Ziggy security pack

Describes Ziggy at commit 256636b8e (2026-10-23)

Contents

1. **Ziggy Security Pack: Read Me First**
   - What this pack is
   - The documents
   - Who is responsible for what
   - How to report a security issue
   - Standing questions
2. **Ziggy security overview**
   - 1. How Ziggy is built
   - 2. Sign-in: password, Google, Microsoft and LDAP
   - 3. Second factor (MFA)
   - 4. Sessions and tokens
   - 5. Roles and permissions, and the two administrator tiers
   - 6. Changes, approvals and access reviews
   - 7. Organisation separation
   - 8. Credentials and secrets
   - 9. Encryption at rest and in transit
   - 10. Audit logging and the SIEM copy
   - 11. Retention
   - 12. The admin portal and the reference deployment
   - 13. How this was checked
   - 14. Design notes
   - 15. Accepted for later
   - 16. Appendix: the audit checklist
3. **Ziggy Secure Configuration Guide**
   - What this guide is
   - 1. Settings that stop a production server from starting
   - 2. Environment and network entry
   - 3. Sign-in, sessions and tokens
   - 4. Multi-factor authentication
   - 5. Encryption keys, database and Redis
   - 6. Logging, the SIEM and retention
   - 7. Override settings
   - 8. Mail, workers, file storage, Reports, the AI agent host and tuning
   - 9. The admin portal
   - 10. Before go-live
   - 11. Hardening the reference deployment
4. **SOC 2 responsibility matrix**
   - What this document is
   - Summary
   - Security: logical and physical access (CC6)
   - Security: system operations (CC7)
   - Security: change management (CC8)
   - Availability (A1)
   - Confidentiality (C1)
   - Privacy (P)
   - What this document does not say
5. **Secure development**
   - 1. How a change is made
   - 2. The review gate
   - 3. Tests
   - 4. Automatic checks that block an unsafe change
   - 5. Keeping secrets out of the source
   - 6. Dependencies
   - 7. Vulnerability disclosure and patch policy
   - 8. What this document does not claim
6. **Security assurance report**
   - Part A. What is examined and how
   - Appendix: every security guarantee
7. **Security testing**
   - Testing your own installation
   - How the audit works
   - In-house security scan (complementary self-check)
   - Related documents

Ziggy Security Pack: Read Me First

What this pack is

This pack is the security documentation for a customer who runs Ziggy themselves from the source code. It describes what the product does to protect your data, how to set it up so those protections hold, and which parts of the work belong to you and which to Ziggy.

It describes one exact version of the product: the commit and date above. If you run a different version, treat the pack as a guide only and ask us for the pack that matches it.

The pack rests on a full audit of that version. The audit covered 1581 items: every route, WebSocket, command, setting, secret-bearing field, encrypted field and outbound connection in the source. The audit was AI-assisted: the judgements were made by an AI model (Claude Opus) against a fixed checklist, and every failure it reported was traced a second time through the code before it counted. It was not carried out by an independent human auditor, and it is not a penetration test. The security overview ("How this was checked") describes the method.

The documents

Document	What it is for	Who should read it
`00-README.md`	This page: scope, the documents, who is responsible for what, how to report an issue	Everyone
`security-overview.md`	The controls Ziggy provides, area by area (sign-in, MFA, sessions, roles and permissions, organisation separation, encryption, secrets, audit logging, retention, the admin portal); how the audit was done; the design decisions behind the controls; and the items the vendor has accepted to leave until later	Security reviewers, architects
`secure-configuration-guide.md`	Every security-relevant setting: what it does, its default, the value we recommend in production, what goes wrong if it is wrong, and whether the server refuses to start on an unsafe value. Ends with a "Before go-live" checklist and advice for hardening a deployment	The engineers who deploy and run Ziggy
`soc-responsibility-matrix.md`	For each SOC 2 criterion in the logical access, system operations, change management, availability, confidentiality and privacy groups: what Ziggy provides, what you must do, and where in this pack the evidence is	Compliance and audit teams
`secure-development.md`	How Ziggy is built and changed: a separate AI review of every change, the test tiers, the start-up checks, secret scanning, static checks, dependency handling, and the vulnerability disclosure and patch policy	Security reviewers, procurement
`security-assurance-report.md`	How Ziggy's security is checked, for your security team: what is examined and how, and an appendix of the security guarantees the code enforces and what enforces each	Security reviewers
`security-testing.md`	How Ziggy is security tested: frequent audits, the built-in security scan, and testing a customer commissions	Security reviewers, procurement

A suggested order for a first read: this page, the security overview, the secure configuration guide, then the matrix and the other documents as your review needs them.

Who is responsible for what

You run Ziggy in your own environment, so security is shared.

Ziggy provides the controls built into the software, as described in the security overview:

- sign-in methods, multi-factor authentication, sessions and tokens;
- roles and permissions, and the separation of one organisation's data from another's;
- encryption of sensitive fields and of data in transit between Ziggy's own components;
- protection of credentials and secrets held by the product;
- audit logging, with export to your SIEM;
- the start-up checks that refuse to run on an unsafe configuration;
- the review, testing and release process described in the secure development document;
- security fixes for the product, under the disclosure and patch policy.

You provide and operate everything around the software:

- the servers, network, firewalling, operating system and cloud account Ziggy runs on;
- the database and its roles, backups, disk encryption, restores and disaster recovery;
- the TLS certificates and the generation, storage and rotation of keys and secrets;
- the settings in the secure configuration guide, set to the recommended production values, and the "Before go-live" checklist;
- your identity provider, and the people you grant access to: who gets which role, how often access is reviewed, and removing access when someone leaves;
- monitoring and response: reading the audit log and alerts, and what you do about them;
- applying Ziggy updates and security fixes promptly, and keeping the operating system and the third-party components you run current;
- the use of Ziggy by your own staff and the data you put into it.

Some of these depend on a limit worth knowing before you rely on them:

- Alerts are raised only for the event types that reach the alert listener. The audit log records more than the alerting covers, so do not assume an alert will fire for every recorded act.
- The access-review export and the separation-of-duties report leave out access that is scoped to a single organisation. Use them as part of a review, not as a complete list of who can do what.
- The security overview's "Accepted for later" section lists the known weaknesses the vendor has accepted to leave until later, in plain words, with why each is unlikely and when it will be closed. It states, among other things, the limits of how SQL that has not been approved is restricted, and of asking for a fresh sign-in when a high-risk change is approved. Read it before you rely on either control.

Where a setting in your hands changes a protection (for example a flag that turns off a start-up check), the configuration guide names it, says what it lets through, and says it must not be left on in production.

How to report a security issue

If you find a vulnerability, or think you have, please tell us before you tell anyone else.

- Contact: info@ziggyplatform.com, or Freddy May directly at freddy.may@ziggyplatform.com
- Please include the version (commit) you run, what you saw, and the steps to repeat it, if you have them. Do not include real customer data or secret values.
- We acknowledge a report within two working days and tell you what we will do about it. The full policy is in the secure development document.

Reports about your own deployment (a server you run, an account in your organisation) are best raised with your own security team first; ask us if the cause may lie in the product.

Standing questions

If your security team has a specific concern, we can add it as a standing question. Every later audit then answers it for every item it concerns, and we can send you the answer. Ask through the security contact above.

Ziggy security overview

This document describes the security controls that Ziggy provides, area by area, for the security team of a customer who runs Ziggy from its source code. It is one of the documents of the Ziggy security pack; the others cover the secure configuration of an installation, who is responsible for what, how Ziggy is built and changed, how its security is checked, and how it is security tested.

How to read it:

- A control is described only where the audit of this commit found it holding. Where a control holds with a limit, the limit is stated in the same paragraph, not left for later.
- "Small" means the item gives no one more than their role already allows, or needs a rare state or a second mistake, or shows names, ids or counts only; the item says what is missing and what an attacker would need. "Medium" means a real weakness that needs something more to use: rights already held, data in an unusual state, or a second mistake. "Serious" means one that works for someone who should not have the power, with nothing more needed.
- The audit read the source code. It did not examine a running installation, a customer's network, or the cloud account an installation is deployed into. What an installation does depends on how it is configured; the secure configuration guide says how.
- Names in the Ziggy screens are used: "Project" and "Pipeline".

Contents

1. How Ziggy is built
2. Sign-in: password, Google, Microsoft and LDAP
3. Second factor (MFA)
4. Sessions and tokens
5. Roles and permissions, and the two administrator tiers
6. Changes, approvals and access reviews
7. Organisation separation
8. Credentials and secrets
9. Encryption at rest and in transit
10. Audit logging and the SIEM copy
11. Retention
12. The admin portal and the reference deployment
13. How this was checked
14. Design notes
15. Accepted for later
16. Appendix: the audit checklist

1. How Ziggy is built

Ziggy is a TypeScript monorepo. The pieces that matter for security are these.

Piece	What it is	Trust boundary
Central server	The API, the WebSocket streams, the command-line tool, the background job queues and the database schema	The main application. Every sign-in, permission, organisation and audit control below lives here
Worker	A small separate process that runs Pipelines for the central server. It serves two controllers only; a deny-by-default guard answers 404 to every other route	Authenticates the central server by an API key
Web client	The browser application. It holds no credentials of its own: session tokens are httpOnly cookies set by the server	Untrusted; every decision is made on the server
Admin portal	A separate application that manages deployments (servers, containers, keys, backups) with its own sign-in	A different trust domain from the central server (section 12)
Agent host	The process that hosts the Discovery agent. Ziggy only dials out to it	Separate; the agent cannot reach sign-in, MFA or access-administration routes

Four rules shape everything that follows.

- **One front door.** Every HTTP route passes one guard chain: a request limit, authentication, the policy check, a boundary for the Discovery agent, an organisation check and a feature gate. A route with no policy entry and no explicit "public" mark is refused for everyone, and the server refuses to start when such a route exists. Routes that are public must either authenticate the caller another way or be listed with what stands in for authentication.
- **One data layer.** All data access goes through one database service whose extension chain seals encrypted fields, scopes reads and writes to the organisation, and blocks writes to history tables. The server refuses to start when a model with an encrypted field is not routed through it.
- **One audit writer.** A single service writes the audit trail (section 10).
- **Start-up refusals.** The server refuses to start in production on a list of unsafe settings: a missing or weak token-signing secret, a plain `ldap://` address, sandbox sign-in allowed, a proxy count not stated, plain-text mail, a weak execution callback secret, and an unverified database or Redis link to another server.

The central server runs behind the reverse proxy of the reference deployment; the proxy terminates TLS with automatically obtained certificates and sets strict transport security and a content security policy.

2. Sign-in: password, Google, Microsoft and LDAP

One gate. A password, an LDAP bind and a Google or Microsoft callback each produce only a verified first factor. A single step then decides what comes next: an address-proof code, an MFA challenge, mandatory MFA set-up, or a session. Nothing mints a session on any other path, so LDAP and provider sign-in cannot skip MFA.

Passwords.

- A password has at least 12 characters, with an upper-case letter, a lower-case letter and a special character. It is stored as a bcrypt hash (cost 10). The last five passwords cannot be reused (the number is an installation setting, 1 to 24). Passwords do not expire; this was a recorded decision. There is no maximum length on the sign-in, change or reset forms; bcrypt uses the first 72 bytes.
- Wrong guesses are counted in one database statement before the password is compared, so parallel guesses cannot all pass a lock. The first two wrong guesses add no lock; the third locks the account for 30 seconds, the fourth for 2 minutes, the fifth and every later one for 15 minutes. A right password resets the count. The same counter covers every place a password is typed: sign-in, change password, step-up and the command-line sign-in.
- Sign-in attempts are also limited per account and address (5 per hour by default in production, 20 times that per address), counted whether or not an attempt succeeds. The number of trusted proxy hops must be stated in production, or the server refuses to start; without it every client would share one address.
- An unknown address, an inactive account and a wrong password all answer "Invalid credentials". A locked account is told so, which discloses that the account exists.

Reset and change.

- "Forgot password" answers the same whether or not the address has an account, and caps requests at 3 per hour per address. The reset code is six digits, valid for 10 minutes, stored only as a hash, with its own attempt limit plus per-account hourly and daily caps on wrong codes, claimed before the comparison.
- A reset or a change of password ends every login of the account in the same transaction (a change keeps the session it was made on). Tokens issued before it are refused.
- Limits: the reset code stays six digits (provider-link codes are eight). A reset code is filed under the lower-case form of an address but applied to the account whose stored address matches exactly; see "Accepted for later".

LDAP. A production process refuses an LDAP address that is not `ldaps://`. Sign-in first finds the directory entry for the typed name with the service account, counts the guess against that entry, then binds as the entry with the typed password. The account, link and address come from the entry the directory answered, never from what was typed. LDAP sign-in is limited to 5 per 15 minutes per address. The service account's bind password is a plain environment variable (see the design notes), and links made before an earlier rule change are on the "Accepted for later" list.

Google and Microsoft. Both use a state value and PKCE held in a short-lived httpOnly cookie that binds the callback to the browser that started it. A sign-in that needs MFA or address proof ends on the login screen with a single-use, 120-second hand-off cookie; no token is ever placed in a URL.

- Google must report the address as verified, or no account is created.
- A Microsoft address counts as vouched only when the tenant is a named one (not the shared multi-tenant defaults), and a Microsoft sign-up never proves the mailbox: a code is mailed.
- A provider identity that matches an existing account by email alone is never linked or signed in on that match. A pending link is created, bound to the browser, and the owner must prove the account with its password or a mailed code (and pass its MFA gate, even on a trusted device) before the identity is linked. Proof attempts and codes are capped per account.

Registration and accounts. Installation policy decides who may sign up (the default allows anyone; a domain list is optional), and no self sign-up is possible once two or more organisations exist. An account that has not proved its address gets no session: a code (eight digits, 15 minutes, 5 attempts per code, account caps) is mailed first. The first account on a new installation becomes its administrator; every later account starts with no access until an administrator approves it. The first account's set-up is several writes rather than one transaction, so a database fault part-way could leave it incomplete.

3. Second factor (MFA)

Factors. An authenticator app (TOTP: 6 digits, 30-second step, one step of tolerance either side), codes emailed to the account (6 digits, valid 10 minutes, at most 5 per account per hour, available only when the installation policy allows them), and ten single-use backup codes (stored only as hashes). The authenticator secret is sealed at rest (section 9).

How it holds.

- Every single-use credential (an authenticator time step, a backup code, an emailed code, a challenge, a recovery link, a recovery approval) is claimed in one conditional statement, so two simultaneous uses succeed once.
- A factor works only when it is turned on and confirmed, read from the account's own configuration. A grace period gives time to enrol; it never stands in for a factor that is in place.
- A wrong code at sign-in counts against the challenge (locks at the third wrong attempt: 30 seconds, then 60 seconds, then 5 minutes) and against the account's own run of wrong codes, which makes the account wait even for the right code. Step-up codes are additionally capped at 10 wrong guesses per account per hour. The challenge lock stops growing at the fifth attempt; further guessing is bounded by the account-level run, the route limit and the five-minute life of the challenge.
- **Mandatory MFA.** While the setting `MFA_MANDATORY_ENABLED` is not `false` (it is on by default), anyone who holds a workflow power (Merge, Promote, Rollback, Admin or BreakGlass) by any grant at any scope must use MFA, whatever the in-app policy says. The in-app policy can require it of more roles.
- **Changing a factor** (set up or confirm an authenticator, switch emailed codes on or off, replace backup codes, set the recovery address, switch the authenticator off) needs a fresh proof, made on the same session within 5 minutes: a code from a factor already in place, or the password or a fresh sign-in when none is. The change ends every other login of the account. When an authenticator is being replaced, the old one keeps working until the new one is confirmed.
- **Trusted devices.** A browser can be trusted for 14 days by a random secret held in an httpOnly cookie and stored as a hash. Nothing about the browser string or address decides it. Any change of factor, a password change, a recovery or an administrator reset revokes them.
- **Recovery.** A recovery link is mailed to the recovery address (valid 1 hour; 24 hours when held for approval), stored only as a hash. It can be requested only after the first factor has been proved. When the person holds an administrator grant and another active installation administrator exists, the recovery is held until that other administrator approves it; nobody can approve their own. A sole installation administrator is not held, so no one else oversees that recovery. Completing a recovery removes every factor, ends every login, mails the owner and returns only a set-up token: the person must enrol again before a session is minted.
- Administrators can switch a person's MFA off, reset a grace period or revoke trusted devices; each needs a reason, is recorded, and mails the owner where it removes factors.

Limits (all Small).

- A new recovery address takes effect at once, before anyone shows they can read it, and a recovery link already mailed to the old address keeps working. A recovery link for an account that has since been switched off still removes its factors, though the session it would lead to is refused.
- A failed send of an emailed code is not reported to the person.
- Refusals while a challenge is locked, or of an already-used code, write no audit record.

- The shipped templates do not set the recovery-link address; until an operator sets it, recovery links point to a local address (the secure configuration guide covers this).

4. Sessions and tokens

- **Access token.** A signed token (HMAC, SHA-256) valid for 15 minutes by default, carried in an httpOnly cookie. The server pins the algorithm when it verifies. The signing secret must be at least 32 characters and not a known placeholder, or the central server will not start; a worker only checks that it is present.
- **Refresh token.** 64 random bytes, valid 7 days by default, stored only as a SHA-256 hash, one row per session. Every refresh claims the old row and writes the new one in a single transaction under the person's credentials lock. A replayed cookie after rotation is refused (a replay within 30 seconds is answered "superseded", which confirms a live session; nothing is minted).
- **Session limits.** At most 5 concurrent sessions per person by default; an idle session ends after 60 minutes (setting `SESSION_IDLE_TIMEOUT_MINUTES`; 0 turns the check off).
- **Every request re-reads the account.** The check that accepts an access token also asks that the account is active, that the person is still a member of the organisation it names, that the session is not on the revocation list, and that the token was issued after the account's last "end every login". So deactivating an account, removing a membership, signing out or resetting a password stops tokens at once, not at expiry. The revocation list is held in Redis with a PostgreSQL write-through when Redis is on, and in PostgreSQL alone otherwise.
- **Sign-out** ends the session and revokes the token. "Sign out others" also ends the person's delegated connector grants. Role and organisation claims in the token are for display only; decisions read the grants from the database.
- **Cookies.** The session cookies are httpOnly with `SameSite=Lax`. `Secure` is on by default in a production process, but an operator can set `COOKIE_SECURE=false` and a production process does not refuse it; the secure configuration guide says never to. The cookie issued when a person switches organisation takes its `Secure` flag from the production check alone, not from that setting. Lax cookies are sent on a cross-site top-level navigation, so state changes are confined to non-GET routes.

Limits.

- Deactivating an account revokes nothing; while it is off every use is refused, but on reinstatement an earlier session resumes if it is still inside the idle timeout (and for up to the 7-day refresh life when the idle check is off). An account that must not return should be deleted or have its sessions ended, not only deactivated.
- "Sign out everywhere" writes no "ended since" marker, so a sign-in racing it can land after it.
- A file download link already handed out keeps working for up to an hour after sign-out. An embed token minted for a Discovery agent keeps working for up to 15 minutes after the agent's key is rotated. A Reports (Superset) session is not ended by Ziggy; see "Accepted for later".
- Sessions opened while sandbox mode was on (a development and staging feature that a production process refuses) are not all ended by an administrator's own "end every login".

5. Roles and permissions, and the two administrator tiers

One engine. Every route and every service-level action is decided by one policy engine reached through one API. More than a thousand route policy entries cover the routes. The server refuses to start when a route has none. Authority is held as permission grants, given to a person, a team, a service account or a role (a named bundle of grants). Grants are read from the database on every decision and nothing caches a decision, so a revoked grant takes effect on the next request. Two exceptions are stated: a History list's reach is cached for 10 seconds per viewer, and a View decision is kept for the whole of one long streamed request.

Verbs and powers. Each feature offers three access verbs (View, Author, Operate; an Author grant also satisfies Operate) and there are five workflow powers that are installation-wide and relevant to separation of duties: Merge, Promote, Rollback, Admin and BreakGlass. Ten built-in roles are shipped (Viewer, Auditor, Author, Agent Builder, Engineer, Merger, Promoter, Operator, Admin, Break-Glass-Operator); a custom role can be built from the same vocabulary. A role's priority number is display only: no decision reads it, and no decision reads a role name.

The two administrator tiers. There is no separate "super administrator" role. The tiers are told apart by the scope of the Admin grant, never by a role name or the token:

- An **organisation Admin** holds an Admin grant scoped to one organisation, and acts only inside organisations they both administer and belong to.
- An **installation Admin** holds an Admin grant at any other scope (in practice the built-in Admin role at installation scope).

The tier rules are applied on every path:

- An organisation Admin may hand out only roles made of the three access verbs, only inside an organisation they administer, and may remove only assignments scoped to those organisations. They cannot put a workflow power in a role, an exception grant, a team's role, a service account's role or an invitation. A role an organisation owns may hold only access verbs and applies in that organisation alone.
- **Reach.** A route that names a person, a team or a service account asks whether the caller reaches it; one out of reach is answered as if it did not exist. An installation Admin's account is changed only by another installation Admin. Administering an organisation requires being a live member of it; leaving it revokes the grants and assignments scoped to it, in the same transaction.
- **Last installation administrator.** Every present path that could take installation-wide Admin away (unassigning a role, revoking admin scope, removing an exception grant, replacing a role's grants, deactivating or deleting the account, leaving the last organisation, removing a team member or deleting a team that carries the role) runs its check and its write under one database lock and is refused when nobody who keeps the authority could sign in. Nothing structural forces a future path to take the lock; the present ones are covered by tests.
- No administrator may deactivate or delete their own account.
- **MFA on the authority itself.** The built-in Admin and BreakGlass grants require a fresh second factor when used; so does the Admin grant an organisation Admin is given and every exception grant that carries a workflow power. A service account can never satisfy such a condition, so a machine cannot hold the built-in Admin role's authority.
- **Break-glass.** Using it needs the installation-wide BreakGlass grant, a written justification and a fresh second factor (always). It writes a record, notifies administrators, escalates if unreviewed after the review window (7 days by default), and the review must be done by someone other than the invoker.
- **Exception grants** need a reason (up to 500 characters) and a future expiry. There is no maximum lifetime.
- **Agents.** The routes for sign-in, MFA and access administration refuse a Discovery agent session, and a write route an agent reaches is either staged into a changeset or paused for a person.

Plain statements about administration.

- Role, grant and admin-scope editing is judged by the route's ordinary Admin check and the tier rules above. There is no second approver, no per-edit prompt, and an administrator granting themselves a capability is not refused (workflow powers are an installation Admin's alone, are recorded, and need a fresh second

factor at use). This was a recorded decision.
- A custom role built with an Admin grant that does not ask for a second factor lets credential acts run on the session alone. Only an installation Admin can build such a role; it exists so that a machine can be given administration.
- Allow decisions are recorded as a sample (one in ten by default, counted per server process); every Deny is recorded. Where the full Allow history is required, set the sample rate to 1.
- A built-in role reset and a role save are several statements; a database fault part-way could leave a role without its grants. Giving another organisation's role or team id to an organisation Admin answers "forbidden" where an unknown id answers "not found", which reveals that the id exists.

Start-up override. The refusal to start with an uncovered route or an unguarded anonymous route is turned into a warning when the operator sets `ALLOW_POLICY_COVERAGE_GAP=true`; a feature gate has the same switch, `ALLOW_CAPABILITY_COVERAGE_GAP=true`. Both are off by default, are in no shipped template, and are honoured in any environment. A route with no entry is still refused at request time. They are for emergency boot only and must never be left on in production.

6. Changes, approvals and access reviews

Governed items are changed on branches and merged through changesets. A changeset is merged only after an approval round, and a failed or missing round fails closed: the changeset cannot be merged. A separation-of-duties check refuses a merge by one of the changeset's own approvers. The people allowed to merge, promote or roll back hold those workflow powers; the separation-of-duties report lists people who hold conflicting powers (Author with Merge, Author with Promote, Merge with Promote). The built-in Admin role holds several, so it always appears there by design.

Limits of the controls above:

- SQL that no merge has approved does not always run with the access of the person trying it. For example, when another person resumes a failed or stopped run of an unapproved Pipeline version, it runs with the access of the person who launched it.
- Approving a step of a high-risk changeset does not always ask the approver for a fresh sign-in: the governance inbox accepts a changeset's step without that extra proof. The approver must still be the step's own resolved approver.
- Submitting and re-submitting a changeset, and re-opening one for a stale review, write the approval round outside the claim on its status. A race between two acts on one changeset can leave it with no round or with two (it fails closed), and a re-open racing a merge can leave the separation-of-duties check with no approvers to compare. See "Accepted for later".

Access reviews. The product offers reports to support a periodic review: separation-of-duties overlaps, grants expiring within 30 days, an access-review export (every person by capability with where it comes from, spreadsheet formulas neutralised), the access-decision log, and a leavers list. Two limits apply.

- The access-review export and the separation-of-duties report leave out access that is scoped to a single organisation: organisation-scoped grants and organisation-scoped role assignments are not listed, so an organisation Admin, or a person given a role inside one organisation, can appear with no row. Do not rely on either report as a complete list of access; the per-person effective-access view does include it.
- The reports are tools, not a control with a record: nothing records that a review was done, by whom or what was decided. The customer's process supplies that.

7. Organisation separation

Most customers run a single organisation, and for them this section describes protection they will not need day to day. The ability to create further organisations exists for special circumstances: each is firewalled from the others inside the same installation, with its own people, roles, data and settings, as set out below.

Organisations are the unit of separation inside one installation.

- **Request level.** A guard refuses a request that names an organisation other than the session's in its body or query; the few routes that legitimately name another carry an explicit mark and their service checks again.
- **Policy level.** The engine drops grants scoped to another organisation, or to none, and treats a resource of another organisation as outside any grant. Role assignments carry where they apply; "Admin in Acme" is not an installation Admin.
- **Data level.** Every row is written with the organisation its writer states, judged once, and reads are narrowed to the session's organisation. This read backstop does not apply inside interactive transactions, raw SQL or system work; those name the organisation explicitly. Membership of an organisation has one table; a team is never a source of membership.
- **Roles and teams** owned by an organisation apply in that organisation only, and their names cannot shadow a system role's.

Limits (Small, counted under "Accepted for later"): some answers reveal that an id belongs to another organisation (a different "forbidden" and "not found"); some lists name items, such as Sources, data models or tables, to a reader in the same organisation who may not open them (names and ids only, no rows and no secrets); and a few lookups by id read a row before the organisation is judged. Cases of this kind that the audit confirmed are graded and counted under "Accepted for later".

