Arthur Locke

Prague, Czech Republic | arthur@locke.cz | +420 608 541 742
linkedin.com/in/arthur-locke | github.com/A-Locke

Available Immediately

Summary

DevOps and Cloud Engineer with an 8+ year QA background bringing automation discipline, reliability rigour, and process thinking to infrastructure work. Hands-on, project-based expertise across Microsoft Azure (Bicep, Azure OpenAI, Entra ID, Copilot Studio, Power Platform), Oracle Cloud Infrastructure (OCI), AWS, Kubernetes, Terraform, and GitHub Actions CI/CD pipelines. Built production-grade infrastructure automation including a Bicep-defined Azure platform with dual CI/CD pipelines, one-button Kubernetes deployments, Terraform-managed multi-service clusters, and WireGuard VPN networking on Oracle Cloud Free Tier. Experienced in building MCP-connected agentic systems and driving AI tooling adoption across engineering teams.

Professional Experience

Freelance – DevOps & Automation Engineer Prague, Czech Republic
2023 – Present

- Deployed and managed production Kubernetes workloads on Oracle Cloud Infrastructure for multiple clients: n8n (queue mode), PostgreSQL, pgAdmin, Prometheus + Grafana, Cert-Manager, Ingress-Nginx – full cluster lifecycle ownership.
- Infrastructure as Code: Terraform (HCL) for OCI provisioning; Ansible for WireGuard VPN, dnsmasq, and Cloudflare DNS (DNS01 certificate challenge).
- Built RAG chatbot (OpenAI, Gemini, Supabase vector DB) connected to WhatsApp and Telegram via webhook integrations.
- Developed accounting workflow automation: WordPress web form → n8n webhook → data processing → Czech government-regulated XML/PDF; XML schema validation for regulatory compliance.
- Engineered automated data extraction and processing pipelines for multiple clients (NDA).
- Developed full-stack websites using Next.js and PayloadCMS with API-first headless CMS architecture.

Advanced Quality Assurance Specialist | Junior Release Manager Keen Software House a.s., Prague
Aug 2020 – Jul 2026

AI Tooling & MCP Integration

- Architected *SE Tester AI Agent System* – three-agent pipeline (Planner, Generator, Healer) producing and self-healing XML test scripts for the internal Space Engineers test tool; MCP server provides real-time game control for playbook execution.
- Connected internal n8n instance to Claude via MCP server – team can observe, diagnose, and create automation workflows via natural language, no technical knowledge required.
- Built and deployed the *UseResponse MCP* server – dual transport (stdio locally, streamable-http on Google Cloud Run for scheduled Cloud Routines), OAuth 2.0/PKCE with signed HS256 JWTs, seven independently-gated tool categories (deny-by-default), giving Claude access to Keen's multi-platform support portal for QA bug triage.
- Pioneered AI adoption across the engineering team: led GWS + Claude rollout, defined QA use cases for AI-assisted document and presentation generation, delivered onboarding guides and hands-on mentoring.
- Built *Clawdbot* prototype – AI-driven vision-based regression agent running on a rented RTX 5090 / Windows 11 machine (OpenClaw, mss, Pillow, OpenCV, pyautogui; LM Studio local inference with GLM 4.7 Flash and Nemotron 3 Nano). Successfully navigated Space Engineers UI and executed headed regression tests autonomously. Superseded by the SE Tester AI Agent System.

DevOps, Automation & Observability

- Designed and deployed n8n automation workflows for spam filtering and weekly report generation (API-driven graphics), saving ∼5 man-hours weekly; integrated external REST APIs into QA and reporting pipelines.
- Operated multi-platform log ingestion system (ELK stack: Elasticsearch, Logstash, Kibana) aggregating telemetry from Xbox, PlayStation, and Steam for centralised QA analysis and anomaly detection.
- Led adoption of SE Tester (XML-driven test execution tool): diagnosed failures at source-code level, identified engine changes required for test cases, managed the Jira defect backlog for the tool.
- Collaborated on DevOps initiatives, expanding into cloud deployments and CI/CD automation pipelines.

QA Engineer Charged Monkey, Prague
Oct 2017 – Aug 2020

- Sole QA for most of the tenure, owning end-to-end test strategy, automation pipeline, and release management across iOS, Android, and Amazon platforms; briefly led a small, two-person QA team after a second QA hire joined.
- Established QA practices from the ground up; mentored new team members.

