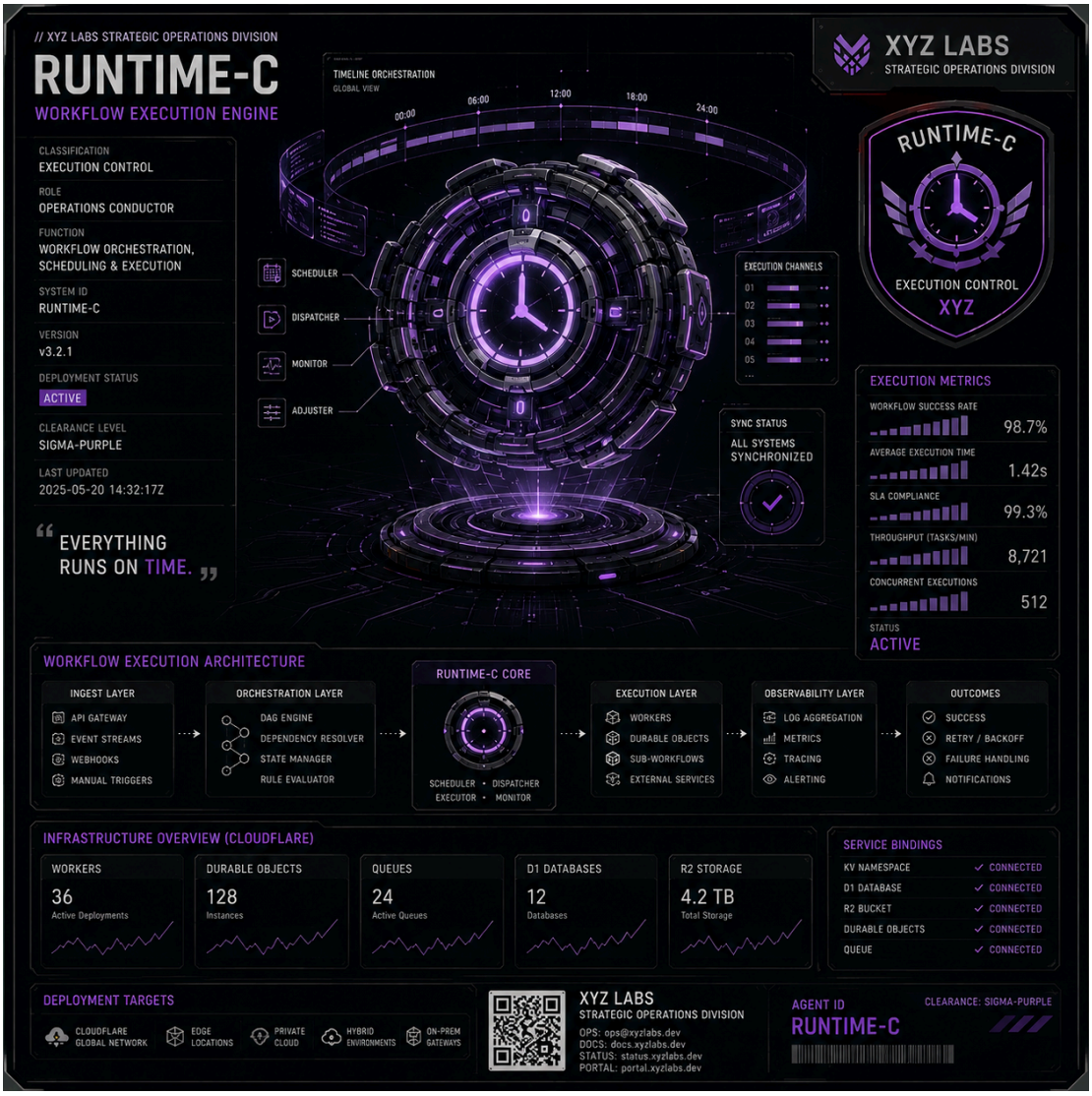
Runtime-C Full Technical Architecture Audit

Comprehensive engineering audit compiled from the Runtime-C architecture inspection, forensic ledger, retrieval configuration, run history, tool manifests, pod configuration, stage matrix, workspace contracts, and extensive audit session.