8. Credentials and secrets

- **Never returned.** Every field the code treats as a credential has a recorded rule: a request-only field is never returned; a shown-once value cannot be retrieved again; a masked value is masked on every route that returns it. The audit checked each such field against that rule and found it holding. API and settings answers show masked values.
- **Stored sealed or hashed.** Passwords, refresh tokens, backup codes, emailed codes, recovery tokens and trusted-device secrets are stored as hashes. Connection credentials, secret values, OAuth tokens, worker API keys, execution data and other sensitive columns are encrypted (section 9). Stored settings secrets (mail password, OAuth client secrets, AI provider keys, the agent-host token, chat integration credentials) are sealed with the personal-data key.
- **A secret goes only where it is bound.** A secret saved for one destination is bound to it, and a change of destination is judged before the secret is sent; the judgement for the stored mail password can be bypassed by an installation Admin (see "Accepted for later"). In the audit, no outbound connection was found to send a credential to a different host through a redirect.
- **Use needs the right to use it.** Naming a secret requires the right to use its value, not an administrator's type-wide right. One route (creating a Source) asks only that the secret exists in the organisation; see "Accepted for later".
- **Logs.** Request bodies are not logged by default. When an operator turns body logging on, it is masked by field name and value shape: a secret under a name no list knows is logged as it stands, and a production process does not refuse the switch. Keep `LOG_REQUEST_BODY` and `LOG_RESPONSE_BODY` off. Audit records never reach the local log files and the copy sent to the SIEM is masked. Log lines that callers write freely, and the raw text of a server error, are not masked by the log sink itself.
- **Execution callback secret.** The execution callback secret that Pipeline runs use is judged by shape at start in production and refused when weak. Wrong attempts at it are counted and warned about, but not capped beyond the global per-address request limit; the value is a long signed secret that cannot be

guessed.
- **Where a secret can still be seen.** The admin portal's table preview shows ten raw rows of any table without the fresh sign-in the portal's shell and tunnel ask (Small); values stored sealed appear as ciphertext, values stored in the clear, such as a restore code, appear as stored. The restore code is held in plain text in its table. The database password appears on the command line of the application's backup and restore commands while they run (see "Accepted for later").

9. Encryption at rest and in transit

Field encryption. Sensitive columns are encrypted by the application, before they reach PostgreSQL, with AES-256-GCM. 41 registered columns across 35 models are covered, in two key types: personal data (emails, IP addresses, user agents; 28 columns, 4 of them deterministic so they can be looked up) and the data-plane key type (connection credentials, secret values, OAuth tokens, worker API keys, execution logs and errors, run debug data; 13 columns).

- Every value carries the id of the key that sealed it, so keys rotate without downtime. Each non-deterministic value uses a fresh random 96-bit initialisation vector, and since a 2026 hardening is also bound to its own column, so a sealed value copied from one column does not open in another. Every string written to a registered column is sealed, including one that already looks sealed.
- Writes are covered through every route the code has: create, update, upsert, nested writes to any depth, and interactive transactions. The audit judged all 41 registered columns holding.
- Authenticator secrets use their own scrypt-derived per-value format.
- Deterministic encryption (emails and the target address of the audit trail) reveals whether two values are equal under one key; that is the price of looking them up.
- A value without an encryption prefix in a registered column is read as it stands (rows written before encryption existed); the key audit reports such values as unencrypted.

Keys. The keys are four sections in one JSON file named by `ENCRYPTION_KEYS_FILE`, validated at start. On a provisioned server the file has mode 600 and is mounted read-only into the application container. **The keys are plain hex in a file on the same host as the database and the application. There is no key management service, hardware module or envelope encryption.** A compromise of the host yields both the ciphertext and the keys. The root disk is encrypted with the cloud account's default disk key, which protects a detached disk, not a running host. The secure configuration guide recommends holding key material in a managed key service. One key type (`hipaa`) is required in the file and generated, but protects no data.

Rotation is manual and run by an operator; there is no schedule, no age warning, and Ziggy does not claim automatic rotation. The tooling is strong: a new key version, then a migration that re-seals every value, checks each re-sealed value by opening it again and again by reading the row back in the same transaction, and stops on any mismatch. Starting and pausing a rotation need a fresh proof from the operator, recorded as an act. A separate audit step counts every value by the key it names and is the only thing that says an old key may be deleted. Limits: that audit does not count a pending authenticator secret (one being replaced), so it can say an old authenticator key may go while a half-finished enrolment still needs it (the enrolment would then fail and be started again); and each key replacement leaves a kept copy of the old keys file on the server, with no retention rule.

In transit.

- Browser to server: TLS terminated by the reverse proxy, automatic certificates, strict transport security with preload.
- Redis: TLS only (the plain port is off) with a password; clients do not present a certificate.
- PostgreSQL: TLS is switched on. The server does not require it of every client; the shipped client addresses use TLS, and production refuses to start when a database or Redis address on another server does not verify the certificate, unless the operator sets an explicit opt-out that is logged at every start. On the same host, the shipped single-server layout encrypts but does not verify the certificate, so the link is encrypted but its certificate is not checked.
- One self-signed 4096-bit certificate valid ten years serves PostgreSQL and Redis, is copied by hand to a second server, and has no rotation procedure.
- Outbound connections to customer systems verify the server's certificate unless the customer opts out for that connection, and the opt-out is recorded. An agent host or worker address can be plain `http`, in which case the bearer or API key travels in clear; nothing refuses it at start. Use `https`.
- Backups are database dumps sent to object storage over HTTPS. Registered columns in a dump are ciphertext; everything else is plain. The keys file is not part of the dump. Encryption of the stored dump depends on the bucket's own default; the guide covers the bucket.

10. Audit logging and the SIEM copy

The trail. Audit records are written by one service (a test fails when anything else writes the table) to a database table. The target email, client address and user agent in each record are sealed. A record carries the actor (a person or a service account), the organisation, the event type (109 types, covering sign-in and sessions, passwords, MFA, users and invitations, roles and grants, governance, agents, break-glass, restores, retention, SIEM changes and key rotation), and metadata with the names of body fields, never their values.

- **Acts are recorded by the route that takes them.** A global interceptor writes the record of any route whose policy declares one before the answer is sent, and a record that cannot be written fails the request. Acts that delete audit data write first and refuse when they cannot. An older path through an event is best-effort: a failed write is logged and the act has already happened.
- Refused sign-ins, step-ups and many refusals are recorded; some refusals are not (wrong reset codes and used-up attempts, refusals after a good LDAP bind, a refused refresh after the race window). A file made public, a partly applied AI runtime settings save and the revoking of several people's exception grants in one call leave thinner records or none (Small).
- **Who may read it.** Reading, searching, and managing log files and levels is limited to installation Admins. An agent session is refused on the sensitive routes.
- **Not append-only.** The table is an ordinary table: no trigger or privilege makes it append-only, and the application's own database role deletes (retention) and updates (anonymising a removed person) rows. Detection of tampering by someone with database write access rests on the SIEM copy. Deleting a person anonymises their audit entries only inside the deleting administrator's organisation, and does not change the SIEM copy, so an erasure is partial.

The SIEM copy. After a row is written, a masked copy is sent to AWS CloudWatch Logs; only CloudWatch is built (naming another provider forwards nothing). Application log lines also go there, at every level.

- Both shipped templates set `SIEM_PROVIDER=cloudwatch` and `SIEM_FAIL_CLOSED=true`. With fail-closed on, a process with no usable provider, or whose provider cannot start, does not start. The code default is off, and an operator can turn it off; in production that only prints a warning. A mistyped value of the setting refuses the start.
- Any change to how the trail is copied is recorded at the next central start, before the server serves anything; if that record cannot be written the server does not start.
- The SIEM settings are read from the process environment only, never from the database.
- Delivery is retried with backoff and entries that failed are kept for the next flush, but the buffer is in memory and bounded (10,000 entries; the oldest are dropped, with the count printed to the console). A crash, or an outage longer than the buffer, loses entries; the database row remains. There is no durable queue and no alert on loss.

- The log-group retention in the shipped templates is 30 days, shorter than the database's. Set it to at least your audit period.

Alerts. Alerts are mailed as digests to installation administrators who asked for them, from a list of event types an Admin chooses (defaults: failed and locked sign-ins, MFA failures, disabled or recovered MFA, an administrator override of MFA, password reset requests, user deletion, role changes, instance, database and worker health, data-quality failures). Alerts are raised only for event types that reach the alert listener. The listener hears the older event path only: event types written directly by the audit writer, which includes every act recorded by the route interceptor (security setting changes, access changes, credential changes, records erased), retention and sweep records, database restores and SIEM changes, can never raise an alert, though the settings screen accepts them. The record and the SIEM copy are not affected. Do not rely on an alert for those events; rely on the trail and the SIEM. Break-glass raises an in-app notification to installation administrators and a review reminder rather than an alert.

Application and request logs. Application logs go to the console and to hourly rotated files kept 14 days, and to the SIEM. The log level defaults to debug and no template sets it, so a provisioned production server logs debug lines locally; set `LOG_LEVEL=info`. HTTP request lines do not name the user (who acted is in the audit trail).

11. Retention

One scheduled sweep (03:00 UTC by default, run by one elected server) removes expired rows. Defaults:

Data	Default
Audit log, ordinary events	365 days
Audit log, security events (failed and locked sign-ins, password resets, restores, SIEM and retention changes, MFA overrides, key rotation, security setting, access and credential changes, records erased, and others)	730 days
Access-decision audit	730 days
Soft-deleted rows (hard flush)	14 days; 90 for high-impact tables
Local log files	14 days
SIEM copy	30 days in the templates; 365 in the code

Other classes (SQL telemetry, notification delivery, provider events, AI logs and others) have their own windows, all editable. An installation Admin can change the windows. Deleting audit data is on the record: a change to a window and every sweep (and the manual hard flush) writes its record first and refuses when it cannot. The API accepts whole days from 1, so an Admin can set the audit window to one day; the environment variables accept 0 or a negative number. No floor protects security events from early deletion except that the change and the sweep are recorded and the SIEM copy sits outside the database. The manual "Run now" deletes only the signed-in organisation's rows (less than asked, never more); the nightly sweep covers all.

12. The admin portal and the reference deployment

The admin portal is a separate application with its own sign-in (Auth0), used to deploy and operate Ziggy servers.

- Tokens are verified against the tenant's published keys, and the audience is checked; the portal refuses to start when sign-in is not configured, unless it is started in development mode. In development mode with no sign-in configured, every guard admits every request and only a loopback bind protects it: never run a reachable portal in development mode.
- A route is administrator-only unless explicitly marked wider. Every act (any route that is not a read) asks a sign-in within the last 5 minutes, and its beginning is recorded before it starts; if the record cannot be written the request is refused. A refused fresh proof is recorded. Reads are not recorded.
- Any account holding a valid token for the portal is at least a viewer: it can read infrastructure and instance summaries, the generated compliance documents and the cloud viewer role. Limit portal accounts to staff.
- Cloud credentials are issued by the portal using its own cloud identity and naming the person as the session's source identity, recorded before any credential is made. **In an environment deployed from earlier infrastructure code, the cloud roles still trust any holder of a portal token until the new trust rule is applied; applying it is a deployment step, the first item under "Accepted for later".**
- Bearer tokens are not revocable and no account state is read, so a token stays good until it expires; one fresh proof serves every act in its 5 minutes. The portal sets no content security headers and has no request limiter of its own; it binds to the loopback address unless told otherwise.
- The portal includes an in-house security self-check that runs 42 probes against a disposable clone of a deployed stack. It is a complementary self-check, not independent testing; see the security testing document.

The reference deployment (AWS, Terraform and Docker Compose; the only cloud covered) is described for hardening in the secure configuration guide. In summary: key-only SSH for one user, opened only to listed operator addresses and refused for the whole internet; the security group opens only ports 80 and 443 to the world; the database and Redis ports are closed to the internet; encrypted root disks; encrypted and locked remote state; image repositories with scan on push. Instances sit in public subnets with their own addresses (see the design notes).

13. How this was checked

Nothing is sampled. A script reads the source and lists every route, WebSocket stream, command, setting, secret-bearing field, encrypted field and outbound connection, and a few shared controls on which many items rest. The audit of this commit listed **1,581 items**: 1,154 routes of the central server, 98 routes of the admin portal, 7 WebSocket gateways, 7 commands, 140 settings, 74 secret-bearing fields, 41 encrypted fields, 52 outbound connections and 8 shared controls (the guard and evaluator, the token check, the organisation scope, fresh proof, logging, the audit writer, global wiring and the admin portal's guards).

How each item is judged. Each item is read end to end, from the route to its permission check, service and data access, against a fixed checklist for its type (the checklist is the appendix to this document). Each answer is pass, fail or not applicable, and cites the lines of code it rests on. The reader is an AI model: Claude Opus (5.5). **The audit is AI-assisted. It was carried out between 23 and 24 October 2026 by AI models reading the code; no human auditor and no independent party examined the code.** Checklist version 2 asked 27 questions of each route and admin-portal route, 7 of each gateway, 9 of each command, 3 of each encrypted field, 2 of each secret-bearing field, 2 of each setting, 6 of each outbound connection and 4 of each shared control.

Nothing is skipped. A coverage gate refuses to finish while any item is unjudged. At the end: 1,581 items judged, none missing, none incomplete, none unsettled.

Failures are not taken on trust. 137 items had at least one failed answer. The failures that could amount to a release blocker (46 verdicts, grouped into 33 causes) were each traced a second time through the code, again by AI reading; 29 causes were confirmed and 4 were traced and set aside as not blockers. The other failures were each reviewed and recorded with the reason they are not blockers. Ziggy's vendor then graded each confirmed cause, and every one is either listed under "Accepted for later" or is a Small counted there.

The gate. Four rules decide what blocks a release:

1. a defect in sign-in, MFA, sessions, permissions, organisation separation, credentials, secrets or keys, or in the integrity of the audit log, that the customer cannot close through configuration;
2. an unsafe setting that a production process accepts without refusing to start, where the safe value is not the default (a shipped template that sets the unsafe value counts as the default);
3. a control that a pack would describe but the code does not apply, or a tool that reports a false assurance;
4. a way for any signed-in user, or no user, to reach another person's data, an administrative action or a secret.

A confirmed blocker is either fixed or, by a recorded decision of the vendor, graded and accepted for later; the pack is not produced while one is undecided. This pack was produced because none is: every confirmed item was decided, and each one that is not fixed appears below.

Track record. Ziggy runs frequent audits, including one before each release. Every Serious blocker they found was fixed before this release, reviewed by a fresh AI review session that did not write the fix, and audited again. The Mediums and Smalls that were not fixed are on the list under "Accepted for later". The one remaining deployment step for installations of the reference cloud setup is also on that list.

Standing questions. A specific concern of a customer can be added as a standing question: every later audit then answers it for every item it concerns. Ziggy's vendor does this on request (info@ziggyplatform.com, or Freddy May directly at freddy.may@ziggyplatform.com).

Its limits. The audit examines Ziggy's own source code. Third-party libraries are outside it (the secure development document says how dependencies are handled). It is not a penetration test, and it did not run the product, examine any deployed environment or the cloud account of the reference deployment.

14. Design notes

These are decisions recorded by Ziggy's vendor, with their rationale, for risks that remain open in the code or behaviour that is intended.

Application instances sit in public subnets. In the reference deployment instances have public addresses and sit in public subnets, directly reachable from the internet. The decision was to keep them behind tightly scoped, fail-closed security groups: SSH, database and Redis ports default to deny, and the Terraform validation rejects a rule open to the whole internet. A planned improvement moves application instances to private subnets behind a dedicated public reverse-proxy instance and a NAT gateway; customers who need it now can deploy that way.

Access tokens are signed with a shared secret (HMAC, SHA-256). The decision is that an asymmetric signature gives no practical benefit here: every application node both issues and verifies tokens, so a split would not reduce how widely the key is distributed. The compensating controls are that the signing secret must be long enough and not a known placeholder (the central server enforces this at start). A worker also holds the signing secret and verifies access tokens at its live-connection door, and it only checks that the secret is present, not its strength; give workers the same strength of secret. The recommendation is to source the secret from a managed secret store with rotation.

The LDAP bind credential is an environment variable. The decision is that the code offers no in-code mitigation: the service account's bind password is supplied through the environment of the server. The recommendation is to inject the bind credentials at runtime from a managed secret store rather than as a static value.

An ML model's editor works with the owner's data access. By design: a person given edit access to a machine-learning model can train it, aim and run its batches and read its training statistics on the owner's access to the source data, including data they could not read directly. This holds while two conditions hold, and the audit found both present: the sharing screen shows the owner, before sharing, that an editor will train and run the model on the owner's access to its data (and the server refuses to grant edit access unless the request says the owner was shown it); and a person with only View access and no right to the data sees a summary with every data-derived value withheld and can predict only on rows they supply.

The earlier recorded note on the cloud roles' trust rule is superseded. An earlier decision accepted that the cloud roles the portal hands out trust the sign-in audience alone. The code no longer does this; it trusts only the portal's own cloud identity. The remaining risk is that an environment has not had the change applied; it is a deployment step, the first item below.

15. Accepted for later

These are the open Serious and Medium items of the vendor's security register at this commit, and the Small items as a count per area. Each says what could happen, why that is unlikely, and the date by which it will be fixed, within the times of the vulnerability policy. None needs to be fixed before release. Where an item is a customer step, it is also a line of the "Before go-live" list in the secure configuration guide.

AL-1. Deployment step: apply the new trust rule for the admin portal's cloud roles. *Applies to:* installations built from the reference AWS infrastructure code before this release, with portal sign-in turned on. *What could happen until it is applied:* someone with any portal account could obtain the portal's administrator cloud role, which makes this Serious for those installations. *Why unlikely:* it needs a portal account, and customers who run their own infrastructure do not use this module. *Status:* fixed in the code at this commit; applying it is in the guide's "Before go-live" list.

AL-2. A maintenance command can write an open changeset's edit onto the main branch (Medium). *What could happen:* someone with shell and database access runs the drift-repair command with its apply option after an upgrade, and an unreviewed edit lands on the main branch. *Why unlikely:* it needs full server access, which could write the main branch directly anyway, and the command is a dry run by default. *To be fixed by:* 19 January 2027.

AL-3. An agent's approved call could run twice (Medium). *What could happen:* the identical approved call, sent twice at the same instant, runs twice. *Why unlikely:* an agent's calls run one after another, and the action was approved, so no new access results. *To be fixed by:* 19 January 2027.

AL-4. Backup and restore put the database password on the command line (Medium). *What could happen:* someone who can list processes on the application's host while a backup runs reads the database password. *Why unlikely:* it needs a foothold on that host at that moment. *To be fixed by:* 19 January 2027.

AL-5. LDAP links made before an earlier rule change are kept (Medium). *What could happen:* in an LDAP installation, a link made under the old rule can stay with, or pass to, the wrong directory entry. *Why unlikely:* it affects only links made before the rule changed, and needs two directory entries with clashing names. *Customer step:* one check of those links by an administrator, in "Before go-live". *To be fixed by:* 15 January 2027.

AL-6. Two submits of one changeset at the same instant can leave its approval round wrong (Medium). *What could happen:* the changeset is left with no approval round or with two. *Why unlikely:* it fails safe, because a changeset without an approval round cannot be merged. *To be fixed by:* 19 January 2027.

AL-7. What a reader is shown is judged from each route's declared answer (Medium). *What could happen:* a future route that declares no answer type could show content unjudged. *Why unlikely:* no current route does this, and a new route passes review. *To be fixed by:* 19 January 2027.

AL-8. A stale-review re-open racing a merge can skip the separation-of-duties check (Medium). *What could happen:* an approver's own merge is not refused. *Why unlikely:* it needs the re-open and the merge to land at the same instant. *To be fixed by:* 19 January 2027.

AL-9. A password reset code treats capital-letter variants of an address as one (Medium). *What could happen:* the owner of a mailbox whose address differs from a person's only in capital letters could reset that person's password. *Why unlikely:* the major mail systems (Gmail, Microsoft 365, Exchange) do not deliver such variants to different mailboxes, and it also needs open self sign-up. *To be fixed by:* 23 January 2027.

AL-10. Moving a data-quality check into a folder can carry an unreviewed edit onto the main branch (Medium). *What could happen:* an organisation Admin brings an unreviewed edit of a colleague's custom-SQL check onto the main branch. *Why unlikely:* it needs the organisation Admin role. *To be fixed by:* 23 January 2027.

AL-11. Renaming the mail settings can send the stored mail password to another server (Medium). *What could happen:* an installation Admin redirects the stored mail password to a mail server of their own. *Why unlikely:* it needs the installation Admin role, the highest the application has. *To be fixed by:* 23 January 2027.

AL-12. An organisation Admin can use a colleague's stored GitHub token for a Source (Medium). *What could happen:* the Admin reads the colleague's private repositories through it. *Why unlikely:* it needs the organisation Admin role and a colleague's token stored as an organisation secret. *To be fixed by:* 23 January 2027.

AL-13. An agent's preview can show a saved query's rows to an editor it was not shared with (Medium). *What could happen:* a person with edit rights on an agent sees rows of a saved query not shared with them. *Why unlikely:* it needs edit rights on that agent, granted by its owner. *To be fixed by:* 23 January 2027.

AL-14. A Reports (Superset) session can outlive sign-out (Medium). *What could happen:* a person who opened Reports and then signs out, is switched off or loses access keeps reading in an open Superset tab. *Why unlikely:* it needs a person who once had access. *Customer step:* set a short session lifetime in Superset. *To be fixed by:* 23 January 2027.

Small items. 141 Small items are open. Each gives no one more than their role already allows, or needs a rare state or a second mistake, or shows names, ids or counts within one organisation only. The vendor accepted all of them. They are:

Area	Open Small items
Sign-in, sessions and second factors	20
Secrets, keys and what travels in clear	39
Permissions, organisation separation and governance	28
What a reader, an email or an AI model is shown	23
Records of acts (audit)	15
Code a pipeline runs	16
Total	141

16. Appendix: the audit checklist

Every item of the audit was asked the questions for its type. Each answer is pass, fail or not applicable, and a question is never skipped silently. The questions are grouped by the type of item.

Routes of the central server and of the admin portal (27 questions each)

Asked of every route:

- Who may call it: its policy entry states the verb and scope a caller needs, and neither the handler nor the service lets a caller with less reach the data or the act.
- Which organisation: every id the route takes, and every row it reads or writes, is held to the caller's organisation or to installation authority, and a foreign id is answered as an unknown one.
- Fresh proof: an act that changes who can sign in or how, grants or removes authority, changes a security setting, reaches credentials or keys or the audit trail, or destroys data wholesale asks for proof that the caller holds the account now, or is refused to everyone it should be.
- Nothing secret goes out: the response, an error message, a log line and the SIEM copy carry no secret value, to any caller.
- The record: an act that changes access, sign-in, security settings, keys, retention or data wholesale writes an audit record through the one writer with its actor, and a refusal of a sign-in or step-up is recorded too.
- Untrusted input: nothing from the request reaches a shell, SQL text, file path, server-side URL fetch or template without being checked.
- State and method: a read changes nothing, and a state change cannot be triggered by a cross-site navigation carrying the session cookies alone.
- Limits: a route that checks a secret, a code or a password counts attempts atomically before it compares, with a cap that a parallel burst cannot pass.
- No value hides another: where a request can carry the same scoping value in more than one place, each is judged against the caller's authority on its own, and the value judged is the one the service uses.
- A widening parameter needs the installation tier: a parameter that widens a read or act beyond one organisation is judged by the installation tier before anything is looked up.
- The caller's own right: an act that opens a connection, credential or data for the caller opens it with the caller's right, never the owner's or the system's, and a query an AI model writes is held to what the caller may read.
- Authority read from the record: an act that decides by an owner or organisation reads it from the governed record's own column, never from a copy inside its saved content or from the request.

Asked where the act concerns sign-in and sessions (setting or changing a password, creating an account or attaching an identity, granting by matching an address, accepting a code, refreshing a session, issuing a stand-in credential, or checking a secret):

- A new password ends the old sessions: it revokes every refresh token of the account and every later use of a token issued before it is refused, and the same holds for deactivating an account.

- The mailbox is proven before the address earns anything: an address is proven by a source that shows control of the mailbox before it gets an invitation's role, a domain's organisation or approval, and every later path refuses an unproven account outside sandbox.
- A factor works only when it is turned on: a second-factor or recovery code is accepted only for a method the account has enabled and confirmed, read from the account's own configuration.
- Renewal refuses what sign-in refuses: refreshing a session refuses an inactive account, tokens issued before a password change, an unproven address and a revoked session, not only an invalid token.
- A stand-in ends with the session: a link token, visitor pass, signed URL or tunnel cookie ends when the session ends, works only for what it was issued for, and refuses use from another site.
- Guesses are counted across challenges: wrong tries are counted per account as well as per challenge, and the number of codes mailed to one account is capped.

Asked where an act should need fresh proof:

- The proof is this session's: the step-up or recent sign-in an act accepts was made on the same session that uses it, and it expires.
- The old one holds until the new is confirmed: an act that replaces a factor, recovery address, key or credential keeps the current one in force until its replacement is confirmed.

Asked where an act writes organisation membership or grants access in more than one write:

- One placement rule: every act that makes a person a member of an organisation goes through the one rule about who may place and a proven address.
- All or nothing: if a later write fails, the earlier one is undone, so nobody is left with part of the access.

Asked where the server later calls or runs something for a user (a URL, a credential-carrying step, user-written code or a query, a job, a schedule):

- The address check covers every call made for a user: every call to a user-supplied URL or host passes the internal-address check on the address actually connected to, after each redirect and on reused sockets.
- A credential stays with its host: a credential set for one host cannot reach another through a redirect, a host built from row data, or a secret resolved later to a different one.
- A user's setting cannot lift the server's bound: a value an author sets cannot raise or turn off a bound the server applies to time, memory, response size or queued work.
- User code holds nothing of the server's: code a user writes cannot read the server's environment, signal its process or reach the network except through the server's checked calls.
- Judged again when it acts: the later act judges organisation and authority again by the same rule the save used, and a thing saved before a rule existed is judged before it acts.

WebSocket streams (7 questions each)

- The connection is authenticated by a token whose revocation, the account's active state and its organisation are checked, and a role or grant is required where the stream is not for every signed-in user.
- Every message handler checks that the caller may act on the resource it names, not only that the connection is signed in.
- Nothing secret is sent down the stream, and nothing from a message reaches a shell, SQL, path or URL unchecked.
- Opening the socket refuses everything a sign-in would, and a stand-in credential ends with the session, works only for what it was issued for and refuses use from another site.
- Where a message or the connection can carry the same scoping value in more than one place, each is judged on its own against the caller's authority and the value judged is the one the handler uses.
- A handler that opens a connection, credential or data for the caller opens it with the caller's right, and a query an AI model writes is held to what the caller may read.
- A handler that accepts a second-factor or recovery code accepts it only for a confirmed, enabled method, and one that accepts fresh proof accepts only proof made on this session that has not expired.

Command-line commands (9 questions each)

- It requires the authority its act needs, read from the grant, with an active account, and says so where the command line's own sign-in is a convenience rather than a boundary.
- It writes an audit record through the one writer for an act that requires one, with the operator as the actor.
- It prints no secret value and quotes none in an error.
- What it reports is true: a "done" or "safe to remove" it prints holds for every value it is about.
- A command that sets, changes or resets a password, or deactivates an account, revokes every refresh token of the account and refuses later use of tokens issued before it.
- A command that creates an account or grants anything by matching an address gives nothing tied to the address before it is proven, and every later path refuses an unproven account outside sandbox.
- A command that makes a person a member of an organisation goes through the one placement rule.
- A command that grants access in more than one write undoes the earlier writes when a later one fails.
- A key-rotation command covers every column the code seals with each key type, including columns sealed by a direct call outside the encryption extension, so nothing says an old key may go while a value still needs it.

