Arthur Locke

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

Available Immediately

Summary

AI Automation Engineer with an 8+ year QA background, now shipping production automation systems – from n8n pipelines to multi-agent LLM architectures. Treats prompt engineering as a production discipline: versioned, context-budgeted, and tested against real outputs before anything goes live. Has shipped automations across Claude, GPT, and Gemini and knows when to reach for which. At Keen Software House, led company-wide LLM adoption, built MCP servers that give non-technical staff natural-language control of live automation infrastructure, and architected a three-agent pipeline that generates and self-heals test scripts autonomously. Designs for failure by default: fallback logic, human escalation paths, and audit trails are part of the spec, not afterthoughts.

Professional Experience

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

AI Tooling & Agentic Systems

- Led company-wide AI adoption: evaluated tools, selected Claude over existing ChatGPT usage, defined the rollout plan for GWS + Claude integration, authored onboarding documentation, and delivered hands-on mentoring to engineers and non-technical staff.
- Connected the internal n8n instance to Claude via MCP server — team members can observe, diagnose, and create automation workflows through plain English with no technical knowledge required.
- Built and deployed the *UseResponse MCP* server — gives Claude direct access to Keen's multi-platform (PC, Xbox, PlayStation, SE2) support portal for Space Engineers bug triage. 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 and never registered when disabled. Shipped custom slash commands (`/triage`, `/submitted`, `/investigating`, `/nmi`, `/search`) plus a generic reusable agent profile.
- Architected the *SE Tester AI Agent System* — a three-agent pipeline (Planner interprets spec files and game source; Generator produces executable XML test scripts; Healer auto-diagnoses and repairs failures from MCP-provided logs). Owned all AI tooling layers; coordinated with programmers who owned the execution environment.
- Built *Clawdbot* prototype — AI-driven vision-based regression agent (OpenClaw, mss, OpenCV, pyautogui; LM Studio local inference) running on a rented RTX 5090 machine. Successfully navigated game UI and executed headed regression tests autonomously before being superseded by the SE Tester system.

Workflow Automation & API Integration

- Designed and shipped n8n automation workflows for spam filtering and automated weekly report generation (with API-driven graphics and formatting) — saving ∼5 man-hours every week with zero ongoing maintenance cost.
- Designed and maintained automated integration workflows covering REST API validation, external service integrations (weather data, QR code generation, report graphics), and data pipeline schema validation.
- Operated multi-platform log ingestion (ELK stack) aggregating telemetry from Xbox, PlayStation, and Steam for cross-platform QA analysis.

Freelance — AI, Automation & Development Prague, Czech Republic
2023 – Present

- **Article Generation Pipeline:** Multi-stage Python pipeline: RSS polling → deduplication against a processed-items database → database context search for prompt enrichment → Claude API with prompt caching (static context in cache, dynamic content injected per run) → Airtable human-in-the-loop review and approval → published markdown output.
- **RAG Chatbot:** Retrieval-augmented generation chatbot (OpenAI, Gemini, Supabase vector DB) with multi-channel delivery via WhatsApp and Telegram webhooks. Full lifecycle: integration design, prompt engineering, client deployment.
- **Tour City Interactive Chatbot:** Stateful n8n conversation logic with PostgreSQL-backed session state

for a tour company's city walking quest. QR code generation for frictionless entry; message routing and branching across separate user journeys. Requirements to live deployment.
- **Accounting Workflow Automation:** End-to-end tax return automation: WordPress form → n8n webhook → Python processing → Czech government-regulated XML and PDF with schema validation for regulatory compliance.

Selected 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 an enterprise multi-agent AI 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 reasoning, not just a stated principle.
- Multi-agent Copilot Studio design: a parent agent using the Connected Agents pattern routes each conversation to one of two specialized sub-agents — a Knowledge agent (RAG over an indexed corpus) and an Enterprise Integration agent (Teams/Outlook/SharePoint tooling) — verified live through real conversations, including catching and documenting a genuine platform-side OAuth consent bug in the Outlook connector across three separate investigation attempts.
- 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.
- 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 the Dataverse business data and the Copilot Studio agent itself between Power Platform environments.
- Root-caused and fixed real, non-obvious issues via live server responses and direct Dataverse API queries rather than guesswork: a GitHub OIDC subject-claim format change, 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 around promoting a bot's custom-connector connection across environments — confirmed against Microsoft's own documented "known issues," not assumed.
- Actual Azure spend through full completion: $0.39 against a $180/month guardrail — every architectural choice evaluated against its free tier first.
- 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-Augmented Customer Support Platform — Self-Hosted Chatwoot + MCP + Claude Support Automation
github.com/A-Locke/ai-game-support-platform

