MrRowie · MrsNanny module

Pull, not push.
MrsNanny

The vigilance module of MrRowie: modular health monitoring for customer sites — roll it out once, extend it with a script toolbox, get a daily AI status report in plain language. No cloud, no off-the-shelf agent software.

Bash + Python
NO FRAMEWORK BLOAT

Restricted SSH
PULL INSTEAD OF PUSH

Local LLM
REPORTS WITHOUT CLOUD

16 collectors
EXTENSIBLE AT WILL

THE TOOLBOX

All aboard: sensors, maintenance, reports

The platform is a toolbox of small, sharply cut building blocks — each deployable on its own, together a complete picture of the fleet's health. This is what ships:

16 collectors + 4 scanners

Measure each node's actual state every day — every number with a traffic-light self-rating, extensible with a script at any time.

- speedtest
- disk
- updates
- reboot
- services
- temperature
- wazuh-agent
- patchmon
- replication
- nextcloud
- wordpress
- openarchiver
- pangolin-stack
- autosnapshot
- system
- dockhand
- fancontrol
- semaphore
- apt-lock
- kvm-spin

Maintenance that runs along

Not just reporting — remedying: automatic security updates fleet-wide and weekly Nextcloud care with a daily status snapshot — including multiple instances per host.

- unattended-upgrades
- occ maintenance
- app:update --all
- core patch/minor
- multi-instance

Reports, checks & alerts

Hourly instant alerts self-sufficient on the node, the LLM daily report per customer every morning — plus the monthly management report, the report archive, the fleet web overview and the NOC view.

- instant alert :30
- daily report 07:10
- monthly mgmt report
- fleet web
- archive 400 days

WHAT THIS IS ABOUT

A thin sensor on the node, the intelligence on the master

MrsNanny answers one simple question every day: *"How are my machines doing?"* — as a mail in plain language, written by a locally running language model, with a traffic-light verdict per customer.

Deliberately little runs on the monitored node: a runner (`export.sh`) that collects a handful of collector scripts and lays out their result as one JSON. All the smart parts — rating, AI summary, HTML mail, trend history — happen centrally on the master. The node needs neither VPN nor LLM nor open ports.

Pull, not push

The master fetches the data via a restricted SSH key (forced command). The customer node never opens a connection to the outside — trust flows only from master to node.

Thin agent, smart master

The node is a passive sensor. AI report, verdict and dispatch sit centrally — the Ollama load stays under control, no LLM and no mail template on the node.

Sensors rate themselves

Every collector ships its own traffic light (green/yellow/red). The master does not need to know new sensors — they simply appear in the next report.

ARCHITECTURE

Two halves, one direction

Customer node /opt/monitoring-agent · user servernanny

- `export.sh` — runner: collectors → one JSON
- `collectors.d/` — the sensors (list below)
- `scanners.d/` — state checks, first one live
- `measure.sh` — speedtest 4×/day via cron
- `alert.sh` — instant alert, self-sufficient via msmtp
- `agent.env` — thresholds per node



restricted SSH forced command pull, read-only

Master on the Wazuh VM

- `monitoring-report.py` — the report engine
  - ├ pull of all sites
  - ├ verdict (traffic light per node)
  - ├ Ollama → plain-language report
  - ├ HTML mail + detail PDF per customer
  - └ history: 13 months retention (trends)
- `sites/*.conf` — one file per node
- `onboard-site.sh` — rollout in a single run
- `build-web.py` — static fleet web overview

If a node is unreachable during the pull, a red outage alert goes out automatically — "no data" is a finding in itself.

A deliberate exception: Nodes that are not always on (workstations) are marked `INTERMITTENT`. An hourly opportunistic pull picks up whatever happens to be running; the daily report rates the last known state and shows its age. Only when the node stays away for days does the absence itself become a yellow finding — a powered-off workstation is no longer a false alarm.

THE SENSORS

Collectors: small scripts, clear responsibility

Every collector is a self-contained script that prints exactly one JSON object — self-rating included.
The number prefix only controls ordering.

10 **speedtest**
bandwidth against target, fixed test server optional

20 **disk**
usage + inodes, instant alert when full

30 **updates**
pending updates (apt), yellow only after a grace period

40 **reboot**
pending restart; on Proxmox via kernel comparison

50 **services**
failed systemd units

60 **temperature**
sensors; NVMe rated at composite; bare metal only

70 **wazuh-agent**
is the security agent running and connected?

75 **patchmon**
is the node reporting to the patch inventory?