Key Projects

Enterprise Multi-Agent AI Platform – Microsoft-Native Multi-Agent AI Platform (Azure, Copilot Studio, Power Platform)
github.com/A-Locke/enterprise-multi-agent-platform

- Built and deployed an enterprise-scale platform across the full Microsoft stack, evaluating a Microsoft-native service (Copilot Studio, Power Platform, Azure OpenAI) against custom pro-code for every architectural capability before defaulting to custom code – 15 documented ADRs recording the trade-off analysis.
- **Two independent, platform-native CI/CD pipelines** in GitHub Actions, each verified end-to-end via live deployments: one provisions and redeploys the entire Azure side from Bicep (Key Vault, API Management, Container Apps, Azure OpenAI, AI Search, Content Safety) via OIDC federated credentials with no stored secret; the other promotes both Dataverse business data and a live multi-agent Copilot Studio deployment between Power Platform environments via a dedicated Dataverse Application User.
- The one deliberate pro-code layer: a Semantic Kernel API behind Azure API Management, Entra ID-authenticated with role-based authorization enforced end-to-end, calling Azure OpenAI and a dedicated Azure AI Content Safety moderation layer via managed identity – zero API keys anywhere in the platform.
- Root-caused and fixed real, non-obvious issues via live server responses and direct API queries rather than guesswork: a GitHub OIDC subject-claim format change, several `azd`-in-CI-specific environment-state gaps, a tenant Security Defaults policy silently blocking an OAuth login flow (fixed by switching to authorization-code + PKCE), and two genuine Dataverse/Copilot Studio ALM limitations confirmed against Microsoft's own documented known issues.
- Cost discipline: actual Azure spend through full completion was $0.39 against a $180/month guardrail – every architectural choice evaluated against its free tier first, backed by a dedicated cost analysis and a directional production-scale cost estimate.
- Stack: Azure (OpenAI, AI Search, Content Safety, API Management, Container Apps, Key Vault, Monitor), Bicep, Azure Developer CLI, Semantic Kernel, Microsoft Entra ID, Copilot Studio, Power Platform (Dataverse, Power Apps), GitHub Actions.

AI QA Automation Platform – Multi-Service MCP Platform (TestRail ↔ Unreal Engine)
github.com/A-Locke/unreal-testrail-qa-platform

- Designed and built a multi-package platform (Python, uv workspace) with clean service boundaries

– a shared core library, two independent MCP servers, an execution engine, and a reporting service – each independently testable and deployable, integrating only through data (a versioned YAML spec, a SQLite database) rather than direct imports.
- Config-driven, not hardcoded, on every axis that matters for reuse: TestRail suite targeting, custom case-template field mapping, and even which Unreal Blueprint class is the player pawn are all settings, not source code – pointing the platform at a different studio's TestRail or Unreal project needs a YAML edit, not a redeploy.
- `testrail-mcp`: from-scratch MCP server (stdio + streamable-HTTP) with config-driven, per-project read/write/delete permission scoping – a project not explicitly listed is unreachable regardless of API credentials – and delete operations defaulting to a soft/preview mode; verified live against a real TestRail Cloud site including the write-rejected/write-allowed/soft-preview/real-delete round trip.
- Containerized the two standalone services (`testrail-mcp`, `qa-regression-dashboard`) with multi-stage Dockerfiles and Compose orchestration, verified live via `docker compose up`; the Unreal-facing services stay host-native since they drive a live desktop editor process rather than a container-friendly one. Live container runs caught real cross-platform bugs a passing test suite couldn't (Windows-style paths breaking inside the Linux container, a relative link that only worked over `file://`).
- Built end-to-end through 12 planned milestones, each shipped with working code, passing tests, and documentation (README per package, ARCHITECTURE, CHANGELOG, a 19-entry ADR decision log, an 11-step reproducible walkthrough) kept in sync as the platform grew – run for real end-to-end, not just unit-tested: 9 manual test cases evaluated, translated, and executed against a live Unreal Engine 5.8 target, with results reported to SQLite and rendered to a static HTML dashboard and PDF.
- Stack: Python, Pydantic, uv workspace, official `mcp` SDK, SQLite, Docker.

AI-Augmented Customer Support Platform – Self-Hosted Chatwoot + MCP + Claude Support Automation
github.com/A-Locke/ai-game-support-platform

- Designed and built a self-hosted AI support-ticket triage platform with strict architectural separation across independently swappable pieces – Chatwoot (unmodified), two self-hosted MCP servers, and Claude as a replaceable intelligence provider – documented across 6 ADRs recording the actual reasoning.
- Docker Compose orchestration across four services (FastAPI webhook pipeline, two MCP servers, Postgres with pgvector) verified live against a real running stack, not just a successful build; found and fixed 10 real cross-service bugs this way, including Chatwoot's SSRF protection silently blocking same-network webhook delivery and an asyncpg connection pool broken by spanning two different asyncio event loops.
- Scheduled Postgres-to-S3 backups (any S3-compatible provider) with a live-verified restore: backed up the real live database, restored it into a completely fresh Postgres instance, and confirmed row counts and message content matched exactly.
- Deployed on Google Cloud Run behind Caddy, with Redis for queueing; evaluated dedicated/managed vector databases and hosted embedding APIs before choosing a zero-infrastructure local-embedding approach (pgvector + fastembed/ONNX) to avoid a new vendor dependency.
- 92 automated tests (pytest, moto for S3 mocking) across four services, all passing; every feature verified live against a running Docker Compose stack.
- Stack: Python (FastAPI, fastmcp, boto3, asyncpg, fastembed), Chatwoot Community Edition, Anthropic Claude API, PostgreSQL (pgvector), Docker Compose, Redis, Caddy, Google Cloud Run.

