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 Quality Assurance Engineer with 8+ years in the games industry, embedded across the full QA lifecycle – from design-doc review through shipped-feature and blocker testing to post-release patch triage – having shipped every Space Engineers DLC since the Wasteland update (13 releases and counting) on Steam. Core strengths in regression, functional, exploratory, and build verification testing, defect triage, and API/integration test automation, with direct experience assisting console certification submissions (Sony TRC, Microsoft XR). Specializes in building QA and support tooling from the ground up – including an AI-driven pipeline connecting test management (TestRail), AI test generation/self-healing, and game-engine test execution, portable across engines (Unreal, Unity, proprietary) – and in process optimization that removes manual QA overhead. Early career foundation in VR hardware/software support.

Professional Experience

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

Support Tooling & AI Automation

- Built and deployed the *UseResponse MCP* server from the ground up, giving Claude direct access to Keen's multi-platform (PC, Xbox, PlayStation, SE2) support portal for 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.
- 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 regression test tool. Healer ingests MCP server logs for root-cause diagnosis; playbooks executed via MCP for real-time game control. Direct precedent for a test-management → AI-generation → engine-execution pipeline (see *TestRail MCP* under Key Projects), portable across engines.
- 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.

DLC Development & Full QA Lifecycle

- Embedded across the full QA lifecycle for every Space Engineers DLC since the Wasteland update (v197 through the current v210 – 13 releases): reviewed design docs pre-development, tested every shipped feature and blocking issue pre-release, supported each Steam release cycle (store page, patch notes, launch-day monitoring), and owned post-release patch triage and prioritization. No feature or blocker shipped across those 13 DLCs without passing through this process.

Regression, Functional & Integration Testing

- Led adoption of SE Tester (XML-driven headless/headed regression test execution tool): diagnosed failures at source-code level, identified engine changes required for targeted test cases, authored and maintained the regression test case backlog, and managed the tool's own Jira defect backlog.
- Designed and maintained automated integration test workflows covering API validation, external service integrations, and Apify data pipeline schema validation, plus REST API contract/schema tests across third-party integrations.

- Conducted playtesting, exploratory, functional, and build verification testing (BVT) across PC, Xbox, and PlayStation builds, translating findings into prioritized, reproducible bug reports.

Cross-Platform Certification & Process Optimization

- 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.
- Operated multi-platform log ingestion system (ELK stack: Elasticsearch, Logstash, Kibana) aggregating telemetry from Xbox, PlayStation, and Steam for centralised QA verification, patch/live-ops regression, and cross-platform anomaly detection.
- Designed and deployed n8n automation workflows for spam filtering and weekly report generation, saving ∼5 man-hours weekly – one of several process-optimization initiatives that cut manual QA/ops overhead across the team.

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.

VR Technician Infobus, Prague
Oct 2016 – Jun 2017

- Supported VR software and hardware systems for interactive public-facing installations – early hands-on foundation in VR hardware/software troubleshooting, built on since through 8+ years of cross-platform QA.

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 Unreal Engine 5.8 regression tests and 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.
- *unreal-mcp-adapter* discovers a running UE 5.8 instance's MCP capabilities at runtime (not hard-coded), normalizing ∼25 optional plugin toolsets into one stable capability API (level loading, character movement, UI interaction, assertions, screenshots, logs) – resolved all 10 target capabilities

live, catching real engine quirks (an unwrapped response envelope, a "wait" call that only ever checks once, UI clicks that silently no-op outside a floating PIE window) along the way.
- AI pipeline normalizes manual TestRail cases, evaluates each against the live-discovered Unreal capabilities to decide automatability (with a confidence score and unsupported-capability reason), generates a versioned YAML regression spec, gates publishing behind explicit human approval, then writes the evaluated suite back into TestRail – run for real against 9 seeded cases, correctly automating 7 and rejecting 2 for a proven engine-input limitation.
- *execution-engine* runs a validated spec against the live Unreal demo project; *qa-regression-dashboard* turns the resulting SQLite history into a static HTML dashboard and PDF report from one shared aggregation layer – run end-to-end for real (6 passed, 2 correctly skipped, 1 correctly-predicted failure).
- *qa-demo-project*: a small, playable UE 5.8 demo (menu, character select, options, gameplay, pause, victory, an interactable door, a collectible) built live entirely through the adapter's Blueprint/UMG tooling – the automation target the pipeline proves itself against.
- Stack: Python, Pydantic, uv workspace, official `mcp` SDK, SQLite, Jinja2, reportlab, Docker; Unreal Engine 5.8 Blueprint/UMG/Slate automation.

