Log Archiving Pre-Flight & Auditor Export: AWS / GCP / Azure / OVHcloud

2026-05-14

Pre-flight checklist for 7-year log archiving: irreversible choices, WORM setup, retention, and tamper-evident auditor export. Part 2 of 6.

This post continues the managed log archiving series with three operational questions that rarely appear in provider documentation: what must be done before writing the first log, what cannot be changed retroactively, and how to export logs to an external auditor in a tamper-evident way.

- Part 1 — Overview & Cost: Services, storage model, 7-year cost comparison
- **Part 2 — Pre-Flight, Flexibility & Auditor Export — this post**
- Part 3 — Operations: Kubernetes integration, ingest pipeline, backup & DR
- Part 4 — Security & Compliance: IAM/RBAC, encryption, WORM, GDPR, compliance certificates
- Part 5 — Query, Dashboards & Recommendations: Query interfaces, dashboards, alerting, cold-tier behaviour
- Part 6 — Production Checklist, Guardrails & Runbooks: Loss detection, privacy, schema, cost guardrails, runbooks, fire drills

Pre-Flight Requirements at a Glance

	AWS	GCP	Azure	OVH
Critical before first log	S3 bucket + Object Lock, Firehose + Glue schema	**Log Sink** (no backfill!), BigQuery dataset	Log Analytics retention, Blob container immutability	S3 buckets + Object Lock, ClickHouse schema + K8s Secrets
Cannot change retroactively	OL Compliance mode; Parquet partition scheme	Log Sink routes future logs only; locked retention	Locked immutability; Archive tier behaviour	Object Lock (cannot enable after creation)
Can change later	Retention period, lifecycle rules, IAM	BigQuery columns (add-only), retention increase	Blob-level immutability, retention increase	ClickHouse schema (ALTER TABLE), lifecycle rules
Auditor read-only access	Pre-signed URLs / IAM policy	Signed URLs / BigQuery IAM	SAS tokens / RBAC	Pre-signed URLs / ClickHouse user
Tamper evidence	S3 Object Lock metadata + CloudTrail	GCS retention lock + Cloud Audit Logs	Immutability policy + Activity Log	S3 Object Lock + ClickHouse query log

Pre-Flight Checklist

These are the decisions that must be made — and infrastructure that must exist — before writing the first log entry. Retroactive setup either loses historical data or requires a full archive rebuild.

AWS

Item	Why before first log
S3 bucket with Object Lock	Object Lock can be enabled on existing versioned buckets, but existing objects are not retroactively protected — configure before the first log write
Versioning	Required for Object Lock; auto-enabled with Object Lock
Firehose delivery stream	Output format (Parquet vs JSON) cannot be changed for historical objects
Glue Data Catalog table	Required for Athena to query Parquet; partition scheme is fixed here
S3 lifecycle rules	Tiering schedule applies to future objects only
IAM roles (writer + reader)	Firehose writer role; Athena reader role
CloudWatch Log Group retention	Default is never expire — you pay for storage. Set before ingest.

```
# Create S3 bucket with Object Lock enabled
aws s3api create-bucket \
  --bucket my-log-archive \
  --region eu-central-1 \
  --create-bucket-configuration LocationConstraint=eu-central-1 \
  --object-lock-enabled-for-bucket

# Set default Compliance retention (7 years)
aws s3api put-object-lock-configuration \
  --bucket my-log-archive \
  --object-lock-configuration '{
    "ObjectLockEnabled": "Enabled",
    "Rule": {"DefaultRetention": {"Mode": "COMPLIANCE", "Years": 7}}
  }'

# Set CloudWatch Log Group retention
aws logs put-retention-policy \
  --log-group-name /aws/eks/my-cluster \
  --retention-in-days 90
```

⚠ Parquet format is a one-way decision

If you configure Firehose with JSON output and later switch to Parquet, historical data stays in JSON. Athena can query both, but JSON queries are 5–10× more expensive due to full-column scans. Decide on Parquet + day-level partitioning before the first log arrives — changing it later means re-writing all historical data.

