Liquibase

LIQUIBASE SECURE · TECHNICAL WHITE PAPER

Change Intelligence: Technical Overview

Architecture, Security, and Governance

Summary

Liquibase Change Intelligence brings deployment visibility, risk remediation, insight, and audit readiness to database delivery without introducing new attack surface, new credential paths to production databases, or new operational dependencies on Liquibase managed infrastructure. It is the visibility module of Liquibase Secure Server, which runs entirely inside the customer's trust boundary. It derives its data from operations the customer is already running with the Liquibase Secure CLI, and it inherits the server's permission model, designed for the access control patterns of regulated environments.

For platform leaders, DevOps teams, DBAs, and security and compliance teams, Change Intelligence provides the visibility required to identify risk earlier, triage failures faster, and capture audit evidence automatically, without slowing delivery.

Executive Summary

Modern database delivery presents enterprise teams with a paradox. The pace and volume of database change continues to accelerate across cloud, hybrid, and on-premise environments, yet most organizations still operate without a unified record of what changed, where, when, and by whom. Failures live in fragmented log systems. Drift surfaces only when it causes an incident. Audit evidence is reconstructed by hand from screenshots, tickets, and email threads.

Liquibase Change Intelligence is the visibility and intelligence layer for governed database delivery. It captures structured metadata from every Liquibase operation, presents it through a single dashboard, and provides the audit, drift detection, and AI assisted analysis capabilities that enterprise teams need to investigate faster, reduce risk, and remain audit ready.

Change Intelligence runs inside Liquibase Secure Server, which is deployed entirely inside the customer's trust boundary. It never connects to monitored databases. All operational data flows from the Liquibase Secure CLI already running in customer pipelines. The result is observability with a minimal security surface: no new credential paths to production data, no required outbound traffic when AI Analysis is configured against a self-hosted model, and no dependency on Liquibase managed infrastructure.

This white paper provides the architectural and security detail required to support enterprise architecture and security reviews of Change Intelligence. Liquibase does not publish a separate Secure Server reference: the chapters on deployment, security posture, access tokens, authorization, the system log, and sizing describe

© 2026 Liquibase Inc. All Rights Reserved

Liquibase Secure Server as a whole, because Change Intelligence shares that infrastructure with the server's Core services and with Change Governance.

Key Outcomes for Enterprise Teams

- Reduce time to resolution by investigating deployment issues with the right context in a single interface.
- Detect drift earlier by monitoring schema, policy, and operational patterns continuously.
- Improve delivery visibility across environments, teams, and projects.
- Strengthen audit readiness with an immutable, queryable record of change activity, and a system log of every permission, access token, and license change.

The Visibility Problem in Database Change

Across most enterprise environments, database change has a structural visibility problem. Deployments move across environments without a unified record. Drift appears outside the change process and goes undetected until it causes a failure. Operational failures live in disparate log systems that require correlation by hand. Audit evidence is stitched together from screenshots, ticket histories, and email threads when a compliance review arrives.

Most teams already have the underlying data. What they lack is a coherent way to see what happened, where risk is accumulating, and what needs attention next.

Change Intelligence addresses this gap by treating Liquibase operations as first class telemetry. Every deployment, policy check, and drift detection run produces a structured operation record that flows into a governed observability layer.

Capabilities Overview

One View Across the Database Estate

Track how changes move from development through test and staging into production. Spot gaps, failed steps, skipped environments, and unexpected divergence in a single interface. DORA metrics, including deployment frequency, change failure rate, rollback rate, and cycle time, are computed continuously from operation data and grouped alongside policy and drift signals.

© 2026 Liquibase Inc. All Rights Reserved

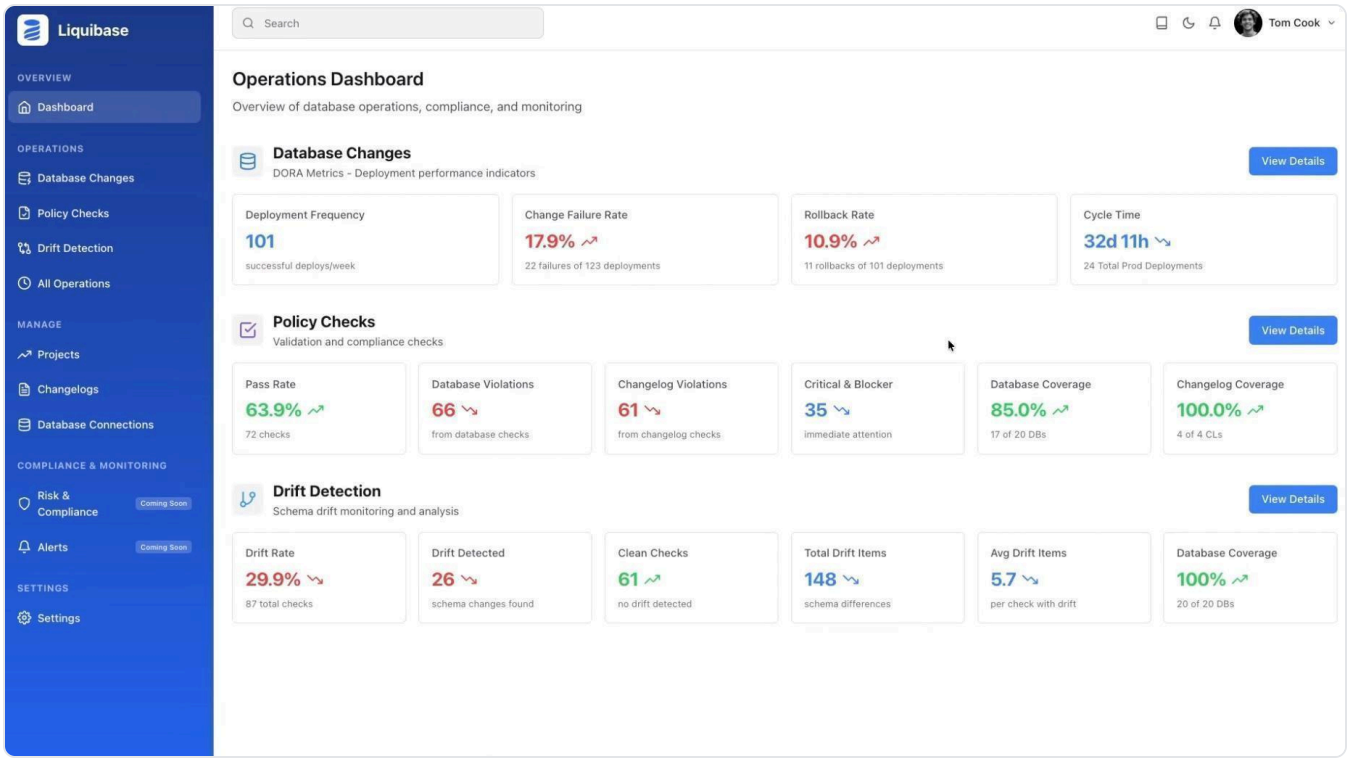


Figure: The Operations Dashboard consolidates DORA metrics, policy check outcomes, and drift detection results into a single surface.

Drilling into Database Changes exposes the DORA trend over time and the underlying operation list, with filters by status, type, environment, and project.