76 **nextcloud**
instance health from the care snapshots, multi-instance

80 **replication**
backup/replication freshness from job spool files

81 **autosnapshot**
are the ZFS auto-snapshots actually running? target from cron, actual from the pool

85 **wordpress**
core/plugin/theme updates via wp-cli, multi-instance

86 **openarchiver**
is the mail archive really ingesting? source status + mail freshness

87 **pangolin-stack**
reverse-proxy stack: container health, detection pulse, bans

88 **system**
RAM (ARC-aware), swap, load, ZFS pools, NVMe SMART — read directly

89 **dockhand**
container updates across the Docker fleet, with a grace period

Updates: report and remedy

30-updates tracks since when each package has been pending and turns yellow only for stragglers (security after 2 days, everything after 14) — freshly available updates are green and normal. Its counterpart `setup-unattended-upgrades.sh` rolls out automatic security updates fleet-wide: idempotent, with a Samba package blacklist, Proxmox kernels stay manual. Yellow now means: something is actually stuck here. For containers **89-dockhand** plays the same role: it reads the update findings of the Docker management for the entire container fleet — new images behind the same tag as well as newer version tags — and reports them, after a grace period, as *one* line in the daily report instead of individual mails.

Backups: silence is red

80-replication reads the spool files the backup jobs (ZFS replication, offsite pull, PBS) leave behind in Checkmk local-check format — and rates line status **and** file freshness. A job that stops delivering turns red, not silent. Without a spool directory: "not applicable", safe to roll out fleet-wide. One layer below sits **81-autosnapshot**: it derives the target from the host's auto-snapshot cron jobs and compares it with the snapshots actually present in the pool — a missing or stale label turns yellow, never red by design.

Nextcloud: care, not just metering

Two systemd timers per node: weekly occ maintenance (db repairs, `app:update --all`, files:scan) and a daily status snapshot. Patch/minor core updates run automatically — with a DB dump before and a verify after; majors are only reported, Docker instances update via their image. Multiple instances per host via `NC_INSTANCES`; collector 76 rates each one, the worst traffic light wins.

Alongside there is `scanners.d/` — technically the same contract, in substance the question *"is a state correct?"*. Four scanners are live: **10-fancontrol** checks fan control not just as a service but measures whether the regulated fans actually spin — born after a control daemon silently died following a kernel update and stayed dead for months. **20-semaphore** watches the Ansible scheduler and flags orphaned long-running tasks and OOM kills — likewise the answer to a silent double outage no alert mail ever reported. **30-apt-lock** flags stuck package-manager processes and watchdog interventions, **40-kvm-spin** spots permanently burning vCPU threads of single VMs on Proxmox hosts — load that drowns in the host total. Further candidates: open ports, certificate expiry, expected processes.

ALERTING

Two levels: instant and daily

LEVEL 1 · INSTANT

Self-sufficient on the node

If a collector with `alert_immediate: true` turns red (disk full, internet collapsed), the node mails **immediately by itself** via its local msmtp — regardless of whether the master is alive.

Hourly cron, dedup to 1 mail/day per sensor, re-arms on recovery.

LEVEL 2 · DAILY REPORT

Central on the master

Every morning at 07:10 — after the nightly update and reboot window — the master pulls all sites, forms a traffic-light verdict per node and has the local LLM (Ollama) write a plain-language report — grouped **per customer**: one HTML mail covering all devices, plus a detail PDF attached (each device on its own page).

Details only appear when ≠ green — the report stays readable.

REPORTING

From daily digest to the monthly management report

Three more views grow out of the same trend history — with no extra access to the nodes:

Monthly management report

Aggregates per customer check coverage, finding days, update trajectory and protection software into a PDF with a management summary — deliberately sober wording, with a deterministic fallback if the LLM is down.

Report archive

Every production run stores HTML and PDF structured by customer/year/month, cleanup automatically after 400 days. No digging through mailboxes.

Fleet web overview

A static website generated from the history: fleet table, per customer the report archive, per device a traffic-light timeline, trend charts and daily detail pages including PDF. Served behind the reverse proxy, access via IP allowlist. On top the **NOC view**: fleet, LibreNMS, Uptime Kuma, Wazuh and open tasks as one live tile overview, refreshed every 5 minutes — internal only.

