

Change management policy

Version	1.0
Effective date	27 October 2026
Owner	Robin Ashford
Applies to	Code, database schema, cloud configuration, secrets and vendor settings for Final Frame
Review	27 October 2027
MPA CSBP	TS-5.0, TS-1.1, TS-1.13, TS-1.15

Approved by Robin Ashford, Founder, 27 October 2026

What counts as a change

- A commit to the repository
- A database migration
- A deploy of an image to a Cloud Run service or job
- A change to Google Cloud configuration: IAM, buckets, keys, load balancer, logging, Cloud SQL
- Adding, rotating or removing a secret
- A setting changed in a vendor dashboard (WorkOS, Postmark, Resend, fal, DNS at Squarespace)

How a code change reaches customers

1. Written on a branch or on main on one of Robin's Macs.
2. Pushed to GitHub. CI runs the checks in the Secure development policy.
3. The commit stamps with the release and a line saying what changed.
4. Robin runs at origin/main. It refuses a dirty tree, anything other than origin/main, a release tag already built, and any deploy inside an open deploy freeze.
5. **Modular deploys.** The deploy works out which parts a release changed from the import graph and the release manifest, and ships only those. A path no rule recognises ships everything. A full deploy runs at least weekly.
6. **The deploy lock.** Every deploy, rollback and roll-forward holds one cross-machine lock, so two machines never ship at once.
7. **The release manifest.** Each part's release, commit and time are written to once that part is verified live. Every rollback and roll-forward is recorded there too.
8. **The canary.** A new API revision is deployed with no traffic, smoke-tested on its own address, and only then given traffic. A failed smoke leaves traffic, the jobs and the manifest where they were.
9. The deploy checks that the API and the worker serve the new release and /readyz answers.

Each stage of a modular deploy has its own switch. With a stage off, that part takes the full deploy path.

Deploy freezes

Inside it nothing deploys, publishes or releases. A spent error budget freezes risky changes as sets out. Rollback, roll-forward and --plan stay allowed. A fix during a freeze needs Robin's say-so.

CI and its budget

- CI runs on every push to main and every pull request that changes more than or brand/**. A newer push cancels the running check; the newest green run covers every commit before it.
- The security workflow (CodeQL, gitleaks, OSV) runs daily and on every pull request. The ZAP baseline runs weekly.
- GitHub Actions minutes are a budget: the plan's allowance plus a fixed monthly spending limit set by Robin. Commits that touch only docs, tickets, the plan or the ledger carry [skip ci]; small changes are batched into one push; no workflow gains a push trigger without Robin's approval; no run is repeated only to see it green.

Rules

1. **Everything through git.** No hot edits on a server.
2. **CI green before deploy.** The GitHub Actions run for the commit has passed before it deploys.
3. **Pull requests for risky changes.** Authentication, authorisation, RLS policies, migrations that touch Org-scoped tables, storage and key code, and anything in the AI use policy go through a pull request with the change described.
4. **Branch protection** on main: required CI, no force push, no deletion.
5. **Migrations** are forward only and run once, as, from a one-off job. A backfill on an Org-scoped table lifts FORCE for that transaction only. Destructive steps (drop, rename) ship in a later release than the code that stops using them.
6. **Secrets** are added with --update-secrets, never --set-secrets.
7. **Emergency changes** may skip rule 3. They still go through git and CI, and get a written note within one working day.
8. **Penetration test windows** are deploy freezes, and the tested build is recorded.

Record

The record of each change is the commit, its CI run, its line, the release manifest, and the Cloud Build and Cloud Run revision history. Admin Activity audit logs (400 days, locked) record cloud configuration changes.

Rollback

Each part rolls back on its own, in one command, inside the deploy lock, recorded in the manifest:

Part	Back
Site	Revert the commit and ship the site
Schema	A forward fix. Restore from backup only for data loss (see the Business continuity and disaster recovery)

A rollback is rehearsed once a quarter on QA.