

Secure Development and Change Management Policy

How code is written, reviewed, tested and deployed, and how it is rolled back.

Version 1.1 · Last reviewed 22 October 2026 · Owner: Security Lead, security@etropo.com · Current version at <https://trust.etropo.com/resources/secure-development-policy>

1. Purpose

Every change to the Etropo application reaches production through the same path: a branch, a pull request, automated checks, a preview, a merge, an atomic deployment that can be undone. This policy describes that path and the practices around it.

2. Source control

- All application code lives in a private GitHub repository. Every change is a commit with an author and a message.
- Work happens on branches. `main` is the production branch; a merge to `main` deploys to production.
- Secrets are never committed. Environment files are ignored by Git, and secrets are held in Vercel environment variables.

3. Change types

Type	Examples	Path
Standard	Bug fixes, copy, dependency patches	Pull request, automated checks, merge
Normal	New features, database migrations, integration changes	Pull request, automated checks, preview deployment tested by hand, merge
Emergency	Security fixes, outages	Same checks, abbreviated review, post-implementation review within 24 hours

4. Automated checks

On every pull request and every push to `main`, GitHub Actions runs:

- **type checking** in TypeScript strict mode. A type error fails the build and blocks deployment;
- **linting** with ESLint;
- **unit and component tests** with coverage thresholds that may only go up;
- **integration tests** against a single confined test organisation in the production database, serialised by a lease so two runs never overlap.

Before a commit, a pre-commit hook lints and runs the tests related to the staged files. Before a push that touches database behaviour, a pre-push hook runs the integration suite.

5. Review

Review depth follows risk. Every change is opened as a pull request and gets automated review: the CI checks above, and an automated AI code review that the author runs on the branch before merge.

Changes to these areas also get a line-by-line human review before merge:

- authentication, sessions and two-factor;
- permissions, organisation membership and Row Level Security;
- billing;
- data access and export;
- database migrations;
- the AI pipeline: prompts, tools and the data they can reach.

A person approves every merge; nothing reaches production on automated review alone. The current GitHub plan cannot require a second reviewer, and the Controls page says so.

6. Security practices in code

- **Input validation.** API routes validate request bodies with typed schemas before any database access.
- **Database access.** Through the Supabase client with parameterised statements. No SQL is built from user input.
- **Authorisation.** Organisation membership is checked on every organisation-scoped request, and Row Level Security is the last line in the database.
- **Database changes.** Every schema change is a migration applied through the Supabase tooling and kept in Git. Changes that touch access rules are covered by conformance tests that fail the build if a second, weaker path appears.
- **Secrets and keys.** Read from the environment only. The token-encryption key is a 256-bit value that exists only in production.

- **Output.** Every response carries a Content Security Policy and standard security headers.
- **Third-party calls.** HTTPS only, with timeouts and retry limits. Webhooks are verified against their signing secrets.
- **AI tools.** Tools that write data are bound only where the caller's authorisation can be checked, and the surface of each chat platform is pinned by tests that fail when a tool is added.

7. Dependencies

- GitHub Dependabot alerts are enabled on the repository and reviewed during regular dependency maintenance.
- A major upgrade is adopted only when everything layered on it declares support.
- Vulnerabilities are patched within 48 hours (critical), 7 days (high) or 30 days (medium).

8. Environments

Environment	Where	Data	Access
Development	Developer laptop	Developer's own test organisation	Developer
Preview	Vercel, per branch	Production environment variables are not used for secrets that write	Vercel team login required
Production	Vercel fra1 + Supabase eu-central-1	Customer data	Public application; consoles per the Access Control Policy

9. Deployment and rollback

- A merge to `main` builds and deploys atomically. Users switch to the new version without downtime.
- The previous deployment stays available and can be promoted back in one click.
- Database migrations are written to be backward-compatible with the deployment before them where possible, so an application rollback does not require a database rollback.
- When a change cannot be rolled back by promotion (for example a migration with data loss), the pull request says so and names the recovery step.

10. Testing with production data

A single test organisation inside the production database is reserved for the integration suite. Tests may only create, change and delete data inside it, a guard fails any test that reaches outside it, and leaked fixtures are reported and cleaned at the end of each run. Developers do not copy customer data to their machines.

11. Review

Reviewed annually with the Information Security Policy. Version 1.1, October 2026: review depth now follows risk (§5).

This PDF is a snapshot. The page at <https://trust.etropo.com/resources/secure-development-policy> is the current version.