PLATFORM · TEMPLATE

Platform Engineering for Regulated Startups: The *Complete* Guide

KAANSYSTEMS.COM/LIBRARY/PLATFORM-ENGINEERING-FOR-REGULATED-STARTUPS · SEPTEMBER 16, 2026

ABOUT THIS TEMPLATE

Platform engineering for a regulated startup is four jobs, not a team you hire. Foundation, delivery, reliability, control — what 'good enough' is for each, and the deep dive that builds it.

THE TEMPLATE

A platform foundation readiness checklist. Work top to bottom, one job at a time — a later job built on an incomplete earlier one is wasted money. Score each item Not started / In progress / Done.

Job 0 — Ownership

- [ ] One named senior owner for the platform (staff or fractional), not a committee
- [ ] Paved-road defaults are the documented path, not one team's private setup
- [ ] Platform work has a place on the roadmap, not just the gaps between features

Job 1 — Foundation (AWS account structure)

- [ ] Confirmed you've crossed a real landing-zone trigger (data / compliance / second env / team size)
- [ ] Multi-account landing zone stood up (min viable: management, log-archive, security, prod, non-prod)
- [ ] Log-archive account with Object Lock — logs are undeletable
- [ ] Build-vs-buy call made (Control Tower vs. Terraform) and recorded, not left open
- [ ] Production isolated by an account boundary, not a tag: a staging credential can't reach prod data

Job 2 — Delivery (how code ships)

- [ ] GitOps controller reconciling prod from git — no hand-run `kubectl apply`

- [ ] Git is the only write path to production; no standing human cluster-admin
- [ ] Controller choice (Argo CD vs. Flux) made and wired to SSO + RBAC
- [ ] Progressive delivery: SLO-gated canaries with automatic rollback
- [ ] Change history for any prod change is produceable without a screenshot tool

Job 3 — Reliability (does it stay up)

- [ ] Cluster scored against a written production-readiness checklist (not a vibe)
- [ ] Resource limits, autoscaling, and network policy in place
- [ ] Observability makes an incident debuggable by someone who didn't build it
- [ ] Backup + a *tested* restore, not just a backup job that's green
- [ ] First honest score recorded; the gap to the target is on the roadmap

Job 4 — Control (correct + least-privilege over time)

- [ ] Drift detection tiered by risk: dangerous drift pages, the rest is a digest
- [ ] Long-lived credential inventory taken; the count is a tracked KPI
- [ ] CI uses OIDC federation; pods use workload identity (IRSA); DBs use dynamic secrets
- [ ] No unexplained static keys in the IAM credential report
- [ ] Access is short-lived and federated by default

The compliance byproduct

- [ ] Account boundary doubles as the isolation control + its evidence
- [ ] Git history doubles as the change-management control + its evidence
- [ ] Identity plane doubles as the access control + its evidence
- [ ] Evidence collection is a byproduct of running the system, not a quarterly drill

Most teams find Job 1 is the unlock and Job 4 is where discipline pays off or quietly rots. Both are engineering problems more than compliance ones — which is the whole point.

WHAT ARE THE FOUR JOBS, AND WHAT DOES "GOOD ENOUGH" LOOK LIKE FOR EACH?

Here's the whole map on one page. Each job has an honest "good enough" bar for a regulated SMB — deliberately not the enterprise bar — and a deep dive that builds it. Work them top to bottom; a later job built on a missing earlier one is wasted money.

THE PLATFORM'S JOB	WHAT "GOOD ENOUGH" LOOKS LIKE FOR A REGULATED SMB	THE DEEP DIVE
1. Foundation — AWS account structure	A real account boundary isolates production; logs are tamper-resistant; one identity plane fronts access. A leaked staging credential can't reach prod data.	Building a multi-account landing zone
2. Delivery — how code ships	Git is the only write path to prod; deploys are reviewed and gated; rollback is automatic; the git log *is* your change-management evidence.	GitOps + progressive delivery
3. Reliability — does it stay up	The cluster clears a concrete production-readiness bar, not a vibe. Incidents are debuggable at 3 AM by someone who didn't build it.	What "production-ready" actually means
4. Control — does it stay correct + least-privilege	Dangerous drift pages; the rest is a weekly digest. No long-lived credentials; access is short-lived and federated.	Drift detection + killing long-lived credentials

The rest of this guide is one section per job.