7-Year Log Archiving in Production: Checklist, Guardrails & Runbooks

2026-05-14

Production checklist for 7-year log archiving: loss detection, privacy, schema, cost guardrails, runbooks, and fire drills. Part 6 of 6.

Choosing the provider and storage tier is only half of the work. A seven-year log archive is a production system: it must prove that logs are complete, protect personal data, keep query costs bounded, survive access-key and restore failures, and give auditors repeatable evidence without turning every audit into an incident.

This final part turns the provider comparison into an operating checklist.

- Part 1 — Overview & Cost: Services, storage model, 7-year cost comparison
- Part 2 — Pre-Flight, Flexibility & Auditor Export: Pre-flight checklist, retroactive flexibility, auditor export
- 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, when to use which
- **Part 6 — Production Checklist, Guardrails & Runbooks — this post**

> ⚠ **Compliance note**
>
> This article is an engineering checklist, not legal advice. Regulations such as DORA, NIS2, GDPR, PCI DSS, and sector-specific retention rules must be interpreted by your legal, compliance, and security teams. Treat the mappings below as implementation prompts: they help you ask the right questions and collect the right evidence.

Production Readiness at a Glance

Area	Minimum production control	Evidence to keep
Completeness	Canary log events, ingest counters, dropped-event alerts	Daily completeness report
Integrity	WORM / Object Lock, checksums, immutable manifests	Object retention metadata, hash manifests
Privacy	Redaction before archive write, PII field policy	Redaction config, sample validation results
Schema	Versioned log contract, additive-only changes	Schema registry / Markdown contract
Access	Read-only auditor role, break-glass role, MFA	IAM export, access review record
Cost	Budgets, query scan limits, lifecycle drift checks	Monthly FinOps report
Recovery	Search/restore/export runbooks	Fire-drill transcript
Compliance	Control mapping to internal policies	Audit pack with screenshots and command output

If one of these rows has no owner, the archive is not production-ready yet.

Loss Detection

The most dangerous logging failure is a quiet one: agents keep running, dashboards look green, but a filter, quota, or overloaded sink drops a slice of production traffic. A seven-year archive needs end-to-end loss detection, not just “agent pod is running”.

Canary Events

Emit a synthetic log event from every cluster, account, project, or subscription at a fixed interval. The event should travel through the exact same path as application logs.

```
{
  "event_type": "log_archive_canary",
  "archive_pipeline": "prod-eu-01",
  "source_cluster": "payments-prod-eu",
  "sequence": 184467,
  "emitted_at": "2026-05-14T00:00:00Z",
  "expected_archive": "logs/year=2026/month=05/day=14/"
}
```

Recommended checks: - **Presence:** every expected canary arrived in the archive - **Latency:** event appeared within the ingestion SLO - **Partition:** event landed under the expected day / namespace / service path - **Queryability:** the archive query layer can find the event

For 100 GB/day pipelines, a five-minute canary interval is enough to detect broken delivery quickly without adding measurable cost.

Emitting Canary Events Per Provider

Provider	Emission method
AWS	EventBridge Scheduler rule → Lambda → `PutLogEvents` (CloudWatch) or `PutRecord` (Firehose)
GCP	Cloud Scheduler → Cloud Functions (2nd gen) → Cloud Logging `write` API
Azure	Timer-trigger Function App → Log Analytics HTTP Data Collector API or Event Hubs SDK
OVHcloud	Kubernetes `CronJob` → Vector or Fluent Bit → ClickHouse (no managed function runtime)

The canary must travel the exact same sink path as application logs. A dedicated Lambda or Cloud Function that calls `PutLogEvents` directly bypasses Firehose and will miss delivery failures on that path.

