**Monitoring & Troubleshooting — Step-by-Step Guide**

**1 — Quick overview**

* **Monitoring** = collecting metrics, logs, and alerts so you know system health before users complain.
* **Troubleshooting** = triage → diagnose → mitigate → resolve → post-mortem.
* Always follow a Runbook: *Detect → Triage → Contain → Fix → Verify → Document*.

**2 — Essential tools (commands you must know)**

# System / process / CPU / memory

top

htop

ps aux | sort -nrk 3,3 | head # top CPU consumers

ps aux | sort -nrk 4,4 | head # top memory consumers

# Disk / I/O

df -h

du -sh /path

lsblk

ls -l /dev/disk/by-uuid

iostat -x 1 3

iotop -o

# Network

ip addr show

ip route show

ss -tulnp # open ports & processes

ss -s # summary

tcpdump -i eth0 -w /tmp/cap.pcap

traceroute, mtr

dig, nslookup

# Logs / journal

journalctl -xe

journalctl -u servicename -f

tail -n 200 /var/log/messages

tail -n 200 /var/log/syslog

# Files / locks / FDs

lsof -i :8080

lsof +D /mountpoint

fuser -vm /mountpoint

# System performance history

sar -u 1 5 # CPU history

sar -d 1 3 # disk activity

# Other

vmstat 1 5

free -m

uname -a

dmesg | tail -n 50

smartctl -a /dev/sda # disk SMART health

systemctl status svc

systemctl list-units --type=service --state=failed

systemd-analyze blame # slow boot services

**3 — What to monitor (key metrics & recommended thresholds)**

* **CPU usage:** sustained > 80% (or > #cores × 1 load) → alert.
* **Memory usage:** free + cached < 10% or swap used heavily → alert.
* **Load average:** > #cores (e.g., load > 8 on 4-core) → alert.
* **Disk usage:** filesystem > 80% (critical at 90+%) and inode usage >90% → alert.
* **Disk I/O wait:** high iowait or high await in iostat → alert.
* **Network:** high packet loss, retransmits, interface errors → alert.
* **Service/process health:** process down or crash loops → alert.
* **Log errors:** error/exception patterns (5xx for web apps) → alert.
* **Hardware health:** SMART failing attributes → alert.

**4 — Setting up basic monitoring (starter stack)**

* **Metrics:** Prometheus + node\_exporter (system metrics) → Grafana dashboards.
* **Alerts:** Prometheus Alertmanager (send to Slack/pager/email).
* **Logs:** rsyslog/journald → central log collector (Fluentd/Logstash → Elasticsearch → Kibana).
* **Host checks:** Nagios/Zabbix/Check-mk for service availability.
* **Application traces:** Jaeger/OpenTelemetry for distributed tracing.

*(If you want sample Prometheus alert rules I’ll add them — quick example later in this note.)*

**5 — General troubleshooting workflow (step-by-step)**

1. **Acknowledge & collect context**
   * Who reported, when, scope (single host vs cluster), error messages, recent changes/deploys.
2. **Check monitoring & alerts**
   * Check dashboards, alert history, recent deploys/patches.
3. **Access the host**
   * SSH to the host (use bastion if required). Preserve logs. Do not immediately reboot.
4. **Gather quick health overview**
5. uptime; top -b -n1 | head -n20; free -m; df -h; ss -tunap | head
6. journalctl -n 200 --no-pager
7. dmesg | tail -n 50
8. **Triaging** — identify whether it’s CPU/MEM/DISK/IO/NETWORK/SERVICE.
9. **Contain** — if user impact large, take mitigation (restart service, scale out nodes, disable ingress) while diagnosing deeper cause.
10. **Diagnosis** — use targeted commands/tools below per symptom.
11. **Remediate** — apply fix (restart, kill, change config, rollback). Prefer graceful actions first.
12. **Verify** — confirm via logs, metrics, and user testing.
13. **Document & Post-mortem** — what caused it, how fixed, steps to avoid recurrence.

