CLAWDBOT

Security Playbook

Privacy-by-Design Setup Guide for a Self-Hosted Automation Server

WasatchWise

Hardware: Dell OptiPlex 7040 | i7-6700 | 16GB RAM | 2TB Storage

February 2026

CONFIDENTIAL

1. Hardware Profile

ClawdBot is a Dell OptiPlex 7040 repurposed as a dedicated automation and hosting server for WasatchWise platforms. It runs from an office location with reliable internet connectivity.

Component	Specification	Security Note
Processor	Intel i7-6700 @ 3.40GHz (4C/8T)	CPU-only inference; no GPU attack surface
Memory	16 GB DDR4	Sufficient for containerized stack; upgrade to 32GB if adding services
Primary Storage	238 GB SSD	OS + containers; enable BitLocker encryption
Secondary Storage	1.82 TB HDD	Backups and model files; encrypt with BitLocker or LUKS in WSL2
Graphics	Intel HD 530 (integrated)	No dedicated GPU; eliminates GPU driver vulnerability class
Network	Office ethernet	Reliable uplink; never expose to public internet directly
OS	Windows 10 Pro	WSL2 + Docker; keep Windows Update active
Power	~67W idle, ~140W peak	~$7/month; viable for 24/7 operation

The OptiPlex 7040 was designed for continuous corporate operation. No special cooling or power supply modifications are needed for 24/7 use.

2. Operating System Strategy

Windows 10 Pro stays intact. All server workloads run inside WSL2 (Windows Subsystem for Linux 2), which provides near-native Linux performance without dual-booting or losing Windows access.

2.1 Why WSL2

- **~95% native Linux performance** for CPU-bound workloads (containers, services, automation).
- **No rebooting** between Windows and Linux. Both run simultaneously.
- **Docker runs natively in WSL2** with dynamic memory allocation.
- **Windows stays accessible** for remote desktop, Windows-specific tools, and updates.

2.2 WSL2 Hardening

CRITICAL: CVE-2025-53788 is a known privilege escalation in WSL2. Keep Windows fully updated to patch this and future WSL2 vulnerabilities.

Required Configuration

- ☐ Limit WSL2 memory to 8GB (reserve remaining 8GB for Windows): set memory=8GB in .wslconfig
- ☐ Disable swap in WSL2 to prevent sensitive data written to disk: set swap=0 in .wslconfig
- ☐ Set Windows Update active hours to avoid surprise reboots during automation runs
- ☐ Enable BitLocker on the SSD (Windows 10 Pro includes this) to encrypt data at rest
- ☐ Disable WSL2 localhost forwarding unless needed: set localhostForwarding=false in .wslconfig

2.3 Windows Update Management

Windows 10 Pro forces 3 to 5 involuntary reboots per month. This is the single biggest reliability risk for an always-on server. Mitigations:

- ☐ Set active hours to cover your primary automation windows (e.g., 6 AM to 2 AM)
- ☐ Use Group Policy to defer feature updates by 30 days and quality updates by 7 days
- ☐ Configure n8n workflows to be idempotent (safe to restart mid-run)
- ☐ Set WSL2 and Docker to auto-start on boot via Windows Task Scheduler

3. Docker Security

Docker runs inside WSL2. The Docker daemon and all containers operate within the Linux subsystem, isolated from Windows. This section covers hardening the Docker layer itself.

3.1 Docker Daemon

NEVER expose the Docker daemon socket to the network. Cryptojacking campaigns actively scan for exposed Docker APIs. A single misconfiguration gives attackers full control of all containers and the host.

- ☐ Bind Docker daemon to Unix socket only (the default in WSL2); never bind to 0.0.0.0 or a TCP port
- ☐ Do not mount /var/run/docker.sock into any application container. Only Portainer needs it, and mount it read-only.
- ☐ Enable Docker user namespaces (userns-remap) so container root maps to an unprivileged host user
- ☐ Enable Docker Content Trust for image pulls (export DOCKER_CONTENT_TRUST=1) where supported, but note this is being deprecated. Migrate to Sigstore/Notation.

3.2 Container Runtime (runc)

Three critical runc vulnerabilities were disclosed in late 2025 (CVE-2025-31133, CVE-2025-52565, CVE-2025-52881). These allow complete container breakout to the host. CVE-2025-52881 bypasses AppArmor and SELinux protections entirely.

Verify your runc version is 1.2.8 or later before deploying any containers. Check with: docker info | grep runc

Container Hardening Rules

- ☐ **read_only: true** on every container. Prevents attackers from writing malware to the container filesystem.
- ☐ **cap_drop: ALL** then add back only what each service needs (typically just NET_BIND_SERVICE).
- ☐ **Never use privileged: true.** This gives containers full host access and makes all runc CVEs trivially exploitable.
- ☐ **Use tmpfs mounts** for /tmp and /var/tmp so temporary data stays in memory and never hits disk.
- ☐ **Set log limits** (max-size: 10m, max-file: 3) to prevent log-based disk exhaustion attacks.

