# 140509\_47.md — Intelligent DevOps Automation Platform

**Theme:** AI in Software Engineering Lifecycle, AI in IT Ops  
**Mission:** Automate CI/CD with ML-driven risk prediction, safe rollouts/rollbacks, infra optimization, and incident automation to maximize delivery velocity and reliability at lower cost.

## README (Problem Statement)

**Summary:** Build an AI-powered DevOps platform that automates deployment pipelines, predicts failures, and optimizes infrastructure resources.  
**Problem Statement:** DevOps workflows are complex and error-prone. The platform must learn from historical deployments to predict risk, automate rollbacks, optimize resources, integrate monitoring, and assist with incident management and RCA.

**Steps:**  
- Intelligent CI/CD automation with failure prediction  
- Deployment risk assessment + automated rollback  
- Infrastructure optimization for cost/performance  
- Monitoring integration for proactive detection  
- Capacity planning + auto-scaling recommendations  
- Incident management automation + RCA

**Suggested Data:** CI/CD logs, deployment histories, test results, infra metrics, incident tickets, SLO/SLI configs.

## 1) Vision, Scope, KPIs

**Vision:** Self-driving delivery pipelines that ship faster with fewer incidents and optimal spend.

**Scope:**  
- v1: CI/CD integration, risk predictor, gated deploys, canary + auto-rollback.  
- v2: Infra right-sizing, predictive autoscale, incident automation + RCA.  
- v3: Cross-service dependency modeling, capacity planning, chaos experiments.

**North-star KPIs:**  
- Failed deployments ↓ **40%**  
- MTTR ↓ **50%**  
- Auto-rollback success ≥ **95%**  
- Infra cost savings ≥ **20%**  
- Lead time for changes ↓ **30%**

## 2) Personas & User Stories

* **DevOps Engineer:** “Block risky releases and roll back safely.”
* **SRE:** “Detect incipient SLO violations and scale ahead of load.”
* **Developer:** “Get actionable feedback in PRs and during deploys.”
* **Engineering Manager:** “See delivery, reliability, and cost KPIs.”

**Representative Stories:**  
- US‑01: Predict risk for a pipeline run before prod deploy; gate if risk > 0.7.  
- US‑07: Trigger automatic rollback if error rate > SLO for 3 mins during canary.  
- US‑12: Recommend instance type change when CPU<30% and mem<35% for 2h.  
- US‑15: Auto-create incident with suspected root cause from logs and metrics.

## 3) PRD (Capabilities)

1. **Risk-aware CI/CD:** ML model predicts deploy risk from code/test/infra signals; policy gates.
2. **Safe Rollouts:** Canary/blue-green with guardrails; auto-rollback on SLO breach.
3. **Infra Optimizer:** Right-size compute, reserve/spot selection, placement hints.
4. **Observability Integration:** Metrics/logs/traces for pre/post deploy verification.
5. **Incident Automation:** RCA summarization, ticketing, comms, and runbooks.
6. **Capacity Planning:** Forecast demand; autoscale recommendations.

## 4) FRD (Functional Requirements)

* **CI/CD Integrations:** GitHub Actions, GitLab CI, Jenkins, ArgoCD, Spinnaker.
* **Feature Extractors:** change size, churn, test pass rate, flaky tests, dependency diffs, perf deltas, owner reputation, prior incident proximity.
* **Risk Model:** Gradient boosting/transformers; calibrated probabilities; per-signal SHAP explanations.
* **Release Strategies:** canary % ladder, blue/green, traffic mirroring; policy DSL for guardrails.
* **Rollback Engine:** SLO monitors (latency P95, error rate, saturation). Threshold + hysteresis.
* **Infra Optimization:** recommend instance types/node pools; spot eligibility; container requests/limits tuning; GPU bin-packing.
* **AIOps:** log template mining, metric change-point detection, causal graph hints; RCA summary.
* **Capacity Planner:** ARIMA/Prophet/Temporal Fusion Transformers; SLO-aware headroom targets.
* **Dashboards:** DORA metrics, cost/perf KPIs, risk heatmaps, optimization backlog.
* **APIs:** REST/GraphQL; webhooks for gates and rollbacks.

## 5) NFRD (Non-Functional)

* **Availability:** 99.95%
* **Latency:** risk inference < 5 s per pipeline; rollback trigger decision < 10 s.
* **Scale:** 10k+ pipeline runs/day; 100k metrics/sec ingest.
* **Security:** OIDC SSO; least-privilege deploy tokens; signed artifacts.
* **Compliance:** SOX-friendly approvals, auditable change logs.

