Challenge

This is a take-home challenge for the role of Trading Systems/Platform/Infra Engineer at Batonics. This assessment is designed to evaluate your production engineering skills. Our industry is where a 1 in 1000 chance of error can lead to disasters, hence the focus.

The challenge includes more requirements than anyone (except for very rare outliers) can complete in a short time, this is intentional. So obviously you don't need to do it all, just choose the parts that best showcase your skills and interests, and tell us how much time it took you and how many steps you completed. You can spend 10 min and do half a step, up to you, just let us know how much time, and what steps you chose.

We allow and encourage using AI tools to accelerate development, but be thoughtful about where you apply them, especially regarding performance, reliability, and security.

Deliverables: Submit your repo, URL, and the reconstructed JSON file (we'll compare it to the 'correct' one), readme and steps done/hours spent

---

The Challenge

Pick the requirements that matter most to you, we care more about doing a few things exceptionally well than attempting everything superficially.

Core Requirements

These form the foundation. We recommend tackling at least 2-3 of these:

1. **Data Streaming**: Stream the MBO file at 50k-500k messages/second over TCP (design for scalability)
2. **Order Book Reconstruction**: Build an accurate order book with p99 latency <50ms and output as JSON (if you don't know how, ask an AI or read here)
3. **Data Storage**: Persist to a time-series database (PostgreSQL/TimescaleDB/ClickHouse/SQLite) with appropriate schema design
4. **Deployment**: Dockerized application with clear setup instructions (docker-compose or similar)

Production Engineering

Add any combination of these based on your strengths:

6. **API Layer**: REST or WebSocket API supporting **10-100+ concurrent** clients reading the order book
7. **Frontend**: Web interface visualizing live order book updates
8. **Configuration Management**: Externalized config with no hardcoded credentials (environment variables, config files)
9. **Reproducible Builds**: Dependency locking and documented build process
10. **Testing**: Unit tests, integration tests, or correctness proofs for order book logic

11. **Performance Optimization**: Achieve higher throughput (targeting 500K msg/sec with p99 <10ms)
12. **Observability**: Metrics (latency percentiles, throughput), structured logging, or distributed tracing
13. **Infrastructure as Code**: Terraform, Pulumi, or similar for deployment automation
14. **Multi-Environment Setup**: Dev/staging/prod configurations with CI/CD pipeline
15. **Resilience Testing**: Demonstrate graceful handling of failures (connection drops, pod kills, etc.)
16. **API Reliability**: Idempotency, retry logic, proper error handling
17. **Security**: Supply chain verification, dependency auditing, SBOM generation
18. **Correctness Verification**: Prove the order book never violates exchange rules (price-time priority, valid quantities)

What We're Evaluating

- How much volume you can produce in what time **without** compromising latency, performance, load, correctness, security, QA, resilience, etc
- Quality over quantity. We'd rather see 3 requirements done exceptionally well than 10 done poorly.
- Provide a personal readme with any thoughts you had (that’s the only part in the assignment where AI is not allowed, you can produce another instructions readme with AI)
- Where you used AI tools and how you validated the output

If unclear about anything please email ahmed@batonics.com