K8s & OpenShift: Compliance

2026-07-01

NIS2, DORA, PCI-DSS, HIPAA, and CRA controls mapped to Kubernetes configuration — audit logging, log retention, and incident reporting. Part 7 of 7.

Part 6 covered the operational loop: GitOps, observability, cost control, and upgrades. This final part maps the cluster configuration covered throughout this series to the regulatory frameworks that require it — NIS2, DORA, PCI-DSS, HIPAA, and the Cyber Resilience Act (CRA) — and covers the two topics that compliance work adds on top of security: audit logging as evidence and log retention as a contractual obligation.

Series navigation: - **Full guide:** The Kubernetes & OpenShift Best Practices Guide (2026 Edition) - Part 1 — The Big Picture - Part 2 — Building workloads right - Part 3 — Resource management deep dive - Part 4 — Scaling & resilience - Part 5 — Security - Part 6 — Day-2 operations & GitOps - **Part 7 — Compliance (this post)**

Compliance is already in the YAML

The most useful framing for compliance in Kubernetes is not "what extra controls do we need" — it is "which controls are already in place, and can we prove it?" The practices from Parts 2 through 6 — RBAC, PSA, NetworkPolicy, image signing, audit logging, GitOps — are the technical implementation of the controls that regulators and auditors require. What compliance work adds is:

1. A mapping from those controls to the specific regulatory language
2. Audit logging that produces evidence that the controls are active
3. Log retention that keeps the evidence available for the required period
4. Incident reporting procedures that use the observability data from Part 6

This part covers five frameworks — the ones most commonly encountered when running Kubernetes in regulated environments.

The five frameworks

Framework	Scope	Origin	Key obligation
NIS2	Essential and important entities (energy, transport, banking, health, digital infrastructure, public administration)	EU Directive, transposed into national law; effective October 2024	Risk management measures, supply chain security, incident reporting (24h/72h/1 month)
DORA	Financial sector (banks, investment firms, payment institutions, insurers, ICT service providers)	EU Regulation, directly applicable; effective January 2025	ICT risk management framework, incident classification and reporting (4h/72h/1 month), resilience testing, third-party risk
PCI-DSS v4.0	Any entity that stores, processes, or transmits payment card data	PCI Security Standards Council; v4.0 mandatory since April 2024	Network segmentation, access control, audit logging with 12-month retention, vulnerability management
HIPAA Security Rule	US healthcare covered entities and business associates handling protected health information (PHI)	US HHSRegulation	Technical safeguards: access control, audit controls, integrity, transmission security; 60-day breach notification
CRA (Cyber Resilience Act)	Manufacturers placing "products with digital elements" (hardware and software) on the EU market — broader and less sector-specific than NIS2 or DORA	EU Regulation, entered into force December 2024; reporting obligations from 11 September 2026, full essential cybersecurity requirements from 11 December 2027	Machine-readable SBOM, vulnerability handling across the product lifecycle, incident reporting for actively exploited vulnerabilities (24h early warning / 72h full notification / 14-day final report)

These frameworks overlap significantly in their technical requirements. The controls they describe — access control, encryption, audit trails, network segmentation, incident response — are the same regardless of which framework applies. The differences are primarily in scope, reporting timelines, and retention periods.

CRA is the odd one out in one respect: NIS2, DORA, PCI-DSS, and HIPAA all regulate the entity *operating* a system, while CRA obligations fall on the *manufacturer* placing a product on the market. A team running Kubernetes purely to operate its own internal systems is not itself a CRA manufacturer — but the moment that cluster runs software the organisation ships to others (a SaaS product, an open-source project, an embedded component), CRA's vulnerability-handling and SBOM obligations apply to that software, and Part 5′s image scanning and signing controls are the technical starting point for meeting them.

Controls mapping

The table below maps the Kubernetes controls covered in this series to their regulatory anchors. "Part" references point to where the control is implemented and explained.