GCP

Item	Why before first log
Log Sink (Cloud Logging → BigQuery or GCS)	**Log Sinks do not backfill** — only logs written after sink creation are exported
BigQuery dataset + table	Log Sink needs a destination; day partition is automatic for streaming sinks
Log bucket retention	Defaults to 30 days; increase before ingest starts
Log Analytics upgrade	Required for SQL queries directly on Cloud Logging data
IAM service accounts	Log Sink writer role for BigQuery destination

```
# Create Log Sink to BigQuery — do this before any logs arrive
gcloud logging sinks create logs-to-bigquery \
  bigquery.googleapis.com/projects/my-project/datasets/log_archive \
  --log-filter='resource.type="k8s_container"'

# Grant Sink service account writer access to BigQuery dataset
gcloud projects add-iam-policy-binding my-project \
  --member="serviceAccount:$(gcloud logging sinks describe logs-to-
bigquery --format='value(writerIdentity)')" \
  --role="roles/bigquery.dataEditor"

# Increase Cloud Logging default bucket retention
gcloud logging buckets update _Default \
  --location=global \
  --retention-days=365
```

⚠ Log Sink does not backfill BigQuery automatically

This is the most critical GCP pre-flight step. If you forget to create the Log Sink before logs start flowing, the data exists in Cloud Logging (queryable via `gcloud logging read` until the retention period expires) but is not in BigQuery. Cloud Logging can batch-copy already stored log entries to Cloud Storage, but a BigQuery archive then needs a load job or custom pipeline. Sinks remain a future-log routing mechanism, so create them before production traffic starts.

Azure

Item	Why before first log
Log Analytics Workspace retention	Default is 30 days; logs deleted before retention is increased cannot be recovered
Blob Storage container with immutability	Recommended before regulated data ingestion; can be added to an existing container, but configuring it before ingestion avoids gaps, inconsistent policy scope, and audit ambiguity
Diagnostic Settings	Routes resource logs to the correct workspace; unrouted logs are lost
Event Hub (bypass path)	Required for the Event Hubs → Blob cost-efficient archiving path
RBAC assignments	Log Analytics Contributor for writers; Storage Blob Data Reader for auditors

```
# Set Log Analytics Workspace retention before first ingest
az monitor log-analytics workspace update \
  --resource-group my-rg \
  --workspace-name my-workspace \
  --retention-time 90

# Create Blob container with immutability before first blob
az storage container create \
  --name logs-archive \
  --account-name mylogaccount

az storage container immutability-policy create \
  --account-name mylogaccount \
  --container-name logs-archive \
  --period 2557  # 7 years in days
```

i Azure retention: set before data arrives

Azure Log Analytics Workspace retention can be increased at any time — but data already deleted by the shorter retention window cannot be recovered. Set the retention period to your target on workspace creation.

OVH (Self-Hosted ClickHouse)

Item	Why before first log
OVH Object Storage S3 buckets	Object Lock on OVH must be enabled at bucket creation — cannot be added later
SSE-C key as K8s Secret	Must exist in ClickHouse storage config before first write; rotating requires re-encrypting all objects
S3 access keys as K8s Secret	ClickHouse reads credentials from Kubernetes Secret at startup
ClickHouse schema + partition key	`PARTITION BY toYYYYMM(timestamp)` is fixed at table creation; changing it requires a full data rebuild
Fluent Bit / Vector DaemonSet	Log collection must start before the first log — no retroactive collection from container stdout

```
# Create OVH Object Storage bucket with Object Lock
# (must be done in OVH Public Cloud console or via S3 API with OVH
credentials)
aws s3api create-bucket \
  --bucket logs-cold \
  --endpoint-url https://s3.rbx.io.cloud.ovh.net \
  --object-lock-enabled-for-bucket

# ClickHouse table with partition key — set this before first INSERT
# (partition key cannot be changed retroactively)
CREATE TABLE logs
(
    timestamp   DateTime64(3),
    namespace   LowCardinality(String),
    pod         String,
    container   String,
    message     String,
    INDEX msg_idx message TYPE tokenbf_v1(32768, 3, 0) GRANULARITY 4
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(timestamp)
ORDER BY (namespace, timestamp)
TTL timestamp + INTERVAL 7 YEAR;
```

What Cannot Be Changed Retroactively

Decision	AWS	GCP	Azure	OVH
Enable Object Lock on existing bucket	Limited — plan at bucket creation	✓ Retention Policy can be added	✓ Blob-level immutability	✗ Must be at bucket creation
Backfill historical logs to new destination	✗ manual S3 copy only	Partial — Log Sink future only; batch copy to GCS possible while logs are retained	✗ Diagnostic Settings: future only	—
Change partition scheme	✗ existing data unaffected; new partitions only	✗ BigQuery partition columns fixed at table creation	—	✗ ClickHouse PARTITION BY fixed at table creation
Shorten Object Lock Compliance retention	✗ including AWS root account	✗ locked retention is absolute	✗ locked immutability is absolute	✗
Change SSE-C encryption key	✗ all existing objects must be re-encrypted manually	—	—	✗ all existing objects must be re-encrypted
Convert log format (JSON ⟶ Parquet)	✗ historical objects stay in JSON	—	—	—

The practical implication: **any compliance-relevant configuration must be in place before the first log write.** A log archive set up without WORM, without correct partitioning, or without a Log Sink cannot be made fully compliant retroactively — at best it requires a manual rebuild from source, which may not be possible if source retention has already expired.

