NOFire.ai

● RECOGNISED · GARTNER MARKET GUIDE FOR AI SRE TOOLING · 2026

The AI Reliability Guide.

For engineering leaders introducing AI agents into production. Patterns, failure modes, governance, and the runtime architecture that makes autonomy safe.

5% today

Enterprise AI SRE adoption

Gartner expects 85% by 2029. Not gradual — driven by necessity.

95%

Enterprise AI pilots fail

MIT NANDA · 300+ deployments. Not a model problem — a production integration problem.

89%

Causal top-1 accuracy

vs 42% SOTA probabilistic — RCAEval · ACM 2025 · N=735 fault-injection scenarios.

FORMAT · REFERENCE GUIDE

AUDIENCE · VP ENG / CTO / CISO / PLATFORM

EDITION · 2026 SPRING

7 CHAPTERS

EXECUTIVE SUMMARY

The operational stack wasn't built for agents.

By 2029, AI agents will commit code, deploy services, and remediate incidents without human review. Most teams introducing agents today are doing so against an operational stack designed for a different premise — a human reading a dashboard, taking the action, learning from the outcome. The premise is shifting. The stack hasn't caught up.

The Gartner Market Guide for AI SRE Tooling (2026) identifies the core limitation: tools optimized for incident response make teams *better at reactively fixing incidents but not at improving system reliability*. The work is moving left — into the change, the deployment, the dependency graph — before the alert fires.

MIT NANDA's analysis of 300+ enterprise AI deployments found 95% fail to reach production. The cause isn't model capability. It's the absence of a runtime layer that can give agents context, enforce policy, and learn from outcomes. This guide is the architecture for that layer.

5% → 85%

enterprise AI SRE adoption · 2025 → 2029

Gartner's projected adoption curve. The shift is demand-driven, not incremental — driven by the math no longer working at scale with human-only SRE.

3

predictable failure boundaries

Context · Action · Memory. Every team we work with hits them in that order. The architecture in this guide addresses all three.

12

policy primitives

The minimum policy surface for safe agent execution in production: from blast radius to kill switch to identity binding. All described in Chapter 05.

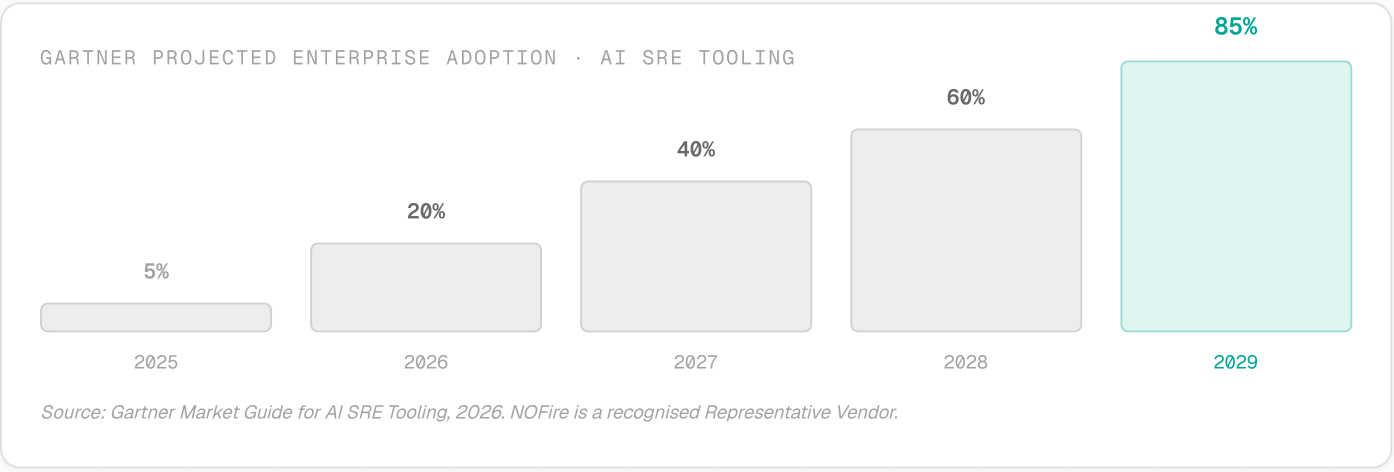
CHAPTERS

CHAPTER 01

The agent-era setup.

By 2029, AI agents will commit code, deploy services, and remediate incidents without human review. Most teams introducing agents today are doing so against an operational stack designed for a different premise — a human reading a dashboard, taking the action, learning from the outcome. The premise is shifting. The stack hasn't caught up.

This guide is for the engineering leader sitting at that exact gap — asked to make agents safe in production while the tooling underneath is still optimized for the previous decade.