Kubernetes control	Series reference	NIS2	DORA	PCI-DSS v4.0	HIPAA	CRA
RBAC least privilege	Part 5	Art. 21(2)(i)	Art. 9	Req 7, 8	§164.312(a)(1)	—
`automountServiceAccountToken: false`	Part 5	Art. 21(2)(i)	Art. 9	Req 8.2	§164.312(a)(2)(i)	—
NetworkPolicy default-deny	Part 5	Art. 21(2)(h)	Art. 9	Req 1	§164.312(e)(1)	—
PSA `restricted` / SCCs	Part 5	Art. 21(2)(b)	Art. 9	Req 2	§164.312(c)(1)	—
etcd encryption at rest	Part 5	Art. 21(2)(h)	Art. 9	Req 3	§164.312(a)(2)(iv)	—
Image scanning + Cosign signing	Part 5	Art. 21(2)(d) supply chain	Art. 28	Req 6, 12.3	§164.312(c)(1)	Annex I Part II — SBOM, vulnerability handling
Kyverno admission enforcement	Part 5	Art. 21(2)(d)	Art. 9	Req 6	§164.312(c)(1)	—
External Secrets / no plaintext secrets	Part 5	Art. 21(2)(h)	Art. 9	Req 3	§164.312(a)(2)(iv)	—
Kubernetes API audit logging	This part	Art. 23 (evidence)	Art. 17	Req 10	§164.312(b)	—
GitOps + immutable delivery	Part 6	Art. 21(2)(e)	Art. 6, 8	Req 6	§164.312(c)(2)	—
PodDisruptionBudgets + HPA	Part 4	Art. 21(2)(e) availability	Art. 11	Req 12.5	—	—
Centralised log retention	Part 6	Art. 23	Art. 17	Req 10.5	§164.530(j)	—

Kubernetes API audit logging

The Kubernetes API server can log every request made against it: who called what, on which resource, from where, with what outcome. This is the primary source of the audit trail that compliance frameworks require — evidence that access controls are working, that changes are traceable, and that security events can be reconstructed.

Audit logging is controlled by an audit policy file provided to the API server, described in full in the Kubernetes auditing documentation. The policy defines which requests are logged and at what level of detail:

Level	What is recorded
`None`	Nothing — the request is suppressed from the audit log
`Metadata`	Request metadata only: user, verb, resource, timestamp, source IP — no body
`Request`	Metadata plus the request body (the object being created or modified)
`RequestResponse`	Metadata, request body, and the response body

A production-grade audit policy captures access to sensitive resources at high fidelity while suppressing high-volume low-value reads that would fill the log with noise:

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  # Suppress routine read-only requests to non-sensitive resources
  - level: None
    verbs: ["get", "watch", "list"]
    resources:
      - group: ""
        resources: ["events", "endpoints", "configmaps"]
      - group: "coordination.k8s.io"
        resources: ["leases"]

  # Full detail for Secret access — the primary sensitive resource
  - level: RequestResponse
    resources:
      - group: ""
        resources: ["secrets"]

  # Capture all writes at Request level — who changed what and to what
  - level: Request
    verbs: ["create", "update", "patch", "delete", "deletecollection"]

