HOOX · DOCUMENTATION

INSTITUTIONS · FUNDS · PLATFORM TEAMS

ENTERPRISE MANUAL

Multi-tenancy, audit, compliance, and institutional scale

2026-07-24 DIN A4 · HTTPS://HOOX.SH/DOCS OPEN SOURCE

CONTENTS

01 · ENTERPRISE MANUAL

HOOX ENTERPRISE

Documentation hub for the institutional / Enterprise layer of HOOX on Cloudflare.

> **This is the commercial / Enterprise extension layer.**
>
> HOOX follows an **Open Core** model. The core architecture, most code, and documentation are open source. This section describes the **commercial Enterprise layer** built on top of the open core.
>
> See:

- OPEN_CORE.md
- OPEN_CORE_FEATURE_SPLIT.md

This section contains **public architecture documentation** for the institutional / paid Enterprise version of HOOX (the commercial extension layer built on the open core). It is designed to run on **Cloudflare Enterprise**.

Note: No proprietary Enterprise source code is included in this public repository.

Documents

- Architecture — Overall topology, bleeding-edge primitives, and evolution from retail HOOX.
- Multi-Tenancy — Workers for Platforms, Dispatch + User Workers, isolation model.
- Observability & Audit — Logpush, Traces, Tail Workers, immutable R2 audit.
- Security & Compliance — Extended 5-layer model with Bot Management, API Shield, Zero Trust, AI Guardrails.

Key Differentiators (vs Retail HOOX)

- Workers for Platforms for true multi-tenancy / platform use
- Cloudflare Workflows for durable, long-running trade & compliance processes
- Full Logpush + R2 for regulatory-grade audit
- Enterprise security features (Bot Management, API Shield, etc.)
- Real-time persistent connections via hibernatable WebSockets in DOs
- Higher limits, custom SLAs, AI Gateway at scale
- Data residency and advanced compliance controls

See the approved architecture plan (internal `plan.md`) for phasing and rationale.

All designs build directly on the existing high-quality retail codebase (Service Bindings, DOs, Smart Placement, shared middleware, etc.).

02 · ENTERPRISE MANUAL

HOOX ENTERPRISE ARCHITECTURE

Bleeding-edge architecture for HOOX on Cloudflare Enterprise, leveraging Workers for Platforms, Workflows, Logpush, AI Gateway, advanced security, and more for institutional-grade algorithmic trading.

HOOX ENTERPRISE ARCHITECTURE

> **Important – Open Core Model**
>
> This document describes the **commercial Enterprise extension layer**.
>
> HOOX follows an **Open Core** model:
>
> - The core architecture, most workers, `packages/shared`, CLI, TUI, and the majority of documentation are **open source**.
> - Advanced multi-tenancy, proprietary features, full compliance stacks, hosted SaaS platform, and certain advanced implementations live in a **separate closed-source commercial repository**.
>
> See:
>
> - OPEN_CORE.md
> - OPEN_CORE_FEATURE_SPLIT.md
>
> The documents in this folder describe **how the commercial layer is built on top of the open core**. They are published publicly for transparency and to support the academic paper.

Status: Proposed / In Design
Target: Cloudflare Enterprise accounts (or hosted SaaS on Enterprise)
Goal: Institutional-grade, multi-tenant, highly observable, durable, secure, real-time edge-native trading platform built as an extension on the open core.

This document expands the approved plan into concrete architecture for the commercial layer, leveraging the full power of Cloudflare Enterprise and 2025-2026 bleeding-edge primitives.

Vision & Principles

- **Everything at the Edge, Amplified**: Preserve the current zero-server, Service Binding, Smart Placement, DO-based model. Enterprise unlocks scale, isolation, compliance, and durability without introducing traditional infra.
- **Multi-Tenancy First**: Support hosted SaaS (funds, prop shops, platforms) or dedicated high-volume Enterprise deployments with strong isolation.
- **Durable + Event-Driven**: Move from cron-heavy to Workflows + real-time WS-driven.
- **Audit & Compliance by Default**: Every action logged immutably.
- **Bleeding Edge but Practical**: Use features that are GA or recently stabilized (WfP dashboard improvements, synchronous User Worker deploys, Workflows enhancements, Logpush one-click + custom fields, etc.).
- **Evolutionary**: Existing single-tenant HOOX remains for retail/self-hosted. Enterprise is additive/parallel.