OVHcloud example (Kubernetes `CronJob`):

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: log-archive-canary
  namespace: platform
spec:
  schedule: "*/5 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: OnFailure
          containers:
          - name: canary
            image: curlimages/curl:8
            command:
            - sh
            - -c
            - |
              printf
'{"event_type":"log_archive_canary","pipeline":"prod-eu-01",
                "cluster":"%s","sequence":%s,"emitted_at":"%s"}\n' \
                "$NODE_NAME" "$RANDOM" "$(date -u +%Y-%m-%dT%H:%M:%SZ)" \
                >> /var/log/canary/canary.log
            volumeMounts:
            - name: canary-log
              mountPath: /var/log/canary
          volumes:
```

```
      - name: canary-log
        hostPath:
          path: /var/log/containers/canary
```

Sequence IDs and Counters

Canaries prove that the path is alive. They do not prove that every application log arrived. For high-value systems, add monotonic sequence IDs at the producer or sidecar level and reconcile them downstream.

Pattern	Works best for	Trade-off
Per-pod sequence counter	Kubernetes workloads	Resets on pod restart; include pod UID
Per-service counter	Centralized app logging library	Needs application changes
Agent read offset	File tailing / stdout collection	Agent-specific and hard to query later
Batch manifest	Firehose / Dataflow / Event Hubs consumers	Proves batches, not individual app events

A daily completeness job should compare source counters with archive counters and write the result to a small, separate audit table.

```sql
-- Example: daily canary completeness in a SQL archive
SELECT
  source_cluster,
  DATE(emitted_at) AS day,
  COUNT(*) AS canaries_seen,
  MIN(emitted_at) AS first_seen,
  MAX(emitted_at) AS last_seen
FROM logs
WHERE event_type = 'log_archive_canary'
  AND emitted_at >= TIMESTAMP_SUB(CURRENT_TIMESTAMP(), INTERVAL 7 DAY)
GROUP BY source_cluster, day
ORDER BY day DESC, source_cluster;
```

Pipeline Alerts

Every provider exposes different metrics, but the alert semantics are the same:

Alert	Meaning	Typical threshold
Oldest undelivered event age	Destination or consumer is falling behind	> 5 minutes warning, > 30 minutes critical
Dropped / failed records	Data loss or retry exhaustion	Any sustained non-zero value
Buffer usage	Backpressure before loss occurs	> 70% warning, > 90% critical
Destination write errors	Auth, quota, schema, or network problem	Any sustained non-zero value
Daily volume deviation	Missing or runaway logs	±30% from 14-day baseline

Do not alert only on infrastructure health. Alert on the data path.

Provider Metric References

Provider	Metric	Alert on
AWS Firehose	`DeliveryToS3.DataFreshness`	Age of oldest undelivered record > 5 min
AWS Firehose	`ThrottledRecords`	Any sustained non-zero value
AWS CW Logs	`IncomingLogEvents` / `ForwardedLogEvents`	Zero count or gap vs. baseline
GCP Logging	`logging.googleapis.com/exports/error_count` (per sink)	Any non-zero value
GCP Pub/Sub	`pubsub.googleapis.com/subscription/oldest_unacked_message_age`	> 5 min
GCP Pub/Sub	`pubsub.googleapis.com/subscription/num_undelivered_messages`	Growing trend
Azure Event Hubs	`IncomingMessages` vs. `OutgoingMessages`	Sustained divergence
Azure Event Hubs	`ThrottledRequests`	Any non-zero value
Azure Log Analytics	`_LogOperation` table, `Level = "Warning"` / `"Error"`	Any ingestion errors
OVH ClickHouse	`system.metrics` (`Query`, `InsertQuery`)	Drop against baseline
OVH Vector	`component_errors_total`, `component_discarded_events_total`	Any non-zero value

Privacy and Redaction

Long-term archives are difficult to delete from by design. That makes privacy decisions front-loaded: secrets and unnecessary personal data must be removed before the archive write, not after an auditor asks about them.

Redaction Order

Prefer this order:

1. **Application code:** do not log passwords, tokens, session cookies, or full request bodies in the first place.
2. **Logging library:** central field allowlist, structured fields, automatic secret masking.
3. **Agent / pipeline:** Vector VRL, Fluent Bit filters, Dataflow, Lambda transform, or Stream Analytics.
4. **Query layer:** last-resort masking for dashboards; not sufficient for archive data.

The archive should receive the redacted form. Query-time masking is useful for UI safety, but the original sensitive bytes still exist in storage.

Field Policy

Create a small field policy and keep it next to the schema contract:

Field type	Archive policy	Notes
Access tokens, API keys, passwords	Never store	Replace with `[REDACTED]` before archive
Session IDs / cookies	Hash or drop	Salt rotation must be documented
IP addresses	Keep, truncate, or hash by legal basis	Often needed for security investigations
User IDs	Prefer internal pseudonymous ID	Avoid email address as primary identifier
Request body	Drop by default	Allowlist only specific safe fields
User agent	Usually keep	Can still be personal data in rare cases

> ⚠ **Object Lock and erasure requests**
>
> If logs are stored under Compliance-mode WORM retention, deleting individual records may be technically impossible until the retention period expires. That must be covered by the legal basis for retention, the records of processing activities, and the privacy notice. Do not rely on "we can delete it later" for immutable archives.

Redaction Tests

Add synthetic secret strings to non-production logs and verify they never appear in the archive:

- `Authorization: Bearer archive-redaction-test-token`
- `password=archive-redaction-test-password`
- `session=archive-redaction-test-cookie`
- `api_key=archive-redaction-test-key`

Run this test after every logging-agent upgrade and every pipeline rule change.

Provider-Specific Pipeline Examples

AWS — Firehose Lambda data transformation:

