Project Brief/Economics

Deterrence Through Non-Repudiation
Once actions are anchored immutably, any tampering becomes both detectable and provable.

- Admins know they can't edit logs without detection, even if they control the main servers.
- It's not just forensic—it's preventative by design.

High Level Architecture:

Server A (Source)

Server B (Publisher)

Server C (Overwatch over A and B)

1. **Hashing Logs**: **Server A** hashes its logs periodically, creating a Merkle root representing the log data - sends to C every 5min and B every 1hr.
2. **Super Merkle Tree**: **Server B** collects these individual roots and aggregates them into a **super Merkle tree**, maintaining traceability across all servers.
3. **Blockchain Anchoring**: The root of the super Merkle tree is anchored immutably to **Polkadot's Asset Hub**, ensuring it cannot be altered or erased without being provable.
4. **Verification**: An independent verifier (**Server C**) is watching: Any tampering with the logs can be detected by verifying the Merkle root against the blockchain. If the root doesn't match, it's an instant red flag. **Server C** does not need to rebuild the entire **super Merkle tree** because of the **hashing properties** of Merkle trees. It only needs to verify that the **Merkle root** it receives from **Server A** is **consistent with the super Merkle root** stored on the blockchain. This process relies on **mathematical consistency** between the roots (the **super Merkle root** and individual **Merkle roots**), leveraging the **immutable** and **verifiable nature of the blockchain**. Think of Server C as the local truth engine. It doesn't trust Server B — it uses it *as a timestamp oracle*, not a source of truth. If log tampering happens on a Server A (client-side agent), Server C will still detect a mismatch in the Merkle chain. If tampering is suspected **after anchoring**, Server C can walk down its stored Merkle path and **validate that the root hash matches the one anchored** on-chain.

Key Notes
Servers can be On-prem or Cloud

- **Frequent checks (5 min) on Server A** to catch tampering early, so

problems are detected *before* an hourly blockchain anchor.
- **Hourly checks** keep overhead low while still protecting the publishing pipeline.
- Ensures integrity at both the **data origin (A)** and **anchor point (B)**.

This gives us a tamper-evident chain of custody **end to end**, without hammering the network or chain.

Economics

Ballpark Figures:
(There are a lot of economic variables at play and I'm just an engineer—but in my view, this solution only really becomes cost-effective at scale.**) Medium to large organisations** see the most value due to shared infrastructure and economies of scale.

Compute Savings:
Using AWS Lambda for local log hashing (Server A) and spot instances for centralised verification (Server C) significantly reduces compute costs.

- **~£50/month for 50 servers' local hashing**
- **~£500/month for centralised verification**

→ Roughly **£10/server/month**

Storage Efficiency:
Server C only needs short-term buffers for recent Merkle roots and logs. Long-term storage remains in your existing S3/Glacier setup.

- **Additional cost: ~£100/month for temporary storage, metrics, and monitoring.**

Blockchain Anchoring:
Merkle roots are **only 32 bytes**, so transactions are light. Chain fees typically include:

- Base fee: flat per-extrinsic cost
- Weight fee: based on compute effort
- Length fee: based on payload size

Cost factor is **anchoring frequency, not the number of servers.**
Polkadot Asset Hub:

- **Daily** anchoring**:** 1 tx/day **= £3/month total**
- **Hourly** anchoring: 24 tx/day **≈** 720/month **= £70/month total**

Cost Summary:

- **Small org (1 server): £250–£320/server/month —** high per-server cost due to fixed infra.
- **Medium (10 servers): £30–£40/server** shared costs make it efficient.
- **Large org (50+ servers): £14–£20/server/month —** shared costs

make it far more efficient.