Encrypted fields (3 questions each)

- Every write path encrypts the field, and no read path returns it to a caller who may not see it.
- Key rotation covers it: the migration re-seals it and the audit counts it under its key type.
- Every write goes through a path rotation knows; none seals it by a direct call that the rotation commands do not read.

Secret-bearing fields (2 questions each)

- The recorded rule for the field is true in the code: a request-only field is never returned, a shown-once value cannot be retrieved, and a masked one is masked on every route that returns it.
- The field is not logged, in a body log or an error message, and is not sent to the SIEM.

Settings (2 questions each)

- If a wrong value weakens sign-in, sessions, permissions, organisation separation, secrets, keys, the audit trail or the SIEM, then either its default is the safe value or a production process refuses to start on the unsafe one.
- The shipped templates set it to the safe value, under the name the code reads.

Outbound connections (6 questions each)

- TLS, when used, verifies the server, unless the customer opts out for that connection and the opt-out is recorded.
- A URL or host from a user or a stored row cannot make the server reach an address it should not, such as the internal network or the metadata service, or the reach is the stated purpose and is limited to the people allowed to configure it.
- Credentials it sends go only to the endpoint they belong to and never into a log line, an error or a URL.
- Every call made for a user passes the internal-address check on the address actually connected to, after each redirect and on reused sockets.
- A credential set for one host cannot reach another through a redirect, a host built from row data, or a secret resolved later to a different one.
- A value the author of a step sets cannot raise or turn off a bound the server applies to time, memory, response size or queued work.

Shared controls (4 questions each)

These are cross-cutting controls that many items rest on: the guard and evaluator, the token check, the organisation scope, fresh proof, logging, the audit writer, global wiring and the admin portal's guards.

- Each claim made for the control is true at this commit, with the lines that make it so; a claim that no longer holds is rewritten or removed.
- Default deny: whatever the control does not recognise, such as an entry with no policy, an unknown verb or scope, a missing token claim or an unparsed organisation id, is refused, never passed.
- No way round: it is applied to everything it claims to cover, no handler-level override skips it, and every exemption it allows is listed by name.
- It fails closed: an error inside it, such as a thrown lookup, a database miss or a timeout, refuses rather than allows.

Ziggy Secure Configuration Guide

What this guide is

This guide lists every security-relevant setting of a Ziggy installation: what it does, its default, the value we recommend in production, what goes wrong if it is wrong, and whether the server refuses to start on an unsafe value. It ends with a "Before go-live" checklist and hardening advice for the reference deployment.

It was written from the source at the commit above: the configuration code under `app/server/src/config/`, the environment templates shipped with the reference deployment, and the admin portal's configuration. The audit that the rest of the pack rests on judged each of the 140 settings the server and the admin portal read from their environment. The audit was AI-assisted (see "How this was checked" in the security overview); this guide is a reading of the code, not a test of your installation.

Three kinds of setting appear here:

- **Environment settings** are read once, when a process starts, from its environment (the `.env` file the reference deployment writes beside the containers). Changing one needs a restart.
- **Settings kept in the application** (sign-up rule, password rule, sign-in attempt limit, MFA policy, alerting, retention windows, mail, sign-in providers) are changed by an installation administrator on screen, take effect without a restart, and are recorded in the audit trail. Where one of these also has an environment value, the saved setting wins.
- **Deployment settings** belong to your infrastructure code, not to Ziggy (the reference deployment's Terraform variables, for example).

"Refuses to start" means the process prints the reason and exits, so a bad value is found at the first start and not later. Many of those checks apply to a **production process** only. A process counts as production when `APP_ENVIRONMENT` or `NODE_ENV` is `production`. A process that is neither gets none of the production-only refusals, so set one of them (the reference templates set `APP_ENVIRONMENT=production`).

Values are never reproduced here. Where a setting is a secret, generate it fresh; do not copy a placeholder from a template or an example file.

1. Settings that stop a production server from starting

These are the checks to know first. A production server refuses to start when any is true.

Check	What it protects
`JWT_SECRET` is missing, under 32 characters, or a known placeholder (refused in every environment, central server)	Forged sign-in tokens
`ENCRYPTION_KEYS_FILE` is not set, cannot be read, is not valid, lacks any of the four key sections, or has a section without a valid current key	Encrypted data being stored in clear or unreadable
`DATABASE_URL` is missing or not a `postgres://` or `postgresql://` address (every environment)	Starting against nothing
A number setting is not a number, or `TRUST_PROXY_HOPS` is not a whole number from 0 to 5 (every environment)	Silent mis-settings
`TRUST_PROXY_HOPS` is not stated	Sign-in attempt limits counting the wrong caller
`LDAP_URL` does not start with `ldaps://`	A directory password sent in clear
`SANDBOX_MODE_ALLOWED=true`	The switch that lets an administrator sign in as another person
`SMTP_ALLOW_PLAINTEXT=true` (central server and worker)	Mail login and sign-in codes sent unencrypted
`EXECUTION_CALLBACK_SECRET` is set but is not a generated random value (under 32 characters, under 128 bits by kind of character, contains a guessable word or a published stand-in, or is a repeated or patterned value)	Callbacks naming any organisation being guessed
`MOVEMENT_FAULT_INJECTION=true`	A debug button that stops the server for every organisation
`ANTHROPIC_BASE_URL_FOR_TESTS` is set at all (outside production it must name this machine)	Prompts and the organisation's AI key going to another address
A database or Redis on another server is reached without checking its certificate, and the matching `..._ALLOW_UNVERIFIED` switch is not set	A network peer posing as your database
`WORKER_API_KEY` of a production worker is under 32 characters or a published stand-in	Anyone admitting requests to the worker
`SIEM_FAIL_CLOSED` is not one of `true`, `1`, `yes`, `false`, `0`, `no`; or is on with no usable log provider; or its provider cannot start (every environment)	Running unrecorded when the operator said not to
`SIEM_CLOUDWATCH_RETENTION_DAYS` is not a period CloudWatch accepts, nor 0 (every environment)	A typo deciding how long the audit copy lives
The admin portal has no `AUTH0_DOMAIN` outside development, or a domain with no `AUTH0_AUDIENCE`	A portal that admits anyone, or any token of your tenant

A refusal is not a substitute for review. The sections below say where a value can still be unsafe and the server will start.

2. Environment and network entry

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`APP_ENVIRONMENT`	Names the environment the process runs as and marks it production when `production`. Its value is also the first-boot hint for the installation's environment record	none	`production`	A process not marked production gets no production-only refusals, serves the interactive API documentation page, and defaults cookies to not Secure	No
`NODE_ENV`	A second way to mark a process production (the worker template sets it). Also decides whether the interactive API documentation page is served and whether the development database switch is honoured	`development`	`production` on every server and worker	As above. An unset value counts as development for the development database switch (section 7)	No
`TRUST_PROXY_HOPS`	How many proxies in front of the server are believed for a caller's address. The sign-in attempt limits count by that address	none	`1` for the shipped Caddy proxy; `2` with your own load balancer in front of it (and list the balancer as trusted in the proxy). `0` only with no proxy at all	Too high: a caller can write the address the limits count, so the limits can be dodged. Too low: every caller looks like the proxy and shares one limit	Yes: unset in production, or outside 0 to 5 anywhere
`CORS_ORIGIN`	The one origin given credentialed cross-origin access, and the consent page's origin in the AI-agent sign-in hand-off	`http://localhost:3000`	The `https` address people use for Ziggy	A wrong origin is trusted or the browser refuses the client. The reference portal replaces the template value when it provisions a domain; a manual deployment must	No
`CLIENT_URL`	Base of invitation links and of redirects after a provider sign-in; the origin a connection sign-in's return address must match	falls back to `OAUTH_SUCCESS_REDIRECT`, then `CORS_ORIGIN`, then a local address	Leave unset (so it follows the `https` redirect) or set to the same `https` address	Mailed links point at the wrong place; return addresses refused	No
`OAUTH_SUCCESS_REDIRECT`	Where a browser is sent after a Google or Microsoft sign-in	`/`	The `https` address people use for Ziggy	Sign-in ends on the wrong page	No
`COOKIE_SECURE`	Whether the sign-in cookies carry the Secure flag (sent only over HTTPS)	unset: Secure when the process is production	Leave **unset** in production	`false` is honoured in production and is not refused: sign-in cookies could then travel over plain HTTP. On a non-production deployment that sets `true`, the cookie issued when a person switches organisation follows the production test and not this setting, so use production identity on any deployment that serves real users	No
`ZIGGY_CALLBACK_URL`	The address that pipeline code and other callers use to call back into Ziggy	local address	The address those callers can reach, over `https` where the path leaves your network	Callbacks fail, or travel unencrypted	No
`PORT`	The port the server listens on	`3000`	As your proxy expects; do not publish it beyond the proxy	An exposed port bypasses the proxy's TLS	Yes if not a number
`LEADER_ELECTION_ENABLED`	One server at a time runs scheduled jobs (retention sweeps, integrity checks)	on	Leave on	`false` makes every server its own leader, so scheduled jobs run on each	No

The templates ship development addresses (`localhost`) for `CORS_ORIGIN`, `OAUTH_SUCCESS_REDIRECT` and the two sign-in callback addresses, and the edge proxy file ships a placeholder host and contact address. The reference portal replaces them when it provisions a domain; if you deploy by hand, replace every one before the first start.

3. Sign-in, sessions and tokens

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`JWT_SECRET`	Signs sign-in tokens (HS256)	none	32 random bytes, for example from `openssl rand -hex 32`, held in a managed secret store	Anyone who learns it can make a valid token for anyone. A worker must hold the same value as the central server; the worker only checks that it is present, so its strength rests on you	Yes (central): missing, under 32 characters, or a known placeholder
`JWT_EXPIRES_MINUTES`	Life of the access token	`15`	`15` or less	A longer life lengthens the window in which a stolen token works	Yes if not a number
`JWT_REFRESH_EXPIRES_DAYS`	Life of a session's refresh token	`7`	`7` or less	Longer sessions without a new sign-in	Yes if not a number

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`SESSION_IDLE_TIMEOUT_MINUTES`	Ends a session idle for this long	`60`	`60` or less. Do not set 0 or below	0 or below turns the idle check off	Yes if not a number
`MAX_CONCURRENT_SESSIONS`	Sessions a person may hold; a new sign-in ends the oldest over the number	`5`	`5` or less	More live sessions per person	Yes if not a number
`LDAP_URL`	Turns directory sign-in on	off	`ldaps://` address, or empty	A plain address would send the service account and users' passwords in clear	Yes in production if not `ldaps://`
`LDAP_BIND_DN`, `LDAP_BIND_CREDENTIALS`, `LDAP_SEARCH_BASE`	The service account and where to search	empty	A read-only service account limited to the search base. Inject the password from a managed secret store	The bind password is a plain environment variable, so anyone who can read the process environment can read it	No
`LDAP_SEARCH_FILTER`	Finds the entry for the typed name	`(uid={{username}})`	As your directory needs; keep it matching exactly one entry	The typed name is escaped; a filter that matches several entries signs no one in	No
`GOOGLE_CLIENT_ID`, `GOOGLE_CLIENT_SECRET`, `GOOGLE_CALLBACK_URL`	Google sign-in	empty	Your own client; callback on your `https` address	A development callback address breaks sign-in or sends the code to a local address. Settings saved on screen replace these	No
`MICROSOFT_CLIENT_ID`, `MICROSOFT_CLIENT_SECRET`, `MICROSOFT_CALLBACK_URL`, `MICROSOFT_TENANT_ID`	Microsoft sign-in	empty; tenant `common`	Your own client; your tenant id if you want to limit who may sign in	Under `common` the address Microsoft gives is treated as unproven. An address alone never links an existing account: linking needs its owner to prove it	No
`SANDBOX_MODE_ALLOWED`	Allows the sandbox, in which an administrator can sign in as another person	off	Leave unset (see section 7)	See section 7	Yes in production

Settings kept in the application (installation administrator):

Setting	What it does	Default	Recommended in production
Sign-up rule	`anonymous` lets anyone with an email address ask for an account; `domain_restricted` limits it to listed domains. A new account holds no role until an administrator approves it. Self sign-up is available only while the installation holds fewer than two organisations; with two or more, people arrive by invitation	`anonymous`	`domain_restricted` to your domains, or invitations only. **The first account to register on a new installation becomes its installation administrator**, so create that account yourself before the installation is reachable by anyone else
Password rule	Strength: at least 12 characters with upper case, lower case and a special character. Reuse history is kept and checked	history on, last 5 passwords (1 to 24)	Keep history on; there is no password expiry; no maximum length is enforced, and the hash reads only the first 72 bytes of a password
Sign-in attempt limit	Attempts per hour for one account from one address (1 to 100); twenty times that is allowed from one address across all accounts. After five wrong passwords an account is locked for 15 minutes per further wrong attempt	`5`	`5` or less. The limit counts by caller address, so it depends on `TRUST_PROXY_HOPS` being right

4. Multi-factor authentication

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`MFA_MANDATORY_ENABLED`	The server floor: anyone who holds a workflow power (Merge, Promote, Rollback, Admin, Break glass) must use MFA, whatever the on-screen policy says	on	Leave on	`false` is the deliberate opt-out and leaves MFA entirely to the on-screen policy. It is not refused in production	No
`MFA_RECOVERY_URL`	Base of the link mailed to a person recovering a lost second factor	`http://localhost:3000/mfa-recovery`	`https://<your Ziggy host>/mfa-recovery`	**The shipped templates do not set it, and it does not follow `CLIENT_URL`.** An installation built from the templates mails recovery links to a local address, which works for no one, so recovery by email does not work until you set it. A wrong host, or plain `http`, would carry the recovery token there; production does not check the value	No
`MFA_ISSUER`	The name shown in the authenticator app	`Ziggy`	Your own name	Cosmetic	No

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`MFA_PENDING_TOKEN_EXPIRES_MINUTES`	Life of the step between password and second factor	`5`	`5` or less	A longer window to finish a sign-in	Yes if not a number
`MFA_EMAIL_CODE_EXPIRES_MINUTES`	Life of an emailed code	`10`	`10` or less	As above	Yes if not a number
`MFA_DEVICE_TRUST_DAYS`	How long a trusted browser skips the second step	`14`	Your policy; shorter is safer. An administrator can revoke all trusted devices	A stolen trusted-device cookie works longer	Yes if not a number
`MFA_TRUST_REMINDER_DAYS`	Days before expiry to remind	`3`	Default	Cosmetic	Yes if not a number
`MFA_GRACE_PERIOD_DAYS`	Days a person the floor covers may take to enrol	`7`	`7` or less	Longer without a factor. Grace is time to enrol; it never stands in for a factor already in place	Yes if not a number
`MFA_MAX_VERIFICATION_ATTEMPTS`	Read, but **has no effect**: the lock steps on wrong codes are fixed in the code	`5`	Leave alone	Changing it changes nothing, so do not rely on it for lockout	Yes if not a number

Setting kept in the application: the MFA policy (Settings, Policies) sets, per role, whether MFA is required and the grace days. The default policy leaves MFA off on screen for ordinary roles. We recommend requiring MFA for every role, because the server floor covers only people who hold the listed powers.

5. Encryption keys, database and Redis

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`ENCRYPTION_KEYS_FILE`	Path of the file holding the encryption keys. Four sections: `pii`, `mfa`, `hipaa`, `zn`, each with a current version and one or more keys	none	A file readable only by the service owner (the reference deployment writes it mode 600 and mounts it read-only into the container). The same file on every server over one database. Never use a sample or test key file from the source tree	Keys in a file on the application host are protected only by that host's access controls. There is no key-management service in the product; the `hipaa` section is required but protects no field	Yes: missing, unreadable, malformed, a type missing, or no valid current key
`KEY_ROTATION_BATCH_SIZE`, `KEY_ROTATION_DELAY_MS`, `KEY_ROTATION_MAX_RETRIES`	Pace of a key migration	`100`, `10`, `3` (retries capped at 10)	Default	Slower or heavier migrations; nothing is exposed	Yes if not numbers
`DATABASE_URL`	The application database	none	An address with TLS. On the same host as the application: `sslmode=no-verify` encrypts but does not check the certificate. On **another server**: `sslmode=verify-full` and `sslrootcert=<the CA that signs it>`; add `DATABASE_TLS_SERVERNAME` when the host is an address and the certificate names something else	A database across a network without certificate checking can be posed as by anyone on that network	Yes: missing or not a postgres address. In production, an unverified link to another server is refused unless `DATABASE_ALLOW_UNVERIFIED` is set (section 7)
`LOCAL_DATABASE_URL` (worker)	A worker's own database for run records	none	As `DATABASE_URL`; the worker's certificate name is set with `LOCAL_DATABASE_TLS_SERVERNAME`	As above	As above
`DB_POOL_MAX`, `DB_POOL_MIN`, `DB_POOL_IDLE_TIMEOUT_MS`, `DB_POOL_CONNECTION_TIMEOUT_MS`, `DB_POOL_KEEPALIVE`, `DB_POOL_KEEPALIVE_DELAY_MS`	Connection pool tuning	`20`, `2`, `30000`, `5000`, on, `10000`	Default	Availability only	Yes if not numbers

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`DB_STATEMENT_TIMEOUT_MS`	Longest a database statement may run	`30000`	Default	A very large value lets a slow query hold a connection	Yes if not a number
`REDIS_ENABLED`	Uses Redis for throttling and the token revocation list	on	Leave on	`false`: revocations and break-glass tokens are kept in the database, and attempt limits are counted in memory per process, which on more than one server multiplies the caps	No
`REDIS_HOST`, `REDIS_PORT`, `REDIS_DB`, `REDIS_PASSWORD`	Where Redis is and how to sign in to it	`localhost`, `6379`, `0`, none	A password (the reference Redis requires one); keep Redis closed to the internet	Redis without a password lets anyone who reaches it read or change counters and revocations	Yes if the numbers are not numbers
`REDIS_TLS`	Encrypts the Redis connection	off in code; `true` in the templates	`true`	Throttle counters, revocations and queued work in clear on the network	No
`REDIS_TLS_CA`, `REDIS_TLS_SERVERNAME`	The CA that signs Redis's certificate and the name to check	unset	Set both for a Redis on another server	Without a CA, an unverified Redis link is accepted only on the same host	Yes in production for an unverified link to another server, unless `REDIS_ALLOW_UNVERIFIED` is set
`REDIS_KEY_PREFIX`, `REDIS_TOKEN_BLACKLIST_PREFIX`, `REDIS_DIRECT_MAIN_BREAK_GLASS_PREFIX`	Names under which Redis keys are filed	`throttle:`, `token:blacklist:`, `versioning:direct-main-bgt:`	Default; give each installation sharing one Redis its own prefix	Two installations on one Redis would read each other's counters	No
`INSTALLATION_SIGNING_KEY_DIR`	Where the installation's bundle-signing keys live	`.installation-keys` beside the process	A persistent folder only the service owner can write (the templates put it under the files mount)	A folder that is lost loses the signing identity; a folder others can read exposes it. The server creates it owner-only	No
`FILES`	Folder for local log files and database backups	`./files`	A persistent folder only the service owner can read; not served over the web (nothing in the product serves it)	Backups and logs readable by others	No

6. Logging, the SIEM and retention

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`SIEM_PROVIDER`	Where a masked copy of every audit record and log line is sent. Only `cloudwatch` is built; `azure`, `gcp` and `datadog` forward nothing	none; `cloudwatch` in the templates	`cloudwatch`	Without a provider no copy exists outside the database. A name that is not built forwards nothing	Only with `SIEM_FAIL_CLOSED` on; otherwise a production start prints a warning
`SIEM_FAIL_CLOSED`	Refuses to run without a working SIEM copy	off in code; `true` in the templates	`true`	Off: a SIEM that is down or misconfigured lets the server run with no outside copy of the audit trail	A value that is not on or off is refused
`SIEM_CLOUDWATCH_LOG_GROUP`, `SIEM_CLOUDWATCH_LOG_STREAM`, `AWS_REGION`	Destination of the copy	`/app/logs`, `application`, `us-east-1`; the templates set a group per project and `eu-west-1`	Your own group, in a region you intend, in an AWS account whose logs only your security team can delete	A copy that the same people can delete as the data defeats its purpose	No
`SIEM_CLOUDWATCH_RETENTION_DAYS`	How long the log group keeps the copy	`365` in code; **`30` in the templates**	At least the period your regime requires for audit evidence (365 or more)	The copy that survives tampering with the database is the shortest-lived of the three places the audit trail is kept	Yes: only periods CloudWatch accepts, or 0 to leave the group's own
`SIEM_BATCH_SIZE`, `SIEM_FLUSH_INTERVAL_MS`, `SIEM_RETRY_ATTEMPTS`, `SIEM_RETRY_DELAY_MS`	Delivery tuning	`100`, `5000`, `3`, `1000`	Default	Timing only. Undelivered entries are held in memory (at most 10,000; the oldest are dropped and counted when it is full), so a crash or a long outage can lose entries from the copy; the database rows remain	Yes if not numbers
`AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`	A static AWS key for CloudWatch and file storage	unset	Leave unset and use the server's instance role	A static key is a long-lived secret on the host	No
`LOG_LEVEL`	Level written to the console and local files (the SIEM copy receives every level whatever this is)	`debug`; no template sets it	`info`	`debug` writes the fullest local log, kept 14 days, readable by installation administrators	No (an unknown value falls back to `info`)
`LOG_REQUEST_BODY`, `LOG_RESPONSE_BODY`	Writes request and response bodies into the log lines	off (`false` in the template)	**Off**	See the warning below	No: production is not refused for turning them on
`LOG_MAX_BODY_SIZE`, `LOG_TRUNCATE_AFTER`	Largest body considered, and where a logged body is cut	`10240`, `2000`	Default	Larger logs	Yes if not numbers
`RETENTION_AUDIT_LOG_DAYS`	Days ordinary audit records are kept	`365`	Your regime's period	A very small value deletes audit evidence early	Yes if not a number
`RETENTION_AUDIT_SECURITY_DAYS`	Days security-event records (failed or locked sign-ins, resets, restores, key changes, access and credential changes) are kept	`730`	Your regime's period, no shorter than the line above	As above	Yes if not a number
`RETENTION_ACCESS_DECISION_AUDIT_DAYS`	Days the record of access decisions is kept	`730`	Your regime's period	As above	Yes if not a number

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`RETENTION_SOFT_DELETE_DAYS`, `HARD_FLUSH_RETENTION_DAYS`, `RETENTION_CLASSIFICATION_APPROVAL_DAYS`, `RETENTION_SQL_TELEMETRY_DAYS`, `RETENTION_NOTIFICATION_DELIVERY_DAYS`, `RETENTION_PROVIDER_EVENT_DAYS`, `RETENTION_OUTPUT_PORT_EVENT_DAYS`, `RETENTION_AI_LOG_DAYS`, `RETENTION_INTEGRITY_AUDIT_RUN_DAYS`, `RETENTION_SEARCH_INDEX_RUN_DAYS`, `RETENTION_LISTENER_DEDUP_DAYS`, `RETENTION_BATCH_SIZE`, `HARD_FLUSH_BATCH_SIZE`	How long other records are kept, and the sweep's batch sizes	`90`, `14`, `30`, `30`, `90`, `90`, `90`, `365`, `14`, `30`, `30`, `1000`, `1000`	Set to your own retention policy	Shorter than your policy deletes data you need; longer keeps data you may be required to delete	Yes if not numbers
`RETENTION_SWEEP_CRON`, `INTEGRITY_CHECKS_CRON`	When the nightly sweep and integrity checks run (UTC)	`0 3 * * *`, `0 4 * * *`	Default	Timing only	No

No retention setting is refused for being too small. The environment accepts 0 or a negative number for a retention window, which would delete those records up to or past the present at the next sweep. The on-screen retention settings take whole days of at least one, and no minimum protects security records from a short window. Never set a window below your regime's period, and keep the SIEM copy longer than the database's.

Every change to a retention window, and every sweep, is recorded before it acts and refused if it cannot be recorded. The manual "Run now" deletes only the signed-in organisation's rows (it deletes less than asked, never more); the nightly sweep covers the whole installation.

> **Warning: request-body logging can write secrets to your logs.** When `LOG_REQUEST_BODY` (or `LOG_RESPONSE_BODY`) is switched on, the body of each request is written to the console, to the local log files and to the SIEM copy. Bodies are masked before they are written, by field name (names that look like a password, token, key, session or code, personal fields, and any field called `value`) and by the shape of the text (private keys, URL passwords, bearer and basic tokens, tokens in known formats). Masking cannot know every secret. **A secret held under a name the masking does not recognise, for example inside the settings of a connection to an external system, or typed unquoted into free text, is written as it stands.** Some connection secrets can therefore reach the logs, and from there the SIEM, where they are kept for the retention period. Some routes that carry sign-in secrets are excluded from body logging, and uploads and binary content are never logged, but the exclusion is not a promise for every route. Leave both settings off in production. If you must switch one on to diagnose a fault, switch it on for the shortest time, treat the logs and the SIEM copy from that time as holding secrets, and rotate any connection secret that passed through.

Other limits to know. Free-text log lines and the raw message and trace of a server error are written as given and are not masked. The request log line does not name the signed-in person; who acted is in the audit trail. Audit records themselves never go to the local log files.

Alerting (kept in the application). Alerts are mailed as digests to installation administrators who ask for them. Alerts are raised only for event types that reach the alert listener, which is a subset of what the audit trail records. The settings screen accepts every event type, including ones that cannot raise an alert, so an alert you switch on for such a type will not fire. Do not treat alerting as complete detection: read the audit trail and the SIEM copy.

7. Override settings

These settings relax a protection. Every one is **off by default**. None should be on in a production installation. Where the server refuses the setting in production, it says so. Where it does not, nothing but you stops it, and the "Before go-live" checklist asks you to confirm each is unset.