**6 — Symptom playbooks (common incidents)**

**A. Server unreachable (ping/SSH fail)**

1. Ping host: ping -c4 <ip>.
2. If ping fails, check network route: traceroute <ip>.
3. If unreachable from your network, try from another network or jump host.
4. Check firewall rules on host (if reachable via console): iptables -L / firewall-cmd --list-all.
5. Console access (cloud provider / iLO / iDRAC) — view interface, reboot if kernel panic.
6. Check dmesg and journalctl -b for NIC/driver errors.

**B. High CPU usage**

1. top / htop → find PID consuming CPU.
2. ps -p <pid> -o pid,ppid,cmd,%cpu,%mem
3. If process expected, check its logs; if stuck in loop, capture stack:
   * For interpreted apps: inspect app logs or use thread/stack dump (Java: jstack <pid>).
   * For native: strace -p <pid> -o /tmp/strace.$pid (short duration) to see syscalls.
4. If process is runaway: try graceful restart (systemctl restart svc), kill -15 <pid>, then kill -9 <pid> if needed.
5. To reduce impact immediately: renice -n 10 -p <pid> or cpulimit (temporary).

**C. High memory consumption / OOM**

1. Check free -m, top sorted by MEM.
2. ps aux --sort=-%mem | head to find culprits.
3. Check /var/log/messages and dmesg for OOM killer logs.
4. Inspect process memory map: pmap -x <pid> | head.
5. If memory leak suspected, take heap/core dump, restart process, analyze.
   * Enable core dumps: ulimit -c unlimited and set kernel.core\_pattern.
   * gcore <pid> to capture core (if gdb installed).
6. Consider adding swap or restarting service, and plan for root cause fix.

**D. Disk full or nearly full**

1. df -h to see which FS is full.
2. Check largest directories: du -sh /\* 2>/dev/null | sort -h or ncdu /path for interactive.
3. Find large files: find / -xdev -type f -size +100M -exec ls -lh {} \;
4. Common culprits: logs, core dumps, backups. Remove or rotate logs: logrotate, or move backups to remote.
5. If /var or / is full and prevents services, consider:
   * Cleaning package cache (yum clean all or apt-get clean),
   * Removing old kernels,
   * Mounting temporary extra storage,
   * Resizing (LVM) or adding disks (live grow).
6. After cleaning: systemctl restart affected-services and confirm.

**E. High Disk I/O / Slow I/O**

1. iostat -x 1 5 to identify device with high await.
2. iotop -o to see processes causing I/O.
3. dmesg for disk/controller errors (retries).
4. If disk/array failing: check RAID status cat /proc/mdstat, mdadm --detail /dev/md0.
5. Mitigation: move heavy jobs to off-peak, add disks, rebalance LVM, replace faulty disk.

**F. Service failing / Crash loop**

1. systemctl status <svc> → note unit logs & errors.
2. journalctl -u <svc> -b --no-pager for detailed logs.
3. Check application logs under /var/log/....
4. If due to config change, revert and reload: systemctl daemon-reload then systemctl restart <svc>.
5. If DB connectivity issue: check DB status (systemctl status mysqld), network connectivity (ss -tnl | grep 3306), and DB logs.

**G. Network slowness or packet loss**

1. ss -s & ip -s link to check interface errors.
2. ethtool -S eth0 for NIC stats (driver dependent).
3. Use mtr <host> to see packet loss per hop.
4. Capture traffic: tcpdump -i eth0 host <peer> and port <port> -w /tmp/cap.pcap. Analyze in Wireshark.
5. Check MTU mismatches: ip link and test with ping -M do -s <size> <host>.
6. If congestion, rate limit or QoS may be needed, or scale network.

**H. DNS resolution problems**

1. cat /etc/resolv.conf check nameservers.
2. dig @<dns> example.com +short to test specific server.
3. If DNS unreachable, check network to DNS server, firewall, and DNS service health.

**I. Boot failure / kernel panic**

1. Use console/serial or cloud provider’s console to access logs.
2. journalctl -xb to see boot logs for current failed boot.
3. systemd-analyze blame to find slow or failing services.
4. Boot into rescue/rescue.target: systemctl isolate rescue.target via console or grub.
5. If kernel update caused issue: boot previous kernel from GRUB and remove faulty kernel.

