FIELDHASH · CHANGE-BOUND APPROVALS · SIDE BY SIDE

Ticket-bound approvals, without a custom approval check.

An AI agent deploys to production with `kubectl apply`. A person approves each deploy against its change ticket. In scripted hook tests, separate from our live demo, we tested what happens next in 214 situations: tickets cancelled, closed, reassigned or rescheduled; manifests edited and put back; commands reordered, respelled or pointed somewhere new. Every test was written down before it ran.

190 = 190

A separately implemented ticket check matched FieldHash, with 1,200 lines of custom checking code. None was added for FieldHash.

88/88

deploys that needed a person went to one with FieldHash, and none of the 102 still-approved deploys was stopped.

1,926/1,926

FieldHash decisions in the runs carry a signed receipt that verifies offline. The checks leave log lines.

Approach	Right answers	Would proceed without needed review	Extra asks	Custom check code
FieldHash window enforcement on, preview build	**190**/190	0	0	none added
Separately implemented ticket check written from a plain spec by a separate AI agent	**190**/190	0	0	1,200 lines
Quick ticket check our own, minimal	**179**/190	0	**11**	111 lines
Expiring approval (modeled) exact command, 8 h	**129**/190	**50**	**11**	config
Standing allow rule (modeled) kubectl apply:*	**102**/190	**88**	0	1 rule

Tested configuration: FieldHash with window enforcement on, in a preview build that is off by default and not yet released. On default settings it missed the 5 situations where a change window simply ended. Change-system integration and operating effort are excluded for every approach.

How we got here. Round 1 used 14 hand-picked situations. FieldHash got 13 of 14: when a ticket's window was extended, it asked a person it didn't need to. We fixed that the same day, then wrote down round 2 before running it: the original 14 plus 200 more generated from a fixed seed. The fixed FieldHash got all 190 scored situations right.

What FieldHash gives a team

- **The approval is the binding.** The reviewer names the ticket when they say yes. There is no approvals file or mapping table to keep.
- **Exact deploys, read the way kubectl reads them.** An edited manifest, another context or namespace goes back to a person. Flags in another order are still the same deploy.
- **Only that approval goes back.** Cancel one ticket and only the deploys that relied on it return to a person.
- **Evidence an auditor can check.** A signed receipt for every decision, with no ticket ids or file names in it.

When it is a judgment call

- We report 24 situations separately, without scoring, because reasonable teams differ: a ticket cancelled and reopened, an approver changed and changed back, a whitespace-only edit.
- **FieldHash asked a person in all 24.** If a ticket condition the approval relied on changed since the yes, even if it changed back, a person sees it. A manifest restored exactly is the same deploy, so its approval applies again.
- The separately implemented check let 12 of them run. The quick check let 8 run.

How this was run: 2026-11-02. Each round was pre-registered in a commit before it ran (round 1: 417c29c; round 2: d0811dd), with 3 logged deviations, all before any result of the round they touch. FieldHash and both checks ran as real hook processes; FieldHash on its real routes and CLI, a fresh deployment per case, from a prototype build (change-bound approvals are off by default, not yet released). The round 2 fix was designed after seeing round 1. The 200 generated situations come from one seed and one grammar written by the team that builds FieldHash. The separately implemented check was written by a separate AI agent from a plain spec, without FieldHash's code, the cases or any result. The expiring approval and allow rule are modeled from their rules. kubectl was a stand-in and the change system a JSON file; FieldHash and the checks all need the change system wired in, which no line count includes. Every situation and every answer: fieldhash.ai/agent-deploys.