Setting	What it lets through	Refused in production?
`ALLOW_POLICY_COVERAGE_GAP`	The server checks at every start that each route has an access rule or is listed as public, and that each public route authenticates its caller some other way or is listed with a reason. Set to `true`, a failure of either check becomes a warning and the server starts. A route with no rule is still refused at request time, but **a public route with no sign-in check and no listed reason stays open to anyone, with no credential**. It is an emergency switch for an upgrade that would otherwise not start	No: honoured in any environment. Never leave it set
`ALLOW_CAPABILITY_COVERAGE_GAP`	The server checks at every start that each route that carries an access verb is tied to a feature the capability grants can name. Set to `true`, the failure becomes a warning. Such routes are checked at installation scope, so a grant for one feature cannot authorise them: it over-blocks people and does not open the route. It hides a coding mistake, so a feature's grants may not behave as the screen suggests	No: honoured in any environment. Never leave it set
`SMTP_ALLOW_PLAINTEXT`	Mail to a server that offers no encryption. Without it, every mail connection requires TLS (implicit on port 465, STARTTLS required elsewhere) before a password or message is sent	Yes: refused at start, and ignored at every send
`DATABASE_ALLOW_UNVERIFIED`	Accepting, from a database on another server, a certificate that is not checked. Anyone on that network can read the traffic or pose as the database. Every start logs that it is on	Not refused: it is the only way to excuse the refusal, so it is the operator's decision. Do not use it in production; add `sslmode=verify-full` and the CA instead
`REDIS_ALLOW_UNVERIFIED`	The same for a Redis on another server (revocation list, throttle counters, queued work). Every start logs that it is on	As above
`SANDBOX_MODE_ALLOWED`	The sandbox, in which an administrator can sign in as another person, and a command-line session skips the operator's second factor while the sandbox is active	Yes (central server). Never on a server with real people on it
`MOVEMENT_FAULT_INJECTION`	The pipeline editor's debug crash button, which stops the server and so every organisation's sessions and runs	Yes (central server)

Setting	What it lets through	Refused in production?
`ANTHROPIC_BASE_URL_FOR_TESTS`	Sends every AI prompt and the organisation's key to the address named (a test stand-in)	Yes in production; elsewhere the address must be on the same machine
`COOKIE_SECURE=false`	Sign-in cookies without the Secure flag, which can then travel over plain HTTP	**No.** Honoured in production. Never set it there
`MFA_MANDATORY_ENABLED=false`	Removes the server floor that makes anyone holding a workflow power use MFA; leaves MFA to the on-screen policy	No, by design. Leave on
`SIEM_FAIL_CLOSED=false`	Lets a server run with no working SIEM copy of the audit trail	No (a production start with no provider prints a warning). Leave `true`
`REDIS_ENABLED=false`	Moves revocations to the database and counts attempt limits in memory per process	No
`CREDENTIAL_POLICY=relaxed` (first start; later changed on screen)	Lets an installation store secret values for another environment and accept a bundle that carries sealed credentials. The default `strict` holds only this installation's own secret values and accepts none from a bundle	No. Keep `strict` unless you have decided to move credentials between environments
`DB_MIGRATION_ON=true`	Points the server at a fixed local database for development. It is honoured only when `NODE_ENV` is `development` **or unset**, and the reference central template sets no `NODE_ENV`, so a production server that has it set would honour it	No. Never set it; set `NODE_ENV=production` as well
`LOG_REQUEST_BODY=true`, `LOG_RESPONSE_BODY=true`	Request and response bodies in the logs (see the warning in section 6)	No
`NODE_TLS_REJECT_UNAUTHORIZED=0` and the AWS and Node settings that redirect or weaken calls to the SIEM (a custom endpoint, a custom set of trusted authorities)	Turning off certificate checking for the whole process, or sending the audit copy somewhere else. A change in these is recorded at the next central start	No. Never set them in production

Switches that make the server stricter (all off in production unless you set them): `FAIL_ON_ENV_DIVERGENCE=true` aborts start when an environment record has diverged from main (otherwise it only warns); `DELETE_POLICY_DRIFT_FATAL=true` aborts start on a delete-rule mismatch (outside production it always aborts); `FAIL_ON_STRUCTURAL_FK_GAP=true` aborts start on an unmodelled graph relationship. Setting them is a safe choice for a staging installation.

Development conveniences that are not security overrides but should not be set in production: `SKIP_DIRTY_INDEX_BACKFILL` (skips a start-up recount of change badges), `AI_SIDECAR_WARM_AT_BOOT` and the other `AI_SIDECAR_*` pool and timeout settings (tuning of the AI model runner), and `SUPERSET_DB_LOCALHOST_REWRITE` (rewrites a database on the host's `localhost` to a container host alias; empty it where Superset is not in a container on the same host).

Names a stored connection cannot read. The environment secret adapter refuses a `{{vault:NAME}}` reference to any setting listed in this guide, to any name in the families that libraries read for themselves (`AWS_*`, `PG*`, `ANTHROPIC_*`, `OPENAI_*`, `NODE_*`, `NPM_*`), and to a credential-shaped name beside one of the server's own, so a connection reads only a value the installation keeps for its connections and not one the server runs on.

8. Mail, workers, file storage, Reports, the AI agent host and tuning

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`SMTP_HOST`, `SMTP_PORT`, `SMTP_SECURE`, `SMTP_USERNAME`, `SMTP_PASSWORD`, `SMTP_FROM_EMAIL`	Fallback mail settings; mail settings saved on screen are read first	empty (no mail), `587`	A relay you control, with TLS. Port 465 uses implicit TLS; every other port requires STARTTLS whether or not `SMTP_SECURE` is true. Put the password on screen so it is sealed, not in the environment	No mail means no verification, reset or alert emails. A password saved on screen is sealed	Yes if the port is not a number; plaintext refused in production
`DEPLOYMENT_MODE`	`worker` makes the process a remote worker	`central`	As each server's role	A worker started as central, or the reverse, runs the wrong set of routes	No
`WORKER_API_KEY` (worker)	The only thing that admits the central server's requests to a worker	none	32 random bytes, different for each worker, from `openssl rand -hex 32`	Anyone with the key can run work on the worker. A worker's address should be `https`: a plain `http` address would carry the key in clear and no start refuses it	Yes in production: under 32 characters or a published stand-in
`WORKER_SLUG`, `WORKER_CLUSTER_NAME`	The worker's identity	none	Unique per worker	Cosmetic and routing	No
`EXECUTION_CALLBACK_SECRET`	A shared secret that admits SDK callbacks naming any organisation	empty (off)	Leave empty unless an SDK consumer needs it; then `openssl rand -hex 32`	A guessable value lets a caller act for any organisation; guesses are not otherwise blocked	Yes in production if set and not a generated value
`FILESTORE_STORAGE_PROVIDER`, `FILESTORE_S3_BUCKET`, `FILESTORE_S3_REGION`	Where uploaded files go	none	A private bucket with encryption on, public access blocked and no public policy	A public bucket exposes files	No

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`FILESTORE_PRESIGN_EXPIRY`	Life of an upload or download link	`3600` seconds	Default or shorter	A handed-out link works for its whole life, even after the person signs out or loses access	Yes if not a number
`BUNDLE_ATTACHMENT_PRESIGN_EXPIRY`	Life of the attachment links inside an exported bundle	`604800` (seven days)	As short as your transfer window allows	A longer window for anyone holding the bundle	Yes if not a number
`COVERING_BACKUP_S3_PREFIX`	Where a promotion's covering backup is copied	`backups/<environment>`	A prefix in the bucket that your backups use, encrypted	Backups beside non-backup data	No
`SUPERSET_API_URL`, `SUPERSET_PUBLIC_URL`	Where the Reports (Superset) service is, for the server and for browsers	local addresses	Internal and `https` public addresses	Plain-text links, or a service reachable from outside that should not be	No
`SUPERSET_LAUNCH_PRIVATE_KEY`, `SUPERSET_SERVICE_USERNAME`, `SUPERSET_SERVICE_PASSWORD`	The key that signs Reports launch tokens, and the account that keeps Reports in step	unset (Reports off)	Generate your own; hold them in a managed secret store, not in a plain environment file	A leaked launch key signs people into Reports	No
`SUPERSET_LAUNCH_TOKEN_TTL_SECONDS`	Life of a Reports launch token	`60`	`60` or less	A longer window to replay a launch link	Yes if not a number
`AGENT_HOST_URL`, `AGENT_HOST_TOKEN`, `AGENT_SESSION_ZIGGY_URL`	The AI agent host and the address it calls back on	local addresses; no token (agent off)	`https` addresses; a token of at least 32 random characters (the host refuses a shorter or placeholder one); set the token and its addresses together	A plain `http` address carries the bearer in clear; no start refuses it	No
`CONN_*` (pool limits, client lifetimes, socket limit, cap-alert threshold)	Tuning of the shared pool of connections to external systems	defaults	Default	Availability only	Yes if not numbers
`SERVER_URL`, `COMPANY_NAME`, `APP_NAME`	Read for display or not at all	local / `App` / `RGen`	Your own values	Cosmetic	No
`IS_AUTHORING_ENABLED`, `IS_EXECUTION_ENABLED`, `INSTALLATION_NAME`, `CREDENTIAL_POLICY`	First-boot values of the installation's own record (authoring on or off, running on or off, name, credential policy); later changed on screen	on, on, `Local Installation`, `strict`	Set before the first start; see the credential policy in section 7	A typo in the credential policy refuses the first start	`CREDENTIAL_POLICY`: yes if not `strict` or `relaxed`

Standard proxy and TLS variables that Node and its libraries read (`HTTP_PROXY`, `HTTPS_PROXY`, `NO_PROXY`, `SSL_CERT_FILE`, `SSL_CERT_DIR`, `GOOGLE_APPLICATION_CREDENTIALS`) are honoured as the libraries read them. If you run an outbound proxy, note that the check that blocks calls to internal addresses sees the proxy and not the target; have the proxy itself refuse internal destinations.

9. The admin portal

The admin portal is a separate application that manages the servers, their keys and backups. It has its own sign-in through Auth0 and no shipped environment template.

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`AUTH0_DOMAIN`	Turns portal sign-in on	none	Your Auth0 tenant	Unset turns every portal guard off	Yes outside `NODE_ENV=development`
`AUTH0_AUDIENCE`	The only audience of token the portal accepts	none	The portal's own API identifier	Without it a token issued to any other application of your tenant is accepted	Yes whenever a domain is set
`NODE_ENV`	`development` is the only mode that may run without sign-in	none	`production`. **Never run a portal that serves real operators with** `NODE_ENV=development`	With no domain, all three portal guards admit every request; only the loopback bind then protects it	No
`ADMIN_BIND_HOST`	The address the portal listens on	`127.0.0.1`	Leave on loopback and reach it through a tunnel or a private network	A public bind exposes a console that manages servers	No

Setting	What it does	Default	Recommended in production	If wrong	Refuses to start?
`AWS_REGION`, `AWS_ADMIN_ROLE_ARN`, `AWS_VIEWER_ROLE_ARN`, `STS_SESSION_DURATION`	The AWS roles handed to portal users and how long they last	`eu-west-1`, empty, empty, `3600`	The roles created by the reference Terraform; one hour or less	An empty role makes the credential exchange fail, so none are given. A role that trusts too much is the deployment's choice (see the checklist)	No
`ADMIN_AUDIT_LOG_FILE`, `ADMIN_AUDIT_CLOUDWATCH_LOG_GROUP`, `ADMIN_AUDIT_CLOUDWATCH_LOG_STREAM`	The portal's own record of what operators did, in a local file and optionally copied to CloudWatch	a file under the portal's data folder; no copy	Set the CloudWatch group so there is a copy off the host	The copy stops for good after one failed start, so check it after each start	No
`ANTHROPIC_KEY`	An AI key for the portal's own assistant	empty	Empty unless you use it; never in a shared file	A leaked key	No
`portal_principal_arn` (Terraform)	The AWS identity the portal's roles trust	empty	The portal's own identity, with permission to assume both roles and set a source identity	Required when `auth0_domain` is set: a plan refuses an empty value. An environment deployed from an older version of this code still has roles that trust a valid portal token for the audience until you apply this change	Plan refuses an empty value

The portal asks for a fresh sign-in (within five minutes) for every act that changes something, and records the start of each act before it runs. A portal bearer token cannot be revoked and is good until it expires.

10. Before go-live

Work through this list for every installation. Each line is a thing only you can do.

Identity and network

- ☐ `APP_ENVIRONMENT=production` and `NODE_ENV=production` on every server and worker, and on the portal.
- ☐ `TRUST_PROXY_HOPS` stated and correct for the proxies in front of you (`1` with the shipped Caddy; `2` behind your own load balancer, which you list as trusted in the proxy).
- ☐ `CORS_ORIGIN`, `OAUTH_SUCCESS_REDIRECT`, `CLIENT_URL` (if set), the Google and Microsoft callback addresses and `ZIGGY_CALLBACK_URL` are your `https` addresses, not the template's local ones.
- ☐ The edge proxy's host and contact address are real, and TLS certificates are issued for your host.
- ☐ `COOKIE_SECURE` is unset.
- ☐ `MFA_RECOVERY_URL` is set to your own `https` address. The templates leave it unset, so recovery by email does not work until you do.
- ☐ Only ports 80 and 443 are open to the internet; the database and Redis ports are not. SSH is open only to named operator addresses.

Secrets and keys

- ☐ `JWT_SECRET` is a fresh random value of at least 32 characters, held in a secret store, the same on the workers. No placeholder remains.
- ☐ The encryption keys file is generated for this installation (never a sample), mode owner-only, the same file on every server over one database, and stored apart from your database backups.
- ☐ You have written a key-rotation policy (for example yearly, and after any suspected exposure). Rotation is run by an operator: it is safe and verified but never automatic, and nothing warns you when a key is old.
- ☐ Do not remove an old authenticator (`mfa`) key while anyone may be part-way through authenticator set-up: the key audit does not count that half-finished secret, and the set-up would fail and have to be started again.
- ☐ After a key replacement, the old keys files the servers keep beside the new one are deleted once the key audit says the old key is gone. They are kept with no retention and hold old keys.
- ☐ No deploy, version switch or container restart is started while a key replacement is running, and only one portal administrator provisions or replaces keys at a time. A restart in the middle can leave servers on different keys over one database.
- ☐ Before you force a re-provision of a second server, copy any hand-set values from its `.env`: the new file is written from the template and no copy of the old one is kept.
- ☐ `WORKER_API_KEY` is a different random value of at least 32 characters for each worker. `EXECUTION_CALLBACK_SECRET` is either empty or a generated value (`openssl rand -hex 32`).
- ☐ The LDAP bind password, mail password, sign-in provider secrets, Reports keys and the agent host token come from a secret store or are saved on screen so they are sealed, not left in files other people can read.
- ☐ The address of the agent host, of every worker, and of the portal's agent link are `https`. A plain `http` address is not refused, and the key or bearer would travel in clear. The agent host's address and its bearer are saved together in one save, or the bearer retyped, so a stale binding cannot be repaired one address at a time.
- ☐ Shell access to the application host is limited to the people who must have it. A backup or restore run from the application puts the database password on a command line for its seconds, and the portal's remote agent runs any command it is sent.

Database

- ☐ `DATABASE_URL` uses TLS. A database on another server uses `sslmode=verify-full` with the CA, not `DATABASE_ALLOW_UNVERIFIED`.
- ☐ **The database role the application connects with is neither a superuser nor holds** `pg_execute_server_program`**.** A restore loads an archive as that role; a crafted archive could otherwise run a program on the database host. Create a dedicated role with only the rights Ziggy needs.

- [ ] Few people hold the application's database login. The audit tables are protected by the application's checks and not by a database trigger or privilege, so a person with the login can alter or drop them; the SIEM copy is what then shows it.
- [ ] Postgres and Redis accept only TLS from other servers, and their ports are not published beyond the hosts that need them.

Sign-in and access

- [ ] Sign up as the first user to create the installation's administrator, before opening the server to others.
- [ ] Sign-up rule is set to your domains or to invitations only.
- [ ] MFA is required for every role on screen, not only the roles the server floor covers. `MFA_MANDATORY_ENABLED` is not `false`.
- [ ] If you use directory sign-in: `LDAP_URL` is `ldaps://`; and, if you are upgrading an installation that already used directory sign-in, an administrator has checked once, by hand, the directory links made before the entry-matching rule was tightened (an account someone linked by typing a colleague's login stays linked to that colleague).
- [ ] If you add a native device sign-in application to your portal's identity provider, first check that it issues a fresh sign-in time; the portal refuses a stale or missing one either way, and nothing uses a device sign-in today.
- [ ] You have a plan to review access: the access-review export and the separation-of-duties report leave out access that is scoped to a single organisation, so review those organisations' role assignments on screen as well.

Logging and retention

- [ ] `SIEM_PROVIDER=cloudwatch` and `SIEM_FAIL_CLOSED=true`, in an account whose logs the application's operators cannot delete. Retention on the log group is at least your audit period; the template's 30 days is not enough for most regimes.
- [ ] `LOG_REQUEST_BODY` and `LOG_RESPONSE_BODY` are off; `LOG_LEVEL=info`.
- [ ] Retention windows match your policy and none is 0 or negative. The security-event window is at least as long as the ordinary one.
- [ ] You do not rely on the manual "Run now" to clear the whole installation; the nightly sweep does that.
- [ ] You have read the alerting limit in section 6 and have a way to review the audit trail and the SIEM copy that does not depend on an alert firing.
- [ ] The portal's own audit copy to CloudWatch is set, and you check after each portal start that it is running.

Overrides

- [ ] None of these is set: `ALLOW_POLICY_COVERAGE_GAP`, `ALLOW_CAPABILITY_COVERAGE_GAP`, `SMTP_ALLOW_PLAINTEXT`, `DATABASE_ALLOW_UNVERIFIED`, `REDIS_ALLOW_UNVERIFIED`, `SANDBOX_MODE_ALLOWED`, `MOVEMENT_FAULT_INJECTION`, `ANTHROPIC_BASE_URL_FOR_TESTS`, `DB_MIGRATION_ON`, `NODE_TLS_REJECT_UNAUTHORIZED`.
- [ ] `CREDENTIAL_POLICY` is `strict` (or, if you have decided otherwise, the decision is written down).
- [ ] Redis is on (`REDIS_ENABLED` not `false`) and leader election is on.

Outbound connections

- [ ] If you run an outbound proxy, it refuses internal destinations itself, because the server's internal-address check sees the proxy and not the target.

Reports (Superset), if you use it

- [ ] Set a short session lifetime in Superset's own configuration. Ziggy checks the person's right when Reports is opened, but nothing in Ziggy ends that Superset session: signing out of Ziggy, switching a person off or removing their right leaves a session already opened able to read the datasets it was launched with until Superset ends it.
- [ ] After any restore of Superset from a backup, run Ziggy's Reports re-sync. A Superset restored from an older backup can reuse object numbers Ziggy had already linked, and Ziggy could then update or delete someone else's object.

Admin portal and AWS

- [ ] The portal runs with `AUTH0_DOMAIN` and `AUTH0_AUDIENCE` set, bound to loopback, never with `NODE_ENV=development`.
- [ ] In every environment deployed with the reference Terraform and `auth0_domain` set: set `portal_principal_arn`, give that identity permission to assume both roles and to set a source identity, then run the plan and apply it, together with the current portal code. Until you do, an environment deployed from older code has AWS roles that trust any holder of a valid portal token for your audience.
- [ ] When you change mail or sign-in settings in the portal, check each server afterwards. A password field whose mask you partly delete before typing is saved as the leftover mask characters plus your text, and a save with a secret left masked can leave a server that has no value for it with none, while the save reports success on every server.

11. Hardening the reference deployment

The reference deployment is one Terraform root that builds a network, a security group and one to nine Ubuntu instances, each running Docker Compose with Caddy as the only published web entrance, the api, the client, and on the first server a Postgres and a Redis. What it does well: SSH is key-only and open only to listed operator addresses (an address of the whole internet is refused); only ports 80 and 443 are open to the internet; Redis accepts TLS only and requires a password; Postgres has TLS on; root disks are encrypted; the state is encrypted and locked; container images are scanned on push and have immutable tags (except `latest`).

You own the infrastructure, so treat the list below as advice for your own review of it.

Network

- The reference places application instances in public subnets with their own public addresses, behind a security group that allows only the ports above. The vendor has deferred moving them behind a proxy instance and a NAT gateway. If your standard is private subnets, build that in your own deployment.
- By default the machine running the plan has its own public address added to the SSH list. Set `auto_allow_operator_ip` to `false` and list your operator addresses in `ssh_sources` explicitly.
- All instances sit in one subnet and one availability zone. Spread them if you need that resilience. Turn on flow logs and consider restricting egress. There is no web application firewall in front of the proxy.
- The edge proxy's content security policy allows inline and eval scripts; tighten it if your client build allows. Replace the placeholder host and contact address in its file.

Hosts

- Require the instance metadata service to use tokens (`http_tokens = "required"`, hop limit 1). The reference does not, so a forged request from inside a container could reach the instance role's permissions.
- Pin the machine image by id per environment. The reference looks up the newest image each time, so a new or re-created instance is not the image you tested.
- The instance set-up downloads two programs and an apt key without checking a checksum or signature. Add the checks, or build your own image.
- SSH hardening is appended to the server's configuration and can be overridden by a drop-in file; add a drop-in loaded first with `MaxAuthTries`, `LoginGraceTime` and `X11Forwarding no`. The login user is in the `docker` group, which is root-equivalent on the host.
- Keep the Postgres and Redis ports closed to anything but the hosts that need them. Configure Postgres to accept only TLS from remote clients (`hostssl` rules in its access file with `scram-sha-256`). The reference sets TLS on but does not require it, so a client that does not ask for TLS is not forced.
- Redis runs with a password but without client certificates, and its password is part of the container command line, visible to anyone in the `docker` group. Use a secret mount for the password where you can, and client certificates for links between servers.
- The internal certificate is self-signed, 4096-bit, valid ten years, shared by Postgres and Redis, never rotated and copied by hand to each second server. Plan a rotation and consider a private certificate authority.

Storage and backups

- The storage bucket that holds Terraform state and the database dumps is **not created by the reference Terraform**, so none of its protections are in the code. Create it with versioning, a public-access block, default encryption with a key you control, access logging, a policy denying non-TLS requests, and a lifecycle or object-lock rule on the backups prefix. The instance role can delete its own backups.
- The backup upload sets no server-side encryption option, so a dump is encrypted only as the bucket defaults. Set `--sse aws:kms` on the upload or enforce it in the bucket policy.
- A dump holds application data: fields registered for encryption are ciphertext, everything else is plaintext, and the keys file is not in the dump. Keep the keys file and the dumps in separate places with separate access, and test a restore.
- A backup passes the database password on a command line for its seconds. Limit shell access to the host (see the checklist).

Keys

- The encryption keys are plain hex in a file on the same host as the database. There is no key management service in the product. For stronger protection, hold the key material in a managed key service and place it on the host at start, or wrap the file with a key you hold; restrict host access; keep the file out of images and backups.
- Set and write down a rotation cadence. Rotation is run by an operator, verified before it replaces a file, and never automatic.
- During authenticator set-up, a half-finished enrolment is sealed with the authenticator key but not counted by the key audit. Do not remove an old authenticator key while people may be part-way through set-up.

Operator machines and pipelines

- A worker-cluster apply writes the worker key and the central database address (with its credentials) to a file in the operator machine's temporary folder and leaves it there. Delete it after every apply and restrict who can read that folder.
- The instance role's image-registry and log permissions are broader than needed (all repositories for the portal's roles; any region and account in the log permission). Scope them to your own account, region and repositories. The registry pull pattern of the instance role does not match the repository names the tools create, so check which credential your hosts actually pull with.
- The reference repository runs no check of Terraform, compose files or images. Add `terraform validate` and a policy scanner to yours, make the image scan fail on high and critical findings, and sign your images. The tag `latest` can be moved.
- Read the Terraform variable descriptions with care. The description of `ssh_sources` suggests it also gates database and Redis access; the security group has no such rule.

Operations

- Write down who may reach the portal, how, and from where; the portal is a console for the whole installation.
- Make a restore drill part of your plan. A restore is an act by an installation administrator, is recorded, and loads an archive as the database role in the checklist above.

SOC 2 responsibility matrix

What this document is

This matrix maps the SOC 2 trust services criteria that a software product like Ziggy bears on to two columns of responsibility: what Ziggy provides, and what you, the customer who runs Ziggy, must do. Each entry also says where in this pack the evidence is.

You run Ziggy yourself, from the source code, on infrastructure you choose. Ziggy is therefore a set of controls inside your system boundary, not a service. Controls that depend on the host, the network, the cloud account, the people who use Ziggy, or the legal basis for holding personal data are yours. This document does not certify anything. It tells your assessor and your security team which controls the software supplies, which it does not, and where the software's limits are.

It covers the security criteria CC6.1 to CC6.8, CC7.1 to CC7.5 and CC8.1, availability A1.1 to A1.3, confidentiality C1.1 and C1.2, and privacy P1 to P8. Criteria outside that list (for example the control environment, risk assessment and vendor management, CC1 to CC5 and CC9) concern your organisation and are not mapped here.

How the claims were checked

Every statement under "Ziggy provides" was read from the source code at the commit above. The audit that did this was AI-assisted: an AI model read every route, socket, command, setting, secret-bearing field, encrypted field and outbound connection (1,581 items in all) against a fixed checklist. It was not performed or reviewed by an independent human auditor, and it is not a penetration test. "How this was checked" in the security overview describes the method and its limits. Where a control has a known limit, the limit is stated in the same entry. The security overview's "Accepted for later" section lists the open items in full.

Reading the evidence column

Name used below	Document in this pack
Security overview	`security-overview.md` : the controls area by area, "How this was checked", "Design notes", "Accepted for later"
Configuration guide	`secure-configuration-guide.md` : every security setting, "Before go-live", reference-deployment hardening
Secure development	`secure-development.md` : how Ziggy is built, reviewed and changed
Assurance report	`security-assurance-report.md` : how security is checked, and the guarantees the code enforces
Security testing	`security-testing.md` : how Ziggy is security tested
README	`00-README.md` : the shared-responsibility model and how to report a security issue

Summary

Criterion	Topic	Who carries most of it
CC6.1	Logical access security	Shared
CC6.2	Registering and authorising users	Shared
CC6.3	Roles, least privilege, separation of duties	Shared
CC6.4	Physical access	Customer
CC6.5	Ending protections when assets are retired	Shared
CC6.6	Threats from outside the system boundary	Shared
CC6.7	Moving and transmitting information	Shared
CC6.8	Unauthorised or malicious software	Shared
CC7.1	Detecting vulnerabilities and configuration change	Shared
CC7.2	Monitoring for anomalies	Shared
CC7.3	Evaluating security events	Customer, with tools from Ziggy
CC7.4	Responding to incidents	Customer, with tools from Ziggy
CC7.5	Recovering from incidents	Shared
CC8.1	Change management	Shared
A1.1	Capacity and demand	Shared
A1.2	Environmental protection, backup, recovery infrastructure	Customer, with tools from Ziggy
A1.3	Testing recovery	Customer
C1.1	Identifying and protecting confidential information	Shared
C1.2	Disposing of confidential information	Shared
P1	Notice	Customer
P2	Choice and consent	Customer
P3	Collection	Shared
P4	Use, retention and disposal	Shared

Criterion	Topic	Who carries most of it
P5	Access to personal information	Shared
P6	Disclosure and notification	Shared
P7	Quality	Shared
P8	Monitoring and enforcement	Customer, with tools from Ziggy

Three limits recur through the matrix, so they are stated once here:

- **The audit trail is the complete record; alerts are not.** Every audited act is written to the audit log and copied to your SIEM. Alerts are raised only for event types that reach the alert listener. An administrator can switch alerting on for more event types than can in fact raise one, so you must not rely on alerts alone to see every recorded act. Watch the SIEM copy.
- **The access-review export and the separation-of-duties report leave out organisation-scoped access.** They show installation-wide access. Access that was given inside one organisation, and the authority of organisation administrators, does not appear in them. Ziggy also keeps no record that a review took place. Your periodic review must cover both gaps (see CC6.3).
- **Encryption keys are a file on the application host.** There is no key-management service or hardware module in the reference design (see CC6.1 and C1.1).

