THE LINUX FOUNDATION
OPEN SOURCE SUMMIT KOREA

서울

GitAIOps

A 4-Layer Architecture for Predictable AI-Assisted Operations

Hoon Jo CNCF & AAIF Ambassador | AI & Cloud Native Engineer, Megazone

Open Source Summit Korea 2026 | Aug 11 | Open AI + Data

Who is telling you this?

Hoon Jo

CNCF & AAIF Ambassador

advocating open source across cloud native and agentic AI

Author

wrote the book behind today's demo, and four more (AI, k8s, Ansible, Python)

AI & Cloud Native Engineer

ran the production migration in this talk, with one AI agent

Your AI agent has amnesia

Forgetting yesterday's decisions, values, and lessons: gone by the next session

Guesswork so it fills the gaps: re-decides settled choices, invents commands and flags

Unpredictability so the same request gives a different result: production cannot run on that

Every session starts from zero. The infrastructure does not.

서울

Git "AI" Ops

GitAIOps the AI joins GitOps with what it never had: a memory that lasts

The test bed: a real production migration

15
Helm releases

Kafka
ZooKeeper to KRaft

Valkey
replacing Redis

Rebuilt
observability stack

Constraint: the old platform keeps sending messages 24/7. No downtime allowed.

One AI agent, one operator, and a deadline.

Layer 1 work-plans: plans humans read

36 files
23,854 lines

Why + How + trade-offs
every component decision written down

Migration plan and runbooks
infrastructure, platform, data, transition

WHAT BROKE Hand all of this to the AI as-is, and it drowns in context.

Layer 2 claude-context: distilled for the AI

6 files
1,254 lines (19:1)

A project state dashboard
summary, environment values, execution guide

Textbook vs exam notes
work-plans are the textbook, this is the summary

WHAT BROKE The AI understood the state, then improvised its own commands.

Layer 3 command-guardrails: control the hands

117 files
step-by-step commands

Execution order enforced
00-prerequisites through 40-transition, then rollback

Old platform protected
read-only rules: get, describe, logs only

WHAT BROKE Right command, creative flags. Versions and values still drifted.

Layer 4 locked values: remove interpretation

30 files
versions and values pinned

Real incident
unpinned Mimir chart auto-upgraded 5.8.0 to 6.0.5, CrashLoop

Everything explicit
version, values file, namespace, timeout: all fixed

RESULT Same input, same deployment. Nothing left to improvise.

Four layers, four freedoms removed

Layer 1 work-plans	what to build and why	**removes missing context**
Layer 2 claude-context	current state, distilled	**removes stale memory**
Layer 3 command-guardrails	what to run, in order	**removes improvised steps**
Layer 4 locked values	exact versions and values	**removes interpretation**

Predictability is what remains when the freedoms are gone.

What predictability bought us

2 weeks → 2 days
full DEV environment build

1 week → 1 day
PROD build, guardrails reused

Zero downtime
old platform never stopped

The speed did not come from the AI typing faster.

It came from never having to redo an unpredictable step.

서울

Live Demo: Ask the Repository

The production repo? I cannot show you

Private

the migration repo stays behind NDA.
Numbers only.

Rebuilt in public

a book and notiflex-platform: the same pattern, end to end.

gitaiops/
notiflex-platform

LIVE DEMO let's ask the repository three questions

1. Why Alertmanager over Grafana Alerting?
2. Onboard a new engineer.
3. Ask anything: your turn.

gitaiops/
notiflex-platform

Why Alertmanager over Grafana Alerting?

the repository answers from 16 recorded decisions

gitaiops/
notiflex-platform

Onboard a new engineer

an onboarding guide, generated from the repository

gitaiops/
notiflex-platform

Ask anything

your turn: the repository will answer

What you just saw

The repo answered why 16 recorded decisions, alternatives included: no asking around

An onboarding guide, generated derived from the repo. Nobody ever wrote it by hand

Your question, answered live, from the same recorded memory: nothing staged for it

Your repository became the single source of truth. Nobody wrote a wiki.

Make it yours: start small

1. **Start now: record as you go** rules and current state, in whatever file your agent reads

2. **Plan first when the work is big** a migration moves settled decisions: plan it, pin the values

3. **Keep the single source of truth alive** everyone updates it, and the AI keeps it readable

Works with any coding agent. The architecture lives in Git, not in the tool.

The migration ends. The memory keeps working.

서울

Recap

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

GitAIOps in three sentences

Git

Git is the memory
every plan, command, and value lives in the repository

AI

AI is the operator
guided by a 4-layer architecture: context, guardrails, pinned values

Ops

Ops is predictable
AI-assisted operations from a single source of truth

Thank you!

GitAIOps on GitHub
github.com/gitaiops

Hands-on with the book
github.com/gitaiops/_Book_GitAIOps

Hoon Jo
CNCF & AAIF Ambassador

UPCOMING Aug 13
AAIF Ambassador AMA
MCP Dev Summit Seoul, same venue
Open Q&A: agentic AI in production
sched.co/2QScw