End-to-End DevOps Integration Report – ZeroVault Platform

1. Executive Overview (DevOps Perspective)

This document presents a **full-system integration report** for the ZeroVault platform, written explicitly from a **DevOps engineer's viewpoint**. The Word document containing user stories is treated as the **entire product definition**, and every technical component (frontend, backend, browser extension, distributed sync engine, security framework, and risk engine) is mapped into a **single cohesive, deployable, secure, and observable system**.

The objective is to answer four core DevOps questions: 1. **What components exist?** 2. **What tools & frameworks power each component?** 3. **How do these components integrate together?** 4. **How is this system built, tested, secured, deployed, and operated at scale?**

---

2. System-Level Architecture

High-Level Components

- **Frontend Web Application (SPA)**
- **Chrome Extension (Client-Side Vault & Autofill)**
- **Backend API Layer (Auth, Vault, Device Security)**
- **Distributed Sync Engine (Consistency & Offline Support)**
- **Security & Cryptographic Framework (Zero-Knowledge)**
- **Risk Engine Core (Deterministic Policy Engine)**
- **DevOps Tooling (CI/CD, Infrastructure, Observability)**

Each component is independently deployable but **cryptographically and logically coupled**.

---

3. Frontend Integration (Web Application)

Technology Stack

- Backend Runtime: **Node.js + Express**
- Database: **Supabase PostgreSQL**
- Versioning: **Monotonic counters stored per vault row**
- Concurrency Model: **Optimistic Concurrency Control (OCC)**

Sync Model

- **Delta-based synchronization**
- **Monotonic vaultVersion counter**
- **Idempotent replay-safe updates**
- **Last-Writer-Wins (LWW)** conflict resolution

Integration Flow

1. Client sends `{ base_version, added, updated, deleted }`
2. API validates JWT & device
3. Supabase transaction validates version atomically
4. Delta applied in a single SQL transaction
5. vaultVersion incremented
6. Updated encrypted blob returned
7. Client sends `{ base_version, added, updated, deleted }`
8. Server validates version
9. Applies delta atomically
10. Increments vaultVersion
11. Returns new state

Failure Handling

- `409 Conflict` enforced at API + database constraint level
- Supabase transactions guarantee atomicity
- Retry-safe and network-loss tolerant
- `409 Conflict` → client rebase required
- Retry-safe (network loss tolerant)

---

7. Security & Zero-Knowledge Framework

Cryptographic Stack

- Key Derivation: **PBKDF2 / Argon2id**
- Encryption: **AES-GCM / XChaCha20**
- Hashing: **SHA-256**

Security Principles

- Zero-Knowledge (Server cannot decrypt vault)
- Forward secrecy via re-derived session keys
- Encrypted-at-rest & encrypted-in-transit

Device Security

- Device IDs tracked server-side
- Revoked devices blocked at middleware

---

8. Risk Engine Integration

Technology Stack

- Language: **C11**
- Build: **Makefile → static library (librisk.a)**
- Crypto Dependency: **libsodium**

Integration Pattern

- Risk engine remains **pure & deterministic**
- Wrapped by Vault Controller for state
- Enforced anti-rollback via version counter

DevOps Build Responsibilities

- Deterministic builds
- Static linking
- Memory-zeroing enforcement

---

9. CI/CD Pipeline Design

CI (Pull Requests)

- npm ci
- Lint
- Unit Tests
- Build Verification
- Merge blocked on failure

Nightly / Release

- Playwright E2E
- Artifact retention (screenshots, traces)

Security Gates

- No secrets in repo
- Dependency audit hooks

---

10. Deployment Strategy

Database

- **Supabase (Managed PostgreSQL)**
- Row Level Security (RLS) policies per user
- Encrypted vault blobs stored as opaque data

Frontend

- Vercel / Netlify
- CDN-backed
- Immutable builds

Backend

- Containerized Node.js service

- Horizontal scaling
- Environment-based config

Database

- MongoDB Atlas
- Index-backed consistency

---

11. Observability & Operations

Logging

- Structured JSON logs
- Request ID propagation

Metrics

- Vault sync success/failure
- Conflict rate
- Latency percentiles

Alerts

- Auth failure spikes
- Sync conflict storms

---

12. Tooling Summary Table

Layer	Tools & Frameworks
Frontend	React, Vite, Tailwind, Vitest, Playwright
Extension	Chrome MV3, Web Crypto API, Zustand
Backend	Node.js, Express, Supabase (PostgreSQL), JWT
Sync Engine	OCC, LWW, Versioned Documents
Security	AES-GCM, PBKDF2, Argon2id
Risk Engine	C11, Makefile, libsodium
DevOps	GitHub Actions, Vercel, Docker

---

13. Final DevOps Verdict

This system is: - **Cryptographically sound** - **Operationally scalable** - **Failure-resilient** - **Audit-ready**

From a DevOps standpoint, ZeroVault is **production-grade by design**, with security and consistency treated as first-class operational concerns.