Source: Gartner Market Guide for AI SRE Tooling, 2026. NOFire is a recognised Representative Vendor.

WHO THIS IS FOR

VP ENGINEERING / HEAD OF PLATFORM

Owns production reliability while introducing agents to deploy and remediate.

The stack they built assumed human review. That assumption is expiring.

CTO

Evaluating whether the org can scale AI in ops without trading reliability for autonomy.

The 95% enterprise AI failure rate is their risk exposure.

CISO

Blast radius now includes machine-initiated changes, not only human ones.

Identity binding and signed audit are the new perimeter questions.

SENIOR SRE / PLATFORM ENGINEER

Building the runtime substrate this all sits on.

The 12 policy primitives in Chapter 05 are theirs to implement.

CHAPTER 02

Where reliability breaks.

Production reliability fails in agent-driven environments at three predictable boundaries. We see them, in this order, in every team we work with. Each is architectural — not a configuration gap, not a model quality issue.

B1

The context boundary.

Agents reason from the telemetry they can retrieve, and telemetry returns snapshots. Production behaves causally — A caused B caused C. The boundary between snapshot and causal chain is where agents make confident wrong decisions. Correlation engines surface adjacent symptoms. They don't traverse the chain that produced them.

CONFIDENT WRONG DECISIONS

B2

The action boundary.

Once an agent decides, the runtime does not enforce. Most platforms allow the action to proceed and audit it after. **Audit is not enforcement.** Enforcement is the policy refusing the action at execution time, with the predicted blast radius attached. Without enforcement, autonomy is risk without constraint.

UNCONSTRAINED EXECUTION

B3

The memory boundary.

Every incident teaches something, then walks out the door. Postmortems are filed; the next on-call rediscovers the same root cause six months later. Agents inherit none of it, because the runtime doesn't remember. Without a memory layer, every agent starts from zero — repeating failures the team already solved.

REPEATED FAILURE PATTERNS

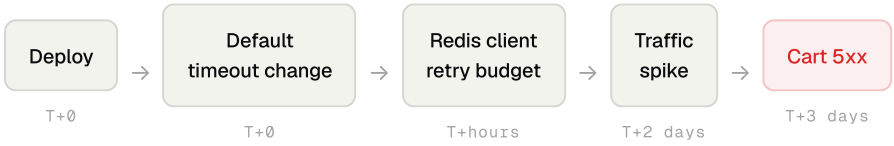
> *"We had three identical incidents in eighteen months. Three different humans solved them. Three different postmortems. The pattern was sitting in our data the whole time. We just didn't have a layer that could read it."*
>
> HEAD OF PLATFORM · 2026 CUSTOMER CALL

CHAPTER 03

Causal, not probabilistic.

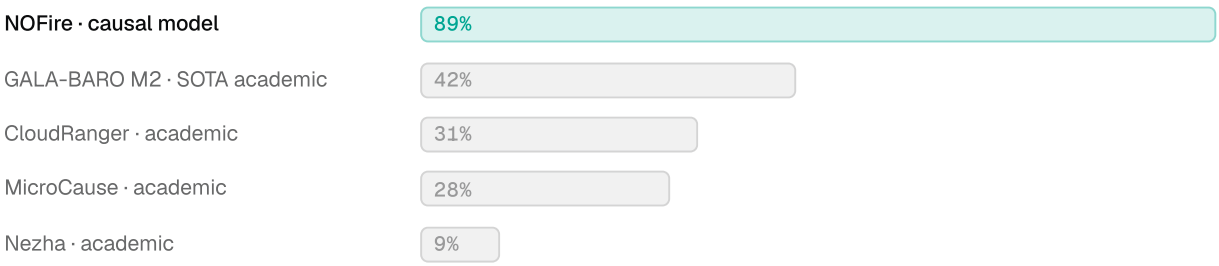
The dominant paradigm for AI in operations today is correlation: an LLM with retrieval, ranking, and a polished surface. It works for adjacent symptoms. It breaks on causal chains. Production runs on chains.

A REAL CHAIN — WHAT CORRELATION ENGINES DON'T SURFACE



No correlation engine surfaces this chain. The cause-and-effect path has to be modeled, not searched.

TOP-1 ROOT CAUSE ACCURACY · RCAEVAL · ACM WEB CONFERENCE 2025 · N=735 FAULT-INJECTION SCENARIOS



Probabilistic systems cluster between 9% and 42%. The causal model crosses 89% — 2.1× the best academic baseline.

WHAT "CAUSAL" MEANS HERE — TECHNICALLY

Typed nodes representing services, deploys, configs, dependencies. Edges representing confirmed causal relations. A time dimension that lets you replay any timestamp. **Root cause analysis becomes graph traversal, not metric search.** It is not an LLM producing a narrative — it is a structured model producing a path.