Current HOOX (retail PoC) already uses advanced primitives (Service Bindings ~0.4ms, SQLite DOs for idempotency, Smart Placement, Queues, D1/R2/Vectorize, Browser Rendering). Enterprise takes this to the next level for 10x-100x volume, regulatory needs, and platform offerings.

Core Levers (Enterprise / Paid Unlocks)

1. **Workers for Platforms (WfP)**
   - Dynamic Dispatch Worker routes to isolated "User Workers" (per tenant, per strategy, per fund).
   - Namespaces, tags for metering/billing/isolation.
   - User Workers can have their own bindings, DOs, Queues (isolated).
   - Recent improvements: Dashboard support for namespaces/dispatch/templates/tags (Dec 2025), static assets support, synchronous first-time uploads (Feb 2025) so deploys are immediately routable.
   - Perfect for "bring your own strategy" or multi-fund platforms.
   - Enterprise: higher scale, custom limits, better support for customer code execution.
2. **Workflows (Durable Execution)**
   - Long-running (hours/days), stateful, automatically retrying, pause-for-external-events.
   - Step-level persistence, input/output gates.
   - Integrates with Agents (real-time) and existing DOs.
   - Use cases:
     - Full trade lifecycle (validate → risk → execute → reconcile → report).
     - Kill switch + position flattening sequences that must complete atomically.
     - Daily compliance/report generation with human approval gates.
     - Post-trade reconciliation across exchanges/D1/R2.
   - Recent enhancements: step context (retry count), dynamic workflows, AgentWorkflow integration.
   - Enterprise: higher execution volume, longer durations, more steps.
3. **Observability & Audit (Logpush + Traces + Observability suite)**
   - Full Workers Trace Events, HTTP, WAF, Queue, DO, etc. pushed to R2 (or external SIEM like SentinelOne).
   - One-click R2 setup, custom fields (raw/transformed headers), daily partitions.
   - Workers Logs (dashboard), Real-time logs, Tail Workers for sampling/filtering/enrichment.
   - End-to-end traces across Service Bindings, DOs, Queues, Workflows.
   - Analytics Engine remains for high-cardinality metrics; Logpush for compliance/audit.
   - Enterprise: higher log volume, longer retention, dedicated support, integration options.
4. **Enterprise Security Stack**
   - **Bot Management** (Enterprise): bot scores, verified bots, corporate proxy detection, JA4/JA3, JS detection. Protect signal sources (TradingView webhooks, Telegram, email).
   - **API Shield**: schema validation, JWT validation, mTLS for clients (dashboard or partner APIs), sequence mitigation.
   - **Advanced WAF + Rulesets**: 100+ rate limit rules, custom high-cardinality rules, managed rules for trading threats.
   - **Zero Trust / Access**: Enforce identity for dashboard, CLI mgmt, internal tools. SCIM, device posture, etc.

- **mTLS / Mutual TLS**: Between services or for outbound.
- **Data Loss Prevention / Gateway** if needed for sensitive configs.
- Existing 5-layer model (WAF → Gateway auth → Service Bindings → Internal key → DO mutex) is extended with Ent controls.