Visualise cost (small-med-large):

Cost Component	Small Org(1 server)	Medium Org(10 servers)	Large Org(50 servers)
🧮 Local Hashing (Server A)	£1	£1/server	£1/server
🖥️ Verification Infra (Server C)	£100	£100 total → £10/server	£500 total → £10/server
📊 Metrics & Monitoring	£100	£100 total → £10/server	£100 total → £2/server
🗂️ Temp Storage / Buffers	£50	£50 total → £5/server	£50 total → £1/server
🛡️ 5-min Spot Checks (Server C - challenge Server A)	+£30–£50	+£50–£75 total → £5–£7.5/server	+£75–£150 total → £1.5–£3/server
⛓️ Blockchain Anchoring (hourly)	£3–£70 (shared or solo)	£3–£70 total → up to £7/server	£3–£70 total → ~£0.06–£1.40/server
🔥 **Total (Typical Range)**	**£250–£320/server**	**£38–£61/server**	**£14–£20/server**

Target Market Gov Infra/Large-Corp:

Component	Total Cost (Monthly)	Per Server (1,000 servers)
🧮 Local Hashing (Server A)	£50	£0.05
🖥️ Verification Infra (Server C)	£1,000	£1.00
🗂️ Temp Storage & Monitoring	£100	£0.10
🛡️ 5-min Spot Checks (Enhanced)	£150	£0.15
⛓️ Daily Anchoring (Shared)	£3	£0.003

🔥 **Total (per month)**	**£1,303**	**£1.30/server**

Benefits over Pure Web2

Feature	Web2 Storage Alone	Hybrid Approach (Web2 + Minimal Web3)
Tamper Evidence	❌ Weak (can be changed silently)	✅ Strong (Merkle roots + blockchain anchoring)
Audit-ability	❌ Requires trust in internal systems	✅ Public, verifiable, timestamped
Vendor Independence	❌ Locked to cloud logs/SIEMs	✅ Anchor in public ledger, agnostic of infra
Cost Scaling	✅ Cheap, predictable	✅ Still low cost, especially at scale
Security Posture	🟡 Centralised (attack surface = single point)	🟢 Decentralised trust layer
Compliance / Evidence	🟡 Requires trust chain docs	✅ Cryptographic evidence of integrity

Key Deductions:
Whilst operational burden is low, this isn't a substitute for traditional security controls—it's extra plate armour. You still need gold-standard web2 sec (IAM, detection systems, behaviour monitoring) to catch attacks in real time.
This system kicks in *after the fact*, providing:

- A provable audit trail
- Cryptographic integrity
- The ability to pinpoint tampering down to a specific log entry (Merkle leaf)

"This isn't about replacing Web2 — it's about **bulletproofing it**.
You get the **cost-effectiveness of Web2**, plus the **integrity and trust guarantees of Web3** - for pennies per server."

The agents are designed to be safe, contained, and low-risk to deploy—even in sensitive environments. Here's the breakdown:

1. Minimal Attack Surface

- **Outbound-Only Communication:**
  Agents only **send** Merkle roots to Server C—no open ports, no inbound listening services.
- **No Privileged Access Required:**

They can run as non-root, read-only users. The only permission they need is read access to log files (or to your logging pipe/stream).
- **Cryptographic Isolation:**
  Each agent computes hashes locally. It never sends raw logs—only 32-byte Merkle roots. No sensitive data leaves the server.

2. Tamper-Evidence, Not Tamper-Resistance

- The agent **doesn't block tampering** but it **makes tampering provable**.
  If someone tries to manipulate logs after the fact, the Merkle tree won't match and you'll catch it downstream.
- This supports **zero-trust infrastructure** principles: even if an admin or attacker gains local access, they can't forge valid past entries without being detected.

3. Lightweight, Auditable Codebase

- The agents are small and purpose-built—can be open-sourced or audited easily. No opaque complexity or surprises.
- Written in secure-by-default languages (e.g. Go or Rust), avoiding memory corruption bugs and runtime risks common in lower-level systems.

4. Optional Integrity Enhancements

If you want higher assurance, we can:

- **Digitally sign Merkle roots** with a local HSM or TPM-backed key
- **Encrypt the roots in transit** (e.g. TLS with mutual auth)
- **Deploy with your existing EDR/SIEM tooling** to monitor for anomalies