CHECKLIST

NoSQL Database Security Checklist for Access Control and Encryption

A practical, verifiable checklist for securing NoSQL databases with access control and encryption. Use it to validate permissions, TLS, encryption at rest, key handling, logging, and production readiness before rollout.

Databases - July 05, 2026

Purpose

NoSQL databases often fail security reviews for the same operational reasons: overly broad permissions, missing transport encryption, weak key handling, and incomplete audit trails. Those gaps are risky because a single compromised service account, misconfigured network path, or exposed backup can create immediate data exposure across large document, key-value, column-family, or graph datasets.

This checklist helps you verify whether a NoSQL deployment is ready for production from an access control and encryption standpoint. By the end, you should be able to decide whether the security design fits the workload, validate the controls that matter most, capture evidence for review, and confirm what still needs to be fixed before go-live.

How to use this checklist

Use this checklist during design review, pre-production validation, and periodic re-assessment. Work through each operational phase in order, capture evidence as you go, and record any failed checks as remediation items with an owner and due date.

Treat each checkbox as a verifiable action, not a statement of intent. If you cannot produce evidence for a control, assume the control is not ready. When role scoping is a major control boundary, the companion workflow in How to Secure NoSQL Databases with Role-Based Access Control can help you validate role design and permission boundaries before rollout.

1) Scope and data classification

Purpose

Confirm what the database stores, who uses it, and what security controls are required based on data sensitivity and access patterns. This phase prevents you from applying weak defaults to sensitive data or overengineering controls for low-risk data.

Checklist items

- ? Confirm the data classes stored in each collection, table, bucket, or namespace, including regulated, confidential, and internal-only data.
- ? Review which applications, batch jobs, operators, and integration services must access the database.
- ? Validate whether the deployment contains any production data, replicated production snapshots, or sensitive test data.
- ? Document the trust boundaries between client applications, admin networks, backup systems, and replication peers.
- ? Assign an explicit data owner and system owner for each database instance or cluster.
- ? Confirm whether the deployment is single-tenant, shared, or multi-environment, and whether that affects access segregation.
- ? Validate whether the platform requires encryption, auditing, or key management based on policy, contract, or regulation.

Evidence to capture

Record the data classification register, architecture diagram, application dependency list, and the owner matrix. Include any policy references that make encryption or access restrictions mandatory.

Acceptance criteria

The team can explain what data is stored, who needs access, and which control requirements apply. Every sensitive dataset has an owner and an identified protection level.

Owner and review cadence

Owner: system owner or data owner. Review cadence: at onboarding, before production changes, and quarterly.

Common mistakes

The most common failure is assuming a non-relational database is automatically less sensitive than a relational one. Another frequent issue is allowing developers, support staff, and automation services to share the same access path without documenting why.

2) Authentication and access control

Purpose

Ensure that only named identities can connect, and that each identity is limited to the minimum required permissions. This is the control that reduces the blast radius of leaked credentials, compromised hosts, and misused admin access.

Checklist items

- ? Confirm that every human and non-human identity uses a unique account or identity provider binding.
- ? Validate that shared accounts are disabled or formally exception-approved with compensating controls.
- ? Review role definitions to ensure they map to job functions rather than individual users or ad hoc tasks.
- ? Confirm that write access, schema changes, administrative functions, and backup access are separated where the platform supports it.
- ? Validate that application service accounts cannot perform interactive administrative actions.
- ? Test denied-access behavior for at least one prohibited operation per major role.
- ? Document how privileged access is requested, approved, time-limited, and revoked.
- ? Confirm that newly provisioned accounts inherit only the expected baseline permissions.
- ? Review whether direct database access is restricted for engineers who only need operational observability.

Evidence to capture

Keep role mappings, permission exports, deny-test results, approval records, and provisioning screenshots or API output. If the platform supports fine-grained access control, capture examples showing read, write, and admin boundaries.

Acceptance criteria

Every identity has an identifiable purpose, privilege is limited to the minimum required scope, and denied actions fail as expected. If you are still defining who should get which role, the design should be aligned with How to Secure NoSQL Databases with Role-Based Access Control.

Owner and review cadence

Owner: security engineer or database administrator. Review cadence: before production enablement, after any role model change, and monthly for privileged roles.

Common mistakes

Avoid giving application teams broad admin access ?temporarily? and never removing it. Also avoid using one service identity for multiple applications, because that makes audit trails and containment far less reliable.

3) Network exposure and transport protection

Purpose

Reduce the chance that credentials or data are exposed while in transit or through unnecessary network paths. Access control is much weaker if clients can connect from anywhere over an untrusted path.

Checklist items

- ? Confirm that the database listens only on approved interfaces and ports.