What Can Be Changed Retroactively

Change	AWS	GCP	Azure	OVH
Increase retention period	✓	✓	✓	✓
Add / modify S3 lifecycle tier rules	✓ future objects only	✓ future objects only	✓ future objects only	✓ future objects only
Add IAM users / roles / policies	✓	✓	✓	✓
Add table columns	Glue: add columns	BigQuery: add-only	—	ClickHouse: ALTER TABLE ADD COLUMN
Enable Object Lock on existing bucket	Limited — new objects only	✓	✓ blob-level	✗
Change log group / workspace routing	✓ future logs only	✓ future logs only	✓ future logs only	—
Rotate S3 access keys	✓	✓	✓	✓ update K8s Secret + rolling restart
Add KMS / CMEK encryption	✗ existing objects stay under original key	✗ existing objects stay under original key	✗ existing objects stay under original key	✗ SSE-C: re-encrypt all objects

Auditor Export

An external auditor needs three things: **read access** to the relevant log data, **proof that the data has not been tampered with**, and ideally **no dependency on your internal systems** (no VPN, no internal accounts).

Read-Only Access Without an Auditor Account

Time-limited pre-signed URLs or SAS tokens require no account creation and expire automatically.

```
# AWS: pre-signed URL valid for 7 days
aws s3 presign s3://my-log-archive/2025/04/15/logs.parquet \
  --expires-in 604800

# GCP: signed URL (7 days)
gcloud storage sign-url \
  --duration=7d \
  gs://my-log-archive/2025/04/15/logs.json

# Azure: blob SAS token (read-only, 7 days)
az storage blob generate-sas \
  --account-name mylogaccount \
  --container-name logs-archive \
  --name "2025/04/15/logs.json" \
  --permissions r \
  --expiry $(date -u -d '+7 days' +%Y-%m-%dT%H:%MZ) \
  --output tsv

# OVH (S3-compatible): pre-signed URL
aws s3 presign s3://logs-cold/2025/04/15/logs.parquet \
  --endpoint-url https://s3.rbx.io.cloud.ovh.net \
  --expires-in 604800
```

Read-Only Account for Multi-Day Auditor Access

For auditors who need to query data themselves over multiple days.

```
# AWS: read-only IAM policy scoped to archive + Athena
aws iam create-policy --policy-name auditor-log-read --policy-document '{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "s3:GetObject", "s3:ListBucket",
      "athena:StartQueryExecution", "athena:GetQueryResults",
      "glue:GetTable", "glue:GetDatabase"
    ],
    "Resource": [
      "arn:aws:s3:::my-log-archive",
```

```
      "arn:aws:s3:::my-log-archive/*",
      "arn:aws:athena:eu-central-1:123456789012:workgroup/primary"
    ]
  }]
}'

# GCP: read access to GCS bucket + BigQuery dataset
gcloud projects add-iam-policy-binding my-project \
  --member="user:auditor@external.com" \
  --role="roles/storage.objectViewer"

bq add-iam-policy-binding \
  --member="user:auditor@external.com" \
  --role="roles/bigquery.dataViewer" \
  my-project:log_archive

# Azure: Storage Blob Data Reader scoped to archive container
az role assignment create \
  --role "Storage Blob Data Reader" \
  --assignee auditor@external.com \
  --scope "/subscriptions/<sub>/resourceGroups/<rg>/providers/
Microsoft.Storage/storageAccounts/mylogaccount/blobServices/default/
containers/logs-archive"

# OVH / ClickHouse: read-only SQL user
CREATE USER auditor IDENTIFIED WITH sha256_password BY '<strong-
password>';
GRANT SELECT ON logs.* TO auditor;
```