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-assisted player/customer support triage and ticket-classification platform on top of Chatwoot (unmodified) – a direct extension of the support-portal automation work at Keen Software House (see *UseResponse MCP* above) into a fully self-hostable, platform-agnostic system – architected as independently swappable pieces (two self-hosted MCP servers, Claude as a replaceable intelligence provider) and documented across 6 ADRs.
- 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 player/customer directly, regardless of confidence score – captured as its own ADR after catching a simulated demo "known issue match" and following that honesty through to its actual design consequence.
- Three interchangeable ways to run the identical ticket-classification logic: an automated webhook pipeline, an on-demand CLI, and a fully API-key-free mode where a Claude Code session drives the MCP servers' tools directly – validated by manually classifying and triaging real seeded tickets before the automated pipeline was even built.
- 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, including Chatwoot's SSRF protection silently blocking same-network webhook delivery and an inconsistent JSON envelope between two Chatwoot search endpoints.
- 92 automated tests (pytest) across four services, all passing, plus a live-verified Postgres backup/restore round trip confirming row counts and message content matched exactly.
- 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, 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.
- Verified every capability live rather than trusting documentation: caught and documented a genuine platform-side OAuth consent bug in the Outlook connector across three separate investigation attempts, and confirmed two Dataverse/Copilot Studio ALM limitations against Microsoft's own documented known issues.
- Two independent, live-verified CI/CD pipelines in GitHub Actions: one redeploys the entire Azure infrastructure from Bicep via OIDC federated credentials with no stored secret; the other promotes

both business data and a live multi-agent Copilot Studio deployment between environments.
- Stack: Azure (OpenAI, AI Search, Content Safety, API Management, Container Apps, Key Vault), Bicep, Semantic Kernel, Microsoft Entra ID, Copilot Studio, Power Platform (Dataverse).

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, with the full dispatch → approval-pause → apply → destroy cycle verified 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 and missing model-attribute extraction on two of the three vendor ingestion adapters.

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. Paired with the *SE Tester AI Agent System* experience above, this forms a working blueprint for a TestRail → AI-generation/self-healing → engine-execution regression pipeline – portable across engines (Unreal, Unity, proprietary).
- 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), Healer (simulates selector drift and auto-repairs broken locators).
- 39 Playwright TypeScript tests across 5 suites, all passing; demonstrates self-healing regression infrastructure – locators repair themselves on UI changes, eliminating manual test maintenance.

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

Testing Methods	Regression, functional, exploratory, smoke, build verification (BVT), playtesting, live-ops/patch, compatibility (console generations), automated integration, end-to-end, API integration, schema/contract validation, continuous testing; test case design, test plan authoring, defect/bug triage; Agile/Scrum, ISTQB, shift-left
Test Automation	Playwright (E2E, API testing, AI-augmented test generation, self-healing locators), Postman (REST API integration testing), TestRail (test case/run/result management, custom MCP integration)
Platforms	PC (Steam), mobile (iOS, Android, Amazon), console (Xbox, PlayStation)
Console Certification	Sony PlayStation TRC (Technical Requirements Checklist), Microsoft Xbox XR (Xbox Requirements)
AI & Agentic Systems	Multi-agent pipeline design (Planner/Generator/Healer) for automated test generation & self-healing, portable across game engines; Claude Code, MCP server development; LM Studio (local inference); vision-based UI agents (OpenClaw, OpenCV, pyautogui)
Game Engine Automation	Unreal Engine 5.8 (Blueprint/UMG/Slate automation via MCP capability discovery), manual-to-automated regression migration, TestRail ↔ engine execution pipelines
Workflow & Support Automation	n8n (custom REST API integrations, multi-service orchestration), support-portal automation built from the ground up (UseResponse MCP), Apify (data pipelines, schema validation)
Monitoring	OpenTelemetry (OTLP pipelines, custom Collector configuration, multi-vendor schema normalization), ELK Stack (Elasticsearch, Logstash, Kibana), multi-platform log aggregation (Xbox, PlayStation, Steam)
Tools	JIRA, Git/GitHub, REST APIs, Node.js, TypeScript, PostgreSQL, Supabase, Docker

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 | Certified in Cybersecurity (CC) – ISC2

Languages

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