**7 — Log analysis best practices**

* Centralize logs (rsyslog/Fluentd -> ELK/Graylog) so you can search across hosts.
* Use structured logs (JSON) where possible.
* Use log levels and alerts on error/warn patterns.
* Common grep patterns:

grep -i -E "error|fail|exception|trace" /var/log/app.log -n

journalctl -u servicename | grep -i error

* Save relevant logs for incident post-mortem.

**8 — Capturing evidence (if you need to forensics)**

* Save volatile info:

ps aux > /tmp/ps.aux

ss -tunap > /tmp/ss.txt

netstat -rn > /tmp/netstat.txt

ip addr > /tmp/ip.addr

dmesg > /tmp/dmesg.txt

journalctl -b > /tmp/journal.boot.txt

* Copy core dumps and config files to a safe location for analysis.

**9 — Automated remediation & runbooks**

* **Automated restart:** systemd can auto-restart: Restart=on-failure in unit file. Use with caution.
* **Monit** or simple cron scripts can auto-restart services and email on failure.
* Maintain runbooks for common incidents: include detection query, quick mitigation commands, escalation contacts, rollback steps.

**Example runbook snippet: Disk full on /var**

1. Alert triggers: >85% /var.
2. SSH to host.
3. Command to run:

df -h /var

du -sh /var/log/\* | sort -h | tail -n 20

journalctl --vacuum-size=200M # shrink journal if needed

rm -f /var/log/old-log-file.log

systemctl restart rsyslog

1. If not resolved: mount temp disk and move /var/log, open incident.

**10 — Prometheus example alert (simple)**

groups:

- name: linux.rules

rules:

- alert: HighCpuUsage

expr: 100 - (avg by (instance) (irate(node\_cpu\_seconds\_total{mode="idle"}[5m])) \* 100) > 85

for: 5m

labels:

severity: critical

annotations:

summary: "High CPU on {{ $labels.instance }}"

description: "CPU > 85% for 5 minutes"

**11 — Post-incident: What to record**

* Incident summary (timeline with exact timestamps).
* Root cause (technical).
* Immediate fix & permanent fix.
* What worked / didn’t work in runbook.
* Action items (who, what, when).
* Add tests to ensure recurrence cannot happen.

**12 — Troubleshooting checklist (quick)**

1. Check alerts/dashboards.
2. Check host health: uptime, top, df -h, free -m.
3. Check logs: journalctl and app logs.
4. Check network: ss, ip route, ping, tcpdump.
5. Check storage: iostat, lsblk, mdadm.
6. Check processes: ps, lsof.
7. Apply mitigation (restart/scale/migrate).
8. Verify and document.

**13 — Interview Q&A (common)**

* **Q:** How do you find the process causing high CPU?  
  **A:** top / ps aux --sort=-%cpu | head then strace or check app logs.
* **Q:** What do you check when disk is full?  
  **A:** df -h, du -sh top directories, find / -type f -size +100M.
* **Q:** How to debug slow web app?  
  **A:** Check server CPU/memory/IO/network, app logs, DB health, instrument with APM/traces.
* **Q:** How to capture packets for network debugging?  
  **A:** tcpdump -i eth0 host <peer> and port <port> -w /tmp/cap.pcap.
* **Q:** What is iostat used for?  
  **A:** Identify disk throughput and latency (await, util) to detect I/O bottlenecks.

**14 — Practical tips & safety**

* Always run read-only commands first (no destructive ops).
* If restarting a service, prefer systemctl restart over killing processes directly.
* Keep an unchanged terminal connected while testing (don’t close SSH session you are using to fix).
* When in doubt, escalate early — don’t spend hours alone on a production outage.

**📘 Monitoring & Troubleshooting in Linux**

**🔹 1. What is Monitoring?**

Monitoring means continuously observing system resources, performance, and processes to ensure availability, stability, and reliability of servers.

👉 It helps detect issues **before** they affect end-users.