```
import base64, re

SECRET_PATTERN = re.compile(
    r'(?i)(password|token|api_key|session|secret)\s*[=:]\s*\S+')

def handler(event, _ctx):
    output = []
    for rec in event['records']:
        payload = base64.b64decode(rec['data']).decode('utf-8',
errors='replace')
        payload = SECRET_PATTERN.sub(r'\1=[REDACTED]', payload)
        output.append({
            'recordId': rec['recordId'],
            'result': 'Ok',
            'data': base64.b64encode(payload.encode()).decode(),
```

```
        })
    return {'records': output}
```

Attach the function as a Firehose `ProcessingConfiguration` transform. For structured JSON use a dedicated DLP library instead of regex.

GCP — Cloud Logging exclusion filter (entry-level) + Dataflow (field-level):

```
# Log Router exclusion — drops the entire entry; use only if the whole
entry must not reach the archive.
# For field-level redaction, route through a Dataflow pipeline with Cloud
DLP.
jsonPayload.password=~"." OR jsonPayload.token=~"."
OR httpRequest.requestUrl=~"(token|api_key|password)="
```

Azure — Stream Analytics REGEXMATCH filter:

```
SELECT EventEnqueuedUtcTime, ServiceName, Level, Message
INTO [log-archive-output]
FROM [event-hubs-input]
WHERE REGEXMATCH(Message,
      '(?i)(password|token|api_key|session)\s*[=:]\s*\S+') = 0
```

For field-level masking instead of drop, use Azure Data Factory with a Microsoft Purview DLP rule.

OVH — Vector VRL transform:

```
[transforms.redact_secrets]
type   = "remap"
inputs = ["kubernetes_logs"]
source = '''
  .message = replace(string!(.message),
    r'(?i)(password|token|api_key|session|secret)\s*[=:]\s*\S+',
    "[REDACTED]")
  if exists(.fields.authorization) {
    .fields.authorization = "[REDACTED]"
  }
'''
```

Schema and Format

Seven-year retention is longer than most application lifetimes. Without a schema policy, the archive becomes a pile of almost-compatible JSON.

Minimal Log Contract

Every archived event should have at least:

Field	Why it matters
`timestamp`	Primary time filter and partition key
`ingested_at`	Pipeline latency and backfill detection
`service.name`	Service-level search and cost pruning
`deployment.environment`	Separates prod/stage/dev
`cloud.provider`	Multi-cloud correlation
`cloud.account.id` / project / subscription	Ownership and IAM investigation
`k8s.cluster.name`	Kubernetes source attribution
`k8s.namespace.name`	Query pruning and tenancy
`severity`	Alerting and triage
`message`	Human-readable fallback
`schema.version`	Safe evolution over years

OpenTelemetry Logs are a good neutral naming baseline, even if the underlying store is CloudWatch, BigQuery, Azure Monitor, or ClickHouse.

JSON vs. Parquet

Format	Use when	Avoid when
gzip JSON	Simplicity and portability matter most	You run frequent Athena/Synapse scans
Parquet	SQL scan cost matters	Schema changes are chaotic
Native BigQuery / Kusto / ClickHouse	Query performance matters	You need full provider portability

For cold archives, the best practical compromise is often:

- Store raw-ish redacted JSON for legal portability.
- Convert to Parquet or columnar tables for query efficiency.
- Keep a manifest linking source objects to converted objects.

Cost Guardrails

The worst archive bill is usually not storage. It is ingest accidentally routed through the expensive path, or a broad query that scans years of data.

Budget and Quota Controls

Provider	Guardrail
AWS	Budgets on CloudWatch, Firehose, Athena, S3; Athena workgroup scan limits
GCP	Budgets on Cloud Logging, Pub/Sub, BigQuery; custom query quotas; billing export
Azure	Cost Management budgets; Log Analytics daily caps; Synapse / ADX budget alerts
OVHcloud	Project budget alerts; object storage lifecycle checks; ClickHouse query limits

Minimum alerts:

- Daily ingest cost exceeds expected baseline by 30%
- Query cost exceeds daily budget
- Archive bucket grows faster than expected raw ingest model
- Lifecycle transition did not occur for yesterday's objects
- BigQuery / Athena / Synapse query scanned more than expected partition range

Setting Guardrails

AWS — Athena workgroup with scan cutoff:

