GhostLedger

Deployment & Operations Runbook

From intake form to chain anchor — the operational backbone

Version 1.0 — February 2026

CONFIDENTIAL

Table of Contents

1. System Overview

GhostLedger is a distributed system composed of seven core services, 27 autonomous agents, and a Solana-based chain anchor layer. The system operates 24/7, ingesting noise from across the internet, detecting harm, managing financial claims, and coordinating human legal representation.

Service	Port	Purpose
`gl-api`	`8080`	Public REST API gateway
`gl-intake`	`8081`	Google Form webhook receiver
`gl-nie`	`8082`	Noise Intelligence Engine
`gl-sde`	`8083`	Shadow Detector Engine
`gl-litmus`	`8084`	LITMUS Evaluation Service
`gl-governance`	`8085`	Policy engine + decision gate
`gl-anchor`	`8086`	Solana chain anchor service
`gl-bus`	`4222`	NATS event bus
`gl-db`	`5432`	PostgreSQL primary
`gl-cache`	`6379`	Redis cache + pub/sub
`gl-store`	`9000`	MinIO object storage (evidence)

All services communicate through the NATS event bus. The API gateway handles external requests and routes them to the appropriate service. Every mutating operation passes through the governance layer before execution.

2. Infrastructure Requirements

2.1 Minimum Compute

Component	CPU	RAM	Storage	Notes
API + Intake	2 vCPU	4 GB	10 GB	Stateless, horizontally scalable
NIE + SDE	4 vCPU	8 GB	20 GB	CPU-intensive classification
LITMUS	4 vCPU	8 GB	10 GB	Parallel agent execution
Governance	2 vCPU	4 GB	10 GB	Policy evaluation, decision gate
Anchor	1 vCPU	2 GB	5 GB	Solana RPC client
PostgreSQL	4 vCPU	16 GB	100 GB	Primary data store
Redis	2 vCPU	8 GB	20 GB	Cache, session, pub/sub
NATS	1 vCPU	2 GB	10 GB	Event bus with JetStream
MinIO	2 vCPU	4 GB	500 GB	Evidence object storage

2.2 Network Requirements

- Public: HTTPS (443) for API gateway and webhook endpoint only
- Internal: All services communicate over private network via NATS
- Solana: Outbound HTTPS to Solana RPC (mainnet-beta or devnet)
- DNS: api.ghostledger.io → API gateway load balancer

2.3 Solana Requirements

The anchor service needs a funded Solana wallet for transaction fees. At current rates, anchoring costs approximately $0.00025 per transaction. With batched anchoring (every 60 seconds), a $10 SOL balance covers roughly 6 months of normal operation.

```
# Solana wallet setup
solana-keygen new --outfile /etc/ghostledger/anchor-keypair.json
solana airdrop 2 $(solana address)  # devnet only

# Environment variable
SOLANA_KEYPAIR_PATH=/etc/ghostledger/anchor-keypair.json
SOLANA_RPC_URL=https://api.mainnet-beta.solana.com
SOLANA_NETWORK=mainnet-beta
```

3. Intake Pipeline

The intake pipeline is the bridge between the live Google Form and the GhostLedger system. Every claim starts as a form submission. The pipeline validates, converts, and routes it into the case management system.

"The form is already live. This is how it connects."

3.1 Flow

- Step 1 — Claimant fills Google Form (Payment Recovery Intake)
- Step 2 — Google Apps Script triggers webhook to gl-intake service
- Step 3 — gl-intake validates fields, creates IntakeSubmission
- Step 4 — IntakeSubmission.to_claim() generates formal Claim
- Step 5 — Claim is persisted, event bus fires intake.submission.converted
- Step 6 — Case management agent picks up claim, begins execution