Security: logical and physical access (CC6)

CC6.1 Logical access security software, infrastructure and architectures

Ziggy provides

- Default-deny authorisation. Every route needs an entry in a central policy registry or is marked public. A route with neither is refused for everyone, and the server refuses to start when such a route exists. The startup refusal can be switched off with a coverage-gap setting; see the configuration guide. Even then the route is refused at request time.
- Sign-in by local password, LDAP, Google and Microsoft, all ending in one gate that decides between a second-factor challenge, a mandatory set-up demand and a session. No path creates a session outside that gate.
- Sessions: a signed access token of 15 minutes in an HttpOnly cookie, and a refresh token of 7 days whose hash alone is stored. Refresh rotates the token, an idle session ends after 60 minutes, at most 5 sessions per person are kept, and sign-out or any credential change ends the affected sessions at once. The signing key is a shared secret; the server refuses to start without a long, non-placeholder value.
- Passwords of at least 12 characters with upper-case, lower-case and a special character, stored with a salted one-way hash, with reuse of the last five refused by default. Wrong guesses are counted before the comparison, so parallel attempts cannot all pass a lock.
- Separation of organisations inside one installation: reads and writes are scoped to the session's organisation by the data layer, and a request that names another organisation is refused.
- Encryption of sensitive fields in the application before they reach the database (see C1.1), and of credentials and secrets that Ziggy stores.
- An administration portal that is a separate application with its own sign-in, role check and a fresh-sign-in requirement for every act that changes something.

Customer must

- Restrict who may sign themselves up. The default installation allows anyone who can reach it to register. Set the sign-up policy (an allowed-domain list or invitation only) before the address is reachable. Create the administrator by signing up as the first user during installation.
- Set the production values the configuration guide lists: a strong signing secret, the secure cookie setting, the number of proxies in front of the server, and LDAP over `ldaps://` only. The server refuses several unsafe values in production but not all (the secure-cookie switch can still be turned off).
- Hold the application host and its encryption keys file to the standard your data requires. Keys are plain files on the host, readable by that host's administrators and mounted read-only into the application. If you need keys in a managed key service, wrap or fetch them there yourself.
- Never start the administration portal in development mode on a reachable address; in that mode it runs without sign-in.
- Limit portal accounts to staff. Anyone with a valid portal sign-in is at least a viewer, who can read infrastructure and instance summaries and obtain the read-only cloud role.
- Apply the portal's cloud-role hardening on every environment: set the portal's own cloud identity in the Terraform variables and apply, so that the roles it hands out trust only that identity. Environments deployed before this change still trust any holder of a valid portal token. This is a step only you can take (see "Before go-live").

Evidence: Security overview (sign-in, sessions and tokens, roles and permissions, organisation separation, credentials and secrets, the admin portal, "Accepted for later"); Configuration guide ("Before go-live"); Assurance report Part A.

CC6.2 Registration and authorisation of new users; removal of access

Ziggy provides

- Two ways in: an administrator invites an address into an organisation with a role chosen at that time, or a person signs themselves up. A self-registered account holds no role until an administrator approves it, and approval also places the person in an organisation.
- An account that has not proved its address gets no session. A code is mailed first.
- A person who signs in with Google or Microsoft is never linked to an existing account on an email match alone. They must prove ownership of the existing account first and pass its second factor.
- Deactivation by an installation administrator, with a reason required, never of one's own account and never of the last installation administrator who can sign in. A deactivated person is refused on the next request.
- Leaving an organisation removes the person's role assignments and grants scoped to it and their places on its teams.
- Hand-over of a leaver's work to a successor before permanent deletion, and permanent deletion of an account (see C1.2 and P4 for its limits).
- Every provisioning, role-change, deactivation and deletion act is recorded in the audit trail.

Customer must

- Define who approves access and run the process: who may invite, who approves self-registered accounts, and how a leaver is removed. Ziggy enforces the checks; it does not know your joiner and leaver process.

- Treat deactivation as a pause, not as an end for a person who must not return. Deactivation does not revoke sessions. If the account is reinstated, sessions that were still inside their idle timeout resume. For a permanent leaver, delete the account, or end their sessions and change their credentials, in addition to deactivating.
- Keep your identity provider (LDAP, Google, Microsoft) offboarding in step with Ziggy. Ziggy checks the account it holds; it does not learn of a removal at the provider.
- Review portal accounts in the same way (portal access is granted in Auth0, not in Ziggy).

Evidence: Security overview (sign-in, roles and permissions); Configuration guide (sign-up policy, "Before go-live"); README (shared-responsibility model).

CC6.3 Role-based access, least privilege and separation of duties

Ziggy provides

- Authority is held as grants given to people, teams, service accounts or roles. Ten built-in roles ship (Viewer, Auditor, Author, Agent Builder, Engineer, Merger, Promoter, Operator, Admin and Break-Glass-Operator). Custom roles can be built from view, author and operate rights per feature. Grants are read on every decision with no decision cache, so a revoked grant is effective on the next request.
- Two administrator tiers, told apart by the scope of the `Admin` grant and never by a role name: an installation administrator, and an organisation administrator whose authority is limited to the organisations they administer.
- A ceiling on what an organisation administrator may hand out. They cannot put a workflow power (Merge, Promote, Rollback, Admin, Break-Glass) in a role or exception grant, cannot act on the account of an installation administrator or of a person who belongs to an organisation they do not administer, and cannot assign a role in another organisation.
- Separation of duties safeguards: no administrator can deactivate or delete their own account, no path can remove the last installation administrator who can sign in, a built-in role cannot be renamed or deleted, and the workflow powers (Merge, Promote, Rollback) are separate rights that the Merger and Promoter roles carry on their own. Holders of a workflow power must use a second factor, by default.
- Exception grants for a named person, with a required reason and a required expiry.
- Break-glass: an emergency override that needs a second-factor proof, a written justification and a record, notifies installation administrators, and must be reviewed by someone other than the person who used it.
- A report of people who hold conflicting rights, a report of exception grants about to expire, a per-person view of effective access with where each right comes from, and an access-review export.

Customer must

- Apply least privilege in how you assign roles. The built-in Admin role also carries Merge, Promote and Rollback, so it always appears in the separation-of-duties report. Where you need separation, give people the Author, Merger and Promoter roles separately rather than Admin.
- Run a periodic access review and record it. Ziggy provides the reports; it keeps no record that a review was done, by whom, or what was decided.
- Cover the gap in the reports. The access-review export and the separation-of-duties report show installation-wide access only and leave out access given inside a single organisation, including the authority of organisation administrators. For organisation-scoped access, use the per-person effective-access view, which can be asked about a named organisation, and review the appointment of organisation administrators separately.
- Decide who holds installation administration. Role and grant authoring is judged by the `Admin` right and the tier rules; it does not ask a second person to approve an edit, and an administrator is not refused when granting to themselves. If you need a second approver, provide it by process, or limit installation administration to a small named group.
- When you build a custom role with `Admin`, keep the second-factor requirement on its `Admin` grant. A custom role built without it makes that credential act on the session alone.
- Keep the break-glass review honest: assign a reviewer and check the review queue.

Evidence: Security overview (roles and permissions with the two admin tiers, organisation separation, "Design notes", "Accepted for later"); Assurance report appendix (guarantees under permissions and approvals); Configuration guide (mandatory second factor, "Before go-live").

CC6.4 Physical access

Ziggy provides nothing here. Ziggy is software and has no physical component.

Customer must control physical access to the facilities and hardware that host Ziggy, or rely on your cloud provider's attestations for them. In the reference deployment on AWS, physical controls are the cloud provider's; obtain its SOC 2 report.

Evidence: README (shared-responsibility model).

CC6.5 Discontinuing logical and physical protections over assets

Ziggy provides

- Permanent deletion of a person's account, refused until nothing outstanding remains to hand over, and recorded.
- A retention sweep and a manual hard flush that remove expired and soft-deleted rows, recorded before they act.
- A key-rotation process that re-seals data under a new key and reports, only after a verified migration, whether an old key may be deleted.

Customer must

- Decommission servers, disks, snapshots and database backups under your own asset-disposal procedure. Ziggy does not manage the host or the storage.
- Remove old key files when the audit says an old key has gone. When the encryption keys file is replaced, the previous file is kept next to it, and nothing removes it. These copies hold old keys and accumulate until you delete them. The same applies to kept copies of environment files.
- Revoke the cloud roles and the portal accounts of a retired environment.

Evidence: Security overview (encryption at rest and in transit, retention); Configuration guide (retention, key rotation).

CC6.6 Protection against threats from outside the system boundaries

Ziggy provides

- A reverse proxy (Caddy) as the single published web entrance in the reference deployment, with automatic certificates, strict transport security, and security headers including a content security policy.

- Rate limits: a global limit per address, per-route limits on sign-in, registration, password reset, second-factor and recovery routes, and attempt limits per account and per address on sign-in.
- A trusted-proxy setting that the production server requires, so that limits count the real client address.
- Refusal of cross-organisation requests, validation that rejects undeclared fields, and responses that do not name internals on a server error.
- A remote worker that serves two controllers only, behind a deny-by-default route guard and its own API key. A worker's key is required to be at least 32 characters in production.
- The administration portal binds to the loopback address unless told otherwise.
- In the reference deployment, a security group that opens only 80 and 443 to the world, with SSH limited to listed operator addresses and an empty default. A list that includes the whole internet is refused. Database and Redis ports are not opened to the internet.

Customer must

- Configure the network to your own standard. In the reference deployment the instances sit in public subnets with their own public addresses behind the restrictive security group above. Moving them to private subnets behind a proxy instance and a NAT gateway is a hardening step you can take; it is not done by the reference Terraform.
- Place a web application firewall, DDoS protection and network flow logging in front of or around the deployment if your risk requires them. None ships with Ziggy.
- Require the metadata service version 2 on the hosts, and require encrypted connections from every database client in the database's own access rules (the reference sets TLS on but does not make the server refuse unencrypted clients).
- Set the allowed web origins, the trusted-proxy count and the callback addresses to your real addresses. The templates carry development values that must be replaced.
- Keep any worker and the agent host on `https` addresses; Ziggy does not refuse a plain `http` address for them.
- Commission independent penetration testing of your deployment, including the edge, which the built-in scan does not cover (see below and CC7.1).

Evidence: Security overview (sessions and tokens, the admin portal, encryption in transit); Configuration guide (reference-deployment hardening, "Before go-live"); Security testing.

CC6.7 Restricting the transmission, movement and removal of information

Ziggy provides

- Browser to server: TLS terminated by the proxy with strict transport security.
- Redis: TLS only, with a password. Postgres: TLS on. The server refuses to start in production when its connection to a database or Redis on another machine does not verify the certificate, unless an operator sets an explicit opt-out, which is logged on every start.
- The audit-log copy to the SIEM travels over HTTPS. Mail to your server is refused in production over plain SMTP. LDAP in production must be `ldaps://`.
- Secret values, credentials and tokens are masked in log lines and in the SIEM copy of audit data. Tokens never travel in an address.
- Registered sensitive fields come back from raw queries as ciphertext. Cells in the access-review export are neutralised against spreadsheet formula injection.
- Exports and access to data are subject to the same grants as any other read, and AI agents are held to the routes they are allowed and kept out of sign-in, second-factor and access administration.

Customer must

- Verify certificates between servers (`verify-full` for Postgres, a trust file for Redis) and do not use the opt-out settings in production.
- Decide what may leave your boundary. Every outbound connection is to a destination you or your administrators configure (data sources, mail server, sign-in providers, the SIEM, any AI provider). Control who may create them and review them.
- Protect backup transport and storage. Database dumps are copied to an object-store bucket. The upload sets no encryption option of its own, so protection depends on the bucket, which the reference Terraform does not manage. Apply encryption, public-access blocking, versioning, access logging and a deny-unencrypted-transport policy to it.
- Use data-loss controls at the endpoint and email layers that Ziggy cannot provide.

Evidence: Security overview (encryption at rest and in transit, credentials and secrets, audit logging and SIEM); Configuration guide (transport settings, "Before go-live", reference-deployment hardening).

CC6.8 Prevention and detection of unauthorised or malicious software

Ziggy provides

- Every change to the system's behaviour goes through a signed-in, authorised route. Outbound requests made by Pipelines are checked against internal addresses.
- Container images built for the reference deployment are pushed to a registry with scan-on-push enabled and immutable version tags (the `latest` tag can be moved).
- Source-level controls in the vendor's build: automatic checks that block an unsafe change before it is saved, secret scanning, and a separate review of every security-relevant change by a fresh AI review session that did not write it (see Secure development and the Assurance report).

Customer must

- Run endpoint and host protection (anti-malware, file integrity monitoring, patching) on the servers. Ziggy does not.
- Make image scanning a gate. The reference build reports scan results but does not stop a push on findings, and no image signing is applied. Add your own pipeline gate and, if required, signing.
- Verify what you install. The host set-up downloads a command-line tool and a credential helper without checking a published checksum or signature; verify or pin them in your own build.
- Run your own dependency monitoring on third-party packages. Ziggy's audit covers Ziggy's own source, and third-party libraries are outside it (see Secure development).

Evidence: Secure development; Assurance report Part A; Configuration guide (reference- deployment hardening).

Security: system operations (CC7)

CC7.1 Detecting and monitoring new vulnerabilities and configuration changes

Ziggy provides

- A full audit of the source, item by item, repeated for each release in this pack's series (see "How this was checked" in the security overview). It is AI-assisted and audits Ziggy's own code; it is not an independent audit.
- Start-up checks that refuse an unsafe configuration: a missing or weak signing secret, plain `ldap://`, sandbox sign-in allowed in production, an unstated proxy count, unverified cross-server database or Redis connections, an unsafe SIEM setting.
- A record of any change to how the audit trail is copied (written at the next start, before the server serves anything), and a record of every change to security settings made through the application.
- The in-house security test suite in the administration portal, a set of 42 automated probes run against a throwaway clone of a deployment. It is a complementary self-check, not independent testing, and it does not cover every kind of attack (see Security testing).

Customer must

- Commission independent penetration testing and keep its report. The in-house suite does not replace it, and it does not test the administration portal, the edge, or several attack classes (see Security testing).
- Scan your infrastructure and hosts for vulnerabilities and configuration drift; the reference Terraform has no policy scanning or drift detection in a pipeline.
- Track advisories for third-party packages and apply updates on your schedule.
- Take each new release's pack and read its "Accepted for later" list against your risk appetite.

Evidence: Security overview ("How this was checked", "Accepted for later"); Assurance report Part A and appendix; Security testing; Secure development.

CC7.2 Monitoring of system components for anomalies

Ziggy provides

- One audit log written by one writer. Acts that change security-relevant state are recorded by the route that takes them, and a record that cannot be written fails the request. Sign-in events, second-factor events, role and grant changes, account lifecycle, key operations, restore, retention changes and sweeps, and security setting changes are recorded.
- A decision audit of the authorisation engine: every denial is stored, and allowed decisions are sampled.
- A copy of each audit record, with sensitive values masked, sent to a SIEM (AWS CloudWatch Logs) after the row is written. In the shipped environment templates the SIEM is on and fail-closed: a process with no usable provider does not start.
- Application log lines and request lines to the console, rotated local files (kept 14 days) and the SIEM.
- Alerts for a list of event types and for health thresholds of instances, database and workers, mailed as digests to installation administrators, with acknowledge. Alerts are raised only for event types that reach the alert listener (see the limits at the top). Failed and locked sign-ins, second-factor failures and recoveries and password-reset requests are in the default list.
- Metrics for scraping and liveness and readiness endpoints.

Customer must

- Operate the SIEM. Set up the log group in your AWS account, give the server permission to write to it, set its retention to your audit period or longer (the shipped templates set 30 days, less than the database keeps ordinary audit rows), and build the detections and notifications you need on the SIEM copy. Do not rely on Ziggy's alerts alone.
- Keep fail-closed on. The code default is off; only the shipped templates turn it on. Do not set it to false on a regulated deployment.
- Choose a log level for production. The code default is a verbose debug level and no template sets it. Set the level you want, and keep request and response body logging off.
- Remember the limits of the trail. The audit table is an ordinary database table, and the application's own database role deletes expired rows and changes the three sealed columns when erasing a person; nothing at the database makes it append-only. Detecting a change by someone with write access to the database depends on the SIEM copy. In memory, the SIEM buffer holds at most 10,000 entries and drops the oldest first when full, with the count printed to the console, so a long outage or a crash can lose copies. The database row remains.
- Forward the administration portal's audit trail to your SIEM as well. It is a separate application with its own copy, and that copy stops after a failed start until it is restarted and checked.

Evidence: Security overview (audit logging and SIEM, retention); Configuration guide (logging and SIEM settings); Assurance report appendix (audit trail).

CC7.3 Evaluating security events

Ziggy provides

- Severity-graded alerts (low to critical) with a status (pending, sent, acknowledged), who acknowledged and when, and optional notes.
- An audit log that can be filtered by event type, organisation, person and service account, readable by installation administrators only, and a decision log for authorisation outcomes.
- Records of break-glass use with a review deadline, an escalation if a record is not reviewed, and a different reviewer from the invoker.

Customer must

- Define what counts as a security event, who triages it, how fast, and how it is escalated. Ziggy supplies the data; the evaluation procedure and the people are yours.
- Review the audit log and the SIEM on a schedule. Because alerts do not cover every recorded act, schedule a review of the recorded security-setting, access, credential and erasure acts.
- Review break-glass records within the review period.

Evidence: Security overview (audit logging and SIEM); Configuration guide (alerting settings).

CC7.4 Responding to security incidents

Ziggy provides

- The means to contain: deactivate or delete an account, revoke trusted devices, switch off or reset a person's second factor (which ends every login of the account), reset a lockout, remove a role or grant, rotate encryption keys, and change secrets.
- The evidence to investigate: the audit trail, the SIEM copy, the decision log, and request lines with timestamps (request lines do not name the user; the audit trail does).

- A route for security issues in Ziggy itself: report to info@ziggyplatform.com, or Freddy May directly at freddy.may@ziggyplatform.com. The vulnerability disclosure and patch policy is in Secure development.

Customer must

- Hold the incident response plan: roles, communication, legal and regulatory notification, evidence handling and post-incident review. Ziggy does not notify your customers or regulators.
- Know that deactivation alone does not end a person's sessions, and plan containment steps accordingly (for example delete the account or reset its credentials).
- If you use the Reports feature, that feature's own session is separate: signing out of Ziggy does not end it. Shorten its session lifetime in its own configuration and end it directly in an incident.
- Test your plan, and use the audit and SIEM data in the exercise.

Evidence: Security overview (audit logging and SIEM, sessions and tokens); Secure development (disclosure and patch policy); README (how to report a security issue).

CC7.5 Recovery from security incidents

Ziggy provides

- Backup and restore of the database through the application and the administration portal. A restore, and the start and end of one, is recorded, and the retention changes that delete audit data are written before they act.
- Key replacement and re-sealing with verification after each write, and an audit step that says whether an old key may be removed.
- Credential recovery: a password reset or a second-factor recovery ends every other login of the account. A recovery for an administrator is held for approval by another installation administrator, where one exists.

Customer must

- Maintain tested restore procedures and the retention of backups (see A1.2 and A1.3).
- After an incident, rotate every secret that may have been exposed: the signing secret, the encryption keys, database and cache passwords, and provider credentials.
- Record what was restored and why, in your own incident log.

Evidence: Security overview (encryption, credentials and secrets, audit logging and SIEM); Configuration guide (key rotation and restore).

Security: change management (CC8)

CC8.1 Authorising, designing, developing, testing and implementing changes

Ziggy provides (to the customer's own work in Ziggy)

- Changes to governed items (connections, queries, Pipelines, quality checks and others) are made on branches and reach the main version through changesets. Governance workflows with approval steps are configurable per installation, and a relaxation of separation of duties is recorded and alerted.
- Merging, promoting to an environment and rolling back are separate rights that sit outside ordinary authoring, are installation-wide and need a second factor from their holders.
- Version history, a difference view of what changed, and a record of promotion attempts.
- Limits to know. A few paths place a change on the main version without carrying the review state of an open changeset, for example moving a data-quality check into a folder, and an operator's drift-repair command run on the server. Do not treat changeset review as absolute on every path. Likewise, a run of a Pipeline version that no merge has approved does not on every path run with the access of the person who tries it. Ziggy does not guarantee that every approval of a high-risk changeset asks for a fresh sign-in either. Both limits are in "Accepted for later".

Ziggy provides (to the product's own development)

- A review gate on every change, deeper for sign-in, permissions, tenancy and secrets; automatic checks that block unsafe changes before they are saved; tests that are written to fail first; secret scanning; and the policy-coverage start-up check that refuses a route with no authorisation entry. See Secure development and the Assurance report for the test tiers and counts.

Customer must

- Control change to your deployment: the infrastructure code, the images you build and deploy, the environment files, and the application version you run. Ziggy's pack describes one commit. When you upgrade, take the pack for that release.
- Review the Terraform and the compose files before you apply them. The reference build has no automated check for them.
- Separate authoring, merging and promoting by assigning the roles to different people, and set governance approval steps for the items you care about.
- Do not deploy, switch versions or restart containers through the portal while an encryption-key replacement is under way; the portal does not stop you.
- Pin the machine image per environment. The reference Terraform looks up the newest Ubuntu 22.04 image on the day an instance is created.

Evidence: Secure development; Assurance report (Part A and appendix); Security overview (roles and permissions, "Accepted for later"); Configuration guide ("Before go-live").

Availability (A1)

A1.1 Capacity and demand

Ziggy provides

- Health thresholds for instance, database and worker, which raise alerts; a liveness and a readiness endpoint; Prometheus-format metrics; per-route and global rate limits that bound demand from a single address; bounded request body size.
- Scalable layout: additional servers and remote workers can be added, with leader election for scheduled work, and queues that run on Redis.

- Retention sweeps that keep operational tables from growing without bound.

Customer must

- Size the hosts, database, cache and storage for your load; set and monitor capacity thresholds; and plan growth. The reference deployment's instances sit in one subnet and one availability zone.
- Monitor disk, memory and database connections at the infrastructure layer.

Evidence: Security overview (retention, audit logging and SIEM); Configuration guide (reference-deployment hardening).

A1.2 Environmental protections, backup and recovery infrastructure

Ziggy provides

- A backup script that dumps the database, compresses it and copies it to an object-store bucket, with an optional retention in days, and an application backup and restore path with recorded events. Registered sensitive fields are ciphertext in a dump; the encryption keys are not part of a dump.
- Encrypted root volumes for the reference instances, and an encrypted, locked remote state for Terraform.
- Graceful shutdown that flushes the SIEM last.

Customer must

- Take backups of the encryption keys file separately and keep them apart from the database backups. A database backup without the keys cannot be read, and the keys are not in it.
- Control the backup bucket (encryption with your key, public-access blocking, versioning, access logging, object lock or lifecycle rules). The reference Terraform does not manage it, the instance role can delete its own backups, and nothing locks them.
- Set a backup frequency and retention period. If a retention number is not passed to the backup script, nothing is deleted.
- Provide environmental protections (power, cooling, fire, flood) and multi-zone design, or rely on your cloud provider's attestations. The reference deployment places all instances in one availability zone, and disk snapshots are not taken.
- Back up the administration portal's own state and settings.

Evidence: Configuration guide (backup, reference-deployment hardening); Security overview (encryption at rest and in transit).

A1.3 Testing recovery plan procedures

Ziggy provides the restore function, and the audit records around it. It does not schedule or run recovery tests.

Customer must test restores of the database and the encryption keys on a schedule, in an environment of your choice, measure recovery time and recovery point against your targets, and keep the results. Test a key replacement and a worker rebuild as well.

Evidence: Configuration guide ("Before go-live").

Confidentiality (C1)

C1.1 Identifying and maintaining confidential information

Ziggy provides

- Field-level encryption in the application with AES-256-GCM, before data reaches the database. 41 registered fields in 35 data models are sealed, covering email addresses, network addresses and browser identifiers in the audit trail and sessions, connection credentials, stored secret values, OAuth tokens, execution data and errors, worker API keys and dead-letter payloads. Each value names the key that sealed it, and non-deterministic values are bound to their own column so a value copied between columns does not open. Four columns use a deterministic mode so they can be looked up by equality.
- Secrets that Ziggy stores for connections and settings (mail password, OAuth client secrets, AI provider keys) are sealed and never returned in clear. The authenticator secret has its own sealing.
- Classification of data and masking policies, so that classified columns can require a clearance and be masked.
- Access to data is subject to grants and organisation scoping (see CC6.1 and CC6.3).
- Safe key rotation: a new key version, a re-seal with read-back verification, and an audit that counts values by key and says whether an old key may be removed.

Customer must

- Decide what is confidential in your use, classify it in Ziggy, and set who may see it.
- Protect the encryption keys file. It sits on the application host in plain form, with no key management service. A compromise of the host yields both the ciphertext and the keys. Restrict host access, keep the keys file out of backups and images, and consider wrapping the keys with a managed key service. The `hipaa` key type is reserved and protects no data today.
- Set a rotation policy. Rotation is run by an operator and nothing reminds you; a policy such as yearly, and after any suspected exposure, is yours to set. Keep a recovery copy of the keys.
- Know that plain values written before field encryption existed are read as they stand, and run the key audit before relying on a column.
- Protect data in places Ziggy does not seal: database dumps in your bucket, the SIEM copy (which carries email, network address and browser in clear, protected by your CloudWatch encryption), and the data your pipelines move.

Evidence: Security overview (encryption at rest and in transit, credentials and secrets); Configuration guide (keys file, rotation); Assurance report appendix (encryption).

C1.2 Disposing of confidential information

Ziggy provides

- Configurable retention windows for the audit trail (365 days for ordinary events and 730 for security events by default), the access-decision log, soft-deleted rows, tokens, telemetry, notifications and AI logs, applied by a nightly sweep. A change to a window and every sweep is recorded before it acts.
- A manual hard flush of soft-deleted rows, which spares younger rows and a list of protected tables.

- Permanent deletion of a person's account (see P4 for the limits of audit-trail erasure).

Customer must

- Set the retention windows to your policy. An administrator can set an audit window as short as one day through the application, and the environment settings do not refuse zero or a negative number. There is no floor for security events, so control who holds installation administration and watch the retention-change records.
- Dispose of backups, dump files, kept key-file copies, SIEM log groups and server disks under your own policy. Ziggy's retention does not reach them.
- Set the SIEM retention yourself; the shipped templates set 30 days.

Evidence: Security overview (retention); Configuration guide (retention settings).

Privacy (P)

You decide why and how personal information is processed, so you are the controller; Ziggy provides tools that support your obligations. The notices, the lawful basis and the privacy programme are yours. The entries below say what the software supports.

P1 Notice

Ziggy provides no privacy notice content and no consent text. It stores the account data it needs to operate (name, email address, sign-in method, session details).

Customer must publish your privacy notice to the people who use Ziggy and to anyone whose data is processed in it, and describe retention, disclosure and rights.

Evidence: README (shared-responsibility model).

P2 Choice and consent

Ziggy provides no consent capture. Optional features are switched on and off by administrators (for example sign-in providers, emailed codes, AI features).

Customer must obtain and record any consent your lawful basis needs, and switch off features you have no basis for.

Evidence: Configuration guide (feature and provider settings).

P3 Collection

Ziggy provides

- Account data limited to what sign-in and audit need. The email address, the client network address and the browser identifier in accounts and the audit trail are sealed in the database.
- Masking of personal fields in log lines and in the SIEM copy of audit metadata, and body logging that is off by default.
- Hashing, not storage, of credentials and codes where one-way storage is enough.

Customer must

- Limit what personal data goes into data sources, Pipelines and free-text fields, and classify it.
- Know that the SIEM copy of an audit record carries email, network address and browser in clear; protect that log group and set its retention.