5. **Real-time & Persistent Connections**
   - Hibernatable WebSockets in Durable Objects (keep connections alive cheaply while DO sleeps).
   - Evolve `ExchangeConnectionManager` DO to maintain persistent WS to Binance/Bybit/etc. for live fills, partials, order books.
   - Drive risk management, notifications, and analytics from real events instead of (or alongside) 5-min cron.
   - Combined with Workflows for durable handling of streams.
6. **Scale, Limits & Performance (Enterprise Unlocks)**
   - Significantly higher CPU time, memory, subrequests, concurrent connections, Worker size, # of Workers/crons per account.
   - Custom limit increases via account team.
   - Higher D1 storage/reads/writes, R2 operations, Queue throughput, Vectorize dimensions/index size.
   - DO: higher concurrency per namespace, more DOs, better SQLite performance at scale.
   - Smart Placement + Enterprise routing hints for even lower jitter to exchange regions.
   - Hyperdrive for accelerating any external data sources (if hybrid Postgres/etc. used for large historical data).
7. **AI at Institutional Scale**
   - **AI Gateway**: logging, caching, rate limiting, fallbacks, cost control, prompt sanitization in front of Workers AI or external providers (OpenAI, etc.).
   - Workers AI + Vectorize at Enterprise volumes (large strategy corpus, market embeddings, RAG for compliance docs or historical trades).
   - Integration with Workflows/Agents for sophisticated risk agents, anomaly detection, natural language strategy config.
   - Existing simple RAG in telegram-worker becomes production-grade with Gateway + fallbacks.
8. **Storage & Data Sovereignty**
   - R2: Event Notifications (trigger Workflows on new trade logs/reports), jurisdiction controls (data residency), larger scale, lifecycle policies for immutable audit logs.
   - D1: Enterprise storage/throughput, point-in-time recovery, read replicas (if/when available), larger DBs per tenant.
   - KV + DO SQLite for low-latency config/state.
   - Immutable audit: Logpush + R2 WORM-like + cryptographic hashes/signatures in D1.
9. **Other Bleeding Edge**
   - Browser Rendering at scale (more concurrent renders for compliance reports).
   - Builds (CI/CD for User Workers in WfP).
   - Email Workers / advanced routing for signal ingestion.
   - Magic Transit / Spectrum if private connectivity to exchanges or on-prem required (rare for CEX but useful for some prop shops).
   - Queues + Workflows for reliable backpressure and exactly-once-ish semantics at high volume.

High-Level Target Topology (Enterprise)

```
Public Ingress (WAF + Bot Mgmt + API Shield)
          │
          ▼
Dispatch Worker (Workers for Platforms)  ← Tenant/Strategy routing + auth (Zero Trust)
          │
     ┌────┴─────┐
     ▼          ▼
User Workers / System Workers (isolated per tenant/strategy)
     │
     ├──► Service Bindings (still <1µs, now tenant-scoped)
     │
     ├──► Durable Objects (per-tenant DO namespaces)
     │     - Idempotency (SQLite)
     │     - ExchangeConnectionManager (Hibernatable WS for live data)
     │     - Risk state / position managers
     │
     ├──► Workflows (durable lifecycles)
     │     - Trade execution + reconciliation
     │     - Kill-switch sequences
     │     - Daily compliance reports
     │
     ├──► Queues (higher throughput, with Workflows consumers)
     │
     └──► Storage (tenant-partitioned)
           - D1 (per-tenant or sharded)
           - R2 (immutable logs + jurisdiction)
           - Vectorize (filtered by tenant)
           - Analytics Engine (tagged)

Observability Layer (always on)
- Workers Traces (cross everything)
- Logpush → R2 (full audit, custom fields)
- Tail Workers (sampling, enrichment)
- AI Gateway (for any AI calls)
- Dashboard (Zero Trust protected, multi-tenant views)
```

Existing workers (trade-worker, agent-worker, hoox, etc.) become templates or are refactored into dispatchable components + system services.

Key New / Evolved Components

1. Dispatch + User Workers (WfP)

- Central Dispatch Worker inspects request (tenant id from subdomain, header, or JWT), looks up namespace, dispatches to User Worker.
- Each User Worker can contain strategy-specific logic or a full lightweight HOOX instance.
- Static assets support for per-tenant dashboards/UI if needed.
- Tagging for billing/quotas.

2. Workflows Definitions

Examples:

- `TradeLifecycleWorkflow`: validate → pre-risk → execute (with retries) → persist → post-risk → notify → reconcile.
- `ReconciliationWorkflow`: runs daily, compares D1 vs exchange reports (via Browser Rendering or API), flags discrepancies.
- `ComplianceReportWorkflow`: gathers logs (from R2), generates PDF (Browser), signs, uploads, notifies.

Workflows can call existing services via Service Bindings and pause for external events (e.g., manual approval).

3. Enhanced DOs

- `IdempotencyStore` generalized with tenant/strategy prefix.
- `ExchangeConnectionManager` uses `hibernatableWebSockets` + alarms for cleanup.
- New DOs for per-tenant risk engines, session state, etc.
- RPC methods + SQLite for best DX/performance.