3.2 Google Apps Script (Webhook Trigger)

```
// Paste this into Google Apps Script for your Form
function onFormSubmit(e) {
  var responses = e.namedValues;
  var payload = {
    full_name: responses["Full Name"][0],
    email: responses["Email Address"][0],
    phone: responses["Phone Number (optional)"][0] || null,
    platform_or_company: responses["Platform or Company Owing You Money"][0],
    estimated_amount_usd: parseFloat(
      responses["Estimated Amount Owed ($)"][0]
    ),
    last_expected_payment: responses["Date of Last Expected Payment"][0],
    platform_reason: responses["What reason did the platform give (if any)?"][0],
    contacted_support: responses["Have you already contacted support?"][0],
    referral_source: responses["How did you hear about GhostLedger?"][0],
    authorization: responses["Authorization"][0] ? true : false
  };

  UrlFetchApp.fetch(
    "https://api.ghostledger.io/v1/intake/submissions",
    {
      method: "post",
      contentType: "application/json",
      headers: { "Authorization": "Bearer " + INTAKE_API_KEY },
      payload: JSON.stringify(payload)
    }
  );
}
```

3.3 Intake API Endpoint

```
POST /v1/intake/submissions
Authorization: Bearer <intake_api_key>

Request:
{
  "full_name": "Jane Creator",
  "email": "jane@example.com",
  "platform_or_company": "Stripe",
  "estimated_amount_usd": 630.00,
  "last_expected_payment": "2025-12-01",
  "platform_reason": "Payout withheld without explanation",
  "contacted_support": "yes_no_resolution",
  "referral_source": "reddit",
  "authorization": true
}

Response (201 Created):
{
  "submission_id": "sub_a1b2c3",
  "claim_id": "clm_d4e5f6",
  "status": "filed",
  "message": "Claim created. Case management will begin within 24 hours."
}
```

3.4 Form Field Mapping

Google Form Field	System Field	Type
Full Name	`full_name`	string (required)
Email Address	`email`	string (required)
Phone Number (optional)	`phone`	string (optional)
Platform or Company Owing You Money	`platform_or_company`	string (required)
Estimated Amount Owed ($)	`estimated_amount_usd`	float (required)
Date of Last Expected Payment	`last_expected_payment`	date (required)
What reason did the platform give?	`platform_reason`	string (required)
Have you already contacted support?	`contacted_support`	enum (required)
How did you hear about GhostLedger?	`referral_source`	enum (optional)
Authorization checkbox	`authorization_granted`	boolean (required)

4. Service Deployment

4.1 Docker Compose (Development)

```
# docker-compose.yml (core services)
version: '3.8'
services:
  gl-api:
    image: ghostledger/api:latest
    ports: ['8080:8080']
    env_file: .env
    depends_on: [gl-db, gl-cache, gl-bus]

  gl-intake:
    image: ghostledger/intake:latest
    ports: ['8081:8081']
    env_file: .env
    depends_on: [gl-db, gl-bus]

  gl-nie:
    image: ghostledger/nie:latest
    ports: ['8082:8082']
    env_file: .env
    depends_on: [gl-db, gl-bus, gl-cache]

  gl-sde:
    image: ghostledger/sde:latest
    ports: ['8083:8083']
    env_file: .env

  gl-governance:
    image: ghostledger/governance:latest
    ports: ['8085:8085']
    env_file: .env

  gl-anchor:
    image: ghostledger/anchor:latest
    ports: ['8086:8086']
    env_file: .env
    volumes:
      - ./keypair.json:/etc/ghostledger/anchor-keypair.json:ro
```

4.2 Environment Variables

Variable	Example	Required
DATABASE_URL	postgresql://gl:pass@gl-db:5432/ghostledger	Yes
REDIS_URL	redis://gl-cache:6379/0	Yes
NATS_URL	nats://gl-bus:4222	Yes
SOLANA_RPC_URL	https://api.mainnet-beta.solana.com	Yes
SOLANA_KEYPAIR_PATH	/etc/ghostledger/anchor-keypair.json	Yes
SOLANA_NETWORK	mainnet-beta	Yes
API_SECRET_KEY	gl_live_sk_...	Yes
INTAKE_WEBHOOK_SECRET	whsec_...	Yes
ANTHROPIC_API_KEY	sk-ant-...	Yes (agents)
MINIO_ENDPOINT	gl-store:9000	Yes
MINIO_ACCESS_KEY	...	Yes
MINIO_SECRET_KEY	...	Yes
LOG_LEVEL	info	No
ANCHOR_BATCH_INTERVAL	60	No (default 60s)

4.3 Startup Order

Services must start in dependency order. The system is not operational until all infrastructure services are healthy.