Evidence: Security overview (encryption, audit logging and SIEM); Assurance report appendix (logging and encryption).

P4 Use, retention and disposal

Ziggy provides

- Retention windows (see C1.2). An administrator can set an audit window as short as one day, so control who may change them.
- Permanent deletion of a person's account, refused until the work they own is handed over, with their audit entries anonymised (email, network address and browser replaced).
- The limit of that anonymisation: it is applied to audit entries in the organisation the deleting administrator is working in. Entries in other organisations, and entries with no organisation, keep the person's details, and the SIEM copy is not changed.

Customer must

- Define purposes and retention for each category of personal data, and set Ziggy's windows to match.
- State the limit of erasure in your privacy terms, and handle the remainder (other organisations' entries, the SIEM copy and backups) by your own process.
- Erase personal data in backups and the SIEM under your own retention.

Evidence: Security overview (retention, "Accepted for later"); Configuration guide (retention settings).

P5 Access to personal information

Ziggy provides

- A person can see and update their own account name and see and end their own sessions and trusted devices.
- An installation administrator can see a person's account details, organisations, roles and effective access with where each right comes from, and the work the person owns or is involved in.
- Audit entries that name a person can be found by filtering the audit log.

Customer must

- Run the request process: verify the requester, collect the data from Ziggy and from every other system, apply exemptions, and respond in time. Ziggy has no data-subject request workflow.

Evidence: Security overview (roles and permissions, audit logging and SIEM).

P6 Disclosure and notification

Ziggy provides

- Outbound connections go to destinations configured in your installation (data sources, mail server, Google or Microsoft sign-in, the SIEM, an AI provider if you enable one, and the administration portal's Auth0 and cloud accounts). Outbound requests from pipelines are checked against internal addresses.
- Records that help you detect and scope a disclosure: the audit trail, the SIEM copy and the decision log.

Customer must

- Maintain a register of the third parties your Ziggy sends data to, agree terms with them, and keep it current as you add connections.
- Decide what an AI provider sees before enabling AI features, and pass those terms on.
- Detect, assess and notify breaches to regulators and individuals as your law requires. Ziggy sends no such notification.
- Record disclosures that your law requires you to account for.

Evidence: Security overview (audit logging and SIEM, "How this was checked"); README.

P7 Quality

Ziggy provides

- People can correct their own name.
- Data-quality checks as a product feature, for the data you load into it.

Customer must keep the personal data accurate in your sources and in Ziggy, and act on correction requests.

Evidence: Security overview (roles and permissions).

P8 Monitoring and enforcement

Ziggy provides

- Audit and SIEM data, access reports and the decision log, which feed privacy monitoring.
- Monitoring of who holds rights to view data, through the effective-access view and, for installation-wide access, the access-review export (with the organisation-scoped limit noted at the top).

Customer must

- Run the privacy programme: assign accountability, handle complaints, review compliance, and track incidents and remediation. Give your assessor the pack together with your own policies.

Evidence: README; Assurance report; Security overview (audit logging and SIEM).

What this document does not say

- It does not say that the SOC 2 criteria are met. That is an assessment of your whole organisation by your assessor.
- It does not describe controls the audit did not find holding. Where one has a limit, the limit is written beside it, and the security overview's "Accepted for later" section lists the open items with how each could be exploited, why that is unlikely, and when it will be closed.
- It does not replace independent testing (see Security testing).

Secure development

This document describes how Ziggy is built and changed: who reviews a change before it joins the main line, which tests and automatic checks stand between a change and a release, how secrets are kept out of the source, and how third-party packages are handled. It describes the rules the source tree itself carries, so a customer's engineer can find each one in the copy they hold. Where a rule has a limit, the limit is stated next to it.

Companion documents: the security overview ("How this was checked" and "Accepted for later") describes the audit of the finished product; the security assurance report lists the security guarantees and the tests and checks behind them; the secure configuration guide covers the start-up settings named below.

1. How a change is made

- Every change is made on a branch and reaches the main line only through the review gate in section 2. The rules for how changes are planned, built and recorded are written in the repository itself (its guidance files and recorded decisions), so they travel with the source.
- Ziggy's code is written with AI coding assistants working under those rules. The reviewers in section 2 are also AI models. A person, the maintainer, decides what is fixed and what is accepted; the review is not a review by an independent human or an independent firm.
- Every decision that shapes a security-relevant behaviour is written down with its reason when it is made, and the tests and checks below are written from that recorded intent rather than from what the code happens to do. A disagreement between the code and the recorded intent is treated as a defect to report, not as a test to write.

2. The review gate

No change to code merges to the main line without one review at arm's length:

- The reviewers are fresh AI sessions, never the session that wrote the change. They are given the change and the recorded intent for it (the written decisions and the numbered guarantees it touches) as the specification.
- Only text and comments are exempt.
- **Depth follows risk.** Changes to sign-in, permissions, organisation separation, credentials and secrets, approvals and governance, and what an AI model is shown get the deep procedure: reviews at the highest effort level, a review of each stage of the work as it lands, a review of the tests written to lock the behaviour in, and a further review of every round of fixes. Everything else gets one review at medium effort when the work is finished.
- **The rule for findings.** The gate is "every finding fixed or accepted". A finding is fixed when it is reproduced and affects what ships. A finding that is true but latent is recorded as accepted with its reason. In the areas above, a latent finding is never accepted silently: it is raised to the maintainer at once.
- Security fixes found by the audit described in the security overview are made the same way, and are then re-reviewed and audited again before release. The record of what was accepted rather than fixed is the "Accepted for later" section of the security overview.

Limits: the review gate is a rule the maintainers follow and record. Nothing in the source repository enforces it technically. The automatic checks in section 4 are the technical backstop.

3. Tests

Three tiers apply to a change, each at its own point:

Tier	When it runs	What it covers
Unit tests	Written with the code and kept green as work proceeds	One unit's behaviour; collaborators that matter run for real rather than as lists of expected calls
API end-to-end tests	Guaranteed before a change is signed off; a change to a route, a request or response shape is not merged until the end-to-end test for that contract exists and passes	The real server and a real database: the happy path, who may and may not call, and the main error codes
Browser end-to-end tests	Only when asked for	Screens and user flows in a real browser

The unit and API tiers also include suites for the shared packages, the admin portal and the agent host, and a separate suite covers the Python model runner. The suite counts at the described commit, counted from the source tree:

Suite	Test files	About this many test cases
Server unit (source and test folders)	1,686	19,159
Server API	602	6,825
Web app	276	3,001
Admin portal	42	483
Browser tests	64	151
Shared packages	19	241
Python model runner	20	151
Agent host	4	36
Total	**2,713**	**30,047**

The case counts are "about" because a case that runs once per item of a list is counted once. The security assurance report says how many of these tests name a security guarantee.

How the tests are built to be worth trusting:

- **Intent first.** Expected behaviour comes from recorded decisions, the numbered guarantees and the domain model, never from reading the implementation.

- **A new check is seen failing first.** Before a new or changed test or automatic check is trusted, the thing it protects is deliberately broken, the check is seen to fail, the break is undone and the check is seen to pass.
- **Every guarantee is tagged.** Each numbered security guarantee has a stable id, and the tests that guard it carry that id in their title. An automatic check fails if a guarantee has no test naming it or a test names a guarantee that does not exist.
- **Response shape is checked everywhere.** In the API tier every response is checked against the contract its own route declares, strictly: a field the contract does not declare fails the test. A separate check compares the status code each route declares with the one it returns.
- **Every route is reached.** After a full run of the API tier, an automatic check fails on any route no test calls and no recorded reason excuses. It refuses to report on a result from a different commit.
- **Weak test patterns are refused.** Automatic checks reject unit tests that only mirror the list of calls the code makes, hand-built database doubles that nothing checks, and fixed sleeps in end-to-end tests.

Limits: the API and browser tiers are run by the team on their own machines before sign-off. There is no automatic cloud run of them, so a defect that exists only in a clean checkout (for example a build file that exists on one machine only) is not caught by them. The automatic cloud runs described in section 4 cover the static checks only.

4. Automatic checks that block an unsafe change

Two groups of checks run without a test naming them.

Static checks. About three dozen checks read the source (and, for some, the type checker) and fail on a property of the code that a test cannot show. They live in `app/server/scripts` and are the `check:` entries of `app/server/package.json`. In plain words, they catch:

Kind	What it refuses
Access and ownership	A run that builds its owner's access by hand instead of through the one place that lends it; a stored secret read anywhere but the one place that checks where it may be sent; data shown to a reader by a path that skips the one place that decides what a reader may see; something that acts for a person without asking whether that person still has standing; a file read, replaced or removed without asking for the right to it
Organisation separation	A write whose organisation is not stated; a read by identifier on a model that carries an organisation and does not name it; a direct read of the version history that has no recorded reason
Sign-in and credentials	A single-use credential or check-then-change step that is not claimed in one statement; a fresh-sign-in check that is not the shared one
Data handling	A value written into the text of a database statement; an unbounded read over a table that grows with use; a loop that makes one database round trip per item without saying why; a schedule or a person-set run limit that skips the server's ceiling
Contract and structure	A route whose declared status differs from the one it returns; an error raised outside the shared error layer; a change that makes core code depend on application code; a class that takes too many collaborators; a function over the complexity limit; a double type cast that defeats the type checker; a database foreign key without an index
Consistency	An AI model identifier or price outside the one file allowed to name one; a deletion rule that differs from the database schema; a field of a versioned item written without committing; documentation naming a path that no longer exists; unformatted source

A check that has a list of accepted exceptions holds the list to the code: an exception whose reason no longer holds fails, and a baseline list may only shrink.

These checks run in two places:

- **On every push**, from the repository's pre-push hook (`.githooks/pre-push`): 33 server checks plus a plan-format check and a documentation check, held and reported together. The push is refused if any fails.
- **In the cloud**, from `.github/workflows/static-checks.yml` and `.github/workflows/delete-policy-check.yml`, on pull requests that touch the server or documentation and on every push to the main line. This is the backstop for a push that skipped the hook or was made from another machine, and it also confirms the packages install from a clean checkout.

Limits: the pre-push hook can be skipped by the person pushing, with an environment variable or a command-line option, and the cloud run does not include every check the hook runs. A few checks need a prior API test run and run as part of that tier instead.

Start-up checks. The server refuses to start, or reports loudly, when its own wiring is unsafe. Two of them guard the permission system and are part of the rules a developer meets:

- **Policy coverage.** Every route of the server must have a permission entry or be declared anonymous. A route with no entry is refused at request time and also stops the server starting. A route declared anonymous must either carry its own authentication guard or be listed with what stands in for authentication. Setting `ALLOW_POLICY_COVERAGE_GAP=true` turns the start-up refusal into a warning for an emergency start; the uncovered routes are still refused at request time. It is off by default and must not be left on in production.
- **Capability coverage.** Every route of the product (not the core settings and sign-in routes) that carries an access right must resolve to a capability feature, so a per-feature grant can authorise it. Setting `ALLOW_CAPABILITY_COVERAGE_GAP=true` turns the start-up refusal into a warning for an emergency start; such a route is then checked at installation scope, so a restricted user is still refused. It is off by default and must not be left on in production.

The secure configuration guide explains both settings. The security assurance report lists the other start-up checks. A database-deletion rule is also checked at start-up against the schema.

5. Keeping secrets out of the source

- **Secret scanning on every commit.** The repository's pre-commit hook (`.githooks/pre-commit`) runs the open-source Gitleaks scanner over the staged changes, with its full built-in rule set plus an allow-list in `.gitleaks.toml` for placeholders and invented fixture values. A match blocks the commit and is shown with the value redacted. If the scanner is not installed, the commit is refused.
- **Encryption key files** are refused by name at commit time; only a test fixture with non-production keys may be tracked. Real key files and local environment files are also excluded by the ignore rules.
- **Data is not code.** A second hook refuses to add database dumps, log files, real environment files, or anything under a backups or uploads folder.
- **Placeholders and injection.** Secrets are supplied to a running installation at provisioning time, not from the source. The environment templates carry substitution markers, not values, and the test environment files keep their secrets empty.
- **Secrets at runtime** (sealed storage, release only to the destination a secret was bound to, masking in logs and in what an AI model is shown) are described in the security overview and checked by the static checks in section 4.

Limits: the hooks can be skipped by the person committing, for an emergency, and secret scanning has no cloud run behind it. The hook's guard on key-file names does not match an older naming of a kept copy of a key file; the ignore rules and the scanner still cover it, and the follow-up is recorded with the audit's open items. A customer who builds from source and commits their own changes should install the scanner and the hooks (`git config core.hooksPath .githooks`; the root package install sets this up).

6. Dependencies

- **Reproducible installs.** The npm lock file is committed, and the cloud checks install from it with a clean, lock-file-exact install. A build is therefore made from the versions the lock file names.
- **Pinned fixes for known advisories.** The root package file carries 34 overrides that force patched versions of transitive packages (parsers, HTTP and form libraries, build tools) over the versions their parents ask for.
- **Advisory checks on demand.** The server and web app workspaces each have an advisory-audit command that checks the installed tree against the public advisory database. It is run by a developer; it is not a gate and no scheduled job runs it.
- **Not automated.** There is no automatic dependency-update service configured in the repository. Updates to packages are ordinary changes and go through the review gate in section 2.
- **Scope of the audit.** The audit described in the security overview covers Ziggy's own source. Third-party packages are outside it; they are handled as above and are not penetration-tested by it.

A customer running Ziggy from source should run the advisory check against the commit they deploy before each release and on a schedule of their own.

7. Vulnerability disclosure and patch policy

Severity is graded by the CVSS 3.1 base score.

- **Report.** By email to info@ziggyplatform.com, or Freddy May directly at freddy.may@ziggyplatform.com. Please do not disclose the issue publicly until a fix is available.
- **Acknowledge.** Within two working days.
- **Assess and grade.** Within five working days; we tell the reporter the grade and the plan.
- **Fix.** Critical (9.0 and above) within 7 days; high (7.0–8.9) within 30 days; medium (4.0–6.9) within 90 days; low (below 4.0) in the next planned release.
- **Deliver.** A tagged release in the source repository, and an email to every customer naming the commit to move to, since customers run Ziggy from the source.
- **Publish.** A security notice once the fix is available, crediting the reporter if they wish.

8. What this document does not claim

- It does not claim an independent code review, an independent audit or an independent penetration test. The security audit of the finished product is AI-assisted, as the security overview states; how Ziggy is security tested is in the security testing document.
- It does not claim that the cloud runs cover the API or browser tests, that the review gate is technically enforced, or that dependency advisories are watched automatically.
- It does not list the weaknesses the vendor has accepted for later; those are in "Accepted for later" in the security overview.

Security assurance report

This report is for your security team. It says how Ziggy's security is checked and how often. It does not ask you to take adjectives on trust: every figure in it is a count you can reproduce from the source you hold, or a count from the audit record kept for this release. It has two parts.

- **Part A** says what is examined and how, area by area.
- **The appendix** lists every security guarantee the code enforces, with the number of test files that name it, so you can find those tests yourself.

Two statements first, so nothing below is read as more than it is.

- **The audit is AI-assisted.** The full audit described in Part A was carried out by an AI model (Claude Opus 5.5), reading the source against a fixed checklist. It is not a human audit and not an independent third-party audit. How Ziggy is security tested, including that it has not yet had an independent penetration test, is in the security testing document.
- **It covers Ziggy's own source at one commit.** Third-party libraries are outside it (the secure development document says how dependencies are handled). Your own infrastructure, network and configuration are yours; the secure configuration guide says what to set.

Part A. What is examined and how

A.1 Four layers of checking

Security in Ziggy is checked in four layers. Each catches a different kind of mistake, and no layer is relied on alone.

1. **A full audit of every entry point and setting.** Nothing is sampled (section A.2).
2. **Automatic checks that block an unsafe change before it is saved** (section A.3). They read the code itself, not its behaviour, and refuse a push that breaks a rule.
3. **Start-up checks that refuse an unsafe start** (section A.4). A server that would run with an unsafe setting or a missing control does not come up.
4. **Tests, and review of every change** (sections A.5 and A.6). Tests are written to fail first. Every new automatic check is first seen failing on a deliberately planted break. Every code change to security-relevant behaviour is reviewed by a fresh AI review session, never the one that wrote it.

A.2 The full audit

A script lists, from the source, every item of the kinds below. The list is the audit's scope: an item that is on it is judged, and the audit refuses to finish while any is not. For this release:

Kind of item	Count	What it is
Server routes	1,154	every HTTP handler of the application server
Admin portal routes	98	every HTTP handler of the admin portal
WebSocket gateways	7	every socket the server opens
Command-line commands	7	every operator command
Encrypted fields	41	every database column stored encrypted
Secret-bearing fields	74	every request or response field named like a credential
Settings	140	every variable the server reads from its environment
Outbound connections	52	every place the server connects to something outside itself
Shared controls	8	the controls many routes rest on (the permission engine, token check, tenancy scope, fresh sign-in check, audit writer, logging, global wiring, admin portal guards)
Total	**1,581**	

How each item is judged. Each item is read end to end, from the route to its permission check, service and data access, by an AI model, against a fixed checklist for its kind. Every answer is pass, fail or not applicable, and a pass or fail cites the lines of code it rests on. The checklist asks 27 questions of each route, 7 of each socket, 9 of each command, 3 of each encrypted field, 2 of each secret-bearing field, 2 of each setting, 6 of each outbound connection and 4 of each shared control: in all, about 34,800 item-and-question pairs for this release.

Nothing is skipped. A coverage gate counts the items judged against the items listed and refuses to finish while any is missing, incomplete or unsettled. For this release it closed with 1,581 of 1,581 judged and none missing.

A failure is not taken on trust. Every failing answer is traced through the code a second time, by a separate pass, before it counts. Only a failure that is confirmed counts as a finding.

What the checklist asks. In plain words, by area:

Area	What is protected	What the audit asks of it
Sign-in	Who is let in, and on what proof	That every route that grants a session passes through the one sign-in gate; that a new password ends the old sessions; that an address is proved by the mailbox before it earns an invitation, role or organisation; that renewing a session refuses everything a fresh sign-in would refuse
Second factor	Proof beyond a password	That a second-factor or recovery code is accepted only for a method the account has turned on and confirmed; that wrong tries are counted per account, not restarted by asking for a new code; that the old factor holds until its replacement is confirmed
Sessions and tokens	How long a login lasts and what stands in for it	That anything standing in for a session (a link token, a signed address, a proof of fresh sign-in) ends with the session, works only for what it was issued for, and belongs to the session that made it

Area	What is protected	What the audit asks of it
Permissions and the two admin tiers	What each person may do	That every route states the permission it needs and refuses what the caller lacks; that the installation tier and the organisation tier are told apart by the scope of the `Admin` grant, never by a role name; that an organisation administrator cannot hand out installation-wide authority
Organisation separation	One organisation never sees another's data	That every id a route takes and every row it touches is held to the caller's organisation, and a foreign id answers as an unknown one; that a value in the body cannot hide a different value in the query
Credentials and secrets	Passwords, tokens, keys, connection strings	That no secret leaves in a response, an error, a log line or the audit copy; that a stored secret goes only to the host it was saved for, including after a redirect; that the field table that says how each credential field behaves is true
Encryption	Data at rest and in transit	That every write path to an encrypted column encrypts it and no read path shows it to someone who may not see it; that key rotation covers every sealed column; that outbound connections verify the other side unless an opt-out is recorded
The audit trail	The record of who did what	That an act that changes access, sign-in, security settings, keys, retention or data wholesale writes a record through the one writer, with its actor; that a refused sign-in or fresh-proof check is recorded; that secrets never reach the record or its copy
Outbound connections	Where the server may reach	That an address from a user or a stored row cannot send the server to an internal address, including after a redirect; that a credential cannot follow a redirect to another host
Work run for a user	Code, queries, schedules and limits a user sets	That user code cannot read the server's environment or reach the network except through the server's checked calls; that a limit a user sets cannot lift a bound the server applies; that the access a run uses is judged again when it acts
Configuration	The settings that decide whether the above hold	That a wrong value which would weaken any area above either has a safe default or makes a production process refuse to start, and that the shipped templates carry the safe value

Standing questions. A specific concern of yours can be added to the checklist as a standing question. Every later audit then answers it for every item it concerns. This is something the vendor does on request: info@ziggyplatform.com, or Freddy May directly at freddy.may@ziggyplatform.com.

Its limits. The audit reads source, not a running deployment. It is not a penetration test. It does not audit third-party libraries, and it does not audit your own infrastructure. The section "What this report does not claim" below lists specific limits.

A.3 Automatic checks that block an unsafe change

Ziggy carries 40 automatic checks (scripts that read the source and fail on a rule broken). 35 of them run on every push from a developer's machine and abort the push if any fails; most of them run again on the shared repository as a backstop. Every commit is also scanned for secrets before it is saved, and a commit that carries an encryption key file or a data dump is refused.

The checks that guard security fall into these kinds. The appendix names each one and says what it refuses.

Kind	What it catches
Who may do what	A route with no permission entry; an act that borrows another person's access without going through the one place that records it; an act that outlives the standing of the person it acts for; a check followed by a change that is not claimed in one step
Organisation separation	A read the tenancy backstop would miss that does not name its organisation itself; a write with no organisation set; a direct read of shared history without a recorded reason; a stored file read or replaced without asking the right to it
Secrets	A stored secret sent anywhere other than the destination it was bound to; a credential field that does not behave as the field table says
Fresh sign-in	An act that should ask a fresh sign-in and does not; the two time windows drifting apart between the application, the command line and the admin portal
Input and queries	A value turned into SQL text; a query built by joining strings
Limits	A number a person sets that bounds a run's time, rate or size and is not passed through the server's limit rule; a schedule that does not keep the five-field rule; a list read with no bound
Data integrity	A delete rule missing or illegal against the live schema; a canonical field written without the commit that records it
AI models	A model version written outside the registry; an AI tool loop with no bound
Coverage	A system guarantee with no test that names it; a route with no test that exercises it
Hygiene	Format, module boundaries, error shapes, comments, plan records

Each such check is itself held to account: it is first made to fail on a planted break, and only then trusted (section A.5).

A.4 Start-up checks that refuse an unsafe start

A process that starts unsafely would hide the fault until the day it mattered, so the server refuses instead. A production process refuses to start on any of the following (the appendix lists each):

- a missing, short or placeholder signing secret;
- a production setting known to be unsafe: a plain-text directory connection, sandbox mode allowed, plain-text mail allowed, a weak shared callback secret, fault injection switched on, or the number of trusted proxies left unstated;
- a connection to a database or cache on another server that is not verified, unless the operator opts out explicitly (the opt-out is logged at every start);
- a route that has no permission entry and is not on the short, reasoned list of anonymous routes;
- a controller that owns a feature but declares no feature key;
- an encrypted column whose model is not routed through the encryption layer;
- a delete rule that is missing or illegal against the live schema;
- a copy of the audit trail set to fail closed with no usable destination, a mistyped value for that switch, or a retention period the destination does not accept.

Two of these have an override, off by default, for emergencies: the permission and feature-key coverage checks can be turned into warnings by an operator setting. The secure configuration guide says what the settings are, what they let through, and that they are never left on in production. Where a start-up check only warns by default (for example a divergence between the environment file and the deployed branch), the guide says which setting makes it refuse.

A.5 Tests, written to fail first

The tests that serve security are part of a much larger body of tests. The counts below were taken from the repository at this commit.

Kind of test	Test files	Test cases (about)
Server unit	1,666	18,871
Server unit (test folder)	20	288
Server API (end to end, against a real database)	602	6,825
Web app	276	3,001
Admin portal	42	483
Browser tests	64	151
Shared packages	19	241
Python model runner	20	151
Agent host	4	36
Total	**2,713**	**30,047**

"About" because a case that runs once per item of a list is counted once. The appendix says how many of these test files name a security guarantee. Not every test is a guarantee: a guarantee is a rule the product promises and that has tests naming it. The other tests check the details, edge cases, screens and past defects that serve those rules, and the appendix is the named core, not the whole of the testing.

Three working rules apply to every security change:

- **A test that fails first.** A fix to a security weakness begins with a test that reproduces it and is seen to fail, and passes only after the fix.
- **A guard against the bypass.** When the same kind of weakness has appeared more than once, it is closed by one central rule that every site calls, plus an automatic check that fails when a new site bypasses it. The check is broken on purpose, seen failing, restored, and seen passing before it is trusted.
- **The audit's question is asked of every item of the kind.** After a fix, the question that caught it is re-asked of every item of the same kind, not only the one that failed.

A.6 Review of every security change

No code change is merged to the main branch without a separate review by a reviewer who did not write it. The reviewer is given the change and the recorded intent (the decisions and guarantees it must keep) as the specification. These reviews are carried out by AI reviewers in fresh sessions, not by the session that wrote the code. Depth follows risk. A change to sign-in, permissions, tenancy, credentials and secrets, governance (approvals and change control), or what an AI model is shown gets a deep review, in slices, and a second review of the tests written for it. Every finding is fixed or accepted in writing before merge. Comments and documents are exempt.

A.7 What this report does not claim

So that nothing above is read as a guarantee it is not, these limits apply and are stated in the same breath wherever they bear:

- **Reports of access are incomplete for organisation-scoped access.** The access-review export and the separation-of-duties report are built from the installation-level reading of each person's access. They leave out access that is scoped to one organisation. They are not a complete list of everyone's access, and a periodic access review must not rely on them alone.
- **Alerts reach only some event types.** Alerts are raised for event types that reach the alert listener. Some recorded acts do not reach it, so an administrator cannot be alerted on every recorded act, even though the settings screen lets them switch alerts on for more.
- **Two guarantees are not stated as held.** That SQL which no merge has approved always runs with the access of the person trying it is not claimed; and that every approval of a high-risk changeset asks for a fresh sign-in is not claimed. Both are accepted for later; see "Accepted for later" in the security overview. Wherever a guarantee is contradicted by an open item or by this release's audit, it is left out of the appendix or listed with its limit in the same sentence.
- **The deployment is the customer's.** Settings, network, backups and key storage are set by the operator. The audit confirms what the server refuses and what its templates ship, not what a given installation set.

Appendix: every security guarantee

The full extent of the testing

Ziggy is tested at 2,713 test files holding about 30,047 test cases, counted from the source at this commit. A case that runs once for each item of a list is counted once, which is why the case figures are "about".

Kind of test	Test files	Test cases (about)
Server unit tests (one service or rule at a time)	1,686	19,159
Server API tests (a booted server on a real database, over HTTP)	602	6,825
Web app	276	3,001
Admin portal	42	483
Browser tests (the real screens)	64	151
Shared packages	19	241
Python model runner	20	151
Agent host	4	36
Whole platform	**2,713**	**30,047**

Of these, 704 server test files, holding about 10,370 test cases between them, name one of the security guarantees listed in this appendix. The listed guarantees are the named core of the testing, not all of it. A guarantee here is a rule the product promises, written down in the server's testing runbook under an id and guarded by tests that name that id. The other tests check the details, edge cases, screens and past defects that serve those rules: they are not guarantees in this sense, so they are not listed, but they run on every change.