3.3 Network Segmentation

Create two Docker networks. Containers that process sensitive data should never have a route to the public internet.

Network	Internet	Containers	Purpose

	Access		
external	Yes	n8n, Uptime Kuma, Tailscale, Ollama (initial model pull only)	Automation, monitoring, remote access
internal_only	No (internal: true)	Metabase, PostgreSQL	Data processing, dashboards, database

This segmentation means that even if n8n gets compromised through its known RCE vulnerability (CVE-2025-68613), the attacker cannot reach your database or dashboards.

4. Container Image Supply Chain

Docker Hub has hosted millions of malicious and typosquatted images. The XZ Utils backdoor (CVE-2024-3094, CVSS 10.0) propagated silently through Docker images in 2024. Supply chain security is not optional.

4.1 Verified Image Sources

Only pull from these exact image names. Any variation (extra characters, underscores, different publishers) is a potential typosquat.

Tool	Image Name	Publisher Status	Data Exposure
n8n	n8nio/n8n	Verified Publisher	Workflow data, API keys, credentials
Portainer	portainer/portainer-ce	Verified Publisher	Docker management, container access
Metabase	metabase/metabase	Verified Publisher	Database queries, dashboard data
Ollama	ollama/ollama	Docker Sponsored OSS	Model files, prompts, completions
Uptime Kuma	louislam/uptime-kuma	Community (single dev)	Monitoring URLs only (low risk)
PostgreSQL	postgres (Official)	Docker Official Image	All persistent data

Uptime Kuma is community-maintained by a single developer. It only monitors URLs (low data sensitivity), so the risk is acceptable. Do NOT use community images for any component that touches education data.

4.2 Pin Image Digests

Never use :latest tags. Tags are mutable and can be replaced with malicious images. Pin every image to its SHA256 digest, which is immutable.

After pulling each image, run **`docker images --digests [image-name]`** and record the SHA256 hash in your docker-compose.yml.

4.3 Vulnerability Scanning

Scan every image with Trivy before deployment and monthly thereafter. Trivy is open source, runs locally, and does not send data to external servers.

1. Install Trivy in WSL2: sudo apt install trivy
2. Scan before first deploy: trivy image --severity HIGH,CRITICAL n8nio/n8n@sha256:...
3. Generate Software Bill of Materials for audit: trivy image --format cyclonedx [image] > sbom.json
4. Schedule monthly scans via n8n or cron to catch new CVEs in existing images

5. Tool-by-Tool Security Configuration

Every tool in the stack has documented critical or high-severity vulnerabilities. The defaults are not production-safe. This section covers the exact settings to change for each.

5.1 Ollama

CRITICAL: Ollama has zero authentication by default. Over 175,000 Ollama instances are exposed globally. Anyone who can reach port 11434 can pull/push models, generate completions, and delete files.

- ☐ **Bind to localhost only:** Set OLLAMA_HOST=127.0.0.1 in your container environment. Never bind to 0.0.0.0.
- ☐ **No direct internet exposure:** Access Ollama only through other containers on the internal network, or through Tailscale for remote use.
- ☐ **Disable model pull after setup:** Once your models are downloaded, restrict outbound access from the Ollama container.
- ☐ **Monitor model directory:** Check /root/.ollama/models periodically for unexpected files.

5.2 Metabase

CRITICAL: Metabase stores database credentials in plain text by default. It also had a pre-authentication remote code execution vulnerability (CVE-2023-38646, CVSS 9.8).

- ☐ **Set MB_ENCRYPTION_SECRET_KEY:** This encrypts stored credentials. Without it, anyone with filesystem access can read your database passwords in plain text.
- ☐ **Run on internal_only network:** No internet access. Metabase does not need to reach the outside world.
- ☐ **Enable audit logging:** Set MB_AUDIT_ENABLED=true. Required for FERPA compliance.
- ☐ **Disable database query logging:** Set MB_DB_LOGGING=false to prevent student data from appearing in logs.
- ☐ **Update immediately on release:** Metabase has a history of critical pre-auth vulnerabilities. Patch windows should be measured in hours, not weeks.

5.3 n8n

WARNING: n8n had a CVSS 9.9 RCE via expression injection (CVE-2025-68613). Webhooks are open to anyone by default.