Bulk Export for Offline Handover

For handing over a specific time range as files — no ongoing access needed after download.

```
# AWS: Athena query → S3 result → sync locally
QUERY_ID=$(aws athena start-query-execution \
  --query-string "SELECT * FROM logs WHERE year='2025' AND month='04'" \
  --result-configuration "OutputLocation=s3://audit-export/2025-04/" \
  --query "QueryExecutionId" --output text)

# Poll until complete, then download
aws s3 sync s3://audit-export/2025-04/ ./audit-export-2025-04/

# GCP: BigQuery export to GCS, then download
bq extract \
  --destination_format NEWLINE_DELIMITED_JSON \
  'my-project:log_archive.container_logs$20250401' \
  gs://audit-export/2025-04/logs-*.json
```

```
gsutil -m cp -r gs://audit-export/2025-04/ ./audit-export-2025-04/

# Azure: download blob batch for date range
az storage blob download-batch \
  --source logs-archive \
  --destination ./audit-export-2025-04/ \
  --pattern "2025/04/*" \
  --account-name mylogaccount

# OVH / ClickHouse: direct Parquet export
clickhouse-client \
  --query "SELECT * FROM logs WHERE toYYYYMM(timestamp) = 202504 FORMAT
Parquet" \
  > audit-export-2025-04.parquet
```

For compliance handover, generate a SHA-256 checksum manifest alongside the export so the auditor can verify the download was not corrupted in transit:

```
find ./audit-export-2025-04/ -type f -exec sha256sum {} \; > audit-
export-2025-04.sha256
```

Proving Tamper-Evidence

Object Lock proves minimum retention. To prove content has not changed, verify the object metadata and retention status.

```
# AWS: verify Object Lock retention on a specific object
aws s3api get-object-retention \
  --bucket my-log-archive \
  --key "2025/04/15/logs.parquet"
# Returns: {"Retention": {"Mode": "COMPLIANCE", "RetainUntilDate":
"2032-04-15T00:00:00Z"}}

# AWS: verify content has not changed (ETag ≠ MD5 for multipart/Firehose
uploads — use ChecksumSHA256 or a manifest)
aws s3api head-object \
  --bucket my-log-archive \
  --key "2025/04/15/logs.parquet"

# GCP: verify object retention configuration
gcloud storage objects describe \
  gs://my-log-archive/2025/04/15/logs.json \
  --format="json(retentionExpireTime,temporaryHold,eventBasedHold)"

# Azure: verify blob immutability policy and ETag
az storage blob show \
```

```
  --account-name mylogaccount \
  --container-name logs-archive \
  --name "2025/04/15/logs.json" \
  --query "{immutabilityPolicy: properties.immutabilityPolicy, etag:
properties.etag}"

# OVH (S3-compatible): verify Object Lock
aws s3api get-object-retention \
  --endpoint-url https://s3.rbx.io.cloud.ovh.net \
  --bucket logs-cold \
  --key "2025/04/15/logs.parquet"
```

Audit Trail: Who Accessed Which Logs

Object Lock proves content integrity. A separate audit trail proves who accessed or downloaded which objects during the audit window. However, **audit logs themselves have a finite default retention** — for a 7-year compliance scenario, they must be explicitly archived just like application logs, or the access history is permanently lost.

Provider	Audit log source	Default retention	Archive path	Retrieval after archiving
AWS	CloudTrail Event History (management events, free)	**90 days**	CloudTrail Trail → S3 (one-time setup, continuous)	Athena on the S3 trail bucket
AWS	CloudTrail S3 data events (GetObject / PutObject)	**90 days**	Same Trail → S3; must be explicitly enabled ($0.10/100K events)	Athena on S3 trail bucket
GCP	Cloud Audit Logs — Admin Activity	**400 days** (free, automatic)	Log Sink → GCS or BigQuery (same pattern as application logs)	`gcloud logging read` within window; BigQuery SQL after sink
GCP	Cloud Audit Logs — Data Access	**30 days** (must be enabled explicitly, then free)	Log Sink → GCS or BigQuery	BigQuery SQL
Azure	Activity Log	**90 days**	Diagnostic Setting → Storage Account or Log Analytics Workspace	`az monitor activity-log list` (within 90d); blob download or KQL after archiving
OVH / ClickHouse	`system.query_log`	**Days** (ClickHouse default TTL, configurable)	INSERT INTO custom archive table with longer TTL, or increase `query_log` TTL in config	SQL on archive table

⚠ Audit trail logs must be archived separately — before the first external access event

For a 7-year compliance requirement, "who accessed which logs" must be provable for the entire retention period — not just the last 90 days. Archive audit logs with the same discipline as application logs:

- **AWS**: Create a CloudTrail Trail (not just Event History) that writes continuously to a dedicated S3 bucket. Enable S3 data events for the archive bucket. One-time setup, fully automatic afterwards.
- **GCP**: Add a Log Sink for `logName:"cloudaudit.googleapis.com"` to the same BigQuery dataset or GCS bucket as your application log sink. Data Access logs must be enabled first in IAM → Audit Log settings.
- **Azure**: Configure a Diagnostic Setting on the Subscription level to route Activity Logs to a Storage Account or Log Analytics Workspace with sufficient retention. The 90-day window is **not** extended retroactively — configure before the first access.
- **OVH / ClickHouse**: The default `system.query_log` TTL is too short for compliance. Either increase the TTL in `config.xml` or write a scheduled INSERT that copies rows into a separate `query_audit` table with a 7-year TTL.

