Grover Agent Sandbox — Full Build Report & Builder's Guide

Project: SynQc Temporal Dynamics Series (TDS) — Grover Agent Sandbox
Timeframe Covered: Past week (local sandbox build and validation)
Audience: Future builders (human or AI), reviewers, and yourself as historical ground truth
Purpose: Preserve exact context, decisions, failures, fixes, and expectations so the *next* build is faster, calmer, and cleaner.

---

1. Executive Summary (What This Sandbox Is)

The **Grover Agent Sandbox** is a fully local, containerized, agent-based execution environment designed to:

- Run **Grover's algorithm workflows** as a *sandboxed agent*, not as a toy script.
- Exercise the **SynQc TDS hybrid controller** architecture without quantum hardware.
- Validate **agent orchestration, metrics, latency, safety, and observability** before UI or cloud deployment.

This sandbox proves that: - Agents can be isolated, invoked, measured, and supervised. - Quantum-algorithm logic can live inside a governed system. - The SynQc architecture is viable beyond diagrams.

This was a *systems engineering* exercise, not a math demo.

---

2. Goals We Set (and Why)

Primary Goals

1. Build a **real sandbox**, not a notebook demo.
2. Keep everything **local-first** (Windows + Docker + Git Bash).
3. Ensure **agents run deterministically** with clear lifecycle states.
4. Produce **metrics and logs** suitable for Prometheus.
5. Avoid hardware dependencies.

Explicit Non-Goals

- No quantum hardware execution yet.
- No production UI yet.
- No premature optimization or cloud lock-in.

This restraint mattered. It prevented architectural debt.

---

3. High-Level Architecture

Core Components:

- **API Service** – receives job requests and exposes metrics
- **Worker Service** – executes the Grover agent
- **Redis** – job queue, state tracking
- **Sandbox (Grover)** – isolated execution context
- **Prometheus Endpoint** – metrics scrape target

All services run under **Docker Compose**.

The Grover agent is invoked as a *job*, not as a script.

---

4. Tools & Stack Used (Exact)

Core Infrastructure

- **Windows 10/11** (host)
- **Docker Desktop (WSL2)**
- **Docker Compose v2**
- **Redis (container + cloud testing)**
- **Git Bash**
- **VS Code**

Backend / Runtime

- **Python 3.12**
- **FastAPI** (API surface)
- **AsyncIO** (agent execution model)
- **Structured JSON / CSV outputs**

Observability

- **Prometheus-compatible metrics endpoint**
- Custom counters and histograms:
- `synqc_bench_events_total`
- `synqc_bench_latency_ms_*`

Quantum Layer (Sandboxed)

- **Grover algorithm logic** (simulation / dry-run mode)
- No Qiskit hardware calls yet

---

5. What We Built (Step-by-Step, Truthfully)

Step 1 — Container Baseline

- Docker Compose brought up:
- `api`
- `worker`
- `redis`
- `shor` (present but not active for Grover)

Initial success: containers ran.

Hidden problem: services were running, but *not all were reachable the way we assumed*.

Step 2 — Agent Invocation Model

We designed Grover as:

- A **job-based agent**
- Triggered via API endpoint
- Executed asynchronously by worker
- Returning structured results

This decision paid off later when debugging latency and retries.

Step 3 — Dry-Run Execution Path

We implemented a **dry_run=true** path:

- Validates job lifecycle
- Measures latency
- Avoids quantum backend calls

This allowed safe iteration.

Step 4 — Metrics & Prometheus

We exposed:

- Job counters
- Latency histograms
- Sandbox identification labels

This turned the sandbox into a *measurable system*, not a black box.

---

6. What Went Wrong (No Sugar-Coating)

6.1 Docker Compose Confusion

Symptom: - Services appeared to be running - Logs said “no such service” for expected containers

Root Cause: - Stale assumptions about service names - Docker Compose v2 ignoring `version:` field - Mental mismatch between YAML structure and actual service graph

Fix: - Used `docker compose config --services` - Treated output as ground truth - Removed mental shortcuts

---

6.2 Frontend Ghosts

Symptom: - Old frontend at `127.0.0.1:8080` still accessible

Root Cause: - Legacy artifacts not fully removed - Confusion between old and new UI paths

Fix: - Explicitly declared deprecated UI paths - Decided MVP UI must be *authoritative*, not additive

---

6.3 Redis Communication Issues

Symptom: - Backend alive - Jobs queued inconsistently

Root Cause: - Redis config mismatch between container and cloud assumptions

Fix: - Locked Redis usage to sandbox scope - Deferred cloud Redis until post-MVP

6.4 Git Bash Friction

Symptom: - Commands failing unexpectedly

Root Cause: - Windows path translation quirks - Editable installs misused

Fix: - Stopped clever installs - Used explicit paths - Treated Git Bash as a *shell*, not magic

---

7. What the Grover Sandbox Actually Does

In plain terms:

- Accepts a job request
- Spins up a Grover agent
- Simulates algorithm flow
- Measures latency and execution stages
- Emits metrics
- Returns structured output

It **does not**:

- Break encryption
- Claim quantum advantage
- Touch hardware

It proves orchestration, safety, and observability.

---

8. Why This Sandbox Matters

This sandbox:

- Validates SynQc TDS as a *platform*, not a paper
- Shows agents can be governed
- Creates a repeatable build pattern
- Prevents chaos in future algorithm sandboxes (Shor, VQE, etc.)

It is foundational infrastructure.

---

9. Lessons Learned (Hard-Won)

1. **Running ≠ Working**
2. Docker output > human memory
3. Metrics early prevent hallucinations later
4. Dry-run paths are non-negotiable
5. UI must wait for backend truth

These are now standing rules.

---

10. Builder Expectations for Next Sandbox

Anyone continuing this work must:

- Assume nothing
- Verify services explicitly
- Treat logs as first-class data
- Preserve dry-run capability
- Keep sandbox scope tight

No heroics. No shortcuts.

---

11. MVP UI — What Comes Next (Brief, Clear)

What the MVP UI Will Do

- Provide a **single authoritative interface**
- Submit Grover jobs
- Display:
- Job state
- Latency
- Metrics snapshots
- Surface errors clearly

What It Will NOT Do (Yet)

- No hardware toggles
- No fancy animations
- No algorithm editing

---

12. Impact on SynQc TDS Front End

Yes — **design changes are required**, but controlled:

- Remove legacy UI paths
- Introduce a **Sandbox Panel**
- Treat Grover as a *plugin agent*
- Keep visual language consistent

This is additive *only where authoritative*.

---

13. Final Note (Why This Document Exists)

This document is not marketing.

It is: - A memory anchor - A context injector - A speed multiplier

Dropping this into a future chat should instantly align expectations, vocabulary, and standards.

This is how systems stay sane.