SRE Automation Case Study

Stock Exchange Trading Platform

Industry Context

A national stock exchange runs a **high-frequency trading (HFT) platform**. Peak traffic occurs during:

- Market open (9:00–9:30 AM)
- Major announcements
- Sudden buy/sell surges

They must ensure:

- Millisecond latency
- Zero downtime during trading hours
- Fast recovery
- Automated reliability operations

Case Study Scenario

ExchangeX, a major stock exchange, operates a trading engine that processes **50 million transactions/day**.

Current Challenges

1. **Manual deployment approval**, leading to delays.
2. **Frequent CPU spikes** on matching engine (buy/ sell) nodes.
3. **Order processing queue delays**, not detected until traders complain.
4. **Manual rollbacks** that take 20–30 minutes.
5. **Incidents are manually triaged**, slowing MTTR.
6. **Inconsistent logs**, making root-cause analysis difficult.

Business Goals

- Achieve **99.99% availability** during trading hours.
- Reduce **MTTR from 25 minutes to <5 minutes**.
- Automate **scaling, deployment, rollbacks**, and **incident detection**.
- Reduce **manual operations by 70%** within 3 months.

Questions

Q1. What SRE Automation should be introduced to reduce manual deployment delays?

Q2. How can auto-scaling be implemented for a trading engine that cannot go down?

Q3. What automation can detect order queue delays before traders report issues?

Q4. How can automated rollbacks be implemented for critical trading services?

Q5. What SRE automation can reduce MTTR during high-severity incidents?

Q6. What logging and monitoring automation strategy would best fit a stock exchange platform?

Q7. How can SREs use resiliency testing safely in a financial trading system?

Q8. What error budget policies should be set for a 99.99% SLA trading system?

Model Answer

Q No.	Question	Model Answer
A1	What SRE Automation should be introduced to reduce manual deployment delays?	**Automated CI/CD + GitOps + Canary Deployment**: Trigger builds automatically from Git, auto-deploy to staging, then canary deploy to 1% of nodes; auto-rollback if latency/error rate spikes. **Removes manual approvals and accelerates safe releases**.
A2	How to implement auto-scaling for a trading engine that cannot go down?	**Predictive + Layered Scaling**: Pre-scale trading systems before peak hours using historical traffic; gateway layers use horizontal auto-scaling; core matching engines stay static but redundant; use blue–green for safe replacement. **Ensures zero-downtime scale adjustments.**
A3	What automation can detect order queue delays early?	**SLO-based Monitoring Automation**: Track queue depth, order latency, and processing rate; set automated alerts; auto-run actions such as scaling gateways, restarting slow consumers, or notifying teams via Slack. **Detects issues before traders complain.**
A4	How to automate rollbacks for critical trading services?	**Automated Health Check Rollbacks + Feature Flags**: CD pipeline checks latency/error metrics; if 3 checks fail, auto-rollback triggers; maintain versioned images; feature flags instantly disable problematic logic. **Rollback completes within <60 seconds.**
A5	What SRE automation can reduce MTTR in high-severity incidents?	**Runbook Automation + AIOps + ChatOps**: Automatically diagnose issues (CPU spikes, latency, queue backlog); auto-run common recovery actions; incident bot posts logs, root cause hints, and suggests runbooks. **MTTR drops from 25 minutes → <5 minutes**.
A6	What logging/monitoring automation suits a stock exchange?	**Centralized Structured Logging + Auto-Anomaly Detection**: Use JSON logs, OpenTelemetry, and ELK/Loki pipeline; automate correlations (order ID ↔ error ↔ latency); auto-trigger alerts on anomalies; enforce automated log retention. **Speeds RCA and ensures compliance**.
A7	How can SREs use resiliency testing safely in a financial trading system?	**Run tests only during non-trading hours** Perform tests in staging or pre-production, not live markets; test small failures like: one server slowing down, network delay, or a service restart Ensure backup systems take over smoothly Always limit the blast radius (only test one component at a time). This helps the **exchange verify that trading continues even if a part of the system fails**—without risking real-money trades.
A8	What error budget policy fits 99.99% SLA?	**52 min/year downtime → Strict Budget Policy**: Only 40 minutes considered usable; freeze deployments after 70% consumption; allow only canary/feature-flag releases when 80–100% budget is used. **Protects stability during trading hours**.