extra software on the monitored hosts.

## 8) Zabbix agents

However, in order to gather data and create statistics about local services or other specific system resources that run on remote instances, such as CPU, disks, internal system process, RAM, etc, you need to install and configure a Zabbix agent.

great tutorial about ZABBIX, you can find there:

### How to Install Zabbix

Zabbix is an Open Source, high-level enterprise software designed to monitor and keep track of networks



<https://www.tecmint.com/install-and-configure-zabbix-monitoring-on-debian-centos-rhel/>



## Learn more about Linux Security

**Check TecMint and Chris Linux Tech, they have great docs!**

The 3 Biggest Security Mistakes Linux Users Make


Security is a journey, not a destination

 <https://christitus.com/linux-security-mistakes/>



26 Security Hardening Tips for Modern Linux Servers

Everybody says that Linux is secure by default, and to some extent

 <https://www.tecmint.com/linux-server-hardening-security-tips/>



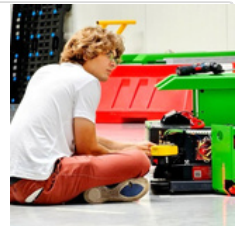
## Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

**Check my GitHub**

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

<https://github.com/MichaelRobotics>



*PS.*

*If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!*