Summary of this appendix

Area	Guarantees listed	Test files behind them	Other tests in the code that serves the area
A. Organisation separation	20	150	not given (code spread too widely)
B. Sign-in, second factor, sessions and tokens	29	58	plus about 890 other test cases
C. Permissions, roles and the two admin tiers	28	130	plus about 450 other test cases
D. Approvals and change control	41	148	not given (code spread too widely)
E. Moving changes between installations	31	42	plus about 130 other test cases
F. Credentials and secrets	11	64	not given (code spread too widely)
G. Encryption and keys	5	15	plus about 90 other test cases
H. Audit trail, logging and the SIEM	16	41	plus about 180 other test cases
I. Outbound connections and queries on customer data	11	29	not given (code spread too widely)
J. What a person, an AI model or an agent is shown	25	70	not given (code spread too widely)
K. AI agents and the assistant	25	58	plus about 1,500 other test cases
L. Data protection: deletion and retention	5	11	not given (code spread too widely)
M. Configuration and start-up	4	10	plus about 90 other test cases
All areas	**251**	**704**	
Automatic checks that block an unsafe change before it is saved	40		
Start-up checks that refuse an unsafe start	14		
Guarantees left out or qualified (see the end of this appendix)	45		

> **How to read this appendix.** A guarantee is a rule Ziggy promises, written down in the server's testing runbook (`app/server/TESTING.md`) under an id of the form `INV-nnn` , and guarded by tests that name that id. The number beside each guarantee is how many test files in the server's source and test folders name that id exactly, and a file that names several guarantees is counted under each, so the area rows do not add up to the all-areas figure, which counts each file once. A higher number is not a stronger guarantee: it shows how many places the rule is exercised. To find the tests for a guarantee, search the source for its id, for example `rg -l 'INV-142' app/server/src app/server/test` ; the search can also find code comments and test helpers that name the id, so it may list a few more files than the number shown. A sentence marked **Limit:** states a case the guarantee does not cover; an entry that points to "Accepted for later" has an open item there, described in full or, for a Small item, counted by area. Each area is followed through to its tables, grouped by theme. The "other tests" figure is given only for an area whose code lives in a clear set of folders (it counts the test cases in those folders that do not name one of the area's guarantees); for an area whose code is spread across the whole server, no figure is given. The tables list the named core of the testing, not the whole of it.

A. Organisation separation

An installation can hold several organisations. This area protects each organisation's data from every other organisation in the same installation: a person, a service account or a background job in one organisation must not read or change another's data, by guessing an id or through shared plumbing. For example, an administrator of one organisation cannot list the people, Pipelines or connections of another.

Every read and write is scoped to an organisation

Guarantee	Id	Test files
No person, service account or background job can read or act on another organisation's data, and approvers are never chosen across organisations. Limit: the automatic scoping layer adds the organisation to list-style reads and writes only; calls by a single id, and any call that passes an absent organisation, rely on the handler naming the organisation itself, and a build check stops any new such call being added.	INV-01	76
A caller with no organisation (an account not yet approved, or a service account bound to none) reads and writes no organisation's data through the scoping layer and cannot name an organisation. Limit: the same one as the line above (calls by id, transactions and system jobs are not covered by the layer).	INV-483	4
Data that carries no organisation of its own (versioning branches and environment pointers) is reached only by joining to something that does, so one organisation cannot read another's through it. Limit: not yet held for two narrow cases, the branch of a revert changeset and conflict checks on an entry that names no organisation.	INV-93	31
An id a caller supplies for another item is checked to belong to the caller's organisation before it is stored; a foreign or unknown id is refused as not found. Limit: a listener row written before the rule is not re-read, and the trigger and the queue worker still read its Project and connection by id alone.	INV-94	14
An event listener chooses the organisation it works in; it does not inherit by accident the organisation of the request that fired it.	INV-86	8
A job id belongs to one organisation: starting a run with an id that another organisation's run already uses is refused. Limit: runs written before this rule in which two organisations already share an id still share one log buffer.	INV-484	2
Whether a data model, a view or a saved query exists for a reader is answered from the reader's own branch and organisation, and an item with no live row hands nothing over.	INV-495	1
A new organisation is set up with its own governance and ontology defaults, and nothing is written into the organisation of the administrator who created it.	INV-254	2

Ids and names supplied by a caller

Guarantee	Id	Test files
A persona named in a request must be an active persona of the organisation the row is written in.	INV-399	4
An id named in an ontology request belongs to the organisation, and an ontology row answers only with its own organisation's people and personas.	INV-402	5
The governance reads any member may make (domain tree, resolved approvers, move previews) answer for the signed-in organisation and for no other.	INV-398	3
A person choosing a persona sees the signed-in organisation's personas and how many people hold each, never who.	INV-390	2
Reading or writing a classification or a tag assignment never reaches another organisation's rows; a create that names another organisation's data model or tag is refused and changes nothing.	INV-267	2
The estate map shows only the signed-in organisation's connections and Pipelines.	INV-126	2

Who belongs to which organisation

Guarantee	Id	Test files
A person belongs to an organisation by an explicit membership, and no read derives membership from a team.	INV-197	7
Adding someone to a team ensures their membership of the team's organisation; removing the last team, or deleting a team, never removes them from the organisation.	INV-198	4
Every way of creating a person for an organisation (accepting an invitation, sandbox creation, first-user set-up) records their membership of it.	INV-199	4
No path leaves a person holding a role but belonging to no organisation.	INV-204	13
Administering an organisation needs belonging to it: a grant counts only while the person has a live membership of a live organisation, and leaving an organisation removes the grants scoped to it. Limit: appointing an administrator for an organisation the person is not in writes a grant that confers nothing until they join.	INV-546	1
Work is handed over only once an account is being offboarded, and only to a member of the same organisation.	INV-216	3

B. Sign-in, second factor, sessions and tokens

This area protects the doors into Ziggy: creating an account, proving who a person is, the second factor, and the sessions and tokens that follow. It guards against guessing a password or code, taking over an account through a provider sign-in, a session that outlives the person's access, and a token used somewhere it was not meant for. For example, a code sent to a mailbox never counts as a second factor on the same sign-in that used the mailbox to get in.

Creating accounts and signing in

Guarantee	Id	Test files
Once an installation holds two or more organisations, accounts are created by invitation only; self-service sign-up is refused because it cannot say which organisation the person wants.	INV-206	9
A sign-up domain restriction applies at every door (password registration and the first Google, Microsoft or LDAP sign-in) and only on an address someone vouched for. Limit: the domain check reads the address as typed; ownership of a typed address is proved by the mailbox step that follows, not by the domain check.	INV-540	2
Every door that sets a password (registration, reset by code, accepting an invitation, a signed-in change) applies the same strength rule.	INV-541	1
A password-reset code allows five guesses, however they arrive; each guess is claimed before the code is compared, so parallel guesses cannot exceed the cap.	INV-226	2
The sign-in attempt limit the installation sets is applied per account from each address and per address, and an attempt made while blocked is not counted, so a block ends. In a production process the server refuses to start unless it has been told how many proxies stand in front of it.	INV-509	4
A provider-link proof claims the account's limits (10 failed proofs and 5 codes an hour, 20 failed proofs a day) before it is judged, so a burst of requests cannot exceed them.	INV-538	1
Accepting an invitation is all or nothing: if the account, its membership or its role cannot be made, the account is removed and the invitation reopened.	INV-232	2
A Google, Microsoft or LDAP identity joins an existing account only after its owner proves it (the account's password or a code sent to the account's address), in the browser that began the sign-in, and after the account's own second factor; the provider's own claim is never enough.	INV-536	2
The mailbox counts as one factor: a sign-in whose first step was a code emailed to the account is offered no emailed code at the second-factor step.	INV-537	2
No bearer credential is put in a URL, and an OAuth sign-in finishes only in the browser that started it (state and a PKCE challenge held in a cookie).	INV-539	2
A caller-supplied redirect target is validated by resolving it against the application's own address and comparing origins, never by matching prefixes.	INV-75	2

Second factor

Guarantee	Id	Test files
Whoever holds a workflow power (Merge, Promote, Rollback, Admin or BreakGlass), by any path and at any scope, must use a second factor unless the operator has deliberately opted out; if the policy cannot be read, only a recent verified challenge passes.	INV-542	2
A person who must use a second factor and is not enrolled is refused, and one whose last challenge is stale is asked again, rather than let through.	INV-34	5
An administrator's second-factor recovery waits for another installation administrator to approve it and cannot be waited out.	INV-543	1
A service account can never satisfy a second-factor-gated grant, whatever the installation's policy; machine administration exists only through a deliberate custom grant that asks no second factor.	INV-13	4
An OAuth token carries no second-factor state of its own: for a delegated person the live step-up status is read on every call, and a machine is floored at not satisfied.	INV-20	2

Ending access

Guarantee	Id	Test files
Deactivating an account ends its live sessions, not just future logins: the account is re-read on every request and a refresh is refused. Limit: a session Ziggy opens in the separate Reports tool (Superset) is not ended by this, and an open live-update connection can stay open for more than about twenty seconds on a large installation (listed under Accepted for later).	INV-214	2

Machine and delegated credentials

Guarantee	Id	Test files
A delegated OAuth token is accepted on the MCP route only, and disabling the client or rotating its secret ends live tokens on the next request, with no expiry wait.	INV-18	1
The OAuth client is a pipe, never a principal: every delegated call runs as the consenting person with their roles; there is no client-credentials grant and no dynamic client registration; codes are single-use.	INV-19	1
An MCP session id is never a bearer: it is bound to the principal that opened it, and any other principal presenting it gets the same answer as for an unknown session.	INV-16	2
An execution callback speaks only for a job its execution token was minted for; an unknown job is refused with the same code as a foreign one.	INV-222	2

Sandbox mode (development and staging only)

Guarantee	Id	Test files
Sandbox mode is active only when every condition holds: the process was started with the switch, the installation runs development or staging, that environment is not flagged production, and sandbox is turned on. Any one false switches every sandbox behaviour off.	INV-182	2
Turning sandbox on needs a destination address and an environment that permits it.	INV-183	2
While sandbox is on, no second factor is asked, because sandbox refuses to create a code at all.	INV-184	3
Signing in as another user needs an Admin, a person (a service account is refused by name) and an active sandbox.	INV-185	2
When an administrator signs in as another user, the real actor is carried on the token and on every refresh, and every such sign-in is audited.	INV-186	2
Sandbox authorisation reads the real actor, not the person being acted as.	INV-187	2
A sandbox session dies with sandbox: it is refused the moment sandbox stops being active, with nobody revoking anything.	INV-188	5
While sandbox is on, password reset, second-factor email codes and recovery are refused before any code is made, and every email goes to the one sandbox address.	INV-189	6

C. Permissions, roles and the two admin tiers

This area protects what each person is allowed to do. Ziggy has two admin tiers, told apart by the scope of an Admin grant: an installation administrator and an organisation administrator. It guards against a route nobody decided, an organisation administrator reaching beyond their organisation, a role that carries more than its author may give, and an installation left with no administrator. For example, an organisation administrator cannot give anyone installation-wide Admin.

Every route is decided

Guarantee	Id	Test files
A route with no registered access policy is denied; every route's access is an explicit registration, and the server refuses to start with a route that has none, unless the operator sets ALLOW_POLICY_COVERAGE_GAP (an emergency switch that is off by default, lets uncovered routes start, and is never to be left on in production).	INV-02	4
A route open without sign-in must authenticate its caller some other way, or be listed with the reason it needs nothing; the server refuses to start otherwise, unless the operator sets ALLOW_POLICY_COVERAGE_GAP (same switch, same limits).	INV-224	3
A route carrying View, Author or Operate resolves to a named capability feature, or its controller is on the exempt list with a reason; the server refuses to start otherwise, unless the operator sets ALLOW_CAPABILITY_COVERAGE_GAP (an emergency switch that is off by default and is never to be left on in production).	INV-218	1
A per-feature grant authorises every route of that feature, sibling controllers included, and nothing outside it.	INV-219	1
The graph feature can be granted on its own, and a graph grant never carries row data.	INV-220	2
Every MCP tool carries an explicit authorisation mapping; an unmapped tool cannot ship.	INV-15	2
An anonymous webhook is accepted only when it is signed; a provider that cannot sign gets no webhooks, and a bad signature is refused and recorded without the caller's body.	INV-227	2

Grants, roles and the two admin tiers

Guarantee	Id	Test files
A grant that is not organisation-scoped (installation, artifact type, resource, environment, folder, persona) keeps working exactly as before, in any organisation and in none.	INV-190	20
An organisation grant satisfies only decisions made inside its own organisation, and never a decision that names no organisation.	INV-191	47
An organisation administrator cannot reach another organisation through any route: lists are narrowed to what they administer, an explicit request for another organisation answers empty, and they cannot appoint an installation administrator or assign a role carrying a workflow power. Limit: an Admin held only through an organisation-scoped role assignment reaches nobody.	INV-192	35
An organisation-owned custom role cannot contain a workflow power (Merge, Promote, Rollback, Admin, BreakGlass); the refusal is at authoring time, and the same ceiling applies to exception grants.	INV-193	4

Guarantee	Id	Test files
An exception grant written by anyone but an installation administrator is scoped to the organisation they are working in. Limit: a grant to a person within reach who is not a member of that organisation is stored but confers nothing.	INV-482	4
A custom role may not take a system role's name; two organisations may each own a role of the same name.	INV-195	2
A role assignment applies in one organisation, or deliberately in all of them.	INV-201	7
The engine admits a role only where its assignment applies, judged against the session organisation, and a bound workflow power is narrowed to it.	INV-202	17
An organisation administrator removes, or puts on a team, no authority beyond the organisations they administer.	INV-544	3
A role an organisation owns is assigned only in that organisation. Limit: assignments written before the rule are not touched.	INV-545	2
Who is an installation administrator is read from the grant on every path, the command line included, never from a role's name or a display claim.	INV-547	2
A person's account is changed only by an installation administrator, or by an organisation administrator of every organisation the person belongs to.	INV-559	3
No path leaves the installation without an administrator who can sign in.	INV-194	13
The last installation administrator who can still sign in cannot be deactivated.	INV-213	1

Rights judged on the instance, not the type

Guarantee	Id	Test files
A saved query is its owner's to edit and to run until it is shared, whatever role the caller holds; a monitor may name a query only where its creator may run it, and its owner must hold that right each time it runs. Limit: a colleague with the right to run monitors can still run or test a monitor and read its count and first number (listed under Accepted for later).	INV-494	4
Content is judged on the instance: ownership or a grant naming the item, so an Admin's role and any other type-wide grant never pass it.	INV-532	1
Linking a connection to a shared OAuth account needs the Operate right on that account. Limit: a promotion apply writes such links under the Promote right.	INV-519	2
Introspecting, running and scheduling use the connection, so they need Operate on it: refused to anyone who neither owns it nor has it shared with them, whatever their role.	INV-522	2
A pipeline run is reached by its id only by a person who may operate its Project, in its own organisation. Limit: a run held by an execution worker, whose row is only in the worker's database, is refused to everyone.	INV-524	4
Reading a Pipeline needs View on the Project; editing it needs Author.	INV-283	4
A subscription request is approved by the product's Owner or Steward persona holder and by nobody by role; seeing and revoking belong to those holders and an Admin of the product's organisation.	INV-319	5

D. Approvals and change control

This area protects what reaches the live system. Changes to reviewed content are made in a changeset, approved by the people the content's domain names, and only then merged; emergency access is possible but leaves a full record. It guards against an unreviewed edit going live, an approver standing in for another, and staged work acting before it is approved. For example, a Pipeline edited inside a changeset does not run on its schedule until the changeset is merged.

Nothing reaches the main line without review

Guarantee	Id	Test files
A write to reviewed content with no changeset and no emergency-access token is refused. Limit: moving a data-quality check into a folder can put an open changeset's unreviewed edit on main, and an operator's drift-repair script can do the same (listed under Accepted for later).	INV-05	5
Work in a changeset (creates, edits, staged deletes) is invisible on main and harmless to live rows until merge, and nothing acts on it by itself: schedules, rules, metrics and runs follow main's definition. Limit: the same two exceptions as the line above (listed under Accepted for later).	INV-07	22
Nothing merges unless every touched item's final content was seen by a valid approval; an edit after approval invalidates the approval. Limit: the same two exceptions as above (listed under Accepted for later).	INV-08	5
An unresolved approver persona blocks the submit rather than skipping the gate.	INV-03	2
Governance gates only what it governs: with no governing domain a changeset is approved automatically, so it can never be an unmergeable dead end.	INV-04	2
Every governing domain must approve; any single rejection rejects the whole changeset.	INV-09	2
Every relaxation of separation of duties emits a record with a non-empty reason, and an automatic approval under a standing rule is not an exception.	INV-10	2
Approval can always progress (widen the pool, escalate, relax with an audit record, or hard-stop for emergency access); a changeset is never silently stuck.	INV-11	5
Emergency access (break-glass) requires a fresh second factor and a justification, and produces a complete audit record.	INV-12	4
Every emergency-access write to main carries the id of its record on the commit, from any route in the product.	INV-268	3
A reviewer holds one standing decision on a changeset; a second one is refused before anything is recorded.	INV-253	2
What a changeset removes is approved under the domain's delete workflow, never under less than an edit would need, and the author is never one of the approvers.	INV-281	6
Renaming the code of a reviewed item is a governed write like any edit.	INV-301	3
On a fully merged main with no open branches, canonical drift (a live row ahead of its committed form) must be zero.	INV-06	2
A ref never loses a commit: concurrent writes to one ref all land, one after another, and a move to a commit not built on the tip is refused.	INV-249	5
A commit that history rests on cannot be deleted, by any route, raw SQL included.	INV-477	1

Guarantee	Id	Test files
Deleting a workflow or a step erases nothing of the approvals that ran through it: who approved, and when, stays.	INV-478	2

Staged work stays staged until it merges

Guarantee	Id	Test files
The catalogue never advertises an item the committed tree does not carry: visibility is asked of the tree on each read and never stored, so an abandoned experiment is not reported as part of the estate.	INV-97	27
A port's publish or deprecate staged in a changeset takes effect only when it merges.	INV-295	3
Anything staged about a port has no effect outside the changeset until it merges; the port list and reads answer to the committed tree.	INV-298	9
A port's contract version, bumped or deprecated, follows the changeset rule, or needs an emergency-access session on main.	INV-299	4
A data product's own edit staged in a changeset takes effect only when it merges.	INV-315	2
A classification staged in a changeset takes effect only when it merges.	INV-321	3
A tag or tag assignment staged in a changeset takes effect only when it merges.	INV-323	4
Every classification and tag-assignment write lands on the working branch, and with no changeset it is refused like any other reviewed write.	INV-264	2
Main never shows an unmerged classification: not in lists, coverage, the graph or search.	INV-265	3
Every write to a critical data element lands on the working branch; a write straight to main is refused.	INV-256	2
A change to a critical data element is reviewed by its governance domain and cannot merge before the domain's approver approves.	INV-257	2
An unmerged critical data element is never marked on main.	INV-263	2
What runs on its own schedule follows main, for every scheduled type: a schedule or definition staged in a changeset is not registered and no job runs it until the changeset merges.	INV-303	12
A Pipeline run nobody launched (a listener's trigger, a monitor's canary, a sub-flow of a scheduled run) runs the Pipeline as main holds it; a person's run uses the branch it was launched on.	INV-305	14
A metric's Run inside a changeset is a trial: it is recorded only when main holds exactly that definition, otherwise the results are returned and nothing is written.	INV-306	3
A scheduled batch prediction reads the model as main holds it.	INV-307	2
Nothing runs or counts a metric off its row: a query runs the metric as the request's branch holds it, and a materialisation runs only the definition main holds.	INV-316	5
The product catalogue answers to main's tree: a product or port only a changeset holds is not seen by manifest sync, impact comments or the in-use scan.	INV-317	2
A contract version records who deprecated it once it lands on main: the person who staged the deprecation, not the merger.	INV-311	2

What a reviewer is shown

Guarantee	Id	Test files
What a changeset is said to have changed is what that changeset changed, at every point of its life, and a merged changeset goes on answering what it carried when it merged.	INV-391	6
A changeset that changes a Pipeline reports what depends on what that Pipeline writes, using the same answer the Explorer gives.	INV-410	3
A changeset carrying a breaking change with a live contract downstream is classed high risk, at submit, at resubmit and when a branch advances under review. Limit: if the changeset's own graph cannot be built, the tier is answered from the stored graph, so it is not fail-closed on that read.	INV-414	3
An approval covers the classification level it was given at; if the level changes later it reads as a breach, never as approved.	INV-124	2
A decided certification approval is recorded as decided, with who and when, the moment it is decided.	INV-287	4

E. Moving changes between installations

Teams promote reviewed content from one installation to another (for example development to production) as signed bundles. This area protects the receiving installation: only trusted, whole, in-order content lands, never a credential, and promoting to production is a person's act, recorded, with a backup. For example, a bundle signed by a key the receiving installation has not been told to trust is refused before any other check.

What a bundle may carry and when it is trusted

Guarantee	Id	Test files
A bundle carries reviewed content only, never a credential, a secret value, a user account, telemetry or other local state, in either direction.	INV-131	5
Identity, membership and role assignments never travel in a bundle, because they are local access facts: promoting development work must not silently grant production access.	INV-203	2
An agent's credentials, subscriptions, grants and budgets never travel in a bundle; the target mints its own identity.	INV-441	2
Nothing acts on an unverified bundle: signature verification against a trusted, active key comes before every other gate; a tampered payload or an unknown or revoked key is refused.	INV-132	5
Importing the installation's own signing key is refused unless self-promotion is deliberately turned on, and a banner shows while it is.	INV-171	7
Every commit in a bundle is reachable from main; the exporter refuses a commit that is not main's tip or an ancestor of it.	INV-130	3

Guarantee	Id	Test files
A bundle is self-sufficient: it carries the commit rows of every ancestor back to the baseline, so any bundle applies to any installation in any order.	INV-143	9
A commit is the same commit everywhere, forever: ids are content-addressed.	INV-144	3
A reference to an introspection snapshot or model resolves locally or arrives attached, or the bundle is refused, naming it.	INV-151	1
A data-residency requirement still refuses a bundle whose required residency differs from the installation's.	INV-150	2
Classification content in a bundle is disclosed to the person exporting it as a warning; it never blocks a deploy.	INV-149	3
Schema compatibility is judged on structure, not on migration history, and the refusal names what differs.	INV-138	4

A person's act, recorded, with a backup

Guarantee	Id	Test files
A production promotion is a person's act: apply and rollback into a production environment demand a fresh second factor from a human while the policy is on, and a machine is refused by name; writing production connection credentials takes the same step-up.	INV-139	10
Only a person with Admin and a fresh second factor decides whether an environment is production, in either direction, and a fresh installation starts protected.	INV-174	3
Whether a bundle may replace a target is decided by the target: an execution installation or a production environment is replaced only by emergency access, with a loss preview, the environment name typed, a justification, a covering backup and a notice to administrators.	INV-173	4
Apply is fast-forward only, and success advances main and the environment pointer together.	INV-145	6
A pristine installation adopts a trusted bundle's history; a non-pristine one keeps the fast-forward refusal, and the refusal says where the disagreement is.	INV-160	4
Apply is atomic: any failure leaves zero rows written and an unmoved pointer, and a refusal is a coded answer, never a stack trace.	INV-133	2
The pre-flight reports every condition that can refuse an apply as its own row, so an all-green report cannot be followed by a refusal on a condition it covers. Limit: ancestry is re-judged inside the apply, so a change between the report and the apply is refused with the same coded reason.	INV-134	6
An apply is covered by a backup it can name, or it does not run: the backup is taken before any write, on every installation, and if it fails the apply is refused with nothing written.	INV-136	4
The covering backup is copied off the machine when a storage provider exists, both locations are recorded, and the last ten local copies are kept.	INV-168	4
Every export and apply attempt is recorded, refusals included, with actor, target, outcome and the coded reason.	INV-135	9
Every move of an environment pointer is a row in the promotion log, kept separate from the commit log.	INV-146	3
The commit-advanced event fires once per successful apply or rollback, after the transaction commits.	INV-147	2
Rollback restores only what this installation actually holds: only points whose content is present are offered, a refused attempt is not offered as a place the environment has been, production rollback takes a fresh second factor, and the justification is mandatory and recorded.	INV-148	7
Hydrated rows are owned by the built-in Promotion principal; the source owner is kept as provenance only, so the person applying never becomes every Pipeline's identity.	INV-162	5
An apply and a rollback target the environment this installation runs; nothing offers a choice, and any other environment is refused.	INV-176	9
Promotion routes check Promote and Rollback against the Environment, so the built-in Promoter role works unmodified.	INV-169	5
Export is authoring and is refused on an execution-only installation; handing back a bundle the installation holds is not, and needs Promote.	INV-140	4
An installation that receives bundles can trust or untrust a signing key and roll an environment back with authoring turned off; trust changes need a fresh second factor from a person.	INV-175	2
Installation settings are changed with a fresh second factor and are audited; environment variables only seed the first boot.	INV-170	2

F. Credentials and secrets

This area protects stored passwords, keys and tokens. It guards against a secret being shown to a person or client, sent to a place it was not bound to, carried between installations, or kept in a form that is easy to steal. For example, a connection's password is shown masked to every caller, including the person who owns the connection.

Stored secrets are write-only

Guarantee	Id	Test files
A stored secret is write-only: every connection read masks the connection type's declared secret fields for every caller.	INV-163	11
Writing credentials alone keeps every field and the mode it leaves out: a field left out keeps its stored value, and the mode changes only when stated.	INV-521	5
The hidden-settings marker is never stored as a real setting.	INV-520	3
A connection's settings and its last error go only to its owner and the people it is shared with; an Admin with no grant reads it as any viewer. Limit: bundle downloads and the Superset sync are not covered.	INV-518	5
Principal credentials are hash-only and shown once: service-account keys and client secrets are bcrypt-hashed, access and refresh tokens are SHA-256 hashed, and plaintext exists in one response only.	INV-24	5
An agent has an account only while it is held by main or an open changeset; one nothing holds has no live account, and none is minted for it.	INV-411	6

Secrets stay with their installation and environment

Guarantee	Id	Test files
Credential policy is one switch per installation, default strict; strict holds only its own environment's secret values and accepts none from a bundle.	INV-164	14
At run time an installation resolves only its own environment's credentials, under either policy; no path falls back to another environment's credential.	INV-165	20
Connection credentials are the installation owner's: reading reports which environments are configured and never returns a secret value; writing needs Admin and targets the installation's own environment only.	INV-141	2
An outbound OAuth grant lives on its own token row, keyed by connection and environment, never in the credentials a person typed, and a grant never crosses an environment boundary: a connection promoted to another environment needs fresh consent there. Limit: queue-listener connections take the first stored credential by environment order, a labelled stand-in until per-environment resolution is built.	INV-73	3
Scope authorises credentials: an external-governance repository receives tokens only for the connections a person scoped to it, never for what the code scanner happened to find.	INV-80	2

G. Encryption and keys

This area protects sensitive values stored in the database and the keys that seal them. It guards against plaintext at rest, a key rotation that strands or exposes data, and a key file that is overwritten or committed. For example, rotating the key does not make any old value unreadable.