**🔹 2. Key Areas to Monitor**

* **CPU usage**
* **Memory usage (RAM + Swap)**
* **Disk usage & I/O performance**
* **Network traffic**
* **Processes & services**
* **System logs**
* **Security (failed logins, sudo usage, etc.)**

**🔹 3. Basic Monitoring Commands**

**🖥️ CPU Monitoring**

top # Real-time CPU, memory, process usage

htop # Better version of top (needs installation)

mpstat 1 # CPU utilization per core (from sysstat package)

uptime # Load average & system uptime

**💾 Memory Monitoring**

free -m # Check RAM & swap usage

vmstat 2 # Memory, CPU, process statistics

cat /proc/meminfo # Detailed memory info

**📂 Disk Monitoring**

df -h # Check filesystem disk usage

du -sh /dir # Size of a directory

iostat -x 2 # Disk I/O statistics (sysstat package)

lsblk # Block devices (disks, partitions, LVM)

**🌐 Network Monitoring**

ifconfig # Interface details (deprecated, use ip command)

ip addr show # Show IP addresses

ss -tulnp # Show listening ports and services

netstat -tulnp # Old command, requires net-tools

ping google.com # Check connectivity

traceroute 8.8.8.8 # Check packet path

**📋 Process Monitoring**

ps aux # All processes

pstree # Process tree

pidstat -p <pid> # Monitor specific process

**📑 Log Monitoring**

journalctl -xe # Check systemd logs with details

tail -f /var/log/messages # General system logs

tail -f /var/log/secure # Authentication logs

**🔹 4. Advanced Monitoring Tools**

* **Nagios / Icinga** → Service & host monitoring
* **Zabbix** → Full monitoring with dashboards
* **Prometheus + Grafana** → Modern metric monitoring & visualization
* **ELK Stack (Elasticsearch, Logstash, Kibana)** → Log monitoring

**🔹 5. Troubleshooting Steps (General)**

1. **Identify the Issue**
   * Collect error messages (logs, dmesg, journalctl).
   * Confirm the symptoms (high CPU, slow network, disk full).
2. **Reproduce the Problem** (if possible).  
   Example: App slow? → Check CPU load, memory, I/O.
3. **Isolate the Cause**
   * Check system resource usage.
   * Identify failed service/process.
   * Check recent changes (patch, config updates).
4. **Apply Fix**
   * Kill unresponsive process (kill -9 PID).
   * Restart failed service (systemctl restart httpd).
   * Free disk space (rm, logrotate).
   * Adjust configs (sysctl, limits.conf).
5. **Verify**
   * Re-check service status.
   * Confirm logs are clean.
   * Ensure users confirm issue is resolved.

**🔹 6. Example Troubleshooting Scenarios**

**🖥️ High CPU Usage**

top

ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head

👉 Find top consuming process → Kill or optimize.

**💾 Out of Memory**

free -m

dmesg | grep -i oom

👉 Check OOM killer logs → Increase swap / add memory.

**📂 Disk Full**

df -h

du -sh /var/log/\*

👉 Clean old logs, compress backups, extend LVM if required.

**🌐 Network Slowness**

ping 8.8.8.8

traceroute google.com

ss -tulnp

👉 Check latency, packet loss, or firewall blocking.

**🔐 User Cannot Login**

tail -f /var/log/secure

last

👉 Check failed logins, password expiry, or account lock.

**🔹 7. Best Practices for Monitoring & Troubleshooting**

* Always check **logs first** before applying fixes.
* Use **monitoring tools** for proactive alerts.
* Document all changes & troubleshooting steps.
* Keep **backup configs** before modifying.
* Apply **root cause analysis (RCA)** after every incident.

**🔹 8. Interview Questions & Answers**

**Q1. How do you check CPU usage in Linux?**  
👉 Use top, htop, mpstat, or uptime for load averages.

**Q2. How do you monitor disk I/O performance?**  
👉 Use iostat, iotop, or sar -d.

**Q3. What steps do you take if a server is running slow?**  
👉 Check CPU (top), memory (free -m), disk (df -h, iostat), and network (ping, ss).

**Q4. How do you troubleshoot a service that won’t start?**  
👉 Use systemctl status <service>, check journalctl -xe, fix config/log errors.

**Q5. How do you monitor logs in real time?**  
👉 tail -f /var/log/messages or journalctl -f.

**Q6. What tools do you use for enterprise monitoring?**  
👉 Nagios, Zabbix, Prometheus+Grafana, ELK Stack.