Justin Hofmann

Platform & Backend Engineer — Java/Quarkus · OpenResty · Docker · Vue/TypeScript

Brandenburg, Germany (remote-first) · hofmannjustinbenedikt@gmail.com · justinhofmann.dev · github.com/Nexinox

PROFILE

Six years building and operating an on-prem microservice platform that makes city bollards smart — access via RFID and phone calls, time-boxed permissions, full audit history — deployed in around ten cities. I built the platform's API gateway, authentication, and deployment layer during its migration from a legacy monolith; more recently I designed and shipped its backup/restore and in-place update systems solo, and review cross-cutting architectural changes across its 15+ services. My strength is orchestration and integration on high-blast-radius systems: coordinated operations across stateful systems without shared transactions, rollback design, and release engineering.

EXPERIENCE

GEMTEC GmbH — Fachinformatiker für Anwendungsentwicklung

08/2021 – present

Full-time; apprenticeship at the same company 09/2018 – 08/2021

On-prem smart-city platform: 15+ Quarkus microservices, 10 Vue frontends, Postgres, MQTT, Graylog/OpenSearch, Proxmox. 1,193 commits across 45 repositories. Scope grew from feature work to platform ownership — organized below by systems owned rather than by year.

Platform infrastructure & migration (2021–2023)

- Built the platform's **JWT authentication** and **Consul service discovery** during the migration from a modular OSGi/Vaadin platform to containerized microservices — starting as an apprentice.
- **Principal author of the OpenResty API gateway** from its first commit to today (49% of all commits over 5 years). Chose OpenResty over HAProxy myself; implemented JWT verification in Lua, per-service routing, and later full CRUD services inside the gateway, plus operational modes (restore mode, login-less status page).
- **Sole author (108/108 commits) of the platform's Nomad/Consul deployment layer**: per-service job specs, service dependency ordering, memory oversubscription, log management, installer scripts. Two years later I argued for **replacing my own layer with Docker Compose** — I was its only maintainer, and a stack only I could run was a liability — and drove that migration; the Compose deployment is still in production.
- Containerized other teams' services and wired integration-test failures into the CI gate.

Backup & restore with cross-version migration (2025–2026, solo)

- Designed and built a platform-wide backup/restore system: a **saga orchestrator** with 14 sequenced restore steps and reverse-order rollback, coordinating six stateful systems (Postgres, Graylog/MongoDB, OpenSearch, Node-RED, gateway config) that share no transactions. Restores onto different machines and **newer product versions**, migrating data between changed backend schemas. In production use with customers since 11/2025.
- Safety design: restore into throwaway temp databases then promote; format-versioned archives with an auto-upgrading descriptor migrator; an availability gate that locks the feature out entirely when preconditions fail.
- Rewrote the working, shipped v2 onto the saga architecture (~13k lines removed, ~16k added over three weeks) before its complexity became unmaintainable.
- Replaced read-then-write with **streaming I/O** throughout, to stop large restores exhausting memory — which also made them **50–100× faster** and materially more reliable.

In-place product updater (2026, solo)

- Designed the system on an architecture board in 2024 (deployment topologies, update protocol, risk pre-mortem); built it from scratch in two months in 2026.
- **Blue-green deployment on Proxmox** (in progress): two VMs on shared NFS storage with only one ever powered on, so an installation keeps a single IP and MAC and needs no load balancer in front of it. Built on Proxmox snapshot/restore and on the backup/restore system above; the hard part is moving installations between Proxmox hosts with differing names and missing ISOs.
- A **GraalVM-native Quarkus service** that updates the entire on-prem stack including Proxmox VMs — and itself, via a versioned "baton" handoff to its successor. Resumable (tus) and online package intake, SSE progress, readiness gating before commit, rollback semantics, disk-space guards. Paired with a standalone login-less status frontend served while the main UI and auth are deliberately down.
- Separately made the platform's last JVM-only backend (Asterisk telephony driver) build as a **native image**: diagnosed a library negotiating its protocol version via runtime reflection over ~6,000 generated classes, made the version configurable, generated 1,216 reflection registrations. Boot ~50 ms; memory ~100 MB vs ~350 MB on the JVM.