- Phase 1 — gl-db, gl-cache, gl-bus, gl-store (infrastructure)
- Phase 2 — gl-governance (must be running before any agents)
- Phase 3 — gl-api, gl-intake, gl-nie, gl-sde, gl-litmus, gl-anchor
- Phase 4 — Agents (start after all services are healthy)

Important: The governance service MUST be running before any other application service starts. Without governance, no policy rules are enforced and no decisions can be gated. The system refuses to process actions without an active governance layer.

5. Agent Deployment

GhostLedger runs 27 autonomous agents across 6 categories. Each agent is a long-running process that subscribes to event bus topics, processes data, and publishes results. All agents operate under the governance layer.

Category	Count	Key Agents
LITMUS Evaluation	7	L-Agent, I-Agent, T-Agent, M-Agent, U-Agent, S-Agent, Orchestrator
Noise Intelligence	5	Ingestion, Classifier, Storm Detector, Front Detector, Escalation
Shadow Detection	4	Pattern Scanner, Harm Recorder, Profile Builder, Consent Manager
Case Management	5	Intake Processor, Evidence Collector, Escalation, Resolution, Reporter
Governance	3	Policy Evaluator, Decision Gate, Audit Logger
Distribution	3	Content Generator, Platform Publisher, Analytics

5.1 Agent Configuration

```
# agents.yaml
agents:
  - name: litmus-L-agent
    category: litmus_evaluation
    model: claude-opus-4-6
    input_topics: [litmus.eval.requested]
    output_topics: [litmus.eval.completed]
    max_concurrent: 3
    enabled: true

  - name: intake-processor
    category: case_management
    model: claude-sonnet-4-5
    input_topics: [intake.submission.received]
    output_topics: [intake.submission.converted, case.claim.filed]
    max_concurrent: 5
    enabled: true

  - name: storm-detector
    category: noise_intelligence
    model: claude-sonnet-4-5
    input_topics: [nie.noise.classified]
    output_topics: [nie.storm.detected, nie.storm.updated]
    max_concurrent: 2
    enabled: true
```

5.2 Agent Health Checks

```
# Each agent exposes:
GET /health          -> { "status": "healthy", "uptime": 3600 }
GET /metrics         -> Prometheus-format metrics

# Agent manager checks every 30 seconds
# Unhealthy agents are restarted after 3 consecutive failures
# All restarts are logged to the audit trail
```

6. Database & Storage

6.1 PostgreSQL Schema

The primary data store holds all system state. Tables map directly to the type definitions in ghostledger_types.py.

```
-- Core tables
CREATE TABLE intake_submissions (...);
CREATE TABLE claims (...);
CREATE TABLE noise_events (...);
CREATE TABLE storms (...);
CREATE TABLE fronts (...);
CREATE TABLE harm_records (...);
CREATE TABLE victim_shadow_profiles (...);
CREATE TABLE consent_unlocks (...);
CREATE TABLE litmus_evaluations (...);
CREATE TABLE evidence_refs (...);
CREATE TABLE policy_rules (...);
CREATE TABLE decisions (...);
CREATE TABLE execution_records (...);
CREATE TABLE chain_anchors (...);
CREATE TABLE hrn_members (...);
CREATE TABLE hrn_engagements (...);
CREATE TABLE audit_log (...);  -- append-only

-- Indexes
CREATE INDEX idx_claims_status ON claims(status);
CREATE INDEX idx_claims_respondent ON claims(respondent_entity);
CREATE INDEX idx_noise_events_storm ON noise_events(storm_id);
CREATE INDEX idx_storms_status ON storms(status);
CREATE INDEX idx_audit_log_ts ON audit_log(timestamp);
```