AI Coding Agent Observatory – OpenTelemetry Observability Platform (Local Docker + AWS)
github.com/A-Locke/ai-coding-agent-observatory

- Designed and built a self-hosted observability platform ingesting real, native OpenTelemetry

telemetry (OTLP over gRPC/HTTP) from AI coding agents via a dedicated OpenTelemetry Collector, normalizing each vendor's distinct schema into one shared ingestion pipeline backed by SQLite – deliberately no synthetic data generator.
- Optional AWS cloud path: 5 Terraform modules (CloudWatch, least-privilege IAM, feature-flagged Lambda/DynamoDB/X-Ray) mirroring the same telemetry into CloudWatch via EMF metrics and log export. Deployed and destroyed for real against a live AWS account across two regions, verified each time with live AWS CLI calls rather than trusting Terraform's own state.
- GitHub Actions CI/CD: lint/typecheck/test/build gate on every push, Terraform fmt/validate/plan on infrastructure changes, and a manual-approval-gated deploy workflow – configured the GitHub Environment's required-reviewers protection rule via the REST API and verified the full dispatch → approval-pause → apply → destroy cycle end-to-end for real, not just written and assumed correct.
- Found and fixed real bugs through live verification rather than assuming correctness: a gzip-compressed OTLP ingest path silently failing, an IAM user needing `force_destroy` since its access key is issued out-of-band, and a Docker Compose `-env-file` resolution gap.
- Stack: Next.js 15, TypeScript, Node.js built-in SQLite, Docker Compose, OpenTelemetry Collector, Terraform, GitHub Actions, AWS CloudWatch/IAM/Lambda/DynamoDB/X-Ray.

TestRail MCP – MCP Server for TestRail Test Management (dual transport, Google Cloud Run)

- Built a from-scratch MCP server wrapping TestRail's REST API v2, mirroring the UseResponse MCP's dual-transport architecture: stdio locally, streamable-http on Google Cloud Run, deployed via `gcloud run deploy -source .`; same OAuth 2.0/PKCE + signed JWT security model.
- Two-tier tool-gating: five category flags (read, case/run/result writes, deletes) stacked with an optional named-tool allowlist that can only narrow, never widen, what a category flag permits – deny-by-default except reads.
- Always-on `list_tool_policy` introspection tool; automated test suite validates the registered tool list matches policy exactly across representative flag combinations.
- Stack: Python, FastMCP, httpx, Docker, uvicorn/Starlette.

kube-dojo – Kubernetes CKA Training Platform with AI Agent Solver
github.com/A-Locke/kube-dojo

- Designed and built a reproducible Kubernetes CKA exam preparation platform running on a local `kind` cluster with deterministic task verification and a local AI agent solver (no cloud API key required).
- Task system: each task injects a broken Kubernetes state (misconfigured Services, bad container images, missing RBAC, stuck PVCs, scheduling taints) into a real cluster; a deterministic `verify.sh` script confirms the fix.
- AI solver: local Claude Code agent explains each `kubectl` command before running and interprets output after; 5/5 tasks passed autonomously in a single suite run.
- `/generate-task` slash command instructs the agent to design, write, and live-test a complete new task (setup, verify, cleanup, prompt) from a one-line description.
- Killercoda integration: generator produces browser-ready scenario packages (index.json, foreground.sh, step markdown, verify delegation) published to companion repo `A-Locke/kube-dojo-scenarios` – zero local setup for end users.
- Stack: Bash, Python, PowerShell, kind (Kubernetes in Docker), kubectl, Claude Code.

OCI Infra Pipeline – Production Kubernetes on Oracle Cloud Free Tier
github.com/A-Locke/n8n_kubernetes

- Designed fully automated GitHub Actions CI/CD pipeline deploying production Kubernetes on OCI Always Free Tier – equivalent infrastructure costs $250+/month on AWS.
- Infrastructure as Code: Terraform (HCL) for OCI resource provisioning; Ansible for WireGuard VPN configuration, dnsmasq, and Cloudflare DNS (DNS01 cert challenge) – replaced Terraform null_resource with Ansible for idiomatic configuration management.