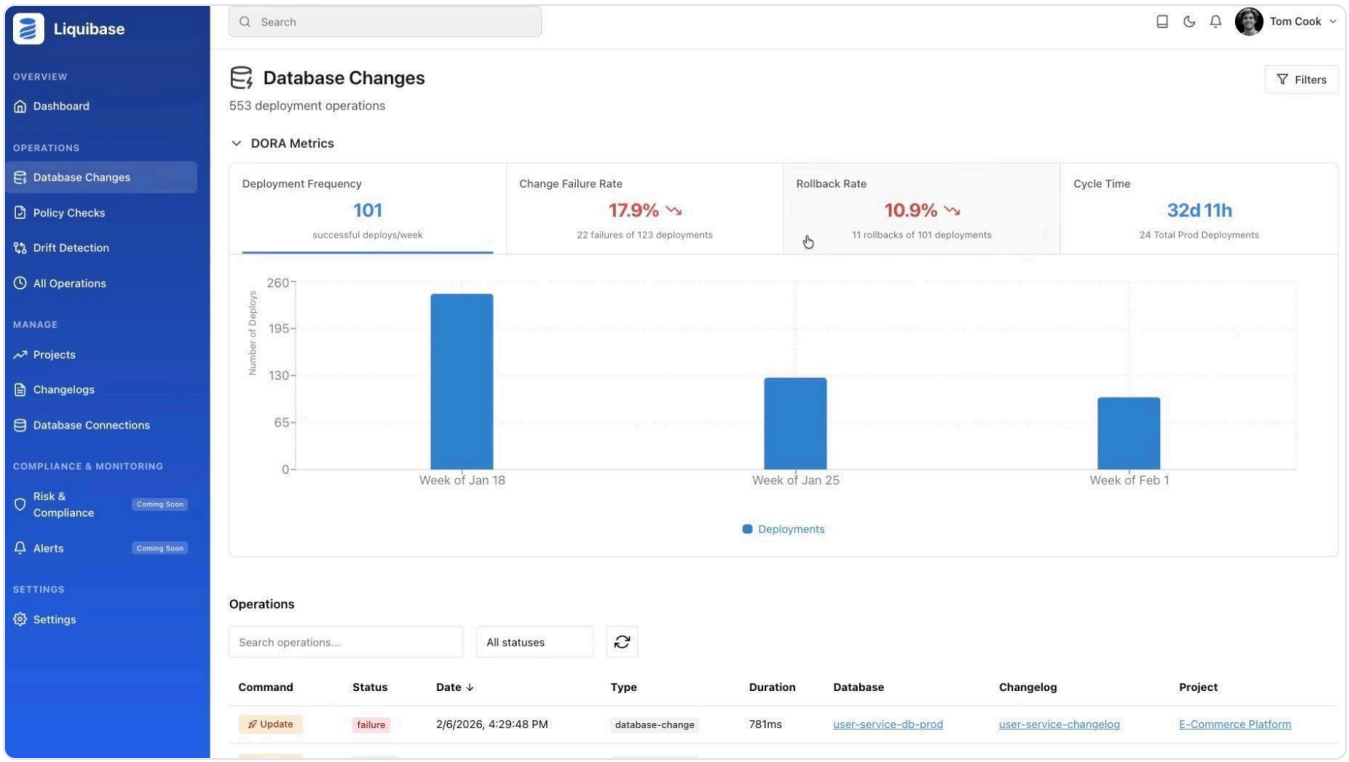


Figure: Database Changes view, showing DORA metric history and the operation log for drill down investigation.

© 2026 Liquibase Inc. All Rights Reserved

Move From Signal to Root Cause, Fast

Investigate failed deployments with full operation history, structured execution logs, and environment context. On-demand AI analysis can summarize a failure, identify likely causes, and propose remediation steps without requiring the user to navigate between disconnected systems.

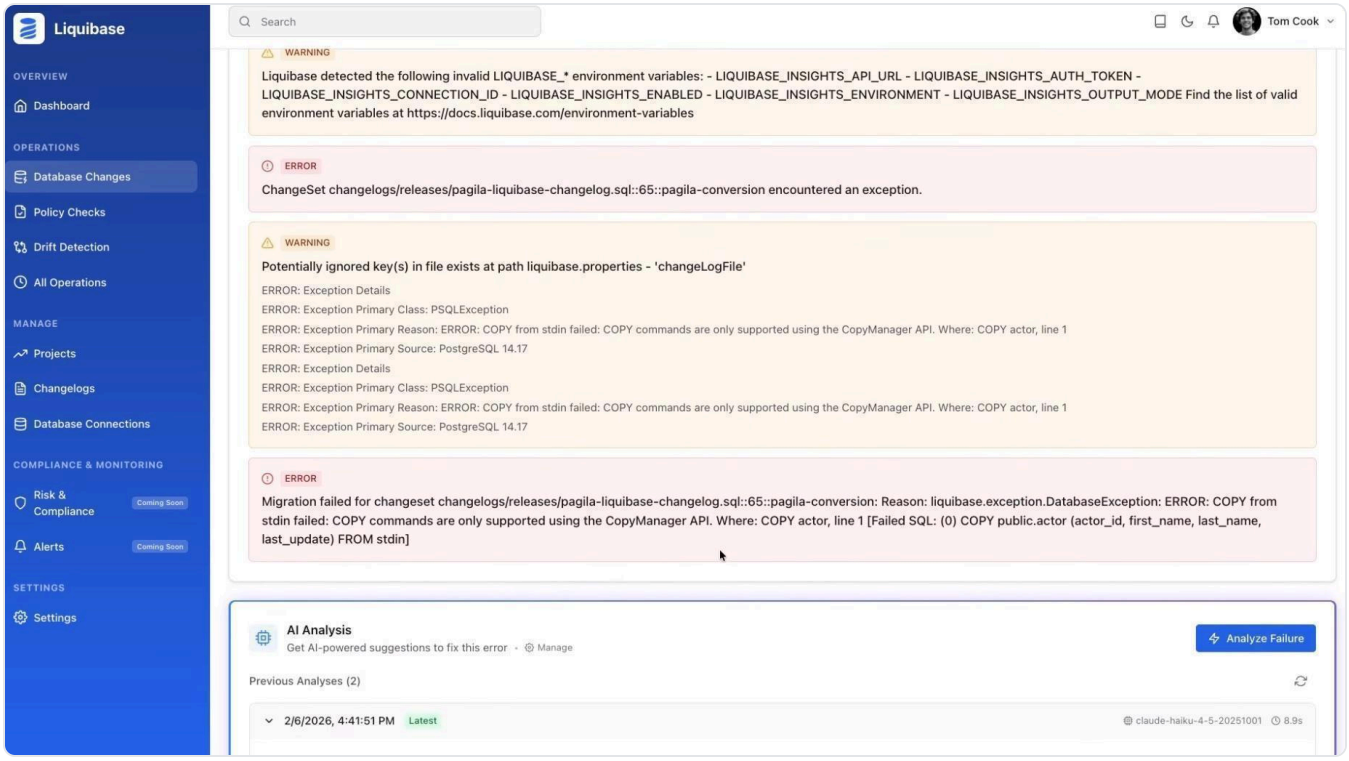


Figure: A failed ChangeSet with warning and error context, ready for AI Analysis. Model and provider are customer configurable.

Catch Drift Early

Monitor schema drift, policy outcomes, and operational patterns continuously. Identify out of process changes before they create delivery, security, or compliance incidents. Drift detection runs at the cadence the customer configures (typically as a scheduled CI/CD job), and the results populate a trend view alongside individual drift findings.

© 2026 Liquibase Inc. All Rights Reserved

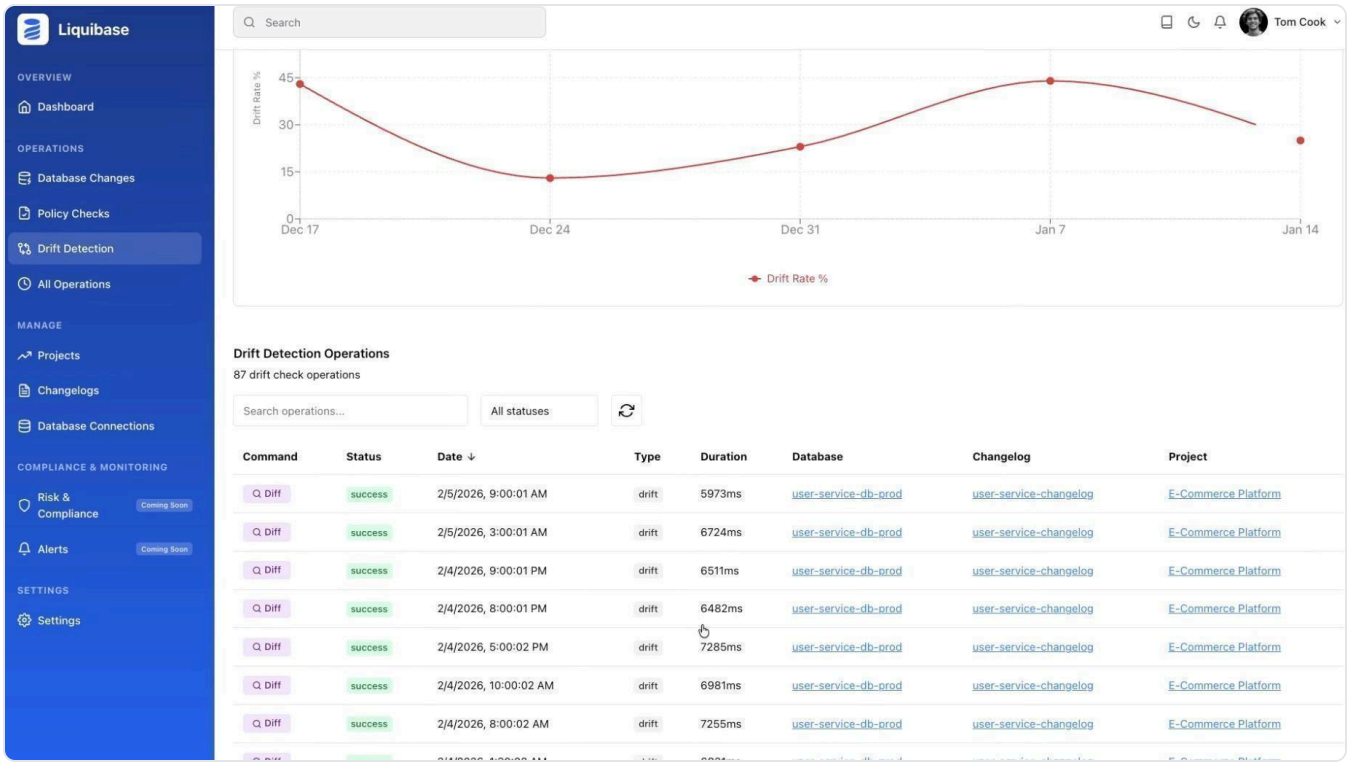


Figure: Drift Rate trend over time, computed from successive drift detection runs across monitored databases.