4. Observability Pipeline

- Every Worker/Workflow/DO emits structured events.
- Logpush jobs (one-click or API) to R2 with `output_options` for relevant fields.
- Tail Worker example for redacting secrets or adding tenant context.
- Full traces for root-causing a missed fill across 5+ Workers.

5. Security Hardening

- Bot score checks early in gateway/dispatch.
- API Shield schemas for all public + internal-ish endpoints.
- Zero Trust policies for dashboard + mgmt APIs.
- Per-tenant secrets (using Workers secrets or Secret Store at scale).
- mTLS where Services talk to trusted partners.

Tenant Isolation Model

- **Namespace per tenant** (or per "fund" / "book").
- DOs, Queues, KV, R2 paths, D1 databases prefixed or bound per namespace.
- User Workers run in isolated isolates with their own limits.
- Billing/quotas via tags + custom dashboards (Enterprise).
- Cross-tenant leakage prevented at Dispatch + binding auth layers (evolved `requireInternalAuth` with tenant claims).

Migration & Coexistence

- Retail HOOX stays as-is (free/paid friendly).
- Enterprise customers get:
  - Dedicated Enterprise CF account + templates, or
  - Onboarded into hosted WfP platform.
- Existing code in `packages/shared`, workers, etc. is reused as libraries/templates.

- Gradual: start with WfP + Logpush + one Workflow, then add real-time WS, Bot Mgmt, etc.

Open Decisions (from plan review)

- Primary target: Hosted SaaS (heavy WfP) vs Power-user self-hosted on Ent account vs both?
- Migration strategy for existing users?
- Specific compliance jurisdictions / external SIEM requirements?
- Depth of "user code" execution (full arbitrary strategies vs safe DSL / config-driven)?

Next Steps / Implementation Order

1. Create `docs/devops/enterprise/` docs (this + security, multi-tenancy, observability).
2. Prototype Dispatch Worker + minimal User Worker + one Workflow (trade-lifecycle).
3. Extend `@jango-blockchained/hoox-shared` with tenant context, Workflow client, AI Gateway wrapper.
4. Add Logpush config examples + a Tail Worker.
5. Update root architecture docs, ADRs, and papers (new section or companion Enterprise paper).
6. wrangler templates in `examples/enterprise/`.
7. (Later) Full implementation, tests, `bun run graph` validation, academic paper updates.

This architecture keeps HOOX's unique edge-native soul while making it suitable for serious institutional use with the full power of Cloudflare Enterprise.

References (use latest docs):

- Workers for Platforms
- Workflows
- Logpush / Observability
- AI Gateway
- Durable Objects (SQLite, WS, Alarms, RPC)
- Enterprise limits & features (higher everything + dedicated support)
- Zero Trust, Bot Management, API Shield

See the approved plan.md for the detailed "why" and phasing.

03 · ENTERPRISE MANUAL

MULTI-TENANCY WITH WORKERS FOR PLATFORMS

How HOOX Enterprise achieves strong multi-tenant isolation using Cloudflare Workers for Platforms, namespaces, dispatch, and per-tenant Durable Objects / bindings.

MULTI-TENANCY WITH WORKERS FOR PLATFORMS

> **This document describes commercial Enterprise capabilities.** Basic Workers for Platforms concepts and examples are part of the open core. Full production multi-tenant SaaS platform, billing, and advanced isolation live in the closed-source Enterprise layer.
>
> See OPEN_CORE_FEATURE_SPLIT.md.

HOOX Enterprise uses **Workers for Platforms (WfP)** as the foundation for multi-tenancy in the commercial offering.

Why WfP?

- Isolated "User Workers" per tenant / fund / strategy book.
- Centralized **Dispatch Worker** performs routing + auth, then dispatches.
- Each User Worker gets its own (or shared-but-namespaced) bindings: DO namespaces, Queues, D1 (per-tenant or sharded), R2 paths, etc.
- Native support for tags (for metering/billing), namespaces, and (as of late 2025) dashboard management.
- Synchronous first deploy + static assets support.
- Perfect for "platform" or "hosted" HOOX where customers can even bring limited custom strategy logic.