Executive Summary

Runtime-C is an execution engine embedded within Factory-XYZ. It accepts buildsheets, resolves execution pods, orchestrates six worker lanes, injects retrieval context, streams execution, validates output, merges artifacts, records proof, telemetry and TPS, and publishes final artifacts.



Architecture Layers

Runtime-C consists of: submit server, queue, runner, pod resolver, pod authority, worker lanes WA-WF, retrieval layer, streaming layer, quality gates, merge layer, promotion layer, telemetry, proof generation, publication.

Execution Contract

The inspected README establishes Runner responsibilities (queue, scheduling, checkpoints, validation, imports) while Pod authority owns factory rules, toolbelt, workspace, LLM resource, artifact generation and proof generation.

Filesystem

Discuss runtime root, sandbox/runs, sandbox/artifacts, restore-points, workspace-seed, pods, tools/runtime-c-lab, warehouse, families, products, web-public, golden, quarantine. Include forbidden runtime roots (/srv/eila, /mnt/eila-hot-sidecar, /opt/eila-os/job_site) from the contract and note historical exceptions in the forensic audit.

Pod System

Document Pod A, B and C. Pod A: qwen3:8b on 100.105.163.20:11435. Pod B: qwen2.5-coder:32b. Pod C: local qwen2.5-coder:14b. Explain handoff scripts, rootfs, golden rules, toolbelt, artifact ownership.

Retrieval

Describe LanceDB configuration, runtime_c_assets table, top_k=5, max chars, resource maps, Qwen indexes, runtime-c-assets warehouse, 5M+ indexed assets, external retrieval sources.

Tool Inventory

Summarize inspected runtime-c-lab inventory including submit server, runner, lane-real, streaming orchestrator, compare-runs, verify tools, publish, metrics, TPS writer, telemetry writer, route context, repair loop, bulk scheduler, context enricher, quality gate, regression gate, export tools and diagnostics.

Subsystem	Representative Tools
Execution	runtime-c-runner, submit-server, resolve-pod, exec-pod-authority
Generation	lane-real, compose, persist-fragments
Streaming	streaming-orchestrator, stream-status, stream-append
Validation	quality-gate, regression-gate, verify-run
Publishing	write-index, publish-full-repo, publish-from-base
Telemetry	write-metrics, write-telemetry, write-tps
Repair	repair-loop, queue-janitor, compare-runs

Run Lifecycle

Describe buildsheet creation, pod resolution, prompt generation, WA-WF execution, receipts, stream logs, merge, verification, metrics, TPS, proof, publish. Include observed run folders Betty's Buns and Big Balls Bowling.

Evidence Produced

List PROMPT_*.json, LLM_RECEIPT_*.json, EVENT_LOG.jsonl, LANE_TELEMETRY.json, TEMPORAL_LINEAGE.json, VERIFY_RUN.json, QUALITY_GATE.json, REGRESSION_GATE.json, RUNTIME_METRICS.json, merged_product, output, proof.

Forensic Findings

Summarize the inspected ledger: TPS fallback bug, premature RUN_DONE, stale telemetry timing, stale /opt roots, missing publish tool, stale latest.json, UI using job_site, duplicate pod truth sources, nginx pollution, retrieval path concerns.

Architecture Assessment

Strengths: auditable execution, reproducibility, separation of concerns, retrieval integration, proof-first design. Weaknesses: legacy path drift, duplicate authorities, backup size, stale publication chain.

Recommendations

Normalize runtime paths, unify pod authority source, separate source from run history, clean publication pipeline, document architecture diagrams, create release packaging profiles, continue graph mapping.

Appendix A: Observed Run Layout

Runtime-C run folders preserve prompts, receipts, streams, metrics, telemetry, merged product, output, proof, worker packets, lineage and TPS. This provides deterministic replay and forensic traceability.

Appendix B: Overall Verdict

Runtime-C is a production-style orchestration engine rather than a simple prompt runner. The inspected architecture demonstrates explicit separation between orchestration, execution authority, retrieval, quality control, proof generation and publication. The next engineering milestone is path normalization and graphing the complete execution DAG.