Estate-wide code review & release engineering (2026)

- **Reviewer of record for cross-cutting changes: 63 merge requests across 18 repositories** — e.g. one architectural pattern reviewed across 8 backends in a single day. Findings included release-blocking API contract gaps across frontends, duplicate DB migration versions, and a global mutex held across network calls.
- **Reverse-engineered and documented the previously undocumented manual release process** spanning 18 repositories (dependency ordering, versioning, registry promotion) and mapped its automation opportunities.

iLEM — the company's first cloud deployment (AWS, ~2022, 6 months)

- Owned the entire deployment of a greenfield smart-energy product (solar, EV charging, storage management): containerized services on **AWS ECS with ECR and RDS**, IAM roles, network isolation — built with no prior in-house cloud experience; advised on the application code in parallel. Set up the customer-showcase environment that still runs. The product was shelved for lack of market demand.

Product & device-integration work (2022–2026)

- **Led the estate-wide Vue 2 → Vue 3 migration** (Webpack → Vite, state management to Pinia) across the application frontends and the shared library chain — re-engineering the frontend build stack to make it possible (2025). With the frontend team, then decided to drop the micro-frontend architecture; the module-based consolidation is ~60% done.
- Full-stack features across 10 Vue frontends and ~10 backends; owner of the shared frontend library chain every application depends on.
- Device integration for physical infrastructure: MQTT messaging, IO state handling, heartbeats and driver abstractions for bollards; adjacent systems include access control, fire alarm panels, nurse call, VoIP/telephony, and PLCs.
- Architecture and interaction design in Figma: a complete next-generation platform architecture (service catalogue, MQTT protocol design with payloads, UML flows — five of the six services it proposed were subsequently built), and a 33-screen UI redesign specified on Material Design 3.
- **Mentored apprentices and working students continuously since 2021**, self-initiated: system onboarding, joint code walkthroughs explaining what's good or bad and why, and architecture design sessions run together with them.

SKILLS

Backend: Java (6 y; Quarkus — Jakarta EE/MicroProfile, directly transferable to Spring Boot, currently porting a service to Spring Boot as a personal project), REST API design & versioning, saga/orchestration patterns, GraalVM native images, SSE, resumable uploads (tus), JWT auth

Infrastructure: Docker & Compose, NGINX/OpenResty + Lua, GitLab CI, HashiCorp Nomad & Consul (scheduling/service-discovery concepts transfer to Kubernetes), AWS (ECS, ECR, RDS, IAM — one full product deployment), release engineering, Proxmox, Postgres operations, Graylog/OpenSearch

Frontend: Vue 2/3 + Vuetify, TypeScript, Pinia (5 y, 10 applications)

Protocols & devices: MQTT (design and implementation), binary device protocols, Node-RED, Asterisk/ARI

Also: Go (two full service reimplementations, unshipped), Python, Bash, SQL/Flyway, Vaadin 7/8 (apprenticeship-era Java UIs; plus a FIDO2/WebAuthn login project on GitHub)

Languages: German (native), English (fluent — daily written use)

PERSONAL PROJECTS

- **centurio-mqtt (Go):** bridge from a home battery/inverter system to MQTT with Home Assistant auto-discovery — hand-built binary protocol layer with unit tests, island-mode control, storage-bottleneck detection.
- **Home automation from firmware up:** custom Python firmware for smart power switches (MQTT, relay control, web UI), solar-data collection, local LLM/RAG experiments.
- **Learning by reimplementation:** rebuilt a production user-management service in Go (complete: JWT access/refresh auth, roles/permissions, migrations, documented API); currently rebuilding it in Spring Boot.

EDUCATION

- **Ausbildung Fachinformatiker für Anwendungsentwicklung (IHK)** — German vocational qualification in software development, GEMTEC GmbH, 09/2018 – 08/2021
- Secondary school certificate (10th grade), 2018