Retrieving audit trail data after archiving:

```
# AWS: query CloudTrail events from S3 archive via Athena
# (requires Athena table over the CloudTrail S3 prefix)
SELECT eventTime, userIdentity.arn, eventName, requestParameters
FROM cloudtrail_logs
WHERE eventName IN ('GetObject', 'PutObject', 'DeleteObject')
  AND requestParameters LIKE '%my-log-archive%'
  AND year = '2025' AND month = '04'
ORDER BY eventTime;

# GCP: query Data Access audit logs from BigQuery sink
SELECT timestamp,
protopayload_auditlog.authenticationInfo.principalEmail,
       protopayload_auditlog.methodName,
protopayload_auditlog.resourceName
FROM `my-project.audit_logs.cloudaudit_googleapis_com_data_access`
WHERE DATE(timestamp) BETWEEN '2025-04-01' AND '2025-04-30'
ORDER BY timestamp;

# Azure: retrieve Activity Log from Storage Account (blob download)
az storage blob download-batch \
  --source insights-activity-logs \
  --destination ./audit-activity-2025-04/ \
  --pattern "*/y=2025/m=04/*" \
  --account-name myauditaccount
```

```
# OVH / ClickHouse: query from custom audit archive table
SELECT query, user, event_time, query_duration_ms
FROM query_audit
WHERE user = 'auditor'
  AND toYYYYMM(event_time) = 202504
ORDER BY event_time;
```

FAQ

Can I retroactively export logs from the last month via CLI if I forgot to set up a sink or export path?

It depends on the provider and configured retention:

- **AWS**: Yes — `aws logs filter-log-events` retrieves CloudWatch Logs data within the configured retention window. For data already in S3, use `aws s3 sync`.
- **GCP**: Partially — `gcloud logging read` with a time filter retrieves data from Cloud Logging within the retention window (default 30 days for the `_Default` bucket). Data not routed to a Log Sink is only in Cloud Logging, but Cloud Logging can batch-copy retained entries to Cloud Storage. BigQuery backfill still requires an extra load job or custom pipeline.
- **Azure**: Yes for hot-tier data via `az monitor log-analytics query`. Long-term retention can be accessed with search jobs for targeted retrieval; full interactive KQL analysis uses restore jobs first.
- **OVH / ClickHouse**: Yes — direct SQL query or `clickhouse-client` export; no additional steps needed.

What export format should I use for an auditor handover?

- **Parquet**: efficient, self-describing schema — preferred for large structured exports from Athena, BigQuery, or ClickHouse
- **NDJSON** (newline-delimited JSON): human-readable, compatible with `jq`, good for spot-check review
- **CSV**: widest compatibility with audit tools and spreadsheets, but loses type information

For compliance handover, always include a SHA-256 checksum manifest alongside the export.

How can I prove to an auditor that logs for a specific incident period have not been deleted?

Three complementary proofs:

1. **Object Lock metadata** (`get-object-retention`): proves a minimum retention period applies until a specific date
2. **Object ETag / checksum**: proves the object content has not changed since it was written (ETag equals MD5 only for single-part uploads; for Firehose or Parquet multipart uploads, use `ChecksumSHA256` or compare against your own SHA-256 manifest)

3. **CloudTrail / Cloud Audit Logs access history**: proves no `DeleteObject` events occurred on the archive bucket during the relevant period

No single proof is sufficient in isolation — combine all three in the evidence package.

Does BYOK (customer-managed keys) complicate auditor access?

With **SSE-KMS** (AWS) or **CMEK** (GCP): the auditor cannot access raw objects without the KMS key. Either export to a separate staging bucket encrypted with a different key, or grant the auditor temporary KMS usage permission via IAM.

With **SSE-C** (OVH): every GET request must include the AES-256 key header. For auditor handover, decrypt and re-upload to a staging location, or export directly via ClickHouse SQL (which holds the key in its storage configuration and returns plaintext results).

Part 1: Overview & 7-year cost comparison

Part 3: Operations — Kubernetes integration, ingest pipeline, backup & DR

Part 4: Security & Compliance — IAM/RBAC, encryption, WORM, GDPR

Part 5: Query, Dashboards & Recommendations

Part 6: Production Checklist, Guardrails & Runbooks