

Monitoring & Server Setup

- [Monitor server load, disk, CPU and memory with email alerts](#)
- [Auto-create home folders for Active Directory users on RHEL / Rocky](#)
- [A production warning banner, safer aliases and a colour prompt for every login](#)
- [List listening ports and the programs using them](#)
- [Email yourself an internet speed test report](#)

Monitor server load, disk, CPU and memory with email alerts

A small shell script, run from cron every 5 minutes, that checks four health numbers and emails you when one goes over its limit. It doesn't need an agent or a monitoring server.

What it checks

Check	How it is read	Default limit
1-minute load average	<code>/proc/loadavg</code>	10.00
CPU use	<code>vmstat 1 2</code> (100 minus idle)	98 %
Root disk use	<code>df --output=pcent /</code>	98 %
Memory use	<code>free</code> (used ÷ total)	98 %

Each limit can be overridden with an environment variable, so one script works on small and large machines:

```
LOAD_THRESHOLD=4 DISK_THRESHOLD=90 RECIPIENTS="you@example.com" /usr/local/bin/loadmon.sh
```

Requirements

`awk`, `df`, `free`, `vmstat` (procps), `bc`, `logger`, a `mail` command (s-nail) with a working mail relay, and optionally `sar` (sysstat) to include recent history in the alert email.

Check for every tool at start-up, including `bc`, so a missing tool fails loudly instead of silently skipping alerts:

```
for c in awk df free vmstat bc mail logger; do
  command -v "$c" >/dev/null || { logger -t loadmon "missing $c"; exit 1; }
done
```

Reading the numbers

```
load=$(cut -d' ' -f1 /proc/loadavg)
cpu=$(vmstat 1 2 | awk 'END {print 100 - $15}')
disk=$(df --output=pcent / | tail -1 | tr -dc '0-9')
mem=$(free | awk '/^Mem:/ {printf "%d", $3 * 100 / $2}')
if (( $(echo "$load > $LOAD_THRESHOLD" | bc -l) )); then ... fi
```

Avoid parsing `top -bn1` for CPU. Its first frame is an average since boot, and its column positions shift.

Schedule it

```
/etc/cron.d/loadmon:
```

```
*/* * * * * root /usr/local/bin/loadmon.sh >/dev/null 2>&1
```

Tips

- **Throttle alerts.** Write a timestamp file per check (for example `/var/lib/loadmon/disk.last`), and only email again after an hour. Otherwise a full disk sends 12 emails an hour.
- **Rotate the log.** One line every 5 minutes is about 100,000 lines a year. Add `/etc/logrotate.d/loadmon`.
- **Check more than `/`.** Loop over `/var`, `/home` and data volumes too.
- Use `hostname -s` in the subject line so you can tell servers apart.

Auto-create home folders for Active Directory users on RHEL / Rocky

When Linux servers are joined to Active Directory (SSSD or winbind), domain users who log in for the first time have no `/home/<user>`.

Recommended: use the built-in feature

RHEL, Rocky and Alma 8/9 ship this already:

```
sudo dnf install oddjob-mkhomedir
sudo authselect enable-feature with-mkhomedir
sudo systemctl enable --now oddjobd
```

The home folder is then created from `/etc/skel` at first login, with the right owner and SELinux context.

Manual fallback (one user)

If you need to create a home folder by hand, quote everything. Group names like `domain users` contain a space.

```
#!/bin/bash
set -euo pipefail
u=${1:?usage: mkhome <user>}
id "$u" >/dev/null          # make sure the user resolves
install -d -m 700 -o "$u" -g "domain users" "/home/$u"
```

```
cp -rT /etc/skel "/home/$u"  
chown -R "$u":"domain users" "/home/$u"  
restorecon -R "/home/$u"
```

Gotcha

Copying commands from Word or email can turn " into curly "smart quotes". The shell treats them as part of the text, so `chown` gets a group name that doesn't exist. Retype the quotes if a command fails mysteriously.

A production warning banner, safer aliases and a colour prompt for every login

Drop-in files in `/etc/profile.d/` apply to every user's login shell. Here are three small ones.

1. Safer aliases: `/etc/profile.d/custom-aliases.sh`

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias vi='vim'
```

These only affect interactive shells, not scripts.

2. Production warning: `/etc/profile.d/custom-prod-warning.sh`

Show it only on hosts that really are production. If every box shows it, people learn to ignore it.