- ☐ **Set N8N_ENCRYPTION_KEY:** Encrypts stored credentials and sensitive workflow data.
- ☐ **Disable public webhook access:** Require authentication on all webhooks, or restrict access via Tailscale.
- ☐ **Set N8N_SECURE_COOKIE=true:** Prevents session hijacking via cookie theft.
- ☐ **Run as non-root:** The n8n container supports running as the node user. Configure this explicitly.

- ☐ **Review community nodes carefully:** Third-party n8n nodes are a supply chain risk. Only install nodes from verified sources.

5.4 Portainer

WARNING: Exposed Portainer instances are actively targeted by the perfctl malware campaign for SSH persistence. Portainer also uses weak encryption (zero initialization vector).

- ☐ **Never expose to the public internet:** Access Portainer exclusively through Tailscale.
- ☐ **Mount Docker socket read-only:** /var/run/docker.sock:/var/run/docker.sock:ro
- ☐ **Set a strong admin password on first launch:** Do not use the default. Use 20+ character passwords.
- ☐ **Enable 2FA:** Portainer supports TOTP-based two-factor authentication. Enable it.
- ☐ **Limit to local environment only:** Do not connect Portainer to remote Docker hosts or Kubernetes clusters.

5.5 Uptime Kuma

WARNING: Local file inclusion via file:// protocol (CVE-2024-56331). Session tokens survive password changes.

- ☐ **Only use http:// and https:// monitors:** Never use file:// protocol in monitor URLs.
- ☐ **Rotate session tokens after any password change:** Manually clear all sessions after updating credentials.
- ☐ **Access through Tailscale only:** Do not expose the Uptime Kuma dashboard to the public internet.

5.6 PostgreSQL

- ☐ **Run on internal_only network:** The database must never be reachable from the internet.
- ☐ **Use strong, unique passwords:** Managed via Docker secrets, not environment variables.
- ☐ **Enable SSL for connections:** Even on the internal network, encrypt database traffic.
- ☐ **Set max_connections conservatively:** Limit to what your applications actually need.
- ☐ **Enable pg_audit extension:** Log all data access for FERPA compliance.

6. Remote Access

6.1 Tailscale (Primary)

Tailscale creates a WireGuard-encrypted overlay network between your devices. Encryption keys never leave your devices. Traffic is end-to-end encrypted, meaning Tailscale's coordination servers can see your network topology but never your traffic content.

> For K-12 data, Tailscale offers a much cleaner FERPA compliance posture than Cloudflare Tunnel because no third party can decrypt your traffic in transit.

1. Install Tailscale on ClawdBot and on every device that needs access
2. Enable Tailscale ACLs to restrict which devices can reach which services
3. Enable MagicDNS for easy access (e.g., clawdbot.tailnet instead of IP addresses)
4. Enable key expiry (90 days) so compromised device keys automatically rotate
5. Consider Headscale (self-hosted Tailscale coordination) if you want zero third-party visibility

6.2 Cloudflare Tunnel (Public Web Only)

> WARNING: Cloudflare Tunnel terminates your TLS. Cloudflare can read all traffic content in transit. Do NOT route any FERPA-regulated data through Cloudflare Tunnel. Threat actors are also actively abusing Cloudflare Tunnels to deliver malware (documented by Proofpoint, ESET, 2024-2025).

Acceptable use: exposing public-facing websites (Adult AI Academy, WasatchVille) that do not handle student data. Not acceptable for Metabase dashboards, n8n admin, or anything touching district information.

6.3 What to Never Do

- Never expose SSH or RDP directly to the internet, even on non-standard ports
- Never use port forwarding on your office router for any ClawdBot service
- Never store Tailscale auth keys in your docker-compose file or git repository

7. Secrets Management

Handling credentials correctly is the difference between a secure server and a data breach. This section covers how to store and manage every password, API key, and encryption key on ClawdBot.

7.1 Never Use Environment Variables for Secrets

> Environment variables are visible via docker inspect, /proc/[pid]/environ, and crash dumps. Any container escape immediately exposes every secret stored this way.

Instead, use Docker secrets (file-based) or a dedicated secrets manager.

7.2 Docker Secrets (File-Based)

Create a /run/secrets/ directory on the host. Store each secret in its own file with 600 permissions (owner read/write only). Reference them in docker-compose.yml via the secrets: directive. Applications read the secret from the mounted file at runtime.

- One secret per file: db_password.txt, n8n_encryption_key.txt, etc.
- File permissions: chmod 600 on every secret file
- Never commit secrets to git. Add the secrets directory to .gitignore.
- Rotate all secrets every 90 days

7.3 Secrets Inventory

Track every secret ClawdBot uses. If you cannot account for a secret, it is a risk.