Each drift finding includes a side-by-side diff between expected and observed state, plus the execution log that produced the finding. The combination provides reviewable evidence for compliance teams and a fast investigation path for engineering teams.

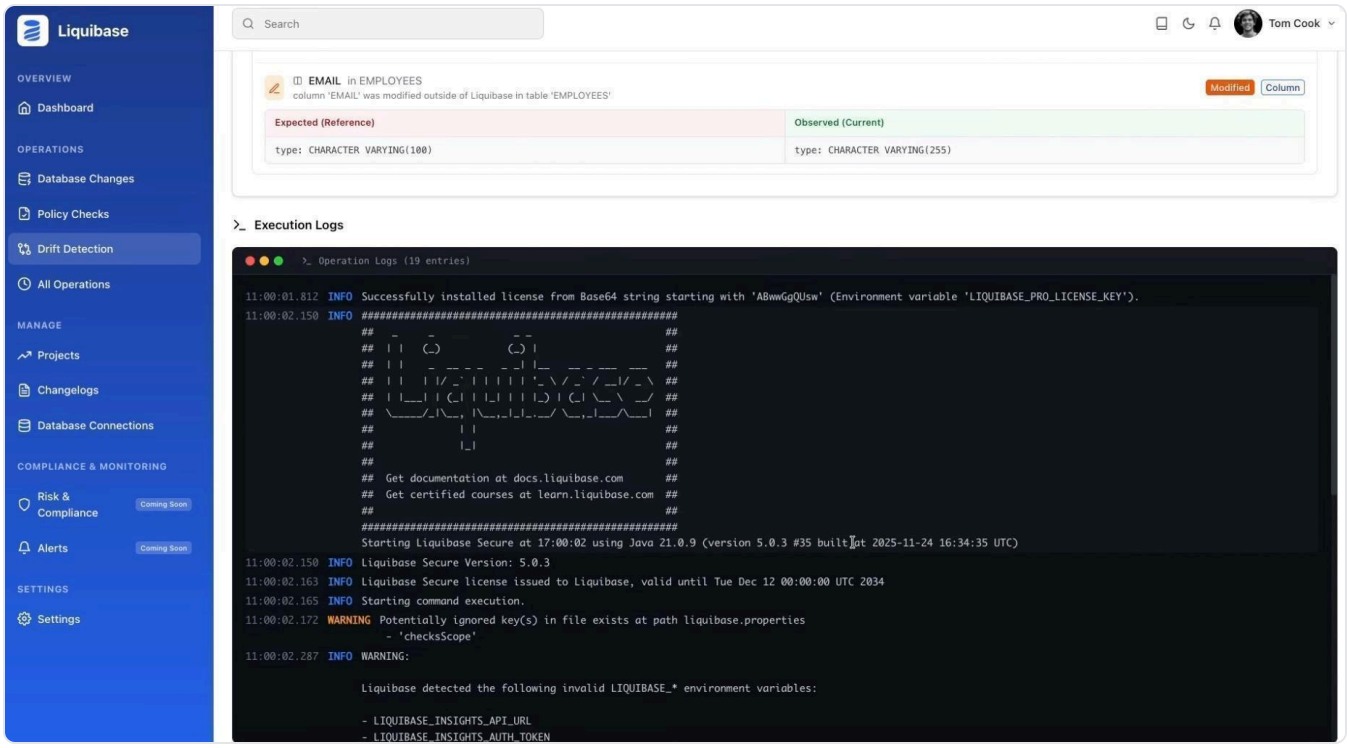


Figure: Drift detail showing an out of process column change (CHARACTER VARYING(100) to (255)) alongside the originating execution log.

© 2026 Liquibase Inc. All Rights Reserved

Audit Ready by Design

Capture an immutable record of permission and configuration changes alongside operation evidence. Security reviewers can answer questions like “who had access to this project on this date” without needing elevated permissions on other systems.

Designed for Enterprise Personas

Platform leaders, DevOps engineers, DBAs, security reviewers, governance leads, and compliance teams each have a default permission template tuned to their workflow. Permissions are governed centrally by Liquibase Secure Server, audited continuously, and extensible without forking the model.

What Change Intelligence Is, and Is Not

Change Intelligence is the observability module of Liquibase Secure Server for Liquibase Secure change governance operations. It captures metadata from each Liquibase Secure CLI run, including deployments, policy checks, and drift detection results, and presents it through the Secure Server web application. The module enables teams to monitor, audit, and triage database change activity across environments, projects, and databases.

Change Intelligence is not a database client, and neither is Secure Server. The server never connects to a customer’s monitored databases. Every operational data point originates from the Liquibase Secure CLI running inside the customer’s existing pipelines and developer workflows.

This architecture is intentional and has important security implications:

- Change Intelligence has no path to query production data.
- There is no requirement for inbound database connectivity from Secure Server.
- Exposure to customer schemas is limited to the metadata Liquibase already generates during normal operation.
- Sensitive Liquibase properties (passwords, secrets, tokens, keys, credentials, and similar patterns) are stripped at the extension before any data leaves the originating host. The extension also omits the absolute working directory, sends the local operating system username only when the operator opts in, and masks credentials in connection strings that appear in error messages and stack traces.

Change Intelligence is also not a standalone application. It has no API, web tier, database, or authentication of its own; it is a licensed module that runs inside Liquibase Secure Server alongside the server’s Core services and Change Governance, and it is administered through the same workspace, groups, templates, and system log.

Liquibase Secure Platform Architecture

Liquibase Secure is delivered as two components. Both run inside the customer’s trust boundary, and Change Intelligence is a module of the second.

Liquibase Secure CLI. The command line tool that installs on developer workstations and CI/CD agents and implements Change Automation: it reads changelogs, previews SQL, runs policy checks, applies and rolls back change, detects drift, and writes reports and structured logs. It makes outbound connections only and holds no

© 2026 Liquibase Inc. All Rights Reserved

credentials beyond the moment of execution. The Secure CLI also hosts the Change Intelligence extension, which is bundled in the distribution and disabled by default, and its checks commands can resolve policy configuration from Change Governance instead of a local file. The Secure CLI is described in the companion Change Automation Technical Overview.

Liquibase Secure Server. A self-hosted, containerized server that the customer deploys and operates. It never connects to a managed database; everything it knows arrives from Secure CLI instances running in the customer's own pipelines or from users in the browser. Secure Server hosts three functional areas behind one API and one web application:

- **Core services**, present in every installation: account and workspace services; project management and the shared entity model (projects, database connections, changelogs, and pipelines); license management and tracking; role-based access control (the permission catalog, templates, groups, overlays, grants and denies, and the resolver); and the other administration services (users, service principals, access tokens (service tokens and personal access tokens), single sign-on configuration, and the system log).
- **Change Intelligence**, the licensed visibility module: it receives operation metadata from the extension, files it under the shared entity model as operations, and presents deployment history, DORA metrics, policy check outcomes, drift findings, and on-demand AI analysis.
- **Change Governance**, the licensed policy module: it holds the organization's policy library as catalogs, packages, and checks, and maps that library to projects, pipelines, connections, and changelogs through Assignments that the Secure CLI resolves at runtime of checks operations.

The modules are not separate applications. They share the Secure Server API and web tiers, the data stores, the authentication layer, and the access control model. Each module registers its own permission domain with the Core RBAC catalog and contributes grants to the seven default templates, so administering access to Change Intelligence uses the same groups, templates, and system log as everything else in the server. Licensing is enforced at the same point: a module the installation is not licensed for has its permissions made inert and its surfaces closed, while the Core services remain fully administrable.