- Built a self-hosted AI support-ticket triage and classification platform with Claude treated as a genuinely replaceable intelligence provider, not a hardwired dependency: strict separation between Chatwoot (unmodified), two self-hosted MCP servers, and the LLM layer, documented across 6 ADRs recording the actual reasoning.
- Three interchangeable ways to run the identical classification logic: an automated FastAPI webhook pipeline calling the Claude API directly, an on-demand CLI reusing the same workflow code with no persistent server, and a fully API-key-free mode where a Claude Code session (or a scheduled routine) drives the MCP servers' tools directly — validated by manually classifying and triaging real seeded tickets through MCP before the automated pipeline was even built.
- Real semantic search, not a flat context dump: pgvector embedded locally via fastembed (ONNX runtime, zero API cost, zero new vendor) — evaluated against dedicated/managed vector databases and hosted embedding APIs before choosing the zero-infrastructure option, then verified live with real relevance scores (0.92 vs. 0.63, genuine match vs. unrelated).
- A permanent, compliance-driven architectural boundary rather than a v1 limitation: no code path anywhere in the system ever sends an AI-generated message to a customer directly, regardless of confidence score, real or self-reported — captured as its own ADR immediately after confirming a demo "known issue match" had been simulated, not computed, and following that honesty through to its actual design consequence.
- Grounded classification against real Jira and Azure DevOps backlogs via their official MCP servers — researched the actual tool names from each project's own source documentation before writing integration

code rather than guessing, and explicit in the code and docs about the one piece not live-verified (no real Jira/Azure DevOps credentials available), rather than presenting it as tested.
- 92 automated tests (pytest, moto for S3 mocking) across four services, all passing; every feature verified live against a running Docker Compose stack, including a live-verified Postgres-to-S3 backup/restore round trip.
- Stack: Python (FastAPI, fastmcp, boto3, asyncpg, fastembed), Chatwoot Community Edition, Anthropic Claude API, PostgreSQL (pgvector), Docker Compose, Redis, Caddy, Google Cloud Run.

AI QA Automation Platform — TestRail ↔ Unreal Engine Regression Pipeline
github.com/A-Locke/unreal-testrail-qa-platform

- Built a multi-package platform where Claude Code itself is the agent driving two custom MCP servers (TestRail, Unreal Engine 5.8) to migrate manual regression suites into AI-evaluated, executable tests — deliberately no server-side LLM API call in this pipeline; the interactive evaluate/generate steps are guided by versioned markdown instruction templates stored alongside the code they call.
- Deliberate scoping decision, not the default "wire an agent to both MCP servers and let it improvise": the LLM's judgment is confined to the one place it genuinely can't be hardcoded — deciding whether a manual case is automatable given what's actually discovered live. Everything downstream (permission enforcement, capability mapping, spec execution, reporting) is deterministic, tested, replayable code, not a prompt re-deriving the same decision differently every session.
- *unreal-mcp-adapter* discovers a running Unreal instance's MCP capabilities at runtime rather than assuming a fixed set, normalizing ~25 optional plugin toolsets onto one stable capability API — lets the AI evaluation step reason against what the engine can actually do right now, not an assumed capability list, and surfaced real engine quirks (an unwrapped response envelope, a "wait" call that only ever checks once) a live-improvising agent would otherwise rediscover inconsistently every run.
- AI evaluation step scores each manual test case's automatability (confidence 0–1) against live-discovered capabilities and records the specific unsupported capability on rejection — run for real against 9 seeded cases: 7 correctly automated, 2 correctly rejected for the same proven engine-input gap.
- Human-approval gate is a real enforced function call, not a documentation reminder — an AI-generated spec must receive an explicit "yes" before anything is written back into TestRail, with an `auto_approve` escape hatch for genuinely unattended runs.
- Shipped end-to-end, not just unit-tested: real execution against a live Unreal Engine target, results recorded to SQLite, rendered to a dashboard and PDF from one shared aggregation layer so the two outputs can't disagree.