```
aws athena create-work-group \
  --name log-archive-read \
  --configuration '{
    "ResultConfiguration": {"OutputLocation": "s3://my-audit-results/"},
    "EnforceWorkGroupConfiguration": true,
    "BytesScannedCutoffPerQuery": 10737418240
  }'
# Queries that would scan more than 10 GB are rejected before execution.
```

GCP — BigQuery dry-run cost estimate:

```
bq query --dry_run --use_legacy_sql=false \
  'SELECT * FROM `project.logs.archive`
   WHERE DATE(timestamp) = "2026-05-01"
     AND service_name = "payments"'
# Prints estimated bytes scanned before committing to the query.
```

Azure — Log Analytics daily ingest cap:

```
az monitor log-analytics workspace update \
  --resource-group rg-logs \
  --workspace-name law-prod \
  --quota 50    # GB/day; ingestion stops when reached
```

OVH — ClickHouse per-user query profile:

```
CREATE SETTINGS PROFILE auditor_limits SETTINGS
  max_execution_time  = 120,
  max_bytes_to_read   = 10737418240,
  max_result_rows     = 100000;

ALTER USER auditor_ro SETTINGS PROFILE 'auditor_limits';
```

Query Safety Rules

Require these defaults for archive users:

- Every query must include a bounded time range.

- Every query must include at least one high-selectivity filter (`service`, `namespace`, `account`, `cluster`, or `tenant`).
- Dashboards must use pre-aggregated tables or views for recurring queries.
- Auditor roles should not have unrestricted ad-hoc query permissions unless required.

For Athena, put users in a workgroup with a per-query scan cutoff. For BigQuery, use custom quotas and dry-run estimates. For ClickHouse, set `max_execution_time`, `max_bytes_to_read`, and per-user profiles for auditor accounts.

Multi-Account and Multi-Project Design

Do not let each application team build its own archive. The usual production pattern is a central logging boundary:

Provider	Central boundary
AWS	Dedicated log archive account with S3 buckets, KMS keys, Glue, Athena
GCP	Dedicated logging project with aggregated sinks and BigQuery / GCS destinations
Azure	Dedicated subscription or resource group with Log Analytics, Storage, Event Hubs
OVHcloud	Dedicated Public Cloud project for archive buckets and ClickHouse access

Principles:

- Application accounts can write, not read or delete.
- Security / SRE can query operationally.
- Auditors get time-limited read-only access.
- Break-glass access is separate, MFA-protected, and reviewed after use.
- Production, staging, and development archives are physically or logically separated.

For WORM buckets, keep ownership with the central platform/security team. Application teams should not be able to reduce retention, change lifecycle rules, or disable sinks.

Compliance Mapping

The same technical controls support several compliance frameworks, but no single log archive “makes you compliant”. Use this as an evidence map.

Requirement family	Log archive control	Evidence
Incident investigation	Searchable logs with reliable timestamps	Query transcript, incident timeline
Audit trail integrity	WORM retention, object versioning, checksums	Retention metadata, hash manifest
Access accountability	IAM/RBAC, audit logs for archive access	Access logs, quarterly review
Operational resilience	Restore/search runbook, fire drill	Drill record, RTO/RPO result
Data protection	Redaction, minimisation, retention policy	Field policy, DPA/ROPA reference
Change management	Schema versioning, pipeline config in Git	Pull requests, deployment history
Cost governance	Budgets, query limits, lifecycle drift checks	Monthly FinOps report