  # Capture all authentication events at Request level
  - level: Request
    users: ["system:anonymous"]
    verbs: ["*"]

  # Catch-all: metadata only for everything else
  - level: Metadata
```

Pass this file to the API server via `--audit-policy-file` and configure an output sink:

- `--audit-log-path` — write to a file on the control plane node (then ship with Fluent Bit)

- `--audit-webhook-config-file` — stream directly to a remote backend (Elasticsearch, Loki, etc.)

On managed clusters, audit logging is configured through the provider's control plane settings, not directly:

- **EKS** — control plane logs are shipped to CloudWatch Logs
- **AKS** — audit logs are sent to a Log Analytics workspace
- **GKE** — audit logs are routed to Cloud Logging
- **OVH MKS** — produces real, object-level `kube-apiserver` audit events under a fixed, non-configurable policy (Secret and ConfigMap changes at `Metadata` level in all namespaces, create/patch/update/delete verbs at `RequestResponse` level, everything else at `Request` or `Metadata` level)
- **STACKIT SKE** — currently only audits the SKE management API itself (cluster creation, node pool changes, credential rotation), not `kube-apiserver` object-level events; per the official SKE FAQ, integrating true Kubernetes API audit logs into the STACKIT Telemetry Router is "still in active development and not generally available yet." The delivery mechanism for the *existing* management-API audit log did move forward: STACKIT's Telemetry Router went GA on 2 June 2026, replacing the old Audit Log API, and can route audit data directly to an external OTLP or S3-compatible destination — not only STACKIT's own services.

Check the provider's documentation for enabling and configuring the audit log. For frameworks that require proof of who accessed or modified a specific Kubernetes object (NIS2, DORA, PCI-DSS Req 10), the SKE gap above is not closed by the platform today — an in-cluster audit webhook or sidecar approach would need to substitute until the managed integration ships.

Both platforms' audit-log tooling are moving targets. Everything in this section reflects what was verifiable on 1 July 2026 — re-check the provider's current documentation before relying on it for a compliance sign-off.

Where these audit logs end up, and what that costs

Producing the audit events is only half the picture. On both platforms, actually storing and querying them requires a separate, dedicated logging service — audit log retention is not bundled into the base Kubernetes service price.

OVH MKS forwards audit events by subscription (`POST /cloud/project/{serviceName}/kube/{kubeId}/log/subscription`) to a stream in OVH's own Logs Data Platform (LDP), a Graylog/OpenSearch-based service in the same OVHcloud account. There is no option to forward audit events directly to an external sink — routing them into a separately-run pipeline (for example the ClickHouse-based stack from the log backend comparison series) means reading them back out through LDP's OpenSearch or Graylog API and re-shipping them yourself. This is not a documentation gap: a direct S3-bucket export option for SIEM integration is an open, unaddressed request on OVH's public roadmap as of December 2025.

STACKIT routes its existing (management-API) audit log through the Telemetry Router, GA since 2 June 2026. Unlike OVH's LDP-only path, the Telemetry Router can send data straight to an external OTLP-compatible endpoint or S3-compatible storage — including a SIEM or bucket

you operate yourself — instead of only STACKIT's own services. That routing itself is billed separately from wherever the data ends up: the Telemetry Router charges for ingestion regardless of destination. If you route it to **STACKIT Logs** (Loki-based) as the destination, that service adds its own ingestion and storage cost on top, and enforces a hard 180-day retention cap that is not configurable per stream; STACKIT's own FAQ describes a dual-write pattern for retention beyond that cap: stream logs to STACKIT Logs and, in parallel, to a separate object storage bucket. Whether the still-pending `kube-apiserver` object-level integration will ship through this same Telemetry Router, and inherit its external-destination flexibility, is not confirmed yet.

	OVH Logs Data Platform	STACKIT Telemetry Router	STACKIT Logs (if used as destination)
Ingestion	Account creation free; Standard tier billed per volume ingested	≈€1.88 / 1024 MB ingested	≈€1.20 / GB ingested
Storage	14 days–1 year indexed, 1–10 years archived	not applicable — routes, does not store	≈€0.30 / GB per 30 days, capped at 180 days total
Fixed-price tier	Enterprise: ≈€299/month + ≈€199 setup fee — dedicated infrastructure, custom retention	not offered	not offered
Beyond the retention cap	Archived storage tier, up to 10 years	n/a	Customer-managed dual-write to a separate object storage bucket
Native export to an external sink	No — a direct S3 export option is an open, unaddressed roadmap request	Yes — routes to external OTLP/SIEM or S3 endpoints natively	n/a

Prices and feature availability as observed on 1 July 2026 — both platforms' audit-log products are under active development and can change on short notice; check current documentation before relying on these figures.

Every use case should document in writing how important its Kubernetes audit logs are and which of the Log retention requirements apply — that is what determines how much effort, and which of the solutions above, the audit log setup actually warrants. Given the per-GB pricing above, retention requirements can turn into a real cost factor once volume and duration add up, not just an engineering effort decision.

i OpenShift API audit log profiles

OpenShift configures the API server audit policy through the `APIServer` custom resource and offers three built-in profiles: `Default` (metadata for most requests, request/response for sensitive resources), `WriteRequestBodies` (writes at RequestResponse level), and `AllRequestBodies` (all requests at RequestResponse — highest volume). Select via:

```
oc patch apiserver cluster --type=merge \
  -p '{"spec":{"audit":{"profile":"WriteRequestBodies"}}}'
```

OpenShift routes audit logs to `/var/log/kube-apiserver/audit.log` on master nodes and exposes them via the cluster-logging stack if the OpenShift Logging Operator is deployed.

What the audit log contains

Each audit event is a JSON object. The fields that matter most for compliance purposes:

```
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Request",
  "auditID": "e9f7a421-...",
  "stage": "ResponseComplete",
  "requestURI": "/api/v1/namespaces/myapp/secrets/db-password",
  "verb": "get",
  "user": {
    "username": "ci-pipeline",
    "groups": ["system:authenticated"]
  },
  "sourceIPs": ["10.0.1.42"],
  "responseStatus": {"code": 200},
  "requestReceivedTimestamp": "2026-07-04T14:32:01.123456Z"
}
```

The combination of `username`, `verb`, `requestURI`, `sourceIPs`, and `requestReceivedTimestamp` answers the auditor's question: who accessed what, from where, and when.

Log retention requirements

Retaining audit logs for the required period is a separate concern from generating them. The retention obligation varies by framework:

Framework	Log type	Retention period	Accessibility requirement
NIS2	Incident-related records	National law varies — typically 1–3 years	Per incident reporting obligations
DORA	ICT-related incident logs	**5 years** (Art. 17)	Must be available on request to competent authority
PCI-DSS v4.0	Audit logs for cardholder data environment	**12 months**	**3 months immediately accessible** (Req 10.5.1)
HIPAA	Policy/procedure documentation	**6 years** from creation or last use	Available to HHS on request
GDPR	Processing activity records	Duration of processing + applicable legal basis	Available to supervisory authority

CRA is deliberately absent from this table: it does not prescribe a fixed log retention period at all, leaving the duration to a risk-based judgment tied to vulnerability handling and incident reporting timelines. What CRA does mandate with a fixed duration is different in kind — technical documen-

tation and the SBOM must be kept for **10 years** after the product is placed on the market, or for the support period, whichever is longer. That is a compliance-documentation retention obligation, not a security-log retention rule, and it should not be conflated with the frameworks above.

PCI-DSS v4.0 Requirement 10.5.1 deserves attention: the 3-month immediate accessibility requirement means that audit logs from the past 3 months must be queryable — not just archived to cold storage. A Loki, Elasticsearch, or ClickHouse cluster with a 90-day hot tier satisfies this; a setup where all logs older than 30 days go to Glacier or equivalent does not.

For the technical implementation of log storage backends and their cost/retention trade-offs — Elasticsearch/OpenSearch, Loki, Quickwit, ClickHouse — the managed log archiving series covers each option in depth, including provider-specific configurations for each managed Kubernetes platform. The log backend comparison covers query performance, ingestion rates, and storage costs across the five backends.

That choice of backend is not fully free on OVH MKS and STACKIT SKE if you rely on the platform's own audit-log service described above. OVH's Logs Data Platform can hold data for the required duration via its archived storage tier (up to 10 years), but archived logs are not queryable — only downloadable as compressed files — so anything past the 1-year indexed retention window loses the same live-accessibility property that Requirement 10.5.1 demands for PCI-DSS. On STACKIT, it depends on where the Telemetry Router points: STACKIT Logs on its own enforces a hard 180-day retention cap that cannot be extended, structurally short of DORA's 5-year and HIPAA's 6-year requirements — but since the Telemetry Router emits to multiple destinations in parallel, routing a second copy straight to your own long-term object storage alongside STACKIT Logs is a native configuration rather than a workaround. Either way, none of this closes the coverage gap from the section above — as of this writing, it only applies to the SKE management-API audit log, not the still-pending `kube-apiserver` object-level events. Check which retention tier a framework actually needs — queryable or merely retained — against what the platform's native service provides before assuming it is covered.

> ⚠ **GDPR and audit log interaction**
>
> Kubernetes audit logs contain IP addresses, usernames, and API call patterns — all of which may constitute personal data under GDPR. Organisations subject to both PCI-DSS (requiring 12-month retention) and GDPR (data minimisation principle) need a documented legal basis for retaining this data, typically legitimate interest or legal obligation. This tension should be resolved with legal counsel, not a technical default.

Incident reporting timelines

Regulatory incident reporting requires knowing what happened, when, and what was affected — which is the same information that the observability stack from Part 6 provides. The audit log and the application logs are the primary evidence for incident reports.

Framework	Initial notification	Intermediate report	Final report	To whom
NIS2	24h early warning	72h incident notification	1 month final report	National CSIRT / competent authority
DORA	4h after classification as major incident	72h intermediate report	1 month final report	Competent financial authority (e.g., ECB, national regulator)
PCI-DSS	Immediately on discovery (Req 12.10)	—	Forensic report after investigation	Acquiring bank / card brands
HIPAA	—	—	60 days after discovery	HHS; affected individuals; if >500 in state: media

The NIS2 24-hour early warning and DORA 4-hour notification are not expected to contain a full root-cause analysis — they are notifications that an incident occurred and is being managed. The final report, due one month later, is where the full timeline, affected systems, impact assessment, and remediation actions are documented.

What the Kubernetes environment contributes to incident reports:

- **API audit log** — proves who had access, what they did, and when; rules out (or confirms) insider access or credential misuse
- **Application logs** — reconstructs the sequence of events from the application's perspective
- **Metrics and alerts** — shows when anomalous behaviour started, which workloads were affected, and what the blast radius was
- **GitOps history** — proves what the desired state was at any point in time and whether unauthorised changes occurred

A cluster without centralised logging and a GitOps audit trail makes incident reporting significantly harder — the timeline has to be reconstructed from fragments rather than read from a structured record.

Compliance scanning

Automated compliance scanning tests the cluster's current configuration against known benchmark profiles. Two tools are widely used:

kube-bench

kube-bench runs the CIS Kubernetes Benchmark checks against the API server, controller manager, scheduler, and nodes. It is the starting point for assessing how much of the CIS benchmark the cluster already satisfies.