Architecture

```
Incoming Request (subdomain, header, or JWT tenant claim)
          |
          ▼
Dispatch Worker (Enterprise account)
  - Resolve tenant from claim / subdomain / tag
  - Enforce Zero Trust / Bot Mgmt / API Shield
  - Lookup namespace + script
  - `env.TENANT_DISPATCHER.dispatch(..., { tenantId, ... })`
          |
          ▼
Tenant / Strategy "User Worker"
  - Runs in strong isolation
  - Own (or prefixed) DOs, Queues, secrets
  - Can still call system services via Service Bindings (with tenant-scoped auth)
          |
    Tenant-specific logic or full lightweight HOOX mesh
```

Dispatch Worker Pattern

```
// workers/dispatch/src/index.ts (Enterprise only)

  async fetch(request: Request, env: Env) {
    const tenantId = extractTenant(request); // subdomain, header, or JWT
    if (!tenantId) return new Response("Missing tenant", { status: 400 });

    // Optional: per-tenant rate limiting, bot score check, etc.
    const namespace = env.NAMESPACES.get(env.NAMESPACES.idFromName(tenantId));

    return namespace.dispatch(request, {
      // pass tenant context
      tenantId,
      // forward internal auth with tenant claim
    });
  }
};
```

User Workers are uploaded under the tenant's namespace/script tag.

Per-Tenant Isolation

- **Durable Objects**: Use `idFromName('tenant:' + tenantId + ':idempotency')` or dedicated namespaces per tenant.
- **Queues**: One queue per tenant or heavily tagged.
- **D1**: One database per tenant (recommended for isolation + billing) or row-level with tenant_id + RLS-like checks.
- **R2**: Keys prefixed `tenants/${tenantId}/...` + separate buckets for strict residency.
- **KV**: Per-tenant or namespaced.
- **Secrets**: Per-script or using Secret Store with tenant scoping.
- **Vectorize**: Metadata filtering by tenant or separate indexes.

Existing `requireInternalAuth` is extended with tenant claims:

```
// in @hoox-shared

  const err = requireInternalAuth(req, env);
  if (err) return err;
  const claim = req.headers.get("X-Tenant-Id");
  if (claim !== tenantId) return new Response("Tenant mismatch", { status: 403 });
}
```

Workers for Platforms Specifics (2025-2026)

- Dashboard support for creating namespaces, dispatch templates, tags.
- Static assets can be attached directly to User Workers.
- First-time User Worker uploads are synchronous (200 means ready to dispatch).
- Tags for cost attribution and quota enforcement (Enterprise advantage).

Hosted SaaS vs Dedicated Enterprise

- **Hosted (SaaS)**: One Enterprise account runs the Dispatch + system workers. Customers get User Workers in isolated namespaces. Billing via tags + external system.
- **Dedicated (Self-hosted on Ent account)**: Customer gets their own Enterprise account + our templates + CLI that sets up WfP (or just higher limits + Logpush + Workflows without full multi-tenancy).

Both modes share the same core architecture.

Limits & Scaling Notes

Enterprise customers get:

- Much higher numbers of scripts / namespaces.
- Higher request volume and concurrency.
- Custom limit increases.
- Better support for customer-provided code (security review processes).

Related

- See architecture.mdx for overall topology.
- observability-audit.mdx for per-tenant log isolation.
- Existing `wrangler.jsonc` patterns will be extended with `dispatch` and namespace bindings in Enterprise templates.

04 · ENTERPRISE MANUAL

ENTERPRISE SECURITY & COMPLIANCE

Extending the 5-layer model with Bot Management, API Shield, Zero Trust, mTLS, AI Guardrails, and immutable audit for HOOX Enterprise.

ENTERPRISE SECURITY & COMPLIANCE

> **Open Core vs Enterprise** The base 5-layer security model and many patterns are part of the open core. Advanced Enterprise-only security features (full Bot Management, API Shield at scale, SCIM, dedicated mTLS setups, etc.) are described in this commercial layer document.

HOOX's existing 5-layer security model is excellent. Enterprise adds several powerful layers that are either only available or significantly stronger on paid/Enterprise plans.

Extended Layers

Layer 0: Bot Management (Enterprise)

- Scores every request (`cf.botManagement.score`).
- Detects verified bots, corporate proxies, automated tooling.
- JA3/JA4 fingerprinting, JS challenge results.
- Early drop of suspicious TradingView/Telegram/email signal sources.
- Combined with WAF custom rules and rate limiting.

