ContractWatch – MVP Technical Design Document

Document Version: 0.1\ **Author:** ChatGPT (draft for Rameez Jhaveri & engineering)\ **Date:** 5 July 2025

---

1. Purpose

This document translates the **Business Requirements Document (BRD)** into a concrete technical plan for delivering the ContractWatch MVP. It defines architecture, components, data flows, tech stack choices, security, and operational concerns.

---

2. High-Level Architecture

```
┌───────────────┐        WebSockets / HTTP(s)         ┌──────────────┐
│   Frontend    │  ←──────────────────────────────→  │ API Gateway │
│  (Next.js)    │                                    │  (Fastify)   │
└───────┬───────┘                                    └───────┬──────┘
        │ REST/GraphQL                                       │ RPC
        ▼                                                    ▼
┌───────────────┐        Pub/Sub         ┌────────────────┐  JSON-RPC
┌─────────────┐
│ Notification  │  ◄─────────────►  │  Event Bus      │  ◄──────────  │  EVM
Nodes  │
│  Service      │ (NATS Streaming)  └────────────────┘
(Alchemy /   │
└───────────────┘                                              QuickNode
                         ▲
RPCs)│
                         │
└────────────┘
                         │
                  ┌─────────────┐
                  │ Chain Scan  │
                  │  Worker     │
                  └──────┬──────┘
                         │ Inserts
                         ▼
                  ┌──────────────┐
                  │  Postgres +  │
                  │  TimescaleDB │
                  └──────────────┘
```

Component Responsibilities

Component	Role
Chain Scan Worker	Streams blocks via WebSocket or polls JSON-RPC; detects `CREATE/CREATE2` tx receipts; enriches with basic proxy fingerprints; publishes `deployment.created` events to NATS.
Event Bus (NATS Streaming)	Reliable, at-least-once pub/sub for internal events.
API Gateway (Fastify + tRPC)	Auth, rate-limit, exposes REST & WebSocket endpoints consumed by the frontend.
Notification Service	Subscribes to `deployment.created`; looks up subscriber wallets; dispatches email (AWS SES) and Discord webhooks.
Database (Postgres + TimescaleDB)	Stores wallets, deployments, alert configs; hypertable on `timestamp` for efficient time-series queries.
Frontend (Next.js + Tailwind)	SPA dashboard: wallet management, timeline view, deployment detail, alert settings, CSV export.

---

3. Technology Choices

Layer	Tech	Rationale
Runtime	**Node.js 20**	Mature libs for web3 + serverless; native WebSocket.
Framework	**Fastify**	Lightweight, high-perf, built-in schema validation.
Data	**Postgres 16 + Timescale**	Relational for wallet/account; time-series for high-volume deployment events.
Event Bus	**NATS Streaming (JetStream)**	Simple, cloud-agnostic, durable message delivery.
Blockchain Access	**Alchemy & QuickNode RPC** (load-balanced)	High reliability and metrics; free tier for testnets.
Emails	**AWS SES (sandbox)**	Cheap, DKIM/SPF support.
Hosting	**Render.com** or **Fly.io** (multi-region Postgres)	Simpler DevOps for MVP.
CI/CD	**GitHub Actions**	Build, lint, unit-test, deploy via Fly/Render.

---

4. Detailed Component Design

4.1 Chain Scan Worker

- **Language:** TypeScript + ethers.js.
- **Networks:** ETH Mainnet, Sepolia, Arbitrum, Polygon.
- **Process:**
- Connect to `eth_subscribe("newHeads")` for each RPC.
- For each block, fetch tx hashes → receipts.
- If `receipt.contractAddress` ≠ null **AND** `tx.from` in `watched_wallets` table → build `Deployment` record.
- Basic proxy detection:
  - Check `slot 0xb531…` for admin; tag as OZ Transparent.
  - Look for `proxiableUUID` selector → tag as UUPS.
- `publish("deployment.created", payload)` to NATS.
- Batch insert to Postgres (COPY every 500 events).
- **Throughput target:** 2 blocks/sec per chain; < 50 MB RAM.

4.2 Notification Service

- Subscribes to `deployment.created`.
- Resolves which users track `payload.deployer`.
- Templates email via MJML → SES; Discord JSON payload.
- Deduplicates multiple deploys in same block (per user) to reduce spam.

4.3 API Gateway

- Fastify plugins: `@fastify/jwt`, `@fastify/rate-limit`, `@fastify/websocket`.
- Endpoints:
- `POST /v1/wallets` (add wallet)
- `GET  /v1/deployments?wallet=…&limit=…`
- `GET  /v1/export.csv`
- WebSocket `ws://…/live?token=…` → push new events.

4.4 Database Schema (simplified)

```sql
CREATE TABLE wallets (
  id UUID PRIMARY KEY,
  user_id UUID,
  address BYTEA UNIQUE,
  created_at TIMESTAMPTZ
);

-- Timescale hypertable
CREATE TABLE deployments (
  ts TIMESTAMPTZ NOT NULL,
  wallet_id UUID REFERENCES wallets(id),
  network TEXT,
  contract_address BYTEA,
  tx_hash BYTEA,
  gas_used BIGINT,
  proxy_type TEXT, -- null | transparent | uups
  PRIMARY KEY (ts, contract_address)
);
SELECT create_hypertable('deployments','ts');
```

---

5. Data Flow (Sequence)

1. **User adds wallet** → API inserts into `wallets` and emits `wallet.added`.
2. **Backfill Job** (one-off) scans last N blocks for historic deploys; populates `deployments`.
3. **Chain Scan Worker** publishes new deploy events.
4. **Notification Service** sends alerts + pushes via WebSocket.
5. **Frontend** receives real-time events, updates UI.

---

6. Security Considerations

- **Read-only chain interaction** (no private keys on server).
- **Least privilege IAM** for SES and DB.
- **JWT auth** with 15-min expiry; refresh tokens stored httpOnly.
- **Rate limiting** 100 req/min per IP to prevent enumeration.
- **Alert spoof prevention**: include tx hash & Etherscan link so user can verify.

---

7. Observability & Monitoring

Metric	Tool
Chain scan lag (blocks)	Prometheus + Grafana
NATS queue depth	NATS Exporter
API latency P95	OpenTelemetry traces to Grafana Cloud
Email/Webhook success	SES metrics; custom webhook retries

---

8. Limitations & Future Work

- **Non-EVM chains** will require chain-specific scanners.
- **Proxy upgrade monitoring** (delegatecall storage diff) deferred post-MVP.
- **Role-based org view** (multi-user teams) not in first release.

---

9. Deployment & Release Plan

1. **Dev** – Fly.io preview app; test RPC via Sepolia.

2. **Staging** – Same infra, but full networks, sample wallets.
3. **Production** – Multi-region Fly.io app + managed Timescale Cloud; SES prod region.
4. **Blue/Green** deploy via GitHub Actions tag.

---

10. Open Questions

1. Do we need API rate-limits per user tier in MVP?
2. Which Discord alert format (embed vs plain)?
3. Will we support CSV exports larger than 10 k rows initially?

End of Technical Design Document