- ? Validate that remote administrative access is restricted to approved management networks or bastions.
- ? Review security groups, firewall rules, network policies, or ACLs to confirm least-exposure rules.
- ? Test that client connections use TLS or an equivalent encrypted transport protocol.
- ? Confirm that certificate validation is enabled on clients and drivers.
- ? Validate that weak protocols, anonymous access, and fallback plain-text modes are disabled.
- ? Document which internal services may connect directly and which must use a proxy, gateway, or private link.
- ? Review whether replication, backup transfer, and administrative APIs also use encrypted transport.

Evidence to capture

Capture listener or binding configuration, firewall or network policy rules, client connection settings, and a test connection showing encrypted transport. Record any exceptions where a trusted internal segment is used and why it is acceptable.

Acceptance criteria

Only approved sources can reach the service, and all supported data paths use encrypted transport. There should be no unexplained exposed listener or open management port.

Owner and review cadence

Owner: network engineer or platform engineer. Review cadence: at deployment, after network changes, and monthly.

Common mistakes

A frequent gap is securing application traffic but leaving backup, replication, or admin traffic unencrypted. Another common issue is relying on IP filtering alone instead of pairing it with authenticated transport.

4) Encryption at rest and key management

Purpose

Ensure that stored data, snapshots, and backups remain protected if disks, exports, or storage accounts are accessed outside the database control plane.

Checklist items

- ? Confirm that encryption at rest is enabled for the database storage layer or underlying volume layer.
- ? Validate whether native database encryption, disk encryption, cloud storage encryption, or a layered approach is in use.
- ? Review whether backups, exports, replicas, and snapshots inherit the same encryption standard.
- ? Confirm that encryption keys are protected in a controlled key management system or equivalent trusted service.
- ? Document key ownership, rotation expectations, recovery responsibilities, and emergency access procedures.
- ? Validate that key rotation does not require an undocumented service outage or manual re-encryption workaround.
- ? Test that a restore or failover path can use the required keys without exposing plaintext secrets to operators.
- ? Confirm that old keys are retired according to policy and that access to retired material is removed.

Evidence to capture

Collect encryption settings, key management policy, backup configuration, restore procedure notes, and any validation output showing that encrypted data can be restored successfully. If the platform offers customer-managed keys, verify the exact service tier or edition required before assuming the feature is available.

Acceptance criteria

Data remains encrypted when stored and during recovery workflows, and keys are managed under a documented control process. Backups and snapshots are not a weaker link than the live database.

Owner and review cadence

Owner: security engineer with storage or cloud platform support. Review cadence: before production use, after key changes, and at least quarterly.

Common mistakes

Teams often validate encryption on the primary database but forget replicas, object storage backups, or exported archives. Another frequent mistake is storing key material in the same administrative boundary as the database itself.

5) Secrets handling and client authentication material

Purpose

Protect the credentials, certificates, tokens, and connection strings used by applications and operators to reach the database. Strong database controls are undermined if secrets are embedded in code or exposed in logs.

Checklist items

- ? Confirm that database passwords, certificates, and tokens are stored in approved secret management systems.
- ? Validate that connection strings do not contain long-lived secrets in plain configuration files or source code.
- ? Review whether certificates are rotated before expiry and whether client trust stores are maintained.
- ? Test that secret retrieval is limited to the identities that need it.
- ? Confirm that secrets are not printed in startup logs, debug output, or exception traces.
- ? Document the rotation process for application credentials and the rollback procedure if a rotation fails.
- ? Validate that revoked credentials stop working within the expected propagation window.

Evidence to capture

Keep secret store policy references, sample redacted configuration, rotation records, and test results showing failed access after revocation or rotation.

Acceptance criteria

No production credential is hard-coded, broadly exposed, or shared across unrelated workloads. Secret access is controlled, auditable, and revocable.

Owner and review cadence

Owner: application owner or platform security engineer. Review cadence: at onboarding, after credential changes, and every release that touches database connectivity.

Common mistakes

A common issue is secure storage for the main password but weak handling for certificates, API keys, or export credentials. Another is leaving debug logging enabled long enough to capture sensitive connection data.

6) Audit logging and monitoring

Purpose

Verify that you can detect unauthorized access, permission misuse, and security-relevant configuration changes. A secure design must produce evidence after an incident, not just prevent a subset of failures.

Checklist items

- ? Confirm that authentication successes and failures are logged.
- ? Validate that privilege changes, role assignments, and admin operations generate audit events.
- ? Review whether data access logs capture enough context to identify the actor, action, resource, and outcome.
- ? Test that critical events reach the central logging or SIEM pipeline without being silently dropped.
- ? Confirm that logs are time-synchronized and protected from alteration.
- ? Document alert thresholds for repeated failures, unusual source locations, and unexpected privilege escalation.

- ? Validate retention periods against investigation and compliance requirements.

Evidence to capture

Retain sample audit entries, log forwarding configuration, alert rules, retention settings, and a test event showing end-to-end delivery to the monitoring system.

Acceptance criteria

Security-relevant actions are logged, forwarded, and retained long enough to support investigation. You can trace who did what, when, and from where.

Owner and review cadence

Owner: security operations or observability owner. Review cadence: continuously for alerting, monthly for log coverage, and after any schema or platform upgrade.

Common mistakes

Do not assume the database?s default logs are sufficient for forensic work. Also do not keep audit events only on the database host if the host itself is a likely target.

7) Backup, restore, and recovery controls

Purpose

Ensure that encrypted data can be restored safely and that backup access does not bypass production controls. Backup workflows frequently become the easiest path to data exposure.

Checklist items

- ? Confirm that backup jobs run under a dedicated identity with only the required permissions.
- ? Validate that backup artifacts are encrypted in transit and at rest.
- ? Review who can download, mount, restore, or export backup content.
- ? Test a restore into a non-production environment using the documented recovery procedure.
- ? Confirm that restored data is handled with the same access restrictions as production data.
- ? Document recovery point objective, recovery time objective, and the last successful restore date.
- ? Validate that recovery key access, backup credentials, and restore approvals are separately controlled.

Evidence to capture

Keep backup job definitions, restore test output, encryption settings, and approval records. Include the date and result of the last successful recovery exercise.

Acceptance criteria

You can restore data successfully without weakening encryption or granting excessive access. Backup and restore paths are controlled as tightly as the primary database.

Owner and review cadence

Owner: backup administrator or platform engineer. Review cadence: after backup changes and at least quarterly restore testing.

Common mistakes

The most common problem is testing backup completion but not restore validity. Another is giving too many people access to backup repositories because they are treated as operational assets instead of sensitive data stores.

8) Pre-production validation and readiness gate

Purpose

Confirm that the controls above work together before the database is exposed to real workload traffic. This phase turns individual checks into an operational go/no-go decision.

Checklist items

- ? Validate that a clean account cannot access data or functions beyond its role.
- ? Confirm that all application paths connect over encrypted transport.
- ? Test that denied network paths fail closed rather than falling back to weaker connectivity.
- ? Validate that audit events appear in the monitoring system after login, privilege change, and data access activity.
- ? Confirm that a backup or restore can be completed with the approved key and credential workflow.
- ? Review open findings and assign each one a remediation owner and due date.
- ? Document the final go/no-go decision and the residual risk accepted for launch.

Evidence to capture

Store the validation script output, test cases, deny results, audit samples, restore evidence, and the final approval record.

Acceptance criteria

The system passes the minimum security checks needed for production and any exceptions are explicitly approved. No critical control depends on tribal knowledge or undocumented manual steps.

Owner and review cadence

Owner: change owner or release manager. Review cadence: every release or infrastructure change that can affect access, encryption, or logging.

Pass/fail criteria

Use simple decision rules so the review is not subjective.

Pass

- All mandatory controls are present and verifiable.
- No critical findings remain open.
- Exceptions are documented, approved, time-bound, and have compensating controls.
- Restore, audit, and denied-access tests succeed.

Fail

- Any production identity has excessive or undocumented privileges.
- Any approved connection path uses unencrypted transport where encryption is required.
- Backup, restore, or replica paths bypass the same protection standard as the primary database.
- Audit logs are incomplete, inaccessible, or not forwarded.
- Key ownership, rotation, or recovery is undefined.

Readiness scoring

Score each phase from 0 to 2:

- 0 = missing or unverified
- 1 = partially implemented or verified only by documentation
- 2 = implemented and tested with evidence

Add the scores from all eight phases for a maximum of 16.

- 0?7: Not ready for production. Fix core control gaps before proceeding.
- 8?12: Conditional readiness. Proceed only with approved exceptions and a dated remediation plan.
- 13?16: Operationally ready, assuming evidence is current and restore/audit tests are within cadence.

Use the score as a decision aid, not a substitute for a failed critical control. One failed encryption or access-control requirement can override a high total score.

Concrete follow-up actions

If the checklist exposes gaps, keep remediation narrow and measurable:

- Reduce privilege first by removing shared accounts, collapsing unused roles, and separating admin from application access.
- Enable encryption paths for both transport and stored data, then re-test client compatibility and restore workflows.
- Fix key ownership so rotation, recovery, and retirement are assigned to named operators with documented procedures.
- Close logging gaps by forwarding audit events centrally and validating alerting on failed logins and privilege changes.
- Repeat restore and deny tests after every significant change until the controls pass consistently.

A NoSQL database is ready when access is explicitly limited, encryption is verifiable end to end, and you can prove recovery and auditability under operational conditions. If you cannot show that evidence, the deployment is not ready yet.

Use this guidance together with ORA-12514 to connect the workflow with related operational context already available on the site.

Use this guidance together with Ubuntu security hardening checklist to connect the workflow with related operational context already available on the site.