Within this architecture, Change Intelligence occupies two places: the extension in the Secure CLI, which is the sole producer of operation data, and the module in Secure Server, which stores and presents it. Everything between them, from authentication and RBAC to the data stores and the reverse proxy, is Core infrastructure that Change Governance uses in exactly the same way. The rest of this document describes the Change Intelligence-specific parts first, then the shared server infrastructure as it applies to both modules.

Solution Architecture

Change Intelligence consists of three cooperating parts, plus one optional external dependency. The first runs in the Secure CLI, the second and third run inside Liquibase Secure Server, and all of them run inside the customer's trust boundary. The fourth, the LLM provider used for AI Analysis, is optional and customer configurable.

Liquibase Extension

The Change Intelligence extension ships bundled inside the Liquibase Secure CLI distribution and is disabled by default. Operators opt in by setting LIQUIBASE_PLATFORM_ENABLED=true and providing other configuration data used in the secure transmission of operation data from the Secure CLI to the Secure Server; there is nothing to

© 2026 Liquibase Inc. All Rights Reserved

download and no manual install step. The extension hooks into the Secure CLI lifecycle, captures operation metadata as JSON, strips sensitive properties, and posts the result to the Secure Server ingest endpoint. It is the sole producer of operational data for Change Intelligence; the server accepts ingest through no other path.

Change Intelligence is a Liquibase Secure capability. The extension reads the runtime license and disables itself when no valid Secure license is present, leaving the underlying Liquibase command to run normally. The server can also be configured to enforce the same rule, rejecting any batch that does not declare a licensed Secure client.

Compatibility: Liquibase Secure CLI 6.0 or newer.

The extension authenticates with a service token. If it is configured with a personal access token, the server refuses the operation data; the extension prints a single notice and the command finishes normally, so a developer's local run is not failed by it. An expired, revoked, or suspended personal access token stops the command before it runs.

When Liquibase writes an operation report, the extension uploads it with the operation so it is available under View Report. Report paths may be local, s3://, or az://; cloud paths are read with the standard AWS or Azure credential chain on the CLI host.

Change Intelligence Module in the Secure Server API

The Secure Server API is a NestJS service written in TypeScript. The Change Intelligence module contributes the ingest endpoint and the operation, dashboard, drift, tag, environment, and AI Analysis routes to that API. It validates and persists operation data under the shared entity model (workspace, project, pipeline, changelog, connection), relies on the Core services for tenancy and authorization, serves the REST endpoints consumed by the web application, and emits real time events over WebSockets via Redis pub/sub. The API tier holds no state of its own; in the supported deployment it runs as a single container behind the reverse proxy.

Change Intelligence Surfaces in the Secure Server Web Application

The Secure Server web application is a Next.js application written in React, served by its own Node process behind a reverse proxy: either a customer-provided reverse proxy or an nginx container managed by the Secure Server Docker Compose configuration. Change Intelligence contributes the Operations Dashboard, the drill down views for operations and drift findings, and the AI Analysis surface to that application. The surfaces for the shared entities (projects, pipelines, connections, and changelogs), together with the system log and the permission management surfaces, belong to the Core services and are shared with Change Governance. The web application communicates with the API over HTTPS for REST traffic and over WSS for real time updates.

AI Analysis (On Demand)

When a user explicitly invokes "Summarize" or "Analyze Failure" on a specific operation, the API calls the configured LLM provider. Supported providers are OpenAI, Anthropic, Google, and any OpenAI-compatible endpoint, which is the path for self-hosted and local models. Important properties of this design:

- The provider, model, and endpoint are set once per Secure Server installation, so a single operator-controlled decision governs where inference happens. AI Analysis is then enabled or disabled per workspace, and is off unless enabled.

© 2026 Liquibase Inc. All Rights Reserved

- There is no automatic outbound LLM traffic.
- No inference runs on operations the user has not intentionally requested.
- Results are persisted with the originating operation, attributed to the provider, model, and timestamp that produced them.
- Because Liquibase operations are immutable, prior analyses do not become stale relative to their subject and do not silently regenerate.

Data Flow

The data flow follows a single direction from producer to consumer:

1. The Liquibase Secure CLI executes a command on a CI/CD agent or developer workstation.
2. The Change Intelligence extension captures the resulting operation metadata.[1]
3. Sensitive properties are stripped at the extension before any network call. The extension does not decide this by name matching: it strips any argument Liquibase itself marks as obfuscated, plus connection URLs, and transmits them as a name with a null value and a sensitive flag. The secret is never materialized on the sending host.
4. The redacted payload is sent over HTTPS to the project scoped ingest endpoint of Liquibase Secure Server, authenticated by a service token (a service principal's credential).
5. The Change Intelligence module validates, persists, and fans out a real time event over Redis to any connected browser sessions.
6. Users authenticated to Secure Server browse, filter, and optionally request AI analysis of individual operations. Policy check operations arrive through the same path, so policy outcomes appear alongside deployment and drift evidence; no Change Governance license or Assignment is required.

Logical Architecture

The diagram below summarizes the two components of Liquibase Secure, the trust boundary, and the small set of external dependencies. Change Intelligence is highlighted.

© 2026 Liquibase Inc. All Rights Reserved

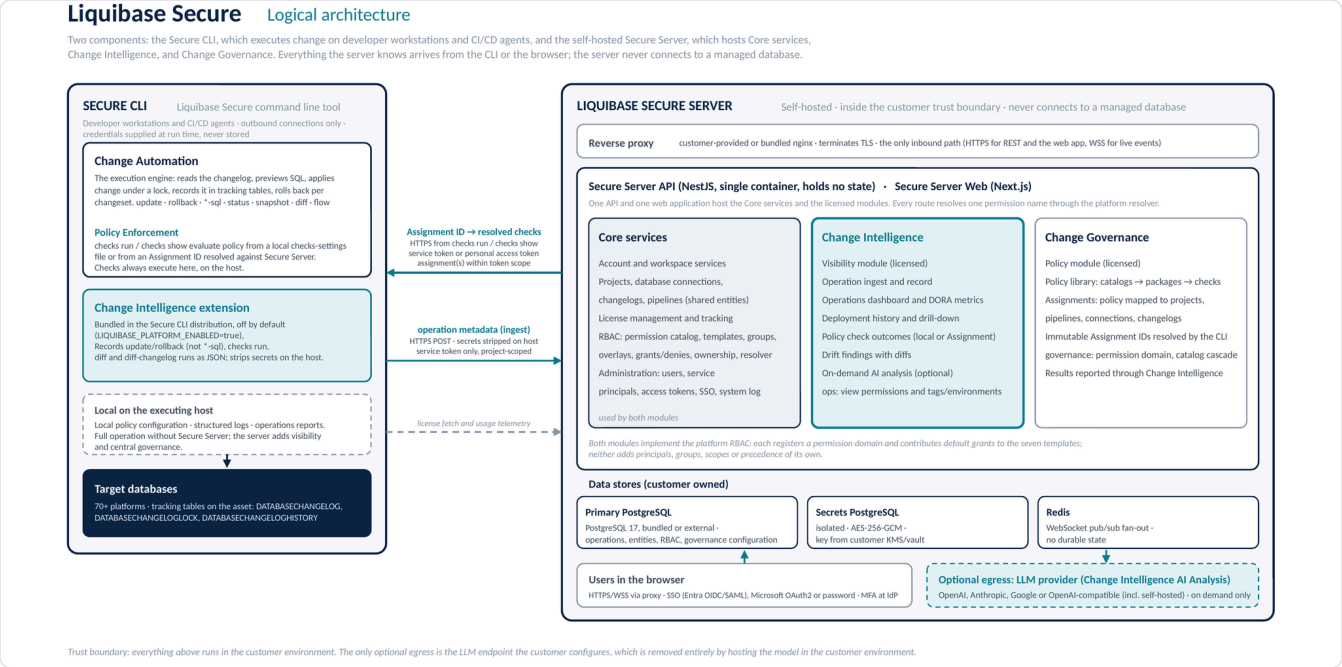


Figure: Liquibase Secure logical architecture. The Secure CLI runs on developer workstations and CI/CD agents; Secure Server hosts Core services, Change Intelligence, and Change Governance behind one API and one web application. Highlighted: the Change Intelligence extension in the Secure CLI, the Change Intelligence module in Secure Server, and the optional AI Analysis egress.