Sealed at rest

Guarantee	Id	Test files
Registered sensitive fields are encrypted at rest and decrypted on read; keys are versioned, so rotation never strands old data. Limit: boot checks that the key is present, not that every model is registered, so a new model that is not added to the registry would store plaintext until it is registered (the new-model checklist is the only enforcement).	INV-31	3
A secret kept in settings (mail password, OAuth secrets, AI keys, agent-host token, Slack and Teams credentials) stays readable across a rotation of the personal-data key.	INV-223	6

Key rotation and the keys file

Guarantee	Id	Test files
Every key type the keys file holds rotates and migrates with the shipped command-line tool, and a migration cannot stop early.	INV-548	2
Only an audit after a completed migration says the old key may be removed; a migration with failed rows ends as failed and names the count. Limit: a half-finished authenticator set-up (a pending authenticator secret) is not yet counted by that audit (listed under Accepted for later).	INV-549	2
The keys file is replaced, never rewritten in place: written to a temporary file, synced and renamed over, with the previous file kept; and no copy of it is committed.	INV-550	2

H. Audit trail, logging and the SIEM

This area protects the record of who did what. It guards against an act that leaves no trace, a record that names the wrong actor or disappears with the account, and a log shipment that fails without anyone knowing. For example, every denied permission decision leaves an audit row. Alerts are raised only for the event types that reach the alert listener; the guarantees below are about the record, not about alerting.

Who acted, and the form it is kept in

Guarantee	Id	Test files
There is one audit writer, the command line's second-factor reset included, and every row it writes is copied to the SIEM after the write, with credentials masked by name and by value.	INV-552	3
Every denied authorisation decision leaves an audit row (best effort: a write failure is logged and never blocks the decision). Allowed decisions are deliberately sampled.	INV-32	3
Machine actions never write the users link: audit rows for service accounts carry the actor type and id only, and human actions carry both.	INV-14	5
MCP business audit rows are written only for executed calls; denials live in the access-decision record.	INV-17	2
A history row records who acted in a form that outlives the account: name and email are captured at creation, beside the user link.	INV-207	4
A history row's user link is cleared, never reassigned to a successor, when an account is offboarded, and the snapshot beside it is never touched by that write.	INV-209	2
Every model call writes exactly one attributed ledger row (feature, organisation, principal), whether or not request detail is stored; the client's own metadata never sets attribution.	INV-380	3

Acts that could erase the record

Guarantee	Id	Test files
An act that can erase audit rows is recorded before it acts: a restore, a housekeeping change and a sweep are written first, and under fail-closed SIEM a record that is not delivered refuses the act.	INV-553	5
No route erases the installation's data wholesale: the system reset is gone.	INV-554	1

The SIEM and request tracing

Guarantee	Id	Test files
SIEM settings are judged at start-up in every process and environment: fail-closed with no usable provider, an unrecognised fail-closed value, or a retention CloudWatch does not accept is refused. Limit: after start-up, a provider outage stops no process.	INV-555	3
The SIEM loses nothing silently: one flush at a time, a failed delivery keeps its entries ahead of newer ones, a drop is printed by count and reason, and a flush says delivered only when nothing was dropped.	INV-556	3
Every log line written during a request or job carries its request id, and a line written outside one carries none.	INV-234	2
A job runs under the request id of the work that enqueued it, and a job enqueued outside a request gets its own.	INV-235	2
Every queue and worker carries the request-id telemetry.	INV-236	2
Every response names its request id, a guard's refusal included; a caller's id is reused only when it is short and plain.	INV-238	2
The operations metrics route needs installation View and never reports an uncountable queue as empty.	INV-237	2

I. Outbound connections and queries on customer data

Ziggy connects out to customer databases, mail servers and providers, and runs queries on them. This area protects those connections and the customer's data behind them. It guards against a query that writes, SQL assembled from a caller's text, a call that falls back to no authentication, a redirect to another site, and a call that hangs forever. For example, a saved query can only read.

Queries only read, and only what they should

Guarantee	Id	Test files
User-authored SQL can never write: every execution passes the read-only validator before it touches a live connection, for saved and ad-hoc queries alike.	INV-23	2
A statement Ziggy runs as a read does no more than read, on every dialect it serves: the validator refuses, per dialect, functions and clauses that write, lock, wait, read a file or run SQL given as text.	INV-493	2
A caller of a metric query never supplies SQL: a time grain is one of a closed set and a filter names a dimension the author declared.	INV-427	2
A batch's input filter and a model's scope filter are one row condition judged by one rule wherever saved or run: no statement break, comment, subquery or write; a filter that cannot be checked is refused, never run unchecked.	INV-424	3
The source reference in a SQL step is resolved to the connection the statement runs on, or refused; another connection, another organisation's or an unknown id is refused alike.	INV-245	1
What runs is the copy its reader holds: a person's run is their changeset's, a machine's run is main's, and nothing runs a saved query's stored row. Limit: no check stops a new caller from reading a saved query's stored SQL from the row, and a pinned view, a rule's view and a Superset sync have no HTTP test.	INV-496	4
A dry run of a Pipeline can never cause a write, anywhere: every side effect is gated by the dry-run flag, a store captures the rows it would have written, and nodes that reach the outside world stop the walk.	INV-40	8
Sending an email reads no file on the server and deletes none: the request has no attachment or path field, and the mail library is told to open no file or address. Limit: the execution send-email route still accepts such a body at its input check and answers with a failure instead of the proper refusal; nothing is read.	INV-485	2

Calls out

Guarantee	Id	Test files
An authenticated connector never falls back to an unauthenticated call: an unsupported flow, an unsaved connection or missing OAuth client settings fail loudly.	INV-74	3
In an OAuth connect flow the signed state is the authentication: it is HMAC-signed, single-use, valid ten minutes, and bound to connection, environment and actor.	INV-76	2
Every direct outbound fetch carries a time limit.	INV-230	2

J. What a person, an AI model or an agent is shown

This area protects what comes back in an answer: column values, settings, errors, conversations and files, and what is sent to an AI model. It guards against a reader seeing more than their standing allows, an Admin reading content only by role, and a value or secret reaching a model. For example, an Admin with no grant on a connection does not see its profiled column values.

Data values

Guarantee	Id	Test files
A connection's data values (top values, minimum, maximum, histograms) and an AI model's source rows go to whoever may Operate the connection; never to an Admin by role alone.	INV-529	1
The search index carries no column value.	INV-530	1
A published output port answers only for the columns it publishes, and a classified column is never returned in the clear nor usable as a filter.	INV-42	11
A profile withholds the values of the columns classified on the table it read, and of no other table.	INV-389	3
A classified column is never lost to a misread expression: a computed column carries the strictest classification of what it may read.	INV-252	1
A classification applies to exactly the table it names, schema included, and never fails open; a bare name takes the strictest of every schema's.	INV-288	10
A port's data is read only with an active subscription or by the data product's Owner or Steward; a platform Admin is not a reader by role and subscribes like anyone.	INV-181	3
A data-quality proposal carries a column's values only for a person who may Operate the connection.	INV-533	2

Guarantee	Id	Test files
An analytical model's suggestions show the AI model values only for a person who may Operate the connection.	INV-534	2
A query's schema text carries values only on a turn of a person who may Operate the connection, and is never stored with them.	INV-535	3
An ML model's editor works with its owner's data access; a View-only reader without data rights sees a safe summary; only the owner shares editing, with an explicit acknowledgement, and a transfer removes editors. Limit: a model deleted on main but still held on a branch can show its scope filter to its former owner (listed under Accepted for later).	INV-560	9

Settings, errors, conversations and files

Guarantee	Id	Test files
A run's error is stored and answered without the connection's whereabouts (host, address, port, user, database), which only the owner and sharers may read. Limit: a value the patterns do not recognise, and rows stored before the rule, keep their text; two small open items are listed under Accepted for later.	INV-523	14
What a run carried (resume cursor, rejected rows, debug payload, failed queue messages) is read by whoever may Operate the Project; an endpoint run's content goes to the person who started it.	INV-525	1
A query's conversation is its owner's and the people it is shared with; every other viewer, an Admin included, keeps the SQL and parameters but sees no conversation.	INV-526	1
An Admin administers who may use an agent and reads none of its use; conversations, cost and test runs go to its managers. Limit: an administrator present when an organisation is created is made its Owner persona holder by the seeder and reads as one.	INV-527	1
A discovery's notes, interview and agent session are its owner's; everyone else, an Admin included, sees the rows without the notes.	INV-528	1
A stored file's bytes are its uploader's and a public file is anyone's; an Admin of either tier is refused another person's private file.	INV-531	1
Organisation-wide AI spend is readable only through a grant that names it; no built-in role, Admin included, carries one.	INV-383	5
Per-person cost on an agent is read only by those who manage it, and My AI usage returns only the caller's own rows.	INV-443	1
An AI chat conversation is never served to a person who does not own it.	INV-37	2

What reaches an AI model

Guarantee	Id	Test files
A failing tool tells the model a database's, driver's or host's words never: only a planned refusal, or an error about input the model itself wrote, is passed on; anything else is one fixed sentence. Limit: a run's own failure message that is returned rather than thrown still reaches the model, and a new handler's catch is found by review, not by a check.	INV-510	2
A Pipeline's documentation is written from every node's configuration and no credential, header value, address login or row of data is in what the model is sent. Limit: a secret typed into a setting whose name does not say so, and a statement's own text, go as written.	INV-511	1
What an agent is answered is redacted by the data model that ran, and where none can be established every value is withheld. Limit: an agent test run's reference figure is one raw number without this withholding.	INV-498	2
The assistant answers from no page it is kept out of: the admin portal, source-code and security sections of the guide are in no search hit unless a page says otherwise.	INV-502	3
What a page tells the assistant it shows is found again by the server, in the person's organisation and for that person, before anything is spent.	INV-500	1

K. AI agents and the assistant

AI agents and the built-in assistant act with tools. This area protects the line between what an agent proposes and what happens: an agent never reaches further than the person it acts for, never runs a tool it was not given, and never writes to the live system without a person's approval. For example, an agent's proposed change goes into a changeset for review rather than onto main.

The agent boundary

Guarantee	Id	Test files
An agent session holds no portable write credential: it is a Viewer service account that works on the MCP route only, and ending a session is two acts.	INV-106	3
An agent write to a route the changeset gate does not cover pauses for a person (428); the default is to pause.	INV-107	2
A route may declare that it writes nothing, and the claim is watched at run time, not believed.	INV-111	2
An approval releases exactly one call: the next matching call (same method and path) spends it, and it is never a standing permission. Limit: the release is read and written back, not claimed in one step, so two identical calls at the same instant could both run (listed under Accepted for later).	INV-112	3
The agent's SQL is select-only and every table it names is shadowed by an organisation-filtered view; an unknown table, a user-defined view or a schema-qualified name is refused.	INV-108	3
MCP offers a principal only the tools it can use.	INV-178	4
The AI provider's address is replaced only for a test lane, only by an address on this machine, and never in a production process.	INV-499	1

What an agent may do and see

Guarantee	Id	Test files
An agent never runs a tool its definition does not offer, whatever the model asks for.	INV-429	4

Guarantee	Id	Test files
A catalogue tool run for an agent returns nothing outside the agent's scope, no data and no metadata.	INV-430	7
An agent used on behalf of a person never reaches further than that person, and a grant on an agent never gives reach to someone who cannot see it. Limit: an editor's preview of an agent's saved-query operation on main answers rows the editor may not have been shared (listed under Accepted for later).	INV-431	11
A person's own settings on an agent only narrow what their grants give, and a block overrides every grant before any model call.	INV-432	3
A tool a caller may not use is never offered to the model on that caller's turn.	INV-433	6
The end-user reference an API caller names is attribution only; it never grants, widens or narrows access.	INV-436	3
An agent's key or embed token authenticates that agent's routes and nothing else; a delegated connector token is accepted only on MCP.	INV-437	4
Creating an agent needs Author on the Agents feature; changing a particular agent needs owning it or a grant naming it, not a type-wide Author.	INV-444	5
An agent's saved-query operation reads the query inside the agent's own organisation, on every channel and with nobody present.	INV-449	3

Nothing an agent proposes goes live unreviewed

Guarantee	Id	Test files
Nothing an agent proposes reaches main without a changeset merge, and an interactive write needs its person's explicit, unexpired approval.	INV-434	9
An agent version that has tests cannot be submitted without a finished test run on that exact version.	INV-440	3
An agent version whose latest test run has a failing value check cannot be submitted, and nobody can acknowledge that failure.	INV-442	2
A hard spending limit, once reached, stops the next model call, not only the next turn.	INV-435	4
Meaning-layer authoring through the assistant is Author-gated and always staged in a changeset, and a delete is confirmed on its card.	INV-67	2
The built-in assistant runs as the person asking and is open to every signed-in person and nobody else; a service account is refused.	INV-490	1
A change the assistant proposes to a page reaches the page only through its person's approval, once, and the server performs nothing.	INV-492	1
A catalogue tool is off while any feature it names is switched off, on every path. Limit: a feature switched off mid-turn reaches an authored agent on its next turn.	INV-516	1
While Classification is switched off, no tool result carries a classification. Limit: describing an output port still gives a column's level and redaction constraint.	INV-517	1

L. Data protection: deletion and retention

This area protects data at the end of its life: deleting an organisation or a person, and erasing soft-deleted rows only after the retention window. It guards against half-deleted organisations, rows erased too early or never, and an account removed while it still owns work. For example, a soft-deleted organisation keeps its 90-day recovery period even though other rows are swept after 14 days.

Deleting and erasing

Guarantee	Id	Test files
Organisation deletion is children-first or not at all: a raw delete of a non-empty organisation is refused rather than half-deleting.	INV-28	2
The generic 14-day hard flush never pre-empts a longer retention window: tables owning a 90-day recovery period are excluded.	INV-29	3
One hard flush erases a soft-deleted parent together with the children soft-deleted beside it, and never a live row.	INV-476	2
Every soft-deleted row can be aged, so every one is erased once past the window; a revoked permission grant is erased like any other.	INV-480	1
Permanent delete of an account is refused until the account is clean, checked fresh, and the refusal is checked before any destructive side effect; the purge is one transaction.	INV-217	5

M. Configuration and start-up

This area protects the installation from starting in, or drifting into, an unsafe state. It guards against an environment that cannot work, a production process started with a setting that weakens sign-in, and an installation whose pointers have diverged. For example, a production process with plain ldap:// refuses to start.

Refusing an unsafe start

Guarantee	Id	Test files
A production process is refused the settings that are refused in production, however it says it is production (APP_ENVIRONMENT or NODE_ENV): plain ldap://, sandbox allowed, plain-text mail (outside production it is allowed only if the operator sets SMTP_ALLOW_PLAINTEXT), a weak execution-callback secret, fault injection, and an unstated proxy count. Limit: an installation flagged production only in its own database record, with neither variable set, is not refused.	INV-508	4
The server refuses to start on an environment that cannot work: DATABASE_URL must be a postgres URL, and every variable the server parses as a number must be a number when set.	INV-229	2
Environment pointers track main: canonical migrations write to main only, and divergence is detected at every boot. It is a warning by default; the server aborts only if the operator sets FAIL_ON_ENV_DIVERGENCE.	INV-30	2
The remote worker starts and serves only its own controller and health; every other route answers 404.	INV-239	2

Automatic checks

The server's own `package.json` holds 36 `check:*` scripts and the repository root holds 4 more, 40 in all. Each reads the source and fails if it finds the pattern named. The column "Before a push" marks the checks the push hook runs; the same set runs in the project's build pipeline. A check that holds a baseline can only shrink it: a new site fails, and a site that disappears must be removed from the baseline.

Checks on security, tenancy and review

Check	What it refuses	Before a push
`check:org-scoped-reads`	A new read or write by a single id on organisation-owned data that does not name its organisation; the count of such calls may only fall.	yes
`check:tree-reads`	A new direct read of the version tree (which has no organisation column) outside the code that owns it, unless it carries a written reason; the count may only fall.	yes
`check:organisation-writes`	A write that sets its organisation by a form that skips the organisation check, or an organisation-authority call outside its one place.	yes
`check:canonical-write-paths`	A write to a field of a reviewed item's committed form that does not also commit, which would leave a live row ahead of what was reviewed.	yes
`check:borrowed-access`	A run that builds its owner's access by hand instead of borrowing it through the one place that judges it.	yes
`check:secret-release`	A new reader of stored secrets outside the one place that releases a secret to the destination it was bound to.	yes
`check:reader-scope`	Content reaching a reader by any route other than the single place that decides each reader's standing.	yes
`check:holder-standing`	Code that acts for a person without asking the one standing check, or that ends logins by any routine other than the shared one.	yes
`check:credential-claims`	A single-use sign-in credential or a session that is not claimed or minted in one locked step, and a write to a credential record outside a registered claim.	yes
`check:fresh-proof`	A fresh-sign-in check written site by site instead of the one shared check used by the app, the command line and the admin portal.	yes
`check:file-writes`	Removing, replacing or moving a stored file other than through the one rule that asks the owner's right.	yes
`check:file-reads`	A read of a stored file's bytes in a place that did not ask the reader's right to them.	yes
`check:schedules`	A schedule a person sets that does not keep the rule of at most once a minute, on save or on run.	yes
`check:raw-sql`	A value becoming SQL text: raw statements built by joining strings, and string literals written into SQL by interpolation.	yes
`check:run-limits`	A number a person sets for how long, how often or how much a run uses that is not held to the server's ceiling.	yes
`check:delete-policy`	A foreign key to a soft-deletable parent with no stated delete behaviour, an invalid entry, or an illegal behaviour.	yes
`check:schema-relations`	A database relation that does not state what happens on delete, a soft-deletable model that cannot be aged for erasure, and a foreign key with no index.	yes
`check:route-coverage`	A controller route that no test reaches over HTTP and that no allowlist entry excuses.	no
`check:invariant-coverage`	A guarantee in the testing runbook that no test names, and a test tag that names a guarantee which does not exist.	yes
`check:ai-models`	An AI model version written as a literal outside the registry, and a model tier that points at an id its provider does not list.	yes
`check:ai-tool-loops`	A second copy of the AI tool loop outside the one that handles truncation, pricing and stored history.	yes
`check:domain-errors`	Server code using the web framework's HTTP exceptions, and a controller throwing a plain error that would answer as an unexplained failure.	yes
`check:api-response-status`	A route that declares a success status other than the one it returns, which would leave its answer unchecked by the contract tests.	yes
`check:module-boundaries`	Core code importing application code, a module importing another module's repository, and an undeclared global module.	yes
`check:unbounded-lists`	A list read over a table that grows with use that does not say how many rows it wants, unless it carries a written reason.	yes

Checks on code quality

Check	What it refuses	Before a push
`check:loop-await`	A loop that makes one database round trip per item without saying why.	yes
`check:unsafe-casts`	A double type assertion that defeats the type checker, unless baselined with a reason.	yes
`check:prisma-doubles`	A unit test that builds its database double by casting a hand-made object.	yes
`check:spec-mirroring`	A unit test that only repeats the list of calls the code made instead of asserting what it did.	yes
`check:e2e-sleeps`	A fixed sleep in an end-to-end test where a wait for the outcome belongs.	yes
`check:class-coupling`	A class that gains collaborators beyond the allowed number.	yes
`check:cognitive-complexity`	A function that becomes harder to read than its baseline.	yes
`check:format`	Code that does not match the project's formatting.	yes
`check:docs-paths`	A path named in the repository's guidance files that does not exist.	yes
`check:quickbooks-schema`	The committed QuickBooks schema artifact drifting from the vendored definitions.	no
`check:catalogues`	Generated documentation catalogues that are out of date.	no
`check:comments`	A source comment that promises future work instead of stating a present fact.	no
`check:plans`	A plan whose phase completion cannot be recorded, or whose completion was recorded but not acted on.	yes
`check:agent-chat-contract`	The published agent-chat contract drifting from its generated source.	no

Check	What it refuses	Before a push
`check:directives`	User-guide pages (which the assistant reads) that break the guide's structural and wording rules.	yes

Checks before a commit

A commit hook also stops a commit that would add a secret (a secret scan over the staged files), a real encryption-keys file (only the test fixture is allowed), a data dump, log, environment file, backup or upload folder, or a deleted plan that should have been archived.

Start-up checks

The server runs these as it starts and refuses to start, or says so loudly, on a failure. The last column names the operator setting that changes the check. ALLOW_POLICY_COVERAGE_GAP, ALLOW_CAPABILITY_COVERAGE_GAP, DATABASE_ALLOW_UNVERIFIED and REDIS_ALLOW_UNVERIFIED lift or soften it: each is off by default, and the secure configuration guide explains what it lets through and when, if ever, it is acceptable (the first two are never to be left on in production). The two FAIL_ON settings and DELETE_POLICY_DRIFT_FATAL do the opposite and turn a warning into a refusal.

Check	Guarantee	What it refuses	Unless the operator sets
Route policy coverage and anonymous routes	INV-02, INV-224	Every route must be registered or deliberately public, and a public route must authenticate its caller or be listed with a reason; the server refuses to start with one that is neither.	ALLOW_POLICY_COVERAGE_GAP
Capability coverage	INV-218	Every route carrying View, Author or Operate must resolve to a capability feature or be on the exempt list; the server refuses to start otherwise.	ALLOW_CAPABILITY_COVERAGE_GAP
Installation-feature gating	-	Every public method of a controller tagged for installation features must carry a route decorator, so no route is silently un-gated; the server refuses to start otherwise.	no override
Delete-policy completeness	-	Refuses to start when the delete policy has drifted from the database schema, outside production and in production when the operator sets DELETE_POLICY_DRIFT_FATAL; otherwise it logs the drift as an error.	DELETE_POLICY_DRIFT_FATAL
Sign-in secret	-	The server refuses to start without a JWT secret, with one shorter than 32 characters, or with a known placeholder.	no override
Encryption keys	INV-31	The server refuses to start without the personal-data encryption key.	no override
Environment values	INV-229	The database address must be a postgres URL and every numeric setting must be a number when set.	no override
Production-process settings	INV-508	A production process refuses to start with plain ldap://, sandbox allowed, plain-text mail, a weak execution-callback secret, pipeline fault injection armed, or an unstated proxy count; plain-text mail is refused in production whatever the switch says.	SMTP_ALLOW_PLAINTEXT (outside production only)
Worker production settings	INV-508	A production worker refuses to start with a weak or placeholder worker API key or with plain-text mail.	no override
Database and cache transport	-	A connection to a database or cache on another server must be verified TLS; a production process is refused an unverified one across servers.	DATABASE_ALLOW_UNVERIFIED, REDIS_ALLOW_UNVERIFIED
SIEM settings	INV-555	Fail-closed with no usable provider, an unrecognised fail-closed value, or an unsupported retention is refused in every environment.	no override
AI provider address	INV-499	The provider address may be replaced only for a test lane, by an address on the same machine, and never in a production process.	no override
Environment divergence	INV-30	A diverged environment pointer is detected at every boot; it is a warning unless the operator sets FAIL_ON_ENV_DIVERGENCE, which makes the server refuse to start.	FAIL_ON_ENV_DIVERGENCE (turns the warning into a refusal)
Structural links of the knowledge graph	-	Every foreign key between graph node types is a registered edge or an explained exclusion; a gap is a warning unless the operator sets FAIL_ON_STRUCTURAL_FK_GAP, which makes the server refuse to start.	FAIL_ON_STRUCTURAL_FK_GAP (turns the warning into a refusal)

Guarantees left out or qualified

The testing runbook lists 483 current guarantees (and 14 retired ids, which are not counted). This appendix lists 251 of them. Of those, 38 carry a limit stated in the sentence and one has a statement left out. The other 232 are not listed, for the reasons in the second part of the table.

Kind	Count	Why
Listed with a limit stated in the same sentence	38	The runbook records a case the guarantee does not cover, or an open item contradicts it in one place; the limit is written beside it.
Listed with one statement left out	1	A statement about a fresh sign-in at the approval step is left out of the second-factor guarantee, because an accepted open item limits it.
Not listed: an accepted open item contradicts it	1	One guarantee about whose access an unapproved query runs with is not stated as held; the open item is described under Accepted for later.
Not listed: its runbook entry records a known residual	1	A guarantee about changeset isolation on the ontology's read surfaces: the entry records that the AI suggestion prompts still read live property names.
Not listed: no test file names it	4	Three concern the admin portal and one a schema-sync backup; they are not claimed here until a test names them.
Not listed: outside the security areas	226	Rules about pipeline run correctness, data-quality mechanics, lineage and impact display, model training, the ontology, scheduling and screens.

Security testing

Ziggy runs frequent audits of every route, setting, secret and outbound connection, including one before each release. Every serious finding is fixed and checked again before release. Each installation also includes a built-in security scan of 42 automated checks, which the operator can run at any time.

Ziggy has not yet had a penetration test by an independent third party. Customers are welcome to commission one of their own installation. We will support it and fix any finding within the times in our vulnerability policy.

Testing your own installation

Give us five working days' notice at info@ziggyplatform.com, or Freddy May directly at freddy.may@ziggyplatform.com, with the dates, the testing firm and the scope, so we can support the test. Test only your own installation. Our website, email, source repository and other customers' systems are out of scope. Share any findings with us in confidence, and we will handle them under our vulnerability policy.

How the audit works

The security overview, "How this was checked", describes a review of every route, socket, command, setting, secret-bearing field, encrypted field and outbound connection in the source code. That review was carried out by an AI model, not by a human or an independent party, and it reads the source; it does not attack a running system. It is not a penetration test and does not replace one.

In-house security scan (complementary self-check)

Ziggy's admin portal includes a built-in security scan. It is a self-check run by the operator of a deployment. It is not independent testing, and its results must not be read as the result of a penetration test.

What it does

- It runs 42 automated checks in seven categories: sign-in (10), multi-factor authentication (5), roles and permissions (5), input handling and injection (6), API protection such as rate limiting (6), encryption and data protection (5), and infrastructure and response headers (5).
- Each check ends as pass, fail, warn or skip, with a severity. The scan gives a weighted score and a letter grade.
- It never attacks the live system. The portal copies the live database into a temporary database, starts a temporary copy of the application with its own cache on the same server, runs the checks against that copy, and then removes everything it created. The test accounts it creates are fictional and exist only in the copy.
- Only a portal user with the administrator role can start it, and the start requires a fresh sign-in and is recorded in the portal audit log.
- Progress is shown live in the portal.

What it does not do

- It tests the main application only. It does not test the admin portal, the background worker, the web client, or the layer in front of the application (reverse proxy, TLS and redirects), and it does not cover every kind of attack. Examples outside its scope include server-side request forgery, file upload handling, cross-site request forgery, single sign-on flows, WebSocket access and abuse of business logic.
- Some of its checks inspect configuration or the shape of the database rather than performing the attack, and its grade is a summary of weighted results, not a statement that the system is free of weaknesses.
- Its reports are held in the portal's memory only. The last 20 are kept and they are lost when the portal restarts, so a scan result is not a retained record of assurance.
- It does not replace a test by a person who chooses attacks according to what they find.

How a customer can use it

Run it against the deployment before go-live and after significant changes, as one input among several. Treat a failed or warned check as a prompt to investigate. Keep a copy of any result you want as evidence, because the portal does not.

Related documents

- Security overview: "How this was checked", and "Accepted for later".
- Secure development: how Ziggy is built and changed, and the vulnerability disclosure and patch policy.
- Security assurance report: how security is examined.
- Secure configuration guide: the settings that affect the checks above, including the "Before go-live" checklist.