6.2 Redis Usage

Key Pattern	Purpose	TTL
`storm:{id}:score`	Cached storm score	60s
`rate:{api_key}`	Rate limit counter	60s

`eval:{id}:status`	LITMUS eval progress	10min
`agent:{id}:heartbeat`	Agent liveness	60s
`session:{token}`	JWT session cache	1hr

6.3 Object Storage (MinIO)

Evidence files (screenshots, documents, exports) are stored in MinIO with SHA-256 content hashing. Every file stored gets a hash that can be anchored on-chain.

```
Buckets:
  evidence/       -> claim attachments, screenshots
  exports/        -> court-ready evidence packets
  backups/        -> daily database dumps
```

7. Monitoring & Alerting

7.1 Health Endpoints

```
# Every service exposes:
GET /health     -> 200 OK or 503 Service Unavailable
GET /ready      -> 200 OK when ready to accept traffic
GET /metrics    -> Prometheus metrics
```

7.2 Key Metrics

Metric	Alert Threshold	Action
API latency p99	> 2000ms for 5min	Scale API pods
Error rate	> 1% for 5min	Page on-call
Event bus lag	> 1000 pending msgs	Scale consumers
Agent heartbeat	Missing 3 consecutive	Auto-restart agent
DB connections	> 80% pool utilization	Scale read replicas
Anchor queue depth	> 100 pending anchors	Check Solana RPC
Claim SLA breach	> 24hr no action	Escalate to HRN
Consent expiry	< 7 days remaining	Notify consent manager

7.3 Dashboard Layout

- System Health — Service status, uptime, error rates
- Intake Funnel — Submissions/hour, conversion rate, pending claims
- Noise Weather — Active storms, fronts, severity heatmap
- Case Pipeline — Claims by status, avg resolution time, amount recovered
- Agent Fleet — 27 agents: health, throughput, error rate per agent
- Chain Anchor — Anchoring rate, Solana balance, pending queue

8. Security & Access Control

8.1 API Key Management

Every API key is scoped to specific permissions. Keys are stored hashed (SHA-256) in the database. Plaintext keys are shown once at creation and never again. Keys can be rotated without downtime.

```
# Key format
gl_live_sk_<32 random chars>     # production
gl_test_sk_<32 random chars>     # staging

# Rotation: create new key, update services, revoke old key
POST /v1/auth/keys { scopes: [...] }  -> new key
DELETE /v1/auth/keys/{key_id}          -> revoke
```

8.2 Encryption

- In transit — TLS 1.3 on all external endpoints
- At rest — AES-256 for database encryption, MinIO server-side encryption
- PII fields — Claimant name, email, phone encrypted at application layer
- Evidence hashes — SHA-256, computed before upload, verified on retrieval

8.3 Network Isolation

Only the API gateway and intake webhook are publicly accessible. All other services run on a private network. The Solana anchor service has outbound-only access to Solana RPC endpoints.

9. Incident Response

Agent stops responding

Check /health endpoint. If 503, check logs for OOM or crash. Restart agent via POST /v1/agents/{id}/restart. If persistent, reduce max_concurrent and investigate.

Solana anchor fails

Check wallet balance (solana balance). Check RPC endpoint status. If RPC is down, switch to backup RPC. Pending anchors queue in NATS and will process when connection restores.

Intake webhook returns 500

Check gl-intake logs. Common causes: database connection timeout, invalid form field format. Submissions are retried by Google Apps Script up to 3 times.

Governance service down

CRITICAL. All agents freeze — no actions are processed without governance. Restart immediately. Check database connectivity. If extended outage, agents buffer events in NATS JetStream (durable).