Example framework prompts:

Framework	Useful archive evidence
ISO/IEC 27001	Risk treatment, access control, logging, monitoring, backup/restore evidence
DORA	ICT incident investigation, operational resilience testing, third-party ICT evidence
NIS2	Cybersecurity risk management, incident handling, supply-chain and monitoring evidence
PCI DSS	Audit trails for cardholder-data environments, log protection, individual accountability
GDPR	Data minimisation, lawful basis, retention documentation, access control
BSI C5 / SOC 2	Access reviews, change management, logical security, availability evidence

Official references worth keeping in the internal control library:

- European Commission — DORA
- European Commission — NIS2 Directive
- ISO/IEC 27001:2022 overview
- PCI Security Standards Council FAQ on audit trails

Runbooks

Runbooks should be short enough to execute during an audit call. Keep provider-specific commands in appendices, but the decision flow should fit on one page.

Auditor Export Runbook

1. Confirm scope: time range, services, clusters, account/project/subscription, fields.
2. Confirm legal basis and approval ticket.
3. Run the query or object export with a bounded time range.
4. Export in Parquet or newline-delimited JSON.

5. Generate SHA-256 checksums for every file.
6. Write a manifest with query, operator, timestamp, file list, and checksum list.
7. Grant read-only time-limited access or deliver via approved secure transfer.
8. Archive the access event and the handover manifest.

Manifest example:

```
{
  "export_id": "audit-2026-05-payments-001",
  "requested_by": "external-auditor@example.com",
  "approved_by": "security-lead@example.com",
  "operator": "platform-sre@example.com",
  "time_range": "2026-05-01T00:00:00Z/2026-05-02T00:00:00Z",
  "query": "service.name = 'payments' AND severity >= 'ERROR'",
  "format": "parquet",
  "files": [
    {
      "path": "exports/audit-2026-05-payments-001/part-000.parquet",
      "sha256": "..."
    }
  ]
}
```

Historical Search / Restore Runbook

1. Decide whether the task is a targeted search or broad analytics.
2. Estimate scan / restore cost before execution.
3. Use the smallest time range that can answer the question.
4. Run the query in a cost-limited workgroup / project / user profile.
5. Save query text, cost estimate, result location, and operator.
6. Delete temporary restored/search-result tables after the approved retention period.

AWS — Athena bounded partition scan:

```
aws athena start-query-execution \
  --query-string "SELECT * FROM logs
    WHERE year='2026' AND month='05' AND day='01'
      AND service='payments' AND severity IN ('ERROR','CRITICAL')" \
  --work-group log-archive-read \
  --result-configuration OutputLocation=s3://audit-results/
```

GCP — BigQuery dry-run then batch query:

```
bq query --dry_run --use_legacy_sql=false \
  'SELECT * FROM `project.logs.archive`
   WHERE DATE(timestamp) BETWEEN "2026-05-01" AND "2026-05-02"
     AND service_name = "payments"'
```

```
bq query --use_legacy_sql=false --batch \
  'SELECT * FROM `project.logs.archive`
   WHERE DATE(timestamp) BETWEEN "2026-05-01" AND "2026-05-02"
     AND service_name = "payments"'
```

Azure — search job (targeted) vs. restore job (broad):

Use a search job when the question is narrow; it does not restore the full table and is billed per GB searched. Use a restore job only when interactive KQL across a wide time window is needed — it is billed at hot-tier rates while active, so delete it after use.