- Automated Helm deployments: Cert-Manager, Ingress-Nginx, PostgreSQL, pgAdmin, n8n v2 (queue mode), Valkey (Redis replacement with AOF persistence), Prometheus + Grafana (kube-prometheus-stack), metrics-server.
- n8n in queue mode: separates webhook listener from workers via Valkey for independent horizontal scaling.
- Kubernetes hardening: RBAC, NetworkPolicy, HPA, ResourceQuota, LimitRange, PodDisruptionBudget via dedicated manifests.
- Zero-trust ingress: WireGuard VPN as the sole access point to n8n, pgAdmin, and Grafana dashboards.
- Claude Code integration: 5 custom Kubernetes skills (`/k8s-status`, `/k8s-debug`, `/k8s-scale`, `/k8s-cost`, `/n8n-queue`) and MCP server for natural-language kubectl access.

AI-Augmented Playwright Test Automation Framework
github.com/A-Locke/ai-agentic-qa-saucedemo

- 5-phase AI-driven test pipeline: Planner → Generator → Debug → Healer; 39 Playwright TypeScript tests, all passing.
- Self-healing locator repair eliminates manual maintenance on UI changes.
- 10 CLI playbooks for repeatable, agent-executable test execution – CI-ready agentic automation.

SpecGuard – OpenAPI Contract Testing Platform with AI Drift Analysis
github.com/A-Locke/OpenAPI

- Designed and built a contract-first OpenAPI testing platform as a TypeScript pnpm monorepo (9 packages/apps): contract runner CLI, drift detection engine, AI drift analyzer, MCP server, mock server, and shared types.
- CI pipeline: type-check → unit tests → Docker multi-stage container build (`node:20-alpine`) → live contract validation – fails on violations, warns on upstream drift; 24/24 unit tests and 8/8 contract tests passing.
- AI layer: calls `claude-sonnet-4-6` with prompt caching (static spec pinned in cache, dynamic drift report injected per run) for semantic severity analysis and remediation actions.
- MCP server (4 tools): pure data providers exposing spec content, drift reports, and test inputs – host Claude performs all reasoning, keeping the server thin.
- Stack: TypeScript strict, Fastify, Zod (boundary parsing + NaN-safe coercion), AJV (OpenAPI schema validation), Vitest, GitHub Actions.

Professional One-Page Website Template
github.com/A-Locke/one-page-web-public

- Developed a Next.js 15 single-page template for professional and business use (TypeScript, Tailwind CSS v4, shadcn/ui, Credly certification badge integration), with environment-based builds configured for Meta Business Verification readiness.

Technical Skills

Cloud Platforms	Microsoft Azure (OpenAI, AI Search, Content Safety, API Management, Container Apps, Key Vault, Entra ID, Bicep, Azure Developer CLI), Power Platform (Dataverse, Power Apps, Copilot Studio), Oracle Cloud Infrastructure (OCI – Compute, OKE, IAM, Networking, Object Storage, Gen AI Services), AWS (EC2, S3, IAM, Lambda, WAF, ALB, Secrets Manager, CloudWatch, DynamoDB, X-Ray)
Container & IaC	Kubernetes (OKE), Docker, Helm, Terraform / OpenTofu (OCI & AWS), Ansible, Bash scripting
CI/CD	GitHub Actions (multi-environment pipelines, infrastructure deployment, container orchestration, automated test integration)
Networking & Security	WireGuard VPN, Cloudflare DNS (DNS01), Cert-Manager, Kubernetes RBAC & NetworkPolicy, secrets management; ISC2 Certified in Cybersecurity (CC)
AI & MCP	Multi-agent pipeline design (Planner/Generator/Healer), Claude Code, MCP server development (stdio & OAuth2/JWT-secured streamable-HTTP on Google Cloud Run), runtime capability discovery (Unreal Engine 5.8); integrated Claude with n8n, kubectl, UseResponse, TestRail, and internal tooling for natural-language system control
Workflow Automation	n8n (queue mode, multi-service orchestration, self-hosted on Kubernetes & Railway), Apify (data pipelines, schema validation)
Monitoring	OpenTelemetry (OTLP pipelines – traces, metrics, logs; custom Collector configuration; multi-vendor schema normalization; CloudWatch EMF export), Prometheus, Grafana (kube-prometheus-stack), ELK Stack (Elasticsearch, Logstash, Kibana), metrics-server
Development	Node.js, TypeScript, Next.js, REST APIs, PostgreSQL, Supabase, Git/GitHub

Education

Czech Technical University in Prague – Faculty of Electrical Engineering
Bachelor's Degree in Computer Science 2014 – 2017

Certifications

- Oracle Cloud Infrastructure 2025 Certified Architect Associate
- Oracle Cloud Infrastructure 2025 Certified Generative AI Professional
- Oracle Cloud Infrastructure 2025 Certified Foundations Associate
- Certified in Cybersecurity (CC) – ISC2
- ISTQB® Foundation Level
- API Test Automation

Languages

English – Full Professional Proficiency (C1) | **Czech** – Professional Working Proficiency