```
[[ $- == *i* && -t 0 && -t 1 ]] || return
grep -qx "$(hostname -s)" /etc/production-hosts 2>/dev/null || return
RED='\033[1;31m'; NC='\033[0m'
```

```

if command -v figlet >/dev/null; then
    figlet -f big PRODUCTION | while IFS= read -r l; do printf "${RED}%s${NC}\n" "$l"; done
else
    echo -e "${RED}***** PRODUCTION SERVER *****${NC}"
fi
echo -e "${RED}This is a LIVE system. Think twice before running commands.${NC}"

```

3. Two-line colour prompt:

/etc/profile.d/custom-prompt-and-colors.sh

```

if [[ $- == *i* && -t 0 && -t 1 && -z ${SSH_ORIGINAL_COMMAND:-} ]]; then
    R=$(tput sgr0); B=$(tput bold); G=$(tput setab 8)
    RED=$(tput setaf 1); MAG=$(tput setaf 5); GRN=$(tput setaf 2); CYN=$(tput setaf 6)
    PS1="\n[$B$G$RED]\u[$MAG]@[$RED]\h[$GRN] : \d \t[$R]\n[$CYN][\w][$R$B] \\$ \[$R]"
    export PS1
fi

```

- Wrap every colour code in `\[ \]`, or line editing gets confused.
- The `-t` and `SSH_ORIGINAL_COMMAND` checks keep `scp`, `rsync` and Ansible sessions clean. Any output from profile scripts breaks them.

Notes

- `tailf` was removed from util-linux; use `tail -f`.
- Files are read in alphabetical order; prefix them (`zz-`) if order matters.

List listening ports and the programs using them

Before you publish a new container port, check what is already listening. `ss` shows everything; this tidies the output into one line per protocol and port.

Quick one-liner

```
sudo ss -Htulnp | awk '{split($5,a,"."); port=a[length(a)]; print $1, port, $7}' | sort -k2,2n -u
```

Example output:

```
tcp 22 users:(("sshd",pid=812,fd=3))
tcp 443 users:(("docker-proxy",pid=1450,fd=4))
udp 53 users:(("dnsmasq",pid=990,fd=5))
```

- `-H` drops the header. `-t`/`-u` select TCP and UDP, `-l` listening sockets only, `-n` numeric ports, and `-p` the process (needs sudo).
- `sort -u` on the port column removes the duplicate IPv4 and IPv6 lines.

Older systems and NAS devices

Where `ss` is missing, use `netstat -tulnp` (net-tools). The columns differ: the local address is field 4 and the program is field 7.

Colour (optional)

If you add colour, **sort first and colour last**. Sorting text that already contains ANSI codes gives the wrong order. Also, `match(str, re, arr)` in awk is a GNU awk (gawk) feature; busybox and mawk don't have it.

Docker-specific view

```
docker ps --format '{{.Names}}\t{{.Ports}}' | column -t
```

Email yourself an internet speed test report

Run Ookla's speed test on a Linux box and email the result, for example to keep a record of your ISP's performance.

Install the Ookla CLI

Install it from Ookla's package repository, rather than piping a web script into `sudo bash`:

```
# Download the repo setup script, read it, then run it
curl -fsSLo /tmp/ookla.sh https://packagecloud.io/install/repositories/ookla/speedtest-cli/script.rpm.sh
less /tmp/ookla.sh && sudo bash /tmp/ookla.sh
sudo dnf install speedtest
```

To check whether a package is installed, use `rpm -q speedtest`. `rpm -qa speedtest` always returns success, even when the package is missing.

Run and send

```
#!/bin/bash
set -euo pipefail
tmp=$(mktemp -d); trap 'rm -rf "$tmp"' EXIT
speedtest --accept-license --accept-gdpr > "$tmp/speedtest.txt"
enscript -q -p - "$tmp/speedtest.txt" | ps2pdf - "$tmp/speedtest.pdf"
mutt -s "Speed test $(hostname -s) $(date +%F)" -a "$tmp/speedtest.pdf" -- you@example.com <
"$tmp/speedtest.txt"
```

Points to note:

- Run `mutt` in the **foreground**. If you background it with `&` and then delete the attachment, the email can go out without it.
- Use `mktemp` rather than fixed names in `/tmp`.

Requirements

`speedtest`, `enscript`, `ghostscript` (`ps2pdf`), `mutt` and a working mail relay.

Schedule it (optional)

```
0 7 * * * root /usr/local/bin/speedtest-report.sh
```