Three paths cross the boundary between the Secure CLI and Secure Server, and all three are initiated by the CLI over HTTPS: the Change Intelligence extension posts operation metadata to the ingest endpoint; the checks commands request the check configuration behind an Assignment ID from Change Governance; and the license endpoints serve the CLI its license and receive usage information. Users reach the web application through the reverse proxy over HTTPS and WSS. The only optional egress from the server is the LLM endpoint the customer configures for Change Intelligence AI Analysis, and it is removed entirely by hosting the model in the customer environment.

Deployment Model

Change Intelligence runs inside Liquibase Secure Server, which is delivered as a self hosted product. The customer owns the deployment, the data, the network, and the egress posture. Liquibase publishes container images and a Docker Compose configuration that runs on any host the customer operates, on premises or on a cloud VM. There is no separate installation for Change Intelligence: the module is part of the Secure Server images and is activated by the license.

Components Inside the Trust Boundary

All runtime components run within the customer environment. None require Liquibase managed infrastructure or external SaaS dependencies for normal operation. The components below are shared by Core services, Change Intelligence, and Change Governance.

© 2026 Liquibase Inc. All Rights Reserved

Component	Role	Stateful
API container	Secure Server API, a NestJS service written in TypeScript. Hosts the Core services and the module routes: validates and persists data, manages tenancy and authorization, serves the REST endpoints consumed by the web application and the Secure CLI, and emits real time events over WebSockets via Redis pub/sub. Holds no state of its own.	No
Web container	Secure Server Web, a Next.js application written in React, served by its own Node process behind a reverse proxy provided by the customer or managed by the product's Docker Compose configuration. Presents the Core administration surfaces and the module user interfaces.	No
Primary PostgreSQL	Application data on standard PostgreSQL 17. Holds the shared entity model, access control configuration, Change Intelligence operation records, and Change Governance policy assets and Assignments. Bundled by default, or an external instance the customer operates, such as Amazon RDS for PostgreSQL.	Yes
Secrets PostgreSQL	Separate, isolated instance for sensitive configuration values.	Yes
Redis	WebSocket pub/sub fan out for live events to connected browser sessions.	No (cache)

Network Posture

- Inbound: limited to the reverse proxy that fronts the API and Web tiers. The proxy terminates TLS for HTTPS (REST and the web application) and WSS (live events). Two kinds of clients arrive through it: users in the browser, and Secure CLI instances on the customer's CI/CD agents and developer hosts, which post operation metadata for Change Intelligence and request Assignment configuration from Change Governance. No other inbound paths exist.
- **Outbound:** required only when Change Intelligence AI Analysis is enabled, and only to the LLM endpoint the customer configures. Customers that point AI Analysis at a self-hosted model can operate Secure Server, and Change Intelligence with it, in a fully air gapped configuration. Installation and updates can also run offline: images can be copied into the customer's own registry, and a verified release bundle can be installed without internet access.
- **Lateral:** no connectivity is required or used from Secure Server to monitored databases. All Change Intelligence operational data arrives via the extension on the CI/CD path.

Data Residency

All customer data, including operation records, properties, policy assets, Assignments, and configuration, remains in customer owned databases. Nothing leaves the customer environment except traffic between the customer's own Secure CLI instances and Secure Server, and, optionally, a customer configured AI provider call made by Change Intelligence.

Security Posture

The security posture described in this chapter belongs to Liquibase Secure Server and applies identically to Core services, Change Intelligence, and Change Governance. Where Change Intelligence adds a control of its own, the text says so.

© 2026 Liquibase Inc. All Rights Reserved

Transport Security

TLS is terminated at the customer's reverse proxy in front of the API and Web tiers. Cipher suites and certificate management follow customer policy at the proxy layer. HTTPS is the recommended configuration for every deployment reachable over a network the customer does not fully control, and is what the documentation directs operators to. Secure Server also accepts plain HTTP for local and fully isolated environments where encryption is not a requirement, so transport encryption is a deployment decision rather than a product constraint. WebSocket connections from the browser follow the same proxy, as WSS wherever TLS is terminated. Secure CLI traffic to the server follows the same rule: HTTPS wherever the proxy terminates TLS.

Authentication

Authentication is a Core service, built on BetterAuth, and supports the controls expected in enterprise environments.

- Email and password authentication.
- Single sign on against Microsoft Entra, over OIDC or SAML 2.0, configured per workspace.
- OAuth2 with Microsoft as identity provider.

Multi-factor authentication is delegated to the customer's identity provider. Secure Server carries no native second factor of its own: where the Entra tenant requires MFA or conditional access, those policies are enforced upstream and apply to every SSO sign-in. This is a deliberate design decision, taken so that the server has no separate authenticator enrollment to manage, no recovery-code custody, and no cryptographic surface of its own to bring into FIPS scope.

Operators can close self-service email and password registration for the whole installation. With registration closed, the sign-up route is refused at the server rather than merely hidden, existing accounts continue to sign in, and new users arrive only through SSO or an administrator's invitation.

Access tokens authenticate Secure CLI and scripted traffic. A service token belongs to a service principal and is the only credential accepted for operation ingest and artifact upload. A personal access token belongs to a user, carries that user's permissions, and is accepted on every route the user can reach except ingest, artifact upload, console sign-in, and token or account management. Every personal access token expires: users choose 30, 60, or 90 days within a ceiling the administrator sets, and each user may hold at most ten active tokens. Administrators can switch personal access tokens off for a workspace, which suspends rather than revokes them, and deactivating a user revokes their tokens. All tokens are displayed once at creation, are stored only as a hash so they cannot be retrieved later, and can be revoked at any time; service tokens can also be rotated. A request made with the wrong kind of credential, or with an expired, revoked, or suspended token, is refused with a distinct error code.

Single Sign On

Single sign on is configured per workspace by a Workspace Admin, in the application, with no operator-level environment changes. The administrator chooses the protocol that matches their Entra enterprise app:

- OIDC, configured with tenant ID, client ID, and client secret.

© 2026 Liquibase Inc. All Rights Reserved

- SAML 2.0, configured by uploading IdP metadata, pointing at a metadata URL, or entering the IdP entity ID, sign-on URL, and signing certificate directly. Secure Server publishes its own service provider metadata for registration in Entra.

One protocol is active per workspace at a time. Client secrets and signing certificates are held in the encrypted secrets store described below.

Sign-in is restricted to the configured tenant. Under OIDC, the tid claim on the ID token is validated against the configured tenant. Under SAML, an assertion is accepted only when the issuer matches the configured IdP entity ID and the signature validates against the configured certificate. Personal Microsoft accounts cannot sign in to a workspace bound to an organizational tenant, and a mismatch is refused with a message that names the cause without exposing configuration detail.

First-time arrivals are provisioned just in time and placed in the system Pending group with zero permissions, where they see an explicit awaiting-access screen until an administrator assigns them. Subsequent sign-ins resolve to the same user record through a stable identifier, without creating duplicates and without changing group membership.

Two lockout guards protect the installation from configuration mistakes: the last enabled SSO provider cannot be disabled while self-service registration is closed, and the last remaining administrator cannot be removed. Disabling or deleting an SSO provider revokes the active sessions that provider issued rather than leaving them to expire.

Scope for this release: user provisioning is just in time only, with SCIM deferred; logout is local to Secure Server, with federated single logout deferred; and Entra group claims are not consumed, so group membership is assigned in Secure Server rather than inherited from the directory.

Secrets Handling

Sensitive configuration values, including database connection credentials and SSO client secrets and signing certificates, are held in a separate, isolated PostgreSQL instance. All values are encrypted at rest with AES-256-GCM using per-value key derivation. The master encryption key, the database passwords, and the authentication secret are supplied to the containers as file-based secrets rather than environment variables, so they do not appear in container metadata such as docker inspect, and the web container receives none of them. Customers typically source the master key from their KMS or vault, and it is never stored alongside the ciphertext. Secrets stored on a connection are write-only through the product: no API response and no permission returns a decrypted value.

Sensitive Liquibase properties never reach the server as values. The Change Intelligence extension strips them on the originating host before any network call: it strips any argument that Liquibase itself marks as obfuscated, together with connection URLs, and transmits them as a name with a null value and a sensitive flag, so the secret is never materialized on the sending host. The API applies a second, independent pass on ingest, redacting by key name against the patterns password, secret, token, key, and credential, and stripping credentials embedded inside URL values. The second pass writes a distinct sentinel, so an operator can tell a value the API caught from one the extension stripped as intended. Neither pass is customer-configurable, and both err toward redacting too much. The UI displays a redacted indicator with no reveal control available under any user action.

© 2026 Liquibase Inc. All Rights Reserved

Change Intelligence ships with no UI affordance to reveal redacted properties. There is zero risk of accidental disclosure today, even by administrators, until a customer explicitly requests a reveal capability and that capability is built behind step up authentication and audit logging.