Storm score spike (false positive)

Check entity list and source events. If false positive, mark storm as resolved via PATCH /v1/storms/{id}. Review classifier thresholds.

Consent expires during active case

Case management freezes all actions on the affected claim. Consent manager agent sends re-consent request. If no response in 7 days, case is paused and HRN member is notified.

Database approaching storage limit

Archive resolved claims older than 2 years to cold storage. Purge audit log entries older than 5 years. Run VACUUM FULL.

10. Backup & Recovery

10.1 Backup Schedule

Component	Frequency	Retention	Method
PostgreSQL	Every 6 hours	30 days	pg_dump to MinIO backups/
Redis	Every 1 hour	7 days	RDB snapshot
Evidence (MinIO)	Continuous	Indefinite	Cross-region replication
Agent configs	On change	90 days	Git versioned
Policy rules	On change	Indefinite	DB + chain anchor
Audit log	Daily archive	Indefinite	Append-only, chain-anchored

10.2 Recovery Procedure

- Step 1 — Stop all application services (agents first, then services)
- Step 2 — Restore PostgreSQL from latest pg_dump
- Step 3 — Verify chain anchors match on-chain records (integrity check)
- Step 4 — Restore Redis from RDB snapshot
- Step 5 — Start services in dependency order (Section 4.3)
- Step 6 — NATS JetStream replays any unprocessed events from durable subscriptions
- Step 7 — Verify all health endpoints return 200

11. Scaling Playbook

GhostLedger scales horizontally at the service and agent level. The event bus (NATS) and database handle the coordination.

Trigger	Action	Limit
API latency > 1s	Add gl-api replicas	10 replicas
Intake queue > 50	Add gl-intake replicas	5 replicas
NIE event lag > 500	Add classifier agents	8 agents
LITMUS eval backlog	Increase max_concurrent	10 per letter-agent
DB CPU > 70%	Add read replicas	3 replicas
Redis memory > 80%	Scale vertically or cluster	Cluster mode
Evidence storage > 80%	Expand MinIO volumes	No hard limit

12. Operational Checklist

12.1 Pre-Launch

- ☐ All environment variables set and validated
- ☐ Database migrations applied, indexes created
- ☐ Solana wallet funded (minimum 0.5 SOL for mainnet)
- ☐ Google Apps Script webhook configured and tested
- ☐ API keys generated for all services and agents
- ☐ Policy rules loaded (7 Laws enforcement rules)
- ☐ SSL certificates provisioned for api.ghostledger.io
- ☐ Monitoring dashboards configured with alert thresholds
- ☐ Backup schedule active and first backup verified
- ☐ Test intake submission processed end-to-end

12.2 Daily Operations

- ☐ Check all service health endpoints (automated)
- ☐ Review agent fleet status — 27/27 healthy
- ☐ Check intake queue depth — all submissions processed within 1 hour
- ☐ Review overnight storm activity — any new critical storms
- ☐ Verify Solana anchor service — wallet balance above 0.1 SOL
- ☐ Check audit log for any policy violations

12.3 Weekly Operations

- ☐ Review claim pipeline — any claims stuck > 7 days without action
- ☐ Audit API key usage — revoke unused keys
- ☐ Database performance review — slow queries, connection pool
- ☐ Storage utilization check — evidence bucket growth rate

- ☐ Agent metrics review — error rates, latency trends
- ☐ HRN engagement status — any briefs unaccepted > 7 days

12.4 Monthly Operations

- ☐ Rotate API keys and webhook secrets
- ☐ Review and update policy rules based on new patterns
- ☐ Archive resolved claims older than retention period
- ☐ Solana wallet top-up if balance below 0.5 SOL
- ☐ Disaster recovery drill — restore from backup
- ☐ Review agent model versions — update if new models available
- ☐ Security audit — review access logs, failed auth attempts

---

"Order from within. The system runs because every piece knows its place."