Example in Dispatch / Gateway:

```
const bot = request.cf?.botManagement;
if (bot && !bot.verifiedBot && bot.score < 30) {
  return new Response("Blocked", { status: 403 });
}
```

Layer 1: Advanced WAF + API Shield

- Enterprise WAF supports 100 rate limiting rules, complex expressions, high cardinality.
- **API Shield**:
  - JSON Schema validation on webhook and internal-ish endpoints.
  - JWT validation.
  - mTLS (client certificates) for dashboard or partner integrations.
  - Sequence / anomaly detection.

Layer 2: Zero Trust / Access

- Dashboard and management APIs protected by Cloudflare Access (SSO, device posture, WARP, etc.).
- Replaces or augments custom passkeys for human operators.
- SCIM for team provisioning in larger organizations.

- Gateway policies for outbound control if needed.

Layer 3: Internal + mTLS

- Existing `INTERNAL_KEY_BINDING` + `requireInternalAuth` remains.
- Add mTLS between Workers where possible (or use Service Binding + strong claims).
- Per-tenant claims propagated in auth headers.

Layer 4: Durable Object + Cryptographic Controls

- SQLite DOs for tamper-evident logs (append-only tables with hashes).
- Signatures on important events (fills, kill switches, config changes).
- Stored in D1 + R2 (dual write for durability).

Layer 5: AI Guardrails (new)

- **AI Gateway** in front of all LLM calls (risk agents, summaries, strategy parsing).
  - Prompt injection detection / sanitization.
  - Logging + caching + rate limits + fallbacks.
  - Cost control and audit of every AI decision.
- Existing prompt-sanitizer in agent-worker is promoted and expanded.

Immutable Audit (Compliance Core)

Every security-relevant event flows to Logpush → R2 (see observability-audit.mdx).

Key events:

- All webhook / signal ingress (with bot score, WAF decision)
- Auth decisions (success + failure)
- Idempotency DO outcomes
- Order placement + exchange responses (full)
- Risk actions (trailing stop, kill switch)
- Config / secret changes (via audit hooks)
- Workflow step outcomes

Retention: typically 7 years for financial services (configurable via R2 lifecycle).

Data Residency & Sovereignty

- R2 jurisdiction controls (e.g., EU, US, APAC only buckets).
- DO location hints + Worker placement for sensitive tenants.
- D1 database location choices.
- Optional: separate Enterprise accounts per major jurisdiction.

Kill Switch & Fail-Closed Enhancements

- Multiple kill switch sources (manual via Zero Trust dashboard, API with mTLS, automated via Workflow or AI anomaly).
- Every activation produces a signed, immutable record.
- Position flattening Workflow must run to completion (durable).

Secrets & Key Management

- Use Cloudflare Secret Store / per-script secrets.
- Rotate via CLI + audit event.
- For very high security: integrate with external KMS via Hyperdrive or mTLS if needed (rare).

Threat Model Updates (Enterprise)

See original threat model (papers/appendices/C-threat-model.tex and docs).

New / amplified threats addressed:

- Malicious or compromised tenant code (mitigated by WfP isolation + review gates).
- Supply-chain attacks on customer strategies (AI Gateway + sandboxing patterns).
- Regulatory "prove you didn't front-run" → full immutable trace + signatures.
- Nation-state / sophisticated bot attacks on signal sources → Bot Management + WAF.

Implementation Notes

- Bot Management and advanced WAF rules are configured at the zone/account level (Enterprise).
- API Shield schemas live with the Dispatch Worker and public ingress points.
- Zero Trust policies defined for `dashboard`, `cli` (via API tokens with context), and internal tools.
- All new Enterprise components must emit audit events using the shared `trackAnalytics` / audit path.

Related Documents

- Root security overview: `docs/devops/security/overview.mdx`
- Architecture: `docs/devops/enterprise/architecture.mdx`
- Multi-tenancy isolation guarantees: `multi-tenancy.mdx`
- Original 5-layer model in the academic paper.