Secure CLI Authentication and Principal Separation

Secure CLI instances authenticate to Secure Server with access tokens. Two kinds exist, service tokens and personal access tokens, and the difference matters for least privilege.

Service tokens belong to service principals, a distinct principal type from human users. Service principals are scoped to one or more projects, cannot sign in to the web application, cannot join groups, and can reach only the routes that explicitly admit them; every other endpoint refuses a service token regardless of possession. Four kinds of route admit them: Change Intelligence operation ingest and artifact upload, where a service principal may write operation data only for entities inside its assigned projects; license fetch; and the Change Governance Assignment resolution endpoint, where a pipeline can retrieve an Assignment when at least one of the Assignment's targets is in one of the service principal's projects, without any governance authoring rights.

A request naming an entity or Assignment outside the principal's scope and a request naming one that does not exist are refused identically, so a CI token cannot be used to enumerate what exists. This separation eliminates the most common abuse path for pipeline credentials, which is repurposing them for read access. Service tokens can be rotated in place and revoked, and both actions are recorded in the system log. Revoking a service token leaves its service principal and project scope in place; deleting the principal is a separate, confirmed action that revokes its tokens.

Personal access tokens are workspace-scoped, owned by a human user, always expire, and carry that user's permissions. They are accepted for Change Governance Assignment resolution, for license fetch, and for interactive and scripted use of the API, but never for ingest: operation data is written only by service principals, so a personal access token cannot be used to feed a pipeline's results into Change Intelligence. Customers should issue a service principal for every CI/CD pipeline and reserve personal access tokens for local and scripted work.

Ingest Authentication

A single ingest endpoint accepts JSON from the extension. Ingest traffic is authenticated by a service token, which belongs to a service principal scoped to one or more projects. A service principal may ingest only for entities inside its assigned projects, and a batch naming anything outside that scope is refused. A batch naming an entity that does not exist and a batch naming an entity outside the principal's scope are refused identically, so a CI token cannot be used to enumerate what exists.

Service principals cannot sign in to the web application and cannot read any operation, dashboard, or permission data, regardless of token possession. The narrow scope is the point: a pipeline token that leaks from a CI log grants the ability to write operation records for one project's entities and nothing else. It cannot be turned into a read credential for the estate.

Personal access tokens are not accepted for ingest. A pipeline that presents a personal access token to the ingest endpoint is refused, so operation records can only ever be written under a project-scoped service principal, and the identity behind every ingested operation is a machine identity the customer created for that purpose. The same service token also authenticates the Secure CLI when it resolves a Change Governance Assignment during a checks run, so a single pipeline credential serves both modules.

© 2026 Liquibase Inc. All Rights Reserved

AI Analysis Controls

All LLM calls originate from the API container, not from the browser. Browsers never call LLM providers directly, and operation data is never proxied through the browser to an external service.

The LLM provider, model, and endpoint are set once for the Secure Server installation, so where inference happens is a single operator-controlled decision rather than a per-workspace one. AI Analysis is then enabled or disabled per workspace, and is off unless enabled, which lets an operator hold it off for workspaces handling regulated data. Customers with strict data exfiltration concerns can point the installation at a local or self-hosted OpenAI-compatible endpoint and operate with no outbound LLM traffic at all.

AI results are persisted with the originating operation. Because Liquibase operations are immutable, prior analyses do not become stale relative to their subject and do not regenerate silently on revisit. Each analysis is a durable, auditable artifact attributed to the model and timestamp that produced it.

Supply Chain

Secure Server images are scanned on every release. Each image is accompanied by a VEX (Vulnerability Exploitability eXchange) document recording Liquibase's exploitability assessment of findings that have no upstream fix, so scanners such as Trivy and Grype can distinguish assessed, non-exploitable findings from actionable ones. Support bundles produced by the installer redact keys, URL passwords, and invitation and password-reset tokens.

System Log

The system log is a Core service. It records events from administering Secure Server, not activity on customer databases; that evidence is the operation record. It captures every permission changing action in Secure Server, across Core services and both modules, in an immutable record. Each entry records the actor, the timestamp, the target, and the before and after state. Logged events include:

- User invitation, deactivation, and reactivation.
- Group create, rename, and delete operations.
- Group membership changes.
- Template assignment, scope change, and unassignment.
- Overlay changes.
- Direct grant create and revoke.
- Direct deny create and revoke.
- Entity ownership added, removed, and transferred.
- Access token lifecycle events for service tokens and personal access tokens: create, rotate, revoke (with reason), expire, suspend and restore (with counts).
- Service principal lifecycle events: create, rotate, revoke.
- SSO configuration changes, including enabling and disabling SSO, protocol changes, tenant and endpoint changes, and secret or certificate rotation. Secret rotations record a reference such as a certificate fingerprint, never the value itself.

© 2026 Liquibase Inc. All Rights Reserved

- License management: license file import, rejection, removal, and re-verification; changes to effective license status and licensed capabilities, recorded under the System actor when a term or grace period ends on its own; downloads and exports of license data.

The log is searchable by actor, target, and date range, and can be filtered by credential type and to license events. It is retained indefinitely in this release; data offloading assistance and retention policy guidance are planned for a future release. Access is gated by the audit view privilege, which is bundled into the Security Reviewer, Governance Lead, and Workspace Admin templates by default. Entries are exportable for handoff to a reviewer or an evidence package.

For Change Intelligence, the system log complements the operation record: operations themselves are immutable once ingested, and AI analyses are persisted with the operation they describe, attributed to provider, model, and timestamp.

Authorization Model

Liquibase Secure Server ships with a permission model designed from a blank sheet against the access control patterns observed in regulated customer environments. The model is a Core service, owned by the platform: one permission catalog, one set of persona-shaped templates, one resolver, and one fixed precedence order. Change Intelligence and Change Governance implement the model rather than inventing their own. Each registers its permissions into the platform catalog and contributes grants to the platform templates. Change Intelligence therefore brings no principals, groups, scopes, or precedence of its own; what it adds is a permission domain and a set of default grants that enrich the seven templates.

The model is built on the concepts of principals, groups, templates, overlays, ownership, and direct overrides. One boundary is worth stating at the outset: the model governs access to Secure Server's own records, which are the server's record of database change and governance configuration, and never access to the databases themselves. Who could connect to a database, and with what rights, remains the province of the database's own privilege system.

Principals

Two principal types exist.

- **Users:** human principals authenticated via password or SSO, with MFA enforced by the customer's identity provider. Users can also call the API with personal access tokens, which resolve to the same user and permissions.
- **Service principals**: non human principals authenticated by service token, used by Secure CLI instances in CI/CD. They cannot log into the web application, cannot join groups, and can reach only the ingest, artifact upload, license fetch, and Assignment resolution routes that explicitly admit them. Ingest accepts service tokens only; a personal access token cannot write operation data.

Groups, Templates, and Template Assignments

Access is granted to groups by default. An individual user can also hold access through three specific paths: ownership of an entity, a direct grant, or a direct deny, each of which names one principal and one entity. Groups are workspace scoped collections of users that mirror the organization. A user can belong to multiple groups, and

© 2026 Liquibase Inc. All Rights Reserved

effective permissions are the union of all sources, modulated by direct grants and direct denies.

A **template assignment** applies one template to one group at one scope: the whole workspace, a list of projects, or a list of specific database connections. Workspace-scoped assignments cover everything in the workspace. Project-list assignments cover a named list of projects, so one org-shaped group can hold one policy across a dozen applications. Connection-list assignments follow specific databases across every project they appear in, which is how database owners are given visibility of the change record for their databases. *(In this document, "template assignment" always means a template applied to a group. Change Governance uses the word Assignment for a different object: a mapping of policy packages to project assets.)*

Seven persona shaped templates ship in every workspace at initialization. Each defines the scope it may be applied at and a default privilege bundle drawn from the Core services and from every module the customer has licensed. Templates are immutable: they cannot be renamed, edited, or deleted, and no endpoint exists to do so. Customers begin from these rather than from an empty permission list.

