Erik Horváth

Nitra · hybrid or remote · open to full-time, contract or B2B
Start: immediately or by agreement

DevOps Engineer – Docker and CI/CD at work, eBPF and KVM introspection in my thesis.

erik@erikhorvath.com · +421 948 177 942 · github.com/ErikHorvath-git · erikhorvath.com

Profile

I build deployment pipelines and environments (Docker, CI/CD), debug problems down to the Linux kernel and am heading towards security long term. My thesis rebuilds a live KVM VM's state from snapshots of its memory.

Experience

DevOps Engineer · Marketeam, s. r. o., Nitra — 02/2026 – present

- Migrated the company's Symfony app from **3.4** to **6.4** (three major versions): APIs, dependencies, config.
- Built a staging environment; currently setting up deployment infrastructure (Docker, CI/CD).
- Cut the Pretix container's memory use from **6.7 GB to ~0.9 GB** (measured at work) by capping Celery worker concurrency with a one-line supervisord change (`--concurrency 2`).

AI Engineer Intern · HOFITECH s. r. o. · Details under NDA — 06/2025 – 01/2026
Frontend Developer (Angular) · freelance — 02/2024 – 09/2024
Programming Instructor for Children · EDU PLAY s. r. o. — 02/2024 – 06/2024

Education

Master's programme, Applied Informatics (in progress) · UKF Nitra — 2025 – present
Bachelor's (Bc.), Applied Informatics · UKF Nitra — 2023 – 2025
Software Engineering (not completed) · Tomas Bata University in Zlín — 2022 – 2023

Projects

`01` **Master's thesis (topic): introspection of a running KVM VM, no in-guest agent during capture** — 2026 (in progress)
private repo `hyptcn3` · AI-assisted · `C` · `eBPF (libbpf)` · `Python` · `KVM` · `x86-64`

- Reconstructed the VM's processes, kernel modules and listening sockets from memory snapshots read on the host (with a per-boot kernel profile taken from the VM beforehand); checked against `ps`/`lsmod`/`ss` inside the VM in one validation session: **81/81** processes stable during the snapshot (0 missing, 0 false positives), **51/51** kernel modules, **10/10** listening sockets (+2 extra, explained).
- Captured the VM's full **2.02 GiB** memory in **~0.39 s** without pausing it (median of 3 runs, checksum off).
- With SHA-256 on, the snapshot cycle went from 13.4 s to 2.5 s (**5.4×**) through a streaming hash with a SHA-NI path (an AI agent wrote this change under my direction); cost: a longer read window, 388 → 2,288 ms.
- Adapted and fixed an existing eBPF collector: resolved the **4** issues that stopped it loading on kernel 7.1 – a kfunc BTF mismatch (fixed with an opaque struct); the other three (verifier complexity, two 1M-instruction limits) with `bpf_loop()`. Custom x86-64 page-table walk (incl. 2 MiB/1 GiB pages) and KASLR shift detection.
- Retracted my own earlier (v2) detection results after finding a labelling flaw (session-level label confound); added `check_claims.sh`, a script that flags retracted numbers and checks source tags against the result JSONs. The TCN model is implemented and tested but deliberately untrained – no accuracy claimed.

`02` **sovereign-ai-poc – MLOps PoC with a signed software supply chain** — 06/2026
PoC · code written by Claude under my direction (8/8 commits) · `Docker Compose` · `Forgejo` · `MinIO` · `MLflow` · `Zot`

- A container image passes only with no CRITICAL vulnerabilities (Trivy), a signature by digest (cosign) and an attested SPDX SBOM (software bill of materials); negative tests verify that unsigned or vulnerable images are rejected. A cgroup-scoped eBPF monitor (bpftrace) distinguishes normal traffic from a deliberate exfiltration test.

`03` **scanforeshop – tracking scripts added to e-shop pages (Magecart-style card-skimming risk)** — 01/2026
solo prototype, AI-assisted (Codex) · ~7.6k lines of Python · `Django` · `DRF` · `Playwright` · `SQLite`

- Opens an e-shop in an automated browser (Playwright) and, before the page's own JavaScript runs, starts recording external scripts inserted at runtime (`appendChild`/`insertBefore`), apart from those in the server HTML; identifies each script by its SHA-256 hash and compares snapshots to produce alerts and a report.

`04` **Internship management system – backend and custom OAuth 2.0 server** — 10/2025 – 01/2026
team project, 3 people · public repo `Bally17/SIprojekt` · `Django` · `DRF` · `PostgreSQL` · `Redis` · `Nginx`

- Wrote **98.75%** of the backend lines by git blame, excl. migrations (96.4% with move/copy detection); teammates built the frontend and SQL schema.
- Implemented the OAuth 2.0 authorization (login) server without an OAuth server library: **4** grant types, PKCE S256, signed-JWT client authentication (`private_key_jwt`, RFC 7523).
- **11**-step GitHub Actions release pipeline that builds backend and frontend images and pushes them to GHCR; signal-based Redis cache invalidation; refactored views into a service layer.

AI workflow

I write the spec with acceptance criteria, AI agents write much of the code; I read, test and, where I can, measure it.

Skills

Programming	Python, C, Bash, SQL, TypeScript, PHP/Symfony (at work), Go*
Linux & security	eBPF (libbpf: verifier, BTF; bpftrace), KVM memory introspection, x86-64 paging, KASLR
Backend	Django/DRF, FastAPI*, OAuth 2.0 / PKCE / RFC 7523, PostgreSQL, Redis, Celery*
DevOps	Docker/Compose, GitHub Actions, GHCR, Nginx, cosign*, SBOM*, Trivy*
ML	PyTorch, scikit-learn* (* mainly through AI-assisted projects)
Languages & other	Slovak – native · English – B2 · category B driving licence · interests: boxing, football