EXTENSIBILITY

The collector contract: drop in a script, done

The heart of the modularity. A new sensor is an executable script in `collectors.d/` printing a JSON object of this shape — neither runner nor SSH key nor master need to be touched:

```
{
  "name":    "disk",
  "title":   "Disks",
  "status":  "green",          // the traffic-light self-rating
  "summary": "root 42 %, data 67 %",
  "alert_immediate": true,     // red → node mails instantly by itself
  "metrics": { "root_pct": 42 }, // for the LLM + trends
  "details": [ "/  42 %  (210 GB free)" ]
}
```

- ✓ **Always valid JSON, always exit 0.** Even on failure — the state lives in the JSON, not in the exit code.
- ✓ **Read-only.** Collectors only read; installing and changing happens exclusively through onboarding.
- ✓ **Fast & cheap.** Runs daily or more often — cap timeout-prone calls yourself.
- ✓ **Thresholds from the environment.** `agent.env` per node (e.g. `DISK_WARN_PCT`), sensible defaults otherwise.
- ✓ **"Not applicable" is green.** A collector that finds nothing to check on a node says so cleanly — that way the same set can be rolled out to the whole fleet without risk; the master hides not-applicable sensors from the report entirely.

ROLLOUT

Onboarding: a new node in a single run

`onboard-site.sh` runs on the master and handles the complete rollout — all questions come upfront, then it runs through. The site config is only created once the pull test over the production path has passed. Exactly **one** passphrase and **one** sudo prompt per run — no matter how many steps follow:

1 / 8	**Questions upfront** — site, host, customer, collector set, speedtest target, mail, optional Wazuh/PatchMon
2 / 8	**Pack the agent bundle** — and copy it to the target host
3 / 8	**Installation on the node** — non-interactive; the self-test must be green or it aborts
4 / 8	**Wazuh agent (optional)** — version automatically matching the manager on the master
5 / 8	**Automatic security updates** — unattended-upgrades, on by default; Samba packages excluded
6 / 8	**PatchMon enrollment (optional)** — with a preflight that cleanly catches missing approval
7 / 8	**Pull test** — over the real production path: restricted key, forced command
8 / 8	**Create the site config** — only after the pull test has passed
9 / 8	**Report trial run** — dry run, optionally followed right away by the first real dispatch

Profiles turn this into serial rollouts: one `.profile` file per node role bundles collector set, Wazuh group and mail defaults — hypervisor, server, server-extern, workstation plus **samba-dc, samba-member and samba-standalone**. The Samba profiles bridge to the security layer: after the Wazuh step they automatically roll out the matching smb.conf audit from the sister repo `wazuh-config`. Precedence: CLI flag → profile → prompt.

```
# fully interactive
./onboard-site.sh

# guided: profile defaults, the rest is asked
./onboard-site.sh --profile server

# one-liner for serial rollouts
./onboard-site.sh -p server --site 'Customer-Meier-DC1' --host 10.20.30.40 -y
```

Optional attachments in the same run: `--trmm` co-installs the Tactical RMM agent (the master creates the install token itself via the TRMM API). `--wireguard` attaches island sites without a site VPN to the platform via a WireGuard tunnel: the node dials **out** — NAT and dynamic IPs on the customer side do not matter, no port forwarding needed. Onboarding creates the peer automatically via the firewall API, only the platform targets are routed, and Wazuh/PatchMon then enroll straight through the tunnel. If the tunnel effect test fails, onboarding aborts.

For operations afterwards: `update-agent.sh` only updates the code — config, data and keys stay untouched; new sensors are added deliberately via `--add-collector`. `uninstall-agent.sh` cleans up properly.

CONTEXT

Part of a bigger platform

MrsNanny is the **health layer** of a three-part security/monitoring platform — and within the MrRowie concept the vigilance module, whose sibling module MrMiyagi handles the sleeping off-site backup. It does not clash with Wazuh and PatchMon — it interlocks with them:

LAYER	TOOL	QUESTION
Health	MrsNanny (this project, repo monitoring-agent)	Is the machine doing fine? Disk, updates, services, temperature, bandwidth, backups.
Security	Wazuh (SIEM) + repo `wazuh-config`	Is something suspicious happening? The onboarding step installs the Wazuh agent right away; collector 70 watches it afterwards. The manager configuration is versioned in the sister repo (next section).
Patch inventory	PatchMon	What is installed, what is missing? Enrollment also happens in onboarding; collector 75 checks the link.

The trick: The neighbouring systems are not just co-installed, they are themselves monitored. If a Wazuh agent dies, the daily report says so — the monitoring layers watch each other.

THE SISTER REPO

wazuh-config: the security layer, versioned just the same

What the monitoring agent is for the health layer, `wazuh-config` is for the security layer: the complete Wazuh manager configuration as a Git repo — rules, decoders, group configs, Samba audit. The source of truth is the repo, not the manager.