Template	Also known as	Applicable scope	Principal type	Default posture (Core and Change Intelligence)
Workspace Admin	Customer Admin	Workspace	User	Full administrative access to everything in the workspace, including user, group, template and overlay administration, access tokens (view and revoke) and service principals, direct grants and denies, license management, the view permissions surface, and the system log. Excludes ingest, which belongs only to service principals.
Application Developer	Application Lead, Team Lead	Project list	User	Operational read plus changelog create and update in assigned projects; all Change Intelligence views including raw logs, and running AI analysis.
DBA	Policy Author, Data Architect	Connection list	User	Operations, drift, and AI analysis (view and run) on specific databases, with project view derived upward from those databases. No changelog modification, nothing administrative.
Platform / DevOps Engineer	Release Manager	Workspace	User	Project, connection, changelog and pipeline creation, cloning, and viewing across the workspace; the service token lifecycle; service principal creation and scoping; the operation views and AI analysis needed to debug a pipeline. Excludes user, group, and permission management.
Security Reviewer	Compliance Lead, Auditor	Workspace	User	Workspace-wide read only across Core and Change Intelligence records, plus the view permissions surface and the system log.

© 2026 Liquibase Inc. All Rights Reserved

Template	Also known as	Applicable scope	Principal type	Default posture (Core and Change Intelligence)
Governance Lead	—	Workspace	User	Workspace-wide read only across Core and Change Intelligence records, plus the view permissions surface and the system log, plus full administration of Change Governance content, including its owners and access.
Service Principal	—	Project list	Service principal	Ingest, artifact upload, and license fetch. No web application session and no view permission over Core or Change Intelligence records.

The default posture column shows the Core and Change Intelligence bundles. Where Change Governance is licensed, its permission domain contributes default grants to the same seven templates; those are described in the Change Governance Technical Overview.

Overlays customize a group's template assignment without forking the underlying template. An overlay is an additive and subtractive permission delta attached to a single template assignment, and the overlay editor enumerates every registered permission, including permissions contributed by modules. When Liquibase ships a template update in a future release, customer overlays are preserved and the new template version applies cleanly beneath them.

Direct grants and direct denies at the entity level handle exceptions, such as temporary cross project investigation rights or containment of sensitive data exposure on a specific operation. They are reachable from the entity's detail page or its Manage Access screen.

Ownership is an independent source of access. Projects, connections, changelogs, catalogs, packages, and governance assignments are ownable; whoever creates an entity owns it initially, ownership is many-owner, groups may own, and an entity is never left ownerless. Owners hold full control of the entities they own even where their template alone would not grant it, which is what lets a platform engineer create a database connection, hand it to the database owners' group, and step away. Deactivating a user or deleting a group that solely owns an entity is blocked, with ownership transfer offered in the same flow, so offboarding never silently orphans a database connection. Owners can share access to what they own, through direct grants and denies limited to the owner's own bundle, without holding the grant permission in their template. Project ownership does not cascade: owning a project gives view access to the connections and changelogs inside it, but not management of their owners. Catalog ownership does extend to every package in the catalog.

Permission Resolution

Permissions are resolved per (principal, permission, entity) tuple using a three layer priority.

1. Direct grants: explicit allow at the entity level. Highest precedence.
2. Direct denies: explicit deny at the entity level.
3. Baseline: the union of everything the principal is allowed before grants and denies adjust it. Each template assignment contributes its template plus its overlay, composed per assignment first (template permissions plus overlay additions minus overlay subtractions) and only then unioned across every applicable assignment;

© 2026 Liquibase Inc. All Rights Reserved

entity ownership contributes the owner's bundle; and a licensed module contributes its own grants. The baseline is a most-permissive union across contributors, never a precedence contest among them.

The effective rule is that a direct grant wins outright, a direct deny overrides the baseline, and the baseline governs everything else. Making a deny beat the baseline is what makes containment possible; making a grant beat a deny is what makes the named exemption possible, for example re-admitting an incident responder to a contained operation. One consequence reads as a defect until explained: a subtraction on one template assignment cannot remove a permission that another assignment grants, because the baseline is most-permissive by definition, and the only mechanism that overrides it is a direct deny.

Ownership contributes to the baseline rather than sitting above the precedence order, so a direct deny still overrides an entity's own owner, and a direct grant can restore them. Ownership is a source of access, not an exemption from the model.

One deliberate exception: denies do not reach a workspace's administrators. A direct deny cannot be used to strip an administrator of access, which prevents a configuration mistake or a malicious deny from locking an organization out of its own installation. The exception exists because a deny is the one statement in the model with no in-product recovery: a deny placed on everyone, or on a group an administrator belongs to, would otherwise leave an entity visible to nobody and require a direct database change to restore it. Administrators here are the members of the workspace's system Administrators group; holding the Workspace Admin bundle by some other route confers no exemption, and the exemption is scoped per workspace, so administering one workspace confers nothing in another. It is enforced on both paths: the write path refuses a direct deny naming the Administrators group or an individual administrator, and the resolver drops any remaining deny in a workspace the asking principal administers, which is what catches a deny placed on everyone or on an ordinary group an administrator also belongs to. An administrator is removed by removing that group membership, which is itself audited, and the last administrator of a workspace cannot be removed.

Every permission check happens at a scope, and containment flows downward, never upward: a workspace assignment reaches the projects, pipelines, and connections inside it, a list-scoped assignment never reaches beyond its listed targets, and a check made with no scope matches no assignment and is denied. Failing closed converts a forgotten scope into a visible denial during testing rather than a silent workspace-wide grant during an incident.

Authorization is always evaluated against a permission, never against a template name, so a customer can restructure templates, overlays, and groups without changing how any endpoint is enforced. Nothing is materialized: there is no precomputed effective-permissions table, so no stored artifact can drift from the configuration that produced it. This resolution model produces predictable, explainable access decisions suitable for audit and review.

Permission Vocabulary and Module Namespaces

Every permission is a three-part name, domain:action:resource, and every authorization decision in Secure Server resolves one exact name at one scope; wildcards are not part of the grammar. Domains are reserved namespaces, each owned by exactly one functional area and governed by that area's license. Seventy-one built-in permissions ship across five platform domains (admin, ops, ingest, audit, owner): fifty-eight owned by the Core services and thirteen owned by Change Intelligence, which are hosted in the ops namespace for continuity with the names that predate the module. The licensing module registers its own domain, licensing, with ten

© 2026 Liquibase Inc. All Rights Reserved

permissions, and Change Governance registers its own domain, governance, with twelve. Registration is validated at startup: a malformed name, a duplicate registration, or a second module claiming an existing domain stops the service rather than letting it boot with a silently wrong catalog.

Licensing is enforced through the same mechanism. A module the installation is not licensed for has its permissions made inert: template assignments, overlays, and direct grants referencing them resolve to nothing, are shown in the view permissions surface as originating from a currently disabled module, and are preserved rather than deleted, so enforcement resumes automatically if the module is re-enabled. The Core namespaces cannot be made inert by any license state, so the permission model can never license away the ability to administer or re-license itself.

Change Intelligence Permissions

Change Intelligence files what it receives under the shared entity model as operations: deployments and rollbacks, policy check runs, drift findings, AI failure analysis, and raw execution logs. Its permissions are chiefly views over that record, plus the tags and environments that organize it and the right to run AI analysis on an operation. The record itself is written only through the Core ingest domain by service principals. The module's thirteen permissions are hosted in the platform ops namespace:

Permission	What it allows
`ops:view:operation`	Access the operations area and read the operation list and detail.
`ops:view:database-changes`	View update and rollback operations.
`ops:view:policy-checks`	View policy check operations.
`ops:view:drift`	View drift operations.
`ops:view:ai-analysis`	View AI failure analysis on operations.
`ops:view:raw-logs`	View raw operation logs.
`ops:generate:ai-analysis`	Run AI failure analysis on an operation.
`ops:view:tag`	Read the workspace's tags.
`ops:create:tag`	Create a tag.
`ops:update:tag`	Rename or recolor a tag.
`ops:delete:tag`	Delete a tag.
`ops:view:environment`	Read the workspace's environments.
`ops:manage:environment`	Create, edit, and reorder environments.

Sensitivity is graded by permission, not by page. Raw execution logs are reached through their own permission, ops:view:raw-logs, separate from the operation record that contains them. It is granted to every human template by default and can be removed from any group by overlay where a customer wants a narrower posture. Direct denies on a single operation contain one incident's record even from principals whose template allows the view; a

© 2026 Liquibase Inc. All Rights Reserved

direct grant re-admits a named responder. An operation is contained, granted, owned, and audited with exactly the same mechanics as any Core entity. Operation views are enforced per type: a principal without ops:view:policy-checks, for example, does not see policy check operations in the general operations list or detail.

Connection properties a customer chooses to store in Secure Server are the one place a secret can exist in the record. They are encrypted at rest and are write-only: no API response returns a decrypted value and no permission reveals one. Access control and secret handling are separate defenses, and the second does not depend on the first being configured correctly.