```
# Run as a Job on the cluster
kubectl apply -f https://raw.githubusercontent.com/aquasecurity/kube-bench/main/job.yaml
kubectl logs job/kube-bench
```

The output lists each check with a PASS/FAIL/WARN status and the remediation for failures. CIS Kubernetes Benchmark is not a regulatory requirement itself, but it maps to the technical controls

required by NIS2, DORA, PCI-DSS, and HIPAA — passing the benchmark provides a documented baseline.

i OVH MKS & STACKIT SKE — EU-native platforms and certification landscape

For organisations in regulated European industries, the platform choice affects the compliance baseline before a single YAML file is applied. Neither OVH MKS nor STACKIT SKE has the OpenShift Compliance Operator — kube-bench is the correct tool for CIS Kubernetes Benchmark checks on both platforms.

Certification	OVH MKS	STACKIT SKE
ISO 27001	✓	✓
SOC 2 Type II	—	✓
BSI C5 (German federal cloud standard)	—	✓
HDS (French health data hosting)	✓	—
SecNumCloud (French national security)	In progress	—

Neither OVH (a French company) nor STACKIT (Schwarz Digits, part of the Schwarz Group) is a US company, and as EU-headquartered providers they are generally not directly subject to the US CLOUD Act in the same way as US-based hyperscalers. The precise legal assessment can still depend on corporate structure and the jurisdictions involved — a definitive answer is a question for legal counsel, not a blanket guarantee. That caveat aside, for NIS2 and DORA, where data residency and third-party ICT risk management are explicit obligations, an EU-headquartered provider reduces the due-diligence burden compared to a US-headquartered one.

Audit logs and compliance scan results generated on these platforms can be stored in the same EU-resident object storage — without data leaving the regulatory jurisdiction.

CRA is deliberately not in the certification table above: it regulates products with digital elements, not the IaaS/PaaS service a managed Kubernetes offering is. Neither OVH MKS nor STACKIT SKE is a CRA-regulated product themselves — CRA becomes relevant for what you build and ship on top of the cluster, not for the cluster as a service.

OpenShift Compliance Operator

The OpenShift Compliance Operator runs OpenSCAP-based scans against cluster nodes and platform components. It ships profiles for:

- CIS Kubernetes Benchmark
- NIST SP 800-53 (moderate and high impact)
- PCI-DSS
- HIPAA

A `ScanSetting` and `ScanSettingBinding` trigger periodic scans:

```yaml
apiVersion: compliance.openshift.io/v1alpha1
kind: ScanSettingBinding
metadata:
  name: pci-dss-scan
  namespace: openshift-compliance
profiles:
  - name: ocp4-pci-dss
    kind: Profile
    apiGroup: compliance.openshift.io/v1alpha1
settingsRef:
  name: default
  kind: ScanSetting
  apiGroup: compliance.openshift.io/v1alpha1
```

Scan results appear as `ComplianceCheckResult` objects. Failed checks surface in the OpenShift console under the Compliance section and can be exported as reports for audit evidence.

> **Use scan results to prioritise remediation**
>
> kube-bench and the Compliance Operator produce long lists of findings. Not every FAIL is equally important — weight findings by the regulatory requirement they map to. A failing check on etcd encryption (PCI-DSS Req 3, HIPAA §164.312) carries more regulatory weight than a failing check on control-plane log rotation. Start remediation from the controls in the mapping table above.

What auditors look at in practice

The controls mapping table and compliance scans produce documentation. Auditors also ask for operational evidence — proof that the controls are not just configured but active and effective. The questions that come up repeatedly in Kubernetes compliance audits:

“Show me who has cluster-admin.”

```bash
kubectl get clusterrolebindings -o json | \
  jq '.items[] | select(.roleRef.name=="cluster-admin") | .subjects'
```

The answer should be a short list of named administrators — not service accounts, not broad groups like `system:authenticated`.

“Show me that privileged pods cannot run in production namespaces.”

```bash
kubectl get ns -o json | \
  jq '.items[] | select(.metadata.labels["pod-security.kubernetes.io/
enforce"] != null) |
    {name: .metadata.name, enforce: .metadata.labels["pod-
security.kubernetes.io/enforce"]}'
```

Every production namespace should show `restricted`. Namespaces without the label are unprotected and represent a finding.

"Show me the last 90 days of access to secrets in this namespace."

This is where the audit log backend is queried. The query should return the audit events for `verb: get, resource: secrets, namespace: myapp` with usernames, timestamps, and source IPs. Without a centralised audit log retained for 90 days, this question cannot be answered.

"Show me evidence that your image signing policy is enforced."

The Kyverno `ClusterPolicy` from Part 5 produces audit events when it rejects a pod. These events, combined with the policy definition itself in the GitOps repository, demonstrate that unsigned images cannot deploy.

"How do you know what changed and who changed it?"

The GitOps commit history. Every change to the cluster's desired state is a git commit with author, timestamp, and diff. The Argo CD or Flux sync events correlate git commits to the time they were applied. This is the audit trail for configuration changes.

Closing the loop

Compliance in Kubernetes is not a separate discipline from the security and operational practices covered in this series. It is the same practices, viewed through the lens of regulatory requirements and evidence production.

The security controls from Part 5 are the technical implementation of the access control and data protection requirements in NIS2, DORA, PCI-DSS, and HIPAA. The observability and audit logging from Part 6 are what turn those controls into verifiable evidence. The GitOps workflow is what makes change control traceable. The log retention setup determines how far back that evidence can be produced on demand.

The practical starting point for any organisation bringing a Kubernetes cluster into a regulated scope:

1. Run kube-bench or the Compliance Operator and document the current baseline
2. Map the gaps to the controls table above and prioritise by regulatory weight
3. Enable and retain Kubernetes API audit logs at the level and duration your framework requires
4. Confirm your log storage backend satisfies the immediate-accessibility requirement for your retention period (particularly PCI-DSS 3-month hot tier)
5. Document the incident response procedure that uses the cluster's audit and application logs
6. Schedule quarterly reviews — compliance is a continuous state, not a one-time certification

---

Partly used sources — specific references for this part:

- NIS2 Directive (EU) 2022/2555 — European Parliament and Council (2022)
- DORA Regulation (EU) 2022/2554 — European Parliament and Council (2022)
- PCI DSS v4.0 — PCI Security Standards Council (2022)
- HIPAA Security Rule — 45 CFR Part 164 — US HHS
- CIS Kubernetes Benchmark — Center for Internet Security
- Kubernetes Best Practices I Wish I Had Known Before — Pulumi (2025, updated 2026)
- kubernetes-best-practices — Diego Lima, Apache-2.0