Changes are committed first and then rolled out via `deploy.sh` (with `.bak` backup, config verify and manager restart) — never edited directly on the manager. A manager rebuild stays reproducible at any time.

manager/rules + decoders

Rule tuning (brute-force denoising, web-scan denoising, filters for the monitoring user) and custom Samba rules with their matching JSON decoders — as XML in the repo, not as handiwork on the VM.

manager/shared

One agent config per Wazuh group (exposed, hypervisor, linux-server, workstation, webserver — the latter with vhost logs and realtime FIM on the web roots): SCA intervals, syscheck. The same roles as the onboarding profiles of the monitoring agent.

samba/*.snippet

Ready-made `smb.conf` blocks for domain controllers (auth audit as JSON) and file servers (file audit on change ops, deliberately without flooding) — rolled out automatically via the Samba onboarding profile.

integrations/custom-signal

Critical alerts additionally go out as a Signal message to the phone — CVE alerts throttled per CVE (24 h quiet) and batched into a 60-minute window — one feed surge across the whole fleet yields one message, not dozens. On top: a generic burst brake per rule and host (30 minutes of quiet).

The result are detections that go beyond stock Wazuh — above all on the Samba file servers. The thresholds are calibrated in production: high enough against alert floods, with repeat suppression per aggregate. De-noising works via a **level ladder** instead of deletion: known noise drops to quiet audit levels below the alert and digest thresholds while aggregates keep counting — nothing disappears, it just gets quieter. This is exactly how the mass aggregates (100120/100121) deliberately run at audit level 7 in day-to-day operation — bulk copies and clean-ups are indistinguishable from an incident from the outside.

RULE	LEVEL	DETECTS
`100103`	10	SSH/Samba brute force: 5+ failed logins per IP in 2 minutes
`100104`	10	password guessing: 5+ failed attempts against one user in 5 minutes
`100112`	5	single file deletion (audit trail)
`100113`	3	recycle-bin emptying — does not count as mass deletion
`100120`	7	**mass deletion**: 50+ deletions per user in 60 s (insider scenario)
`100121`	7	**suspected ransomware**: 100+ file changes per user in 60 s
`100130`	12	**audit outage**: full_audit refuses share connects (fail-closed)

Interplay: The monitoring agent brings the Wazuh agent onto the node and watches it (collector 70); wazuh-config determines, versioned, what the manager makes of it. Health and security layer share the same role logic — two repos, one platform.

OUTLOOK

Distribution as a kit (work in progress)

The next expansion step makes the platform itself deployable: `setup.sh` reads a central `platform.env` and orchestrates modules in `modules.d/` — core, llm, wazuh, patchmon, signal, web. The **module contract** mirrors the collector contract one level up: new module = drop in a file, `setup.sh` does not need to know it.

The core idea is the three-way check of every module: *ready* (install), *not configured* (empty required values = a deliberate no, skip silently) or *broken* (configured but endpoint dead = hard error). Add-ons light up as soon as their dependency is there — an empty section in `platform.env` is not an error, a filled one with a dead endpoint is.

STATUS

What runs, what comes

■ IN PRODUCTION

- agent runner, 16 collectors, 4 scanners, instant alerts — incl. system basics (RAM, load, ZFS pools, NVMe) and ZFS auto-snapshots read directly, third-party agent replaced
- master report: pull → verdict → Ollama → HTML mail + detail PDF per customer
- monthly management report, report archive (400 days), fleet web overview + NOC view
- automatic security updates fleet-wide + update grace period
- Nextcloud care as systemd timers: occ maintenance, patch/minor core updates, multi-instance
- onboarding with 7 role profiles (incl. Samba audit) + one-liner
- optional onboarding steps: TRMM agent, WireGuard tunnel for island sites
- installer, updater (--add-collector), uninstaller
- fleet watchdogs in the daily report: PatchMon stale, Newt version drift, container updates across the Docker fleet
- history: 13 months retention; around 100 nodes, 10 customer groups

□ PLANNED / DELIBERATELY OPEN

- more scanners: ports, certs, expected processes
- automatic update chain for apps and containers: snapshot → update → effect check → rollback on failure (concept in place; detection already runs via collector 89)
- modular distribution: modules + contract in place, production hardening open
- Windows health: decision open (candidate: RMM bridge)
- Alpine Linux: deliberately unsupported (bash/apt/systemd assumptions)