Secret	Used By	Rotation	Storage
PostgreSQL password	Metabase, n8n	90 days	Docker secret file
MB_ENCRYPTION_SECRET_KEY	Metabase	Annual	Docker secret file
N8N_ENCRYPTION_KEY	n8n	Annual	Docker secret file
Portainer admin password	Portainer	90 days	Set on first launch
Tailscale auth key	Tailscale	90 days (auto-expire)	Docker secret file
Backup encryption key	Backup scripts	Annual	Offline (printed, locked)

8. FERPA Compliance Architecture

Self-hosting gives WasatchWise complete control over education data, but that control comes with complete responsibility. This section maps ClawdBot's architecture to FERPA requirements.

8.1 FERPA Requirements Mapping

FERPA Requirement	ClawdBot Implementation
Data Control: Only authorized users access student records	Tailscale ACLs + per-service authentication + no public internet exposure for data services
Encryption at Rest: Educational records protected on disk	BitLocker (SSD) + encrypted volumes for container data
Encryption in Transit: Data protected during transmission	WireGuard (Tailscale) end-to-end encryption + PostgreSQL SSL
Audit Trails: All access logged and reviewable	Metabase audit logging + pg_audit + centralized log collection
Data Residency: Data stays within your control	All data stored locally on ClawdBot. No SaaS dependency for data storage.
Reasonable Security: Technical and administrative safeguards	Container hardening + network segmentation + secrets management + regular patching

8.2 What ClawdBot Should Never Process

Even with these safeguards, some data categories should not live on a repurposed office desktop:

- Bulk student PII exports (Social Security numbers, medical records). These belong in certified systems.
- Active directory or authentication databases. Use proper identity providers.
- Financial transaction records (payment data, billing). Use PCI-compliant services.

> ClawdBot is appropriate for: aggregated compliance metrics, anonymized or de-identified analytics, workflow automation that references but does not store student records, and AI-assisted content generation for training materials.

9. Backup and Recovery

9.1 What to Back Up

Component	Data Location	Frequency	Method
PostgreSQL	Named volume: postgres_data	Daily	pg_dump + encrypted archive
n8n workflows	Named volume: n8n_data	Daily	Volume snapshot + encrypted archive
Metabase config	Named volume: metabase_data	Weekly	Volume snapshot + encrypted archive
Docker Compose files	/home/user/clawdbot/	On change	Git (no secrets in repo)
Secrets files	/run/secrets/	On change	Encrypted USB + printed offline copy

9.2 Backup Encryption

Every backup must be encrypted before it leaves ClawdBot. Use AES-256 encryption. Store the backup encryption key separately from the backups (printed copy in a locked location, not on the same machine).

9.3 Test Restores

A backup you have never tested is not a backup. Schedule a quarterly test restore: spin up a fresh WSL2 instance, load the backup, verify data integrity. Document results.

10. Ongoing Maintenance Schedule

Cadence	Task	Tool
Weekly	Check Windows Update status and pending reboots	Windows Settings
Weekly	Review Uptime Kuma for service health trends	Uptime Kuma dashboard
Monthly	Run Trivy scans on all container images	trivy image --severity HIGH,CRITICAL
Monthly	Check for new versions of all containers	Docker Hub / GitHub releases
Monthly	Review Portainer for unexpected containers or changes	Portainer dashboard
Monthly	Verify backup integrity (spot check)	Restore test on single volume
Quarterly	Rotate secrets (database passwords, auth keys)	Secrets inventory (Section 7.3)
Quarterly	Full restore test from backup	Fresh WSL2 instance
Quarterly	Review Tailscale ACLs and connected devices	Tailscale admin console
Quarterly	Update this playbook with new findings	This document
Annually	Rotate encryption keys (Metabase, n8n, backup)	Secrets inventory
Annually	Review hardware health (SSD wear, fan noise, thermals)	Physical inspection + CrystalDiskInfo

11. Quick Reference: The Rules

Print this page and tape it next to ClawdBot.

Never Do

- Never expose the Docker daemon to the network
- Never run a container with privileged: true
- Never put secrets in environment variables, Dockerfiles, or git
- Never use :latest tags. Pin image digests.
- Never expose SSH, RDP, or any service directly to the internet
- Never route FERPA data through Cloudflare Tunnel
- Never skip a Trivy scan before deploying a new image
- Never store bulk student PII on ClawdBot

Always Do

- Always use read_only: true and cap_drop: ALL on containers
- Always encrypt data at rest (BitLocker + encrypted volumes)
- Always access ClawdBot remotely through Tailscale only
- Always set encryption keys for Metabase and n8n before first use
- Always scan images with Trivy before deployment
- Always encrypt backups before they leave the machine
- Always test restores quarterly
- Always update runc/Docker when security patches are released