05 · ENTERPRISE MANUAL

ENTERPRISE OBSERVABILITY & AUDIT

Full auditability using Logpush, Workers Traces, Tail Workers, R2 immutable storage, and integration with Workflows for compliance in HOOX Enterprise.

ENTERPRISE OBSERVABILITY & AUDIT

> **Note:** Core observability (Analytics Engine, basic traces, Logpush examples) is available in the open core. The full production-grade compliance, SIEM integration, long-term immutable audit, and tenant-aware pipelines described here are part of the commercial Enterprise layer.

One of the biggest jumps from retail HOOX to Enterprise is moving from "some analytics" to **regulator-grade, immutable, queryable audit trail**.

Pillars

1. **Workers Traces** (automatic)
   - End-to-end visibility across Service Bindings, DO RPC, Queues, Workflows, external fetches.
   - No code changes needed for basic tracing.
2. **Logpush** (Enterprise strength)
   - Push Workers Trace Events, HTTP requests, WAF events, Queue events, etc.
   - Destination: R2 (primary), or external SIEM (SentinelOne, Splunk, etc.).
   - One-click R2 setup + advanced options (custom fields, raw vs transformed headers, daily partitioning).
   - High volume support on Enterprise.
3. **Tail Workers**
   - Sample, filter, enrich, or redact logs in real time before they go to Logpush or storage.
   - Example: add tenant context, hash sensitive fields, drop health checks.
4. **R2 as Immutable Audit Store**
   - Combined with R2 Event Notifications → trigger Workflows for post-processing (indexing, compliance checks, retention policies).
   - Jurisdiction controls for data residency.
   - Lifecycle + Object Lock for immutability.
5. **Analytics Engine + Dashboard**
   - Keep high-cardinality real-time metrics (latency, fills, risk events).
   - Tenant-tagged datasets.
6. **Workflows + Observability**
   - Every Workflow step can emit structured events.
   - Workflow execution history is durable and auditable.

Example Logpush Job (R2)

```
{
  "name": "hoox-enterprise-audit",
  "dataset": "workers_trace_events",
  "destination_conf": "r2://hoox-audit-logs/{DATE}?account-id=...&access-key-id=...&secret-access
  "output_options": {
    "field_names": ["EventTimestamp", "EventType", "RayID", "Worker", "Outcome", "TenantId", ...]
    "timestamp_format": "rfc3339"
  },
  "enabled": true
}
```

Use the one-click flow or the API for automation in CLI/deploy.

Tail Worker Example (Tenant Enrichment + Redaction)

```
// workers/tail-audit/src/index.ts

  async tail(events: TraceItem[], env: Env) {
    for (const event of events) {
      // enrich
      const tenant = event.scriptTags?.find(t => t.startsWith("tenant:"))?.slice(7) || "unknown";

      // redact secrets
      const logs = (event.logs || []).map(l => ({
        ...l,
        message: l.message?.replace(/secret:[a-z0-9]+/gi, "secret:[REDACTED]")
      }));

      await env.AUDIT_QUEUE.send({
        tenant,
        ray: event.rayID,
        logs,
        // ...
      });
    }
  }
};
```

Compliance Use Cases

- **Trade reconstruction**: Pull all events for a `clientOrderId` or `traceId` across Workers, DOs, and Queues from R2.
- **Kill switch audit**: Every activation + reason + affected positions logged immutably.
- **Regulatory reports**: Scheduled Workflow reads from R2 + D1, produces signed PDF via Browser Rendering, stores result + manifest in R2.
- **Anomaly detection**: Feed Logpush stream (or sampled Tail) into AI Gateway + Vectorize for real-time risk signals.

Retention & Cost

- R2 is dramatically cheaper than traditional logging at scale.
- Use lifecycle rules: hot (30d) → cold → delete or archive after N years.
- Enterprise customers often keep 7+ years for compliance.

Integration Points

- Existing `trackAnalytics` in shared can be extended to also emit to the audit path.
- All new Workflows and DOs should emit structured `auditEvent` payloads.
- Dashboard (Enterprise) should have "Audit Explorer" views querying R2 (via Worker or R2 SQL if available) + Analytics Engine.

Related

- architecture.mdx
- multi-tenancy.mdx — tenant tagging in logs
- Root `wrangler.jsonc` Enterprise profiles will include Logpush + Tail bindings.