## 6) Architecture (Logical)

[SCM & CI/CD] -> [Event Collector] -> [Feature Service] -> [Risk Model]  
 | |  
 v v  
 [Policy/Gates] [Deploy Orchestrator]  
 |   
 [Canary Controller] [Rollback Engine]  
 |  
 [Obs Integrations: Metrics/Logs/Traces]  
 |  
 [Infra Optimizer] [Incident/RCA Automation]  
 |  
 [Dashboards & APIs]

## 7) HLD (Key Components)

* **Event Collector:** subscribes to CI/CD webhooks; normalizes to common schema.
* **Feature Service:** recent test outcomes, diff stats, ownership, service health; cached for low latency.
* **Risk Service:** model server (LightGBM/XGBoost) with calibration; SHAP server for explainability.
* **Canary Controller:** progressive traffic shifts; reads SLOs from Prometheus; halts/rolls back.
* **Rollback Engine:** maintains rollback graph to last-known-good; tracks blast radius and dependencies.
* **Infra Optimizer:** Prometheus metrics; rules + ML to suggest rightsizing/placement; integrates with cluster autoscaler.
* **AIOps/RCA:** log clustering (Drain3), change-point detection (Bayesian/ADWIN), causal hints (PCMCI/Granger).
* **Planner:** demand forecasts → HPA/VPA recommendations; multi-objective optimization (cost x SLO).
* **Dashboards:** DORA, SLO heatmaps, risk trends, optimization actions.

## 8) LLD (Selected)

**Policy DSL (guardrails):**

rule "block\_high\_risk\_prod":  
 when env == "prod" and risk\_score > 0.7 then block  
  
rule "auto\_rollback":  
 when canary.error\_rate > 2x\_baseline for 3m then rollback

**Canary Ladder:** 5%→25%→50%→100%, promotion requires: error\_rate Δ<0.5%, P95 latency Δ<5%, saturation stable.

**Rightsizing Heuristic:** If avg CPU<30% & mem<35% for 2h → reduce requests by 20%; else if CPU>80% for 15m → increase 15%.

**RCA Summary Template:** Service X error spike after deploy Y; suspect change: handler Z; correlated metric: DB latency; probable cause: index miss.

## 9) Pseudocode (Deploy Flow)

on\_pipeline\_complete(run):  
 feats = feature\_service.extract(run)  
 p = risk\_model.predict(feats)  
 if p > 0.7: gate(block, explain=shap(feats))  
 else: start\_canary(run)  
  
on\_canary\_tick(metrics):  
 if violates\_slo(metrics):  
 rollback()  
 create\_incident(metrics, run)  
 else if ladder\_ready(metrics):  
 promote\_canary()

## 10) Data & Evaluation

* **Training Data:** historical CI/CD runs, test outcomes, deploy results, incidents; label failures/rollbacks.
* **Offline Metrics:** ROC-AUC, PR-AUC, Brier score; feature ablation; SHAP stability.
* **Online:** reduced failed deploys, rollback time, change failure rate; A/B rollout across services.

## 11) Security & Governance

* Signed artifacts (Sigstore/cosign), SBOMs; approvals recorded; least-privilege deployer.
* Immutable audit trail of risk decisions and rollbacks.

## 12) Observability & Cost

* **Metrics:** risk distribution, gate rates, canary stop rates, MTTR, infra $/req, rightsizing savings.
* **Tracing:** OpenTelemetry spans for deploy stages and rollback actions.
* **FinOps:** spot recommendations, idle cluster detection, GPU bin-packing.

## 13) Roadmap

* **M1 (4w):** CI/CD events → risk model baseline → gate in staging.
* **M2 (8w):** Canary controller + auto-rollback; dashboards.
* **M3 (12w):** Infra optimizer + AIOps RCA + ticketing integration.
* **M4 (16w):** Capacity planner + cross-service dependency modeling + chaos drills.

## 14) Risks & Mitigations

* **False positives blocking deploys:** explainable thresholds, override w/ approval.
* **Rollback loops:** cooldown windows & circuit breakers.
* **Model drift:** periodic retrain; feature monitoring.
* **Cultural adoption:** phased rollout, clear KPIs & success stories.