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

- Built a self-hosted observability platform for AI coding agents themselves — ingests real, native OpenTelemetry telemetry from Claude Code, Codex CLI, and Gemini CLI (each vendor's actual documented metric/event/trace schema, deliberately no synthetic data generator), normalized into one shared per-agent-adapter ingestion pipeline.
- OpenTelemetry Collector receives OTLP over gRPC/HTTP into a Next.js 15 dashboard (sessions, cost, token usage, tool calls, per-session timeline); verified end-to-end with a real, live-connected Claude Code session flowing through ingestion into the UI — not a test payload.
- Optional AWS path: 5 Terraform modules (CloudWatch, least-privilege IAM, feature-flagged Lambda, DynamoDB, X-Ray) mirroring telemetry into CloudWatch via EMF metrics — 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 gate on every push plus a manual-approval-gated deploy workflow for the AWS path — configured the required-reviewers rule via the GitHub REST API and verified the full dispatch → approval-pause → apply → destroy cycle end-to-end for real.
- Found and fixed real bugs through live verification: a gzip-compressed OTLP ingest path silently failing, an IAM user needing `force_destroy`, and missing model-attribute extraction on two of the three vendor ingestion adapters.

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

- Built a 9-package TypeScript monorepo: contract runner CLI, structural drift detection engine, AI drift analyzer, and 4-tool MCP server.
- AI layer: `claude-sonnet-4-6` with prompt caching — static spec pinned in the cache, dynamic drift report injected per run — produces semantic severity analysis and remediation actions. Prompt caching chosen

deliberately to minimise token cost on repeated runs.
- AI test generation (`-ai-test-gen`): Claude generates edge-case `TestCase` objects from the internal spec at runtime and merges with spec-driven tests. Full CI: 24/24 unit tests and 8/8 contract tests passing.

TestRail MCP — MCP Server for TestRail Test Management

- Built a from-scratch MCP server wrapping TestRail's REST API v2, mirroring the UseResponse MCP's dual-transport architecture (stdio + Google Cloud Run streamable-http) and 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. Always-on `list_tool_policy` introspection tool; automated test suite validates registered tools match policy across flag combinations.

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

- 5-phase agentic pipeline: Planner (converts PRD to structured test plan) → Generator (produces Playwright TypeScript specs) → Debug (resolves real failures — config mismatches, assertion brittleness) → Healer (simulates selector drift, auto-repairs broken locators).
- 39 tests across 5 suites, all passing; real bugs diagnosed and fixed during the debug phase. 10 CLI playbooks for repeatable, CI-ready agent execution.

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, environment-based builds for Meta Business Verification readiness.

Skills

LLMs & Prompt Engineering	Versioned, context-budgeted prompts tested against real outputs before release; prompt caching (Anthropic), structured outputs, tool use, RAG; deliberate model-tier selection (Haiku/Sonnet/Opus and GPT/Gemini equivalents) for cost and latency
Agentic Systems	Multi-agent architecture design (Connected Agents pattern, Planner/Generator/Healer), Microsoft Copilot Studio, MCP server development (stdio & OAuth2/JWT-secured streamable-HTTP on Google Cloud Run), runtime capability discovery (Unreal Engine 5.8), human-in-the-loop design, fallback and escalation paths, Claude Code, LM Studio (local inference)
Workflow Automation	n8n (self-hosted, queue mode, multi-step workflows, conditionals, error handling), Apify, REST API integration, webhook pipelines
API & Integration	REST APIs, OAuth, JSON parsing and schema validation, pagination, rate-limit handling, OpenAPI contract testing
Scripting & Dev	Python, TypeScript, Node.js, Bash; reads, edits, and debugs AI-generated code
Governance & Security	Prompt injection awareness, least-privilege agent design, secret handling, audit trails; ISC2 Certified in Cybersecurity (CC)
Cloud & Infra	Microsoft Azure (OpenAI, AI Search, Content Safety, API Management, Container Apps, Key Vault, Entra ID, Bicep), Power Platform (Dataverse, Power Apps), OCI (Compute, OKE, IAM, Gen AI Services), AWS (EC2, S3, Lambda, CloudWatch, DynamoDB, X-Ray), Kubernetes, Docker, Terraform, GitHub Actions, OpenTelemetry (OTLP pipelines, Collector configuration, multi-vendor schema normalization)
Enablement	Cross-functional AI adoption, hands-on mentoring, internal documentation and playbooks

Education

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

Certifications

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

Languages

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