CHAPTER 04

The Context & Control Model.

The Context & Control Model is a live model of how production behaves — distinct from the telemetry it draws on. Telemetry stores observations. The runtime model stores the structure that produced them.

WHAT LIVES IN THE LAYER

- **Services, dependencies, and their declared contracts.** Not the metrics — the structure. Which services call which, under what conditions, with what guarantees.
- **Deploys: diff, predicted blast radius, actual outcome.** Every deploy is linked to the services it touched and compared against its prediction after execution.
- **Configs and configmaps, time-versioned.** A config change is a causal event. The model tracks it with the same precision as a code deploy.
- **Incidents, their resolved chains, the policies they generated.** Every investigation feeds back into the model. The next agent inherits the resolution.
- **Agent actions — proposed, evaluated, executed, signed.** Not just logged. Bound to a policy verdict, with a predicted impact and a post-execution health check.

HOW AGENTS USE IT

STEP 1 · READ

Agent reads the runtime model before proposing any action. It receives the predicted impact across the dependency graph.

→

STEP 2 · EVALUATE

Policy evaluates the proposed action against the runtime model. Blast radius, schema impact, network boundary, historical pattern.

→

STEP 3 · PROCEED OR REFUSE

Action executes within policy bounds, routes to human review, or is refused — based on the model, not retrieval.

GARTNER

The 2026 Gartner Market Guide notes: **"By 2030, 60% of new infrastructure designs will be validated by AI using historical failure data before development begins."** The Context & Control Model is the mechanism that makes that validation possible — it holds the historical failure data as a structured causal graph, not a document archive.

CHAPTER 05

Runtime enforcement.

Enforcement is the difference between a copilot and a control layer. A copilot suggests; the human decides. A control layer constrains; the action either falls within bounds or does not execute. Both belong in production. Autonomy needs the second.

COPILOT

Suggests an action. Human reviews and approves. Audit happens after. Good for assisted workflows. Not sufficient for autonomous execution at scale.

CONTROL LAYER

Policy evaluates at execution time. The action proceeds within bounds or is refused — with the predicted blast radius, the policy verdict, and the signed audit record attached.

THE 12 POLICY PRIMITIVES · MINIMUM SURFACE FOR SAFE AGENT EXECUTION

P1
Blast radius bounded
Predicted impact stays under a declared percentile of traffic before execution proceeds.

P2
Schema drift refused
Destructive migrations require explicit human sign-off. Not config — hard gate.

P3
Network boundary
Actions cannot cross declared trust zones without policy approval.

P4
Sandbox grants
Write access is scoped, time-bound, and revocable. No ambient authority.

P5
Signed audit
Every executed action is hash-signed with the policy verdict and model identity attached.

P6
Reversible by default
Actions auto-revert if post-execution health fails within the observation window.

P7
Memory write
Outcomes — successful or not — flow back to the runtime model. The next agent inherits.

P8
Rate limiting
Agent action throughput capped per service per window. Prevents runaway loops.

P9
Quorum gates
High-severity actions require N independent approvals — human or agent.

P10
Identity binding
Every action carries a signed identity, including which model version produced it.

P11
Drift detection
Running agent behavior compared to declared behavior; deviation raises alarm.

P12
Kill switch
Global pause, scoped pause, dry-run mode. One flag. Works at 3 AM.

CHAPTER 06

Governance & sovereignty.

Agent governance is operational reliability rebranded for legal and security audiences. The same enforcement primitives that keep production stable produce the audit trail compliance needs.

The architecture is the security argument: the Context & Control Model runs on your infrastructure, against models you already control — Bedrock, Azure OpenAI, Vertex; private VPC; single-tenant by default. Nothing about agent reasoning needs to leave your perimeter.

WHAT AUDITORS ACTUALLY LOOK AT

- **Identity binding** on every machine-initiated change — which model, which version, which policy scope.
- **Signed audit log** with policy verdicts attached — not just action outcomes, but the decision record.
- **Reversibility evidence** — that an executed action can be reversed, and was reversed when health failed.
- **Quorum policy** for actions that cross the human-review threshold.
- **Memory boundary** — what the runtime model retains, who can read it, how it expires.

CHAPTER 07

Adoption playbook.

Teams who adopt agent-driven operations without losing reliability follow a consistent four-step pattern — with a measurable gate before each promotion.

STEP 1 · READ-ONLY

Connect the runtime model to telemetry. Agents in advisory mode — they propose, humans execute. Calibrate until chain match and time-to-first-hypothesis are stable.

STEP 2 · CONSTRAINED EXECUTION

Allow execution within tight bounds — reversible actions only, blast radius under a low percentile, sandbox grants with expiration. Measure policy refusal rate. A runtime that never refuses isn't doing its job.

STEP 3 · FULL AUