Arthur Locke

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

Available Immediately

Summary

Senior QA Engineer with 8+ years of experience across manual and automated testing on PC, mobile, console, and cloud-native platforms. Recent focus on building test automation frameworks, automated integration test suites, and AI-augmented QA tooling – hands-on expertise in Playwright, API integration testing, and CI/CD-integrated continuous testing. Proven track record of building scalable automation infrastructure from scratch, pioneering AI adoption, and leading QA practices across engineering teams.

Professional Experience

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

AI & Automation

- Architected *SE Tester AI Agent System* – three-agent pipeline (Planner, Generator, Healer) that produces and self-heals XML test scripts for the internal Space Engineers headless/headed test tool. Healer ingests MCP server logs for root-cause diagnosis; playbooks executed via MCP for real-time game control.
- Pioneered AI adoption across the engineering team: led GWS + Claude rollout, defined QA use cases for AI-assisted document, spreadsheet, and presentation generation; delivered onboarding documentation and hands-on mentoring.
- Connected internal n8n instance to Claude via MCP server – team can now observe, diagnose, and create automation workflows via natural language.
- Built and deployed the *UseResponse MCP* server, giving Claude direct access to Keen's multi-platform (PC, Xbox, PlayStation, SE2) support portal for Space Engineers bug triage – dual transport (stdio for local Claude Code sessions, streamable-http on Google Cloud Run for scheduled Cloud Routines), OAuth 2.0/PKCE with signed JWTs, and seven independently-gated tool categories, deny-by-default. Shipped custom slash commands (`/triage`, `/submitted`, `/investigating`, `/nmi`, `/search`) driving the team's daily bug-board workflow.
- 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 with GLM 4.7 Flash and Nemotron 3 Nano for local inference). Successfully navigated Space Engineers UI and executed headed regression tests autonomously. Superseded by the SE Tester AI Agent System.

Test Automation & Integration Testing

- Designed and maintained automated integration test workflows covering API validation, external service integrations (weather data, QR code generation, weekly report graphics), and Apify data pipeline schema validation.
- Implemented REST API integration tests validating service contracts and response schemas across third-party integrations.
- Led adoption of SE Tester (XML-driven headless/headed test execution tool): diagnosed failures at source-code level, identified engine changes required for targeted test cases, managed the Jira defect backlog for the tool itself.

Workflow Automation & Observability

- Designed and deployed n8n automation workflows for spam filtering and weekly report generation (API-driven graphics and formatting), saving ∼5 man-hours weekly.

- Operated multi-platform log ingestion system (ELK stack: Elasticsearch, Logstash, Kibana) aggregating telemetry from Xbox, PlayStation, and Steam for centralised QA verification and anomaly detection.
- Assisted with the PlayStation release pipeline for Space Engineers: supported build submission through Sony's TRC (Technical Requirements Checklist) certification, compliance sign-off, and store publication, working alongside the Sony platform team.
- Assisted with Xbox certification preparation and game validation: supported pre-submission validation against Microsoft's XR (Xbox Requirements), helped diagnose certification-blocking issues, and assisted with resubmission cycles.
- Handled player-facing support functions, translating community-reported issues into test priorities and defect triage.

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

- Built AI-driven agentic workflows including RAG chatbot (OpenAI, Gemini, Supabase vector DB) connected to WhatsApp and Telegram via webhook integrations.
- Developed accounting workflow automation: web form → n8n webhook → data processing → government-regulated XML/PDF; XML schema validation against Czech regulatory requirements.
- Engineered automated data extraction, processing, and media handling pipelines for multiple clients (details under NDA).

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 the full release cycle across iOS (App Store), Android (Google Play), and Amazon/Kindle Fire platforms — build sign-off through store submission, certification, and live deployment; briefly led a small, two-person QA team after a second QA hire joined.
- Established QA practices from the ground up; mentored incoming team members.
- Defined and executed test strategies covering automated regression, functional, integration, and platform-specific testing.

Key Projects

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

- Designed and built a multi-package platform that migrates manual TestRail regression suites into AI-evaluated, executable regression tests against a real Unreal Engine 5.8 target, then reports on real execution results – 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.
- Deterministic pipeline: normalizes raw manual TestRail cases into a structured shape, evaluates each against live-discovered engine capabilities to decide automatability (confidence score + unsupported-capability reason when rejected), generates a versioned YAML regression spec, validates it against its schema, gates publishing behind an explicit human-approval step, then writes the evaluated suite back into TestRail – run for real against 9 seeded cases, correctly automating 7 and rejecting 2 for a proven capability gap.
- Execution engine runs the validated spec against the live target and records every result to SQLite; a shared aggregation layer feeds both a static HTML dashboard and a PDF report so the two can never disagree – run end-to-end for real (6 passed, 2 correctly skipped, 1 correctly-predicted failure), with two rendering bugs caught only by visually inspecting the actual output.
- Custom MCP server (`testrail-mcp`) wraps TestRail's REST API v2 with config-driven, per-project read/write/delete scoping and a soft-delete preview mode, verified live including the write-rejected/write-allowed/soft-preview/real-delete round trip.

- Stack: Python, Pydantic, uv workspace, official `mcp` SDK, SQLite, Jinja2, reportlab, 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 and classification platform on top of Chatwoot (unmodified), architected as independently swappable pieces – two self-hosted MCP servers and Claude as a replaceable intelligence provider – documented across 6 ADRs recording the actual reasoning, not a stated principle.
- 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 – captured as its own ADR after catching and correcting a simulated demo "known issue match" and following that honesty through to its actual design consequence.
- Found and fixed 10 real cross-service bugs by running the full stack against a live Chatwoot instance and real Docker containers rather than trusting unit tests alone: Chatwoot's SSRF protection silently blocking same-network webhook delivery, a labels API with replace-not-merge semantics, an inconsistent JSON envelope between two Chatwoot search endpoints, and an asyncpg connection pool silently broken by spanning two different asyncio event loops.
- Real semantic search over the knowledge base (pgvector, embedded locally via fastembed at zero API cost) evaluated against dedicated/managed vector databases and hosted embedding APIs before choosing the zero-infrastructure option – verified live with real relevance scores (0.92 vs. 0.63, genuine match vs. unrelated).
- Scheduled Postgres-to-S3 backups 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.
- 92 automated tests (pytest, moto for S3 mocking) across four services, all passing; every feature verified live against a running Docker Compose stack, not left as untested documentation.
- Stack: Python (FastAPI, fastmcp, boto3, asyncpg, fastembed), Chatwoot Community Edition, Anthropic Claude API, PostgreSQL (pgvector), Docker Compose, Redis, Caddy, Google Cloud Run.

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, verifying every architectural capability live rather than trusting documentation – including catching and documenting a genuine platform-side OAuth consent bug in the Outlook connector across three separate investigation attempts, and confirming two Dataverse/Copilot Studio ALM limitations against Microsoft's own documented known issues.
- Two independent CI/CD pipelines in GitHub Actions, each verified end-to-end via live deployments rather than assumed correct: one redeploys the entire Azure infrastructure from Bicep, the other promotes both business data and the multi-agent Copilot Studio deployment itself between environments.
- Root-caused and fixed real, non-obvious issues via live server responses and direct API queries: a GitHub OIDC subject-claim format change, several `azd`-in-CI environment-state gaps, and a tenant security policy silently blocking an OAuth login flow.
- Stack: Azure (OpenAI, AI Search, Content Safety, API Management, Container Apps, Key Vault), Bicep, Semantic Kernel, Microsoft Entra ID, Copilot Studio, Power Platform (Dataverse), GitHub Actions.

AI Coding Agent Observatory – OpenTelemetry Observability Platform for AI Coding Agents
github.com/A-Locke/ai-coding-agent-observatory

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

telemetry from AI coding agents (Claude Code, Codex CLI, Gemini CLI) – deliberately no synthetic data generator – normalizing each vendor's distinct metric/event/trace schema into one shared, per-agent-adapter ingestion pipeline backed by SQLite.
- Verified end-to-end with a real, live-connected Claude Code session flowing through the OpenTelemetry Collector (OTLP over gRPC/HTTP) into ingestion and the Next.js dashboard – not a test payload.
- GitHub Actions CI gate on every push (lint/typecheck/test/build) plus a manual-approval-gated deploy workflow for the optional AWS path – 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.
- Found and fixed real bugs through live verification rather than assuming correctness: a gzip-compressed OTLP ingest path silently failing, a Docker Compose `-env-file` resolution gap, and missing model-attribute extraction on two of the three vendor ingestion adapters.
- Stack: Next.js 15, TypeScript, Node.js built-in SQLite, Docker Compose, OpenTelemetry Collector, Terraform, GitHub Actions, AWS CloudWatch/IAM.

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

- Designed and built a contract-first OpenAPI testing platform from scratch as a TypeScript pnpm monorepo (9 packages/apps): wrapper API, contract runner CLI, drift detection engine, AI drift analyzer, MCP server, shared types, and mock server.
- Contract runner validates live API responses against an internal OpenAPI spec using AJV schema validation; distinguishes transport errors, upstream schema drift, contract violations, and test assertion failures – each reported separately.
- Drift detection engine (`drift-core`) performs structural diff between upstream Apify spec and the internal contract, surfacing breaking changes before they reach production.
- 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. `-ai-test-gen` merges Claude-generated edge-case tests with spec-driven tests at runtime.
- CI pipeline: type-check → unit tests → Docker container build → live contract validation; 24/24 unit tests and 8/8 contract tests passing with 0 violations confirmed in CI artifact.
- Stack: TypeScript strict, Fastify, Zod (boundary parsing, NaN-safe coercion via `z.coerce.number().catch(0)`), AJV, undici, Vitest, Docker multi-stage (`node:20-alpine`), GitHub Actions.

TestRail MCP – MCP Server for TestRail Test Management

- Built a from-scratch MCP server wrapping TestRail's REST API v2 (projects, suites, sections, cases, runs, plans, tests, results), mirroring the dual-transport architecture (stdio locally, streamable-http on Google Cloud Run) and OAuth 2.0/PKCE + signed JWT security model of the UseResponse MCP above.
- Two-tier tool-gating scheme: five category flags (read, case writes, run writes, result writes, deletes) stacked with an optional named-tool allowlist that can only narrow – never widen – what a category flag permits; deletes deliberately scoped to case/suite only.
- Always-on `list_tool_policy` introspection tool reports live category flags, effective allowlist, and resolved tool list; automated test suite validates registered tools match policy across representative flag combinations.
- Stack: Python, FastMCP, httpx, Docker, uvicorn/Starlette, deployed via `gcloud run deploy -source ..`

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

- Architected a 5-phase AI-driven test pipeline: Planner (PRD → structured test plan), Generator (Playwright TypeScript specs), Debug (resolves real failures – config mismatches, assertion brittleness), Healer (simulates selector drift and auto-repairs broken locators).

- 39 Playwright TypeScript tests across 5 suites, all passing; real bugs diagnosed and fixed during the debug phase.
- Demonstrates self-healing automated integration test infrastructure – locators repair themselves on UI changes, eliminating manual maintenance.

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

- Built a reproducible Kubernetes CKA exam preparation platform on a local `kind` cluster with deterministic task verification and a local AI agent solver requiring no cloud API key.
- Task system: each task injects a broken Kubernetes state (misconfigured Services, missing RBAC, stuck PVCs, scheduling taints); a deterministic `verify.sh` confirms the fix – no fuzzy grading.
- AI solver: local Claude Code agent explains each `kubectl` command and interprets output at every step; 5/5 tasks passed autonomously in a single suite run.
- `/generate-task` slash command designs, writes, and live-tests a complete new task from a one-line description; Killercoda integration publishes browser-ready scenarios to a companion repo – 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

- One-button GitHub Actions CI/CD deploying production Kubernetes on OCI Always Free Tier ($0/month vs $250+/month on AWS).
- Continuous testing infrastructure: Prometheus + Grafana monitoring, automated certificate management, WireGuard VPN zero-trust access.
- Claude Code integration: 5 custom Kubernetes skills and MCP server for natural-language cluster management.

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

Test Automation	Playwright (E2E, API testing, AI-augmented test generation, self-healing locators), Postman (REST API integration testing)
Testing Methods	Automated integration testing, end-to-end testing, regression, API integration testing, schema/contract validation, smoke, exploratory, continuous testing; Agile/Scrum, ISTQB, shift-left
AI & Agentic Systems	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 for engine automation (Unreal Engine 5.8); LM Studio (local inference); vision-based UI agents (OpenClaw, OpenCV, pyautogui)
CI/CD & DevOps	GitHub Actions, Docker, Kubernetes, Helm, Terraform; automated test execution integrated into CI/CD pipelines
Workflow Automation	n8n (custom REST API integrations, multi-service orchestration), Apify (data pipelines, schema validation)
Monitoring	OpenTelemetry (OTLP pipelines – traces, metrics, logs; custom Collector configuration; multi-vendor schema normalization), ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, Grafana; multi-platform log aggregation (Xbox, PlayStation, Steam)
Cloud Platforms	Oracle Cloud Infrastructure (OCI), AWS (EC2, S3, IAM, Lambda, WAF, CloudWatch)
Tools	TestRail, JIRA, Git/GitHub, REST APIs, Node.js, TypeScript, PostgreSQL, Supabase

Education

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

Certifications

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

Languages

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