```
# Targeted: search job
az monitor log-analytics workspace table search-job create \
  --resource-group rg-logs --workspace-name law-prod \
  --table-name AppLogs_SRCH \
  --search-query "AppLogs | where ServiceName == 'payments' | where Level
== 'Error'" \
  --start-search-time "2026-05-01T00:00:00" \
  --end-search-time "2026-05-02T00:00:00"

# Broad: restore job
az monitor log-analytics workspace table restore create \
  --resource-group rg-logs --workspace-name law-prod \
  --table-name AppLogs_RST \
  --restore-source-table AppLogs \
  --start-restore-time "2026-05-01T00:00:00" \
  --end-restore-time "2026-05-02T00:00:00"
```

OVH — ClickHouse bounded query:

```
SELECT *
FROM logs
WHERE toDate(timestamp) = '2026-05-01'
  AND service = 'payments'
  AND level IN ('ERROR', 'CRITICAL')
SETTINGS max_execution_time = 120,
         max_bytes_to_read  = 5368709120;  -- 5 GB hard cap
```

Key-Loss Runbook

For KMS / CMK / SSE-C paths:

1. Stop new writes using the affected key.
2. Identify object prefixes and tables encrypted with the key.
3. Verify whether the key is lost, disabled, scheduled for deletion, or merely rotated.
4. If disabled: restore access following break-glass approval.
5. If permanently lost: declare affected data unrecoverable and start incident process.
6. Update the risk register and key-escrow procedure.

For SSE-C, losing the key means losing the data. The runbook should say that plainly.

AWS — KMS key state:

```
aws kms describe-key --key-id alias/log-archive-key \
  --query 'KeyMetadata.{State:KeyState,Deletion:DeletionDate}'

aws kms enable-key           --key-id alias/log-archive-key  # if merely
disabled
aws kms cancel-key-deletion  --key-id alias/log-archive-key  # if
deletion was scheduled
```

GCP — Cloud KMS key version state:

```
gcloud kms keys versions describe 1 \
  --key log-archive-key --keyring log-archive-ring \
  --location europe-west3 --format="get(state)"

gcloud kms keys versions enable 1 \
  --key log-archive-key --keyring log-archive-ring \
  --location europe-west3
```

Azure — Key Vault key state:

```
az keyvault key show \
  --vault-name kv-log-archive --name log-archive-key \
  --query
'{enabled:attributes.enabled,expires:attributes.expires,recovery:recoveryLevel}'

az keyvault key set-attributes \
  --vault-name kv-log-archive --name log-archive-key --enabled true
```

OVH — SSE-C: The key is stored entirely client-side. OVH has no recovery path. Document the key storage location (HashiCorp Vault, HSM, or secrets manager) in this runbook. If the key is lost, the encrypted objects are unrecoverable.

Object Lock Proof Runbook

1. Select a sample object from the incident/audit time range.
2. Fetch retention metadata (`retain-until`, retention mode, legal hold).
3. Fetch object checksum or compare with archived manifest.
4. Fetch access/audit logs showing no delete or overwrite events.
5. Export all command output to the audit evidence folder.

This proves both minimum retention and practical tamper evidence.

AWS:

```
aws s3api get-object-retention \
  --bucket log-archive-prod \
```

```
  --key "logs/year=2026/month=05/day=01/part-000.parquet"

aws s3api get-object-attributes \
  --bucket log-archive-prod \
  --key "logs/year=2026/month=05/day=01/part-000.parquet" \
  --object-attributes Checksum

# Confirm no delete events in CloudTrail
aws cloudtrail lookup-events \
  --lookup-attributes AttributeKey=ResourceName,AttributeValue=log-
archive-prod/logs/year=2026/month=05/day=01/part-000.parquet \
  --query "Events[?EventName=='DeleteObject'||
EventName=='DeleteObjects']"
```

GCP:

```
gcloud storage objects describe \
  gs://log-archive-prod/logs/year=2026/month=05/day=01/part-000.parquet \
  --
format="json(retentionExpirationTime,temporaryHold,eventBasedHold,md5Hash,crc32c)"

# Confirm no delete events (Data Access logging must be enabled for
storage.objects.delete)
gcloud logging read \
  'protoPayload.methodName="storage.objects.delete" AND
protoPayload.resourceName:"log-archive-prod/logs/year=2026"' \
  --freshness=7d --format=json
```

Azure:

```
az storage blob immutability-policy show \
  --account-name stlogarchiveprod \
  --container-name log-archive \
  --name "logs/2026/05/01/part-000.parquet"

# Confirm no delete operations in Activity Log
az monitor activity-log list \
  --resource-group rg-logs --start-time 2026-05-01 \
  --query "[?operationName.value=='Microsoft.Storage/storageAccounts/
blobServices/containers/blobs/delete']"
```

OVH (S3-compatible endpoint):

```
aws s3api get-object-retention \
  --bucket log-archive-prod \
  --key "logs/year=2026/month=05/day=01/part-000.parquet" \
  --endpoint-url https://s3.gra.perf.cloud.ovh.net
```

```
aws s3api get-object-attributes \
  --bucket log-archive-prod \
  --key "logs/year=2026/month=05/day=01/part-000.parquet" \
  --object-attributes Checksum \
  --endpoint-url https://s3.gra.perf.cloud.ovh.net
```

Fire Drills

Run a retrieval fire drill at least once per year, and after any major platform change. For regulated environments, quarterly is better.

Drill	Success criterion	Key command / method
Find a canary from 18 months ago	Query returns expected event and partition	Athena / BigQuery / KQL / ClickHouse query with `event_type = 'log_archive_canary'`
Export one day for one service	Manifest, checksums, and auditor access all work	See Auditor Export Runbook
Restore/search old Azure Monitor data	Search or restore completes within expected window	`az monitor log-analytics workspace table search-job create`
Verify Object Lock	Retention metadata proves WORM status	See Object Lock Proof Runbook
Rotate reader credentials	Query users keep access; old credentials stop working	`aws iam delete-access-key` / `gcloud iam service-accounts keys delete` / `az ad sp credential reset` / `ALTER USER … IDENTIFIED BY` (ClickHouse)
Simulate dropped sink	Alert fires before data loss budget is exceeded	Stop Fluent Bit DaemonSet / pause Pub/Sub subscription / disable Event Hubs consumer group; verify alert fires within SLO
Cost spike test	Budget alert reaches the right owner	Athena full-table query without partition filter in test workgroup; or `bq query` without `WHERE DATE(...)` clause

Record:

- Date and operator
- Exact commands / queries
- Time to first result
- Cost estimate and actual cost
- Gaps found
- Follow-up ticket links

The point of the drill is not to pass. The point is to learn while the auditor is not waiting on the call.

Final Production Checklist

Before calling a seven-year archive production-ready, make sure all boxes are checked:

- □ WORM / Object Lock configured before regulated data arrives
- □ Lifecycle rules tested and monitored for drift
- □ Canary events emitted and checked end-to-end
- □ Dropped-event, buffer, lag, and volume alerts active
- □ Redaction policy documented and tested with synthetic secrets
- □ Schema contract versioned and owned
- □ Query cost limits and budgets active
- □ Central logging account/project/subscription owns archive resources
- □ Auditor read-only role tested
- □ Search / restore / export runbooks tested
- □ Key-loss runbook reviewed
- □ Annual or quarterly retrieval fire drill scheduled
- □ Evidence folder template exists

FAQ

Should the archive store raw logs or only normalized logs?

Store redacted raw logs if legal portability matters, and store normalized / columnar copies for query efficiency. The manifest should link raw source objects to derived Parquet tables or ClickHouse partitions.

Can I redact data after it has already been written to WORM storage?

Not reliably. You might be able to write a corrected copy, but the original immutable object remains until retention expires. Redaction must happen before archive write for data that must not be retained.

How often should restore and export runbooks be tested?

At least yearly. Test quarterly for regulated workloads, after major IAM changes, after logging-agent upgrades, and after changing lifecycle or Object Lock configuration.

Who should own the archive?

Usually a platform/security team, not individual application teams. Application teams write logs; platform/security owns retention, WORM configuration, query guardrails, auditor access, and fire drills.

Part 1: Overview & 7-year cost comparison

Part 2: Pre-Flight, Flexibility & Auditor Export

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

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

Part 5: Query, Dashboards & Recommendations