Governance

- **Governed reads throughout.** Every read endpoint that returns operational, configuration, policy, or permission data enforces a corresponding view permission, and list endpoints filter to what the caller may see rather than refusing the whole page. A small set of routes is deliberately unauthenticated because it must function before or without a session: the health and version probes, the pre-sign-in settings the login page reads, the SSO sign-in routes themselves, and the invitation preview a new user opens from an emailed link. The license usage collector is authenticated separately, by a service token or personal access token. None of them return operational, policy, or permission data. Retrieving the signed license file is governed: it requires licensing:access:license, held by Workspace Admin and Service Principal, or a personal access token.
- **View permissions surface.** A Security Reviewer, or any principal with the view grants privilege, can open a who has access to this project view from any project, and a who has direct grants or denies on this entity view from any entity detail page. Seeing who has access is a separate permission from reading operational data and is bundled into Security Reviewer, Governance Lead, and Workspace Admin rather than into the operational templates. These surfaces produce audit evidence on demand.
- **Compliance evidence, and its boundary.** The system log and the view permissions surface together produce reviewable evidence of who could see which Secure Server records at any point in time, suitable for SOC 2, HIPAA, and similar review activities, answerable from inside the product by a reviewer whose own access is read-only. It is not evidence about access to the databases themselves; presenting these reports as an answer to that question would misrepresent the control.

Forward Compatibility

Extensible Today

The authorization model is not a fixed set of permissions with room left over. It is a registry, and extending it is a shipped capability rather than a future one.

Permissions are declared in reserved namespaces, each owned by a named module and governed by that module's license. Four owners are already registered: the Core services, licensing, Change Intelligence, and Change Governance, each contributing its own permissions and its own additions to the default templates without forking them. A namespace collision fails the build rather than surfacing at a customer's boot.

Authorization is always evaluated against a permission, never against a template name. Customers can restructure templates, overlays, and groups, and Liquibase can ship template updates, without changing how any endpoint is enforced. This is what makes the overlay mechanism safe: a customer's overlays are preserved across a template update, and the new template version applies cleanly beneath them.

© 2026 Liquibase Inc. All Rights Reserved

Designed to Absorb

The following extend the model additively, without architectural rewrites or breaking changes for customers deploying today.

- Execution permissions, when Secure Server later drives operations directly, for example ops:execute:deploy and ops:cancel:running-operation. These are new keys in an existing namespace, not a new mechanism.
- Environment as a scope qualifier, narrowing assignments to non-production or production. Grants and denies already bind to scope kinds from workspace down to an individual entity, so this adds a scope kind rather than a scoping mechanism, and existing assignments default to all environments.
- Step-up authentication for sensitive actions. Because Secure Server delegates multi-factor authentication to the customer's identity provider, this will be expressed as a re-authentication requirement satisfied by the IdP, under the customer's own conditional-access policy, rather than as a second factor managed by the server.
- Time limited grants for break glass and on call access scenarios.
- Approver as a distinct principal role, for separation of duties on execution flows.

Customers deploying Change Intelligence today receive an authorization model that will extend to govern execution and approval workflows as those capabilities arrive, without disrupting existing assignments.

Deployment and Sizing Guidance

Liquibase Secure Server ships as a set of containers and a Compose configuration that runs all of them. That configuration is the product's default and its supported baseline: it stands up the application tiers, the databases, the cache, and the reverse proxy together, with the liquibase-platform CLI generating configuration, managing secrets, and running database migrations. Docker Compose 2.20 or newer is required. A customer who wants the product running does not assemble it from parts, and Change Intelligence requires no additional container: it is part of the API and Web images and is activated by the license.

Customers whose standards call for a managed database service can run the primary database on their own PostgreSQL 17 instance; the section below describes how. Nothing in the architecture requires either choice.

The Default Deployment

Every component runs as a container on a single host.

Container	Role	Stateful
Reverse proxy	Terminates TLS and routes to the Web and API tiers	No
Web	Secure Server Web: Next.js application served by its own Node process	No
API	Secure Server API: NestJS service hosting Core services and the module routes, including ingest, Assignment resolution, REST, and WebSocket endpoints	No
Primary PostgreSQL	PostgreSQL 17: application data, operation history, policy assets, and access control configuration. Can be replaced by an external instance (see below)	Yes
Secrets PostgreSQL	Encrypted configuration values, isolated from the primary	Yes
Redis	WebSocket pub/sub fan out	No (cache)

© 2026 Liquibase Inc. All Rights Reserved

Database migrations run as a separate, on-demand container rather than automatically at startup, so schema changes are an explicit operator action. Before update changes anything, it lists any pending upgrade notices and asks for confirmation. Database passwords, the encryption key, and the authentication secret are passed to the containers as file-based secrets.

Evaluation and Small Teams

Suitable for evaluation, pilots, and small teams. Every container on one host.

Component	Resource
Host	1 VM, 4 vCPU, 8 GB RAM, 100 GB SSD, running the full Compose configuration
Storage	Docker volumes on the host; back up the volumes or take database dumps
TLS	Customer provided certificate at the bundled reverse proxy
Liquibase Secure CLI	Version 6.0 or newer on CI/CD agents and developer hosts, with a valid Secure license

Production

Suitable for production deployments with multiple projects, multiple environments, and active use of the web application. The starting point is the same Compose configuration on a larger host, with the primary database either kept in place on dedicated storage or moved to an external PostgreSQL 17 instance.

Component	Resource
Host	8 vCPU, 16 GB RAM to run the application tiers and cache; size storage from retention, below
Primary PostgreSQL	PostgreSQL 17. 4 vCPU, 16 GB RAM, 200 GB SSD to start, with automated backups and point in time recovery. Run as the bundled container on dedicated storage, or as an external instance you operate, such as Amazon RDS for PostgreSQL
Secrets PostgreSQL	2 vCPU, 4 GB RAM, isolated from the primary. The encryption master key is supplied at deploy time and held outside both databases
Redis	Redis 7, bundled. Holds no durable state; losing it drops live updates until it returns and loses no operation or policy data
Database connections	The API opens two pools to the primary database, each capped at LIQUIBASE_PLATFORM_DB_POOL_MAX connections (default 10), plus two session connections, for up to 22 by default; keep that total below the database's max_connections, especially on a managed instance. A request that cannot get a connection within the pool timeout (LIQUIBASE_PLATFORM_CONN_TIMEOUT_MILLIS, default 5000 ms) fails with 503 instead of hanging the API
Reverse proxy	The bundled proxy, or the customer's standard (Nginx, AWS ALB, Azure Application Gateway, or equivalent) terminating TLS in front of it
Backups	Point in time recovery on both databases. The secrets database is small but must be backed up alongside the primary, and its master key must be recoverable independently of either
Egress	Only to the configured LLM endpoint when Change Intelligence AI Analysis is enabled; eliminated entirely by hosting the model in the customer environment

© 2026 Liquibase Inc. All Rights Reserved

These figures are starting points for a deployment of this shape, not the output of a published benchmark. Actual sizing depends on operation volume, retention, and how many users are active at once. For Change Intelligence, storage is the component most sensitive to retention: operation history and execution logs accumulate in the primary database, and a customer keeping several years of history across a large estate should plan storage from their own volume rather than from the starting figure above. Operation history and the system log are retained indefinitely in this release; data offloading assistance and retention policy guidance are planned for a future release.

Using an External Primary Database

The primary database can run on a customer-operated PostgreSQL 17 instance, including Amazon RDS for PostgreSQL. Point the installer at it with the LIQUIBASE_PLATFORM_DB_* settings, or with install --db-host and the related flags; install, update, and migrate check that the database accepts the connection before writing anything, and then apply migrations to it. The TimescaleDB extension is not required. The application tiers do not change, and no code or image differs between the two shapes. What changes is who operates the database, and the customer takes on its backup, patching, and availability. The secrets database and Redis always run as bundled containers.

Two constraints are worth stating before a customer plans this. The primary database must be PostgreSQL 17. And the encryption master key for the secrets database must be supplied from outside both databases, typically from the customer's KMS or vault, whichever way the database itself is run.

Not Required

Direct connectivity from Secure Server to monitored databases is not required and is not used. All Change Intelligence operational data arrives through the extension on the CI/CD path. The server also requires no shared filesystem between containers, and no scheduler or job runner beyond what the container platform already provides.

---

1. Not every Secure CLI command produces an operation record. The extension captures the commands that materially alter or inspect a database: the update and rollback command families (excluding the SQL-preview variants such as update-sql and rollback-sql, which generate statements without executing them), the checks run subcommand, diff, and diff-changelog.↩

© 2026 Liquibase Inc. All Rights Reserved