# Document 1 — DevOps with Conjugate Intelligence (CI)

## Executive Summary

DevOps today excels at automation but loses **context** across tools. CI restores context as a first-class asset. Pipelines, changes, and incidents become **pearls** in an EKR graph with explicit intent, assumptions, and evidence. The result is safer, faster delivery with lower toil.

## Problems Today

- Brittle pipelines; behavior changes per branch/env without clear rationale.
- Context loss in handoffs (Product → Dev → Ops); decisions scattered across chats and tickets.
- Dashboards show **what** happened; teams still hunt for **why** it happened.
- Toil: repetitive diagnostics and release rituals that could be learned once and reused.

## CI Approach (SpiralOS)

- **EKR Graph as Ground Truth:** Commits, tests, releases, incidents, runbooks, and approvals are linked as semantic nodes with versioned history.
- **Vows (Intent Binding):** Each change/release carries explicit goals, risk posture, and guardrails that travel with the artifact.
- **Conjugation:** The system's intended state (self) is continuously compared to live telemetry (other) to drive meaningful actions.

## Key Capabilities

- **Context-Preserving Pipelines:** Pipelines adapt to branch risk, error budgets, and business criticality; rationale is recorded.
- **Adaptive Guardrails:** Pre/post-conditions, safe rollout recipes, and rollback contracts are attached to releases as pearls.
- **Change Risk Scoring:** CI scores a change using code diff features, test coverage, blast radius, and recent incidents.
- **ChatOps with Memory:** Conversational commands operate with full context and produce auditable narratives.
- **Holarchic Replay:** Any change/incident can be replayed: inputs → decisions → effects → outcomes.

## Outcomes & KPIs

- **Lead Time for Changes** ↓ 20–40%
- **Change Failure Rate** ↓ 25–50%
- **MTTR** ↓ 20–35%
- **Toil Hours** ↓ 30–60%
- **Deployment Frequency** ↑ with confidence (guardrailed)

### Integration Path (Low-Friction)

1. **Observer Mode (Week 1–2):** Ingest Git repos, CI/CD logs (e.g., Actions/Jenkins), artifact metadata, and incident tickets. No write access.
2. **Pearl Templates (Week 2–3):** Change/Release/Incident templates with minimal fields: intent, risk, guardrails, evidence.
3. **Guided Actions (Week 3–5):** CI suggests rollout plans, test gates, and safe rollback steps; human-in-the-loop.
4. **Selective Automations (Week 5+):** Approve auto-rollouts/rollbacks under thresholds; all actions narrated to EKR.

### Risks & Mitigations

- **Over-automation:** Thresholded, opt-in automations; explicit approvals (vows) required.
- **Vendor lock-in:** Open graph model, import/export APIs, and human-readable narratives.
- **Model drift:** Continuous evaluation using post-deploy outcomes and incident feedback.

### Example Walkthrough (Hotfix)

A critical fix to Service X touches a high-risk module. CI scores the change, proposes a **canary + shadow traffic** plan, sets SLO-aware abort criteria, and prepares a **one-click rollback contract** . During rollout, latency rises; CI executes rollback, annotates graphs with evidence, and generates a post-release narrative and follow-ups.

### Sector Examples

- **Manufacturing:** Adaptive pipelines ensure MES updates roll out without halting production lines.
- **Mobility:** OTA updates to vehicle fleets are canary-tested against safety guardrails.
- **Energy:** Grid control software patches carry explicit risk vows, linked to SLOs on uptime and safety.

### Talking Points (for Erich & Echo)

- “CI doesn’t replace your pipelines — it **binds** them to intent and evidence.”

- “Your dashboards become **stories** you can replay, not screenshots you forget.”
- “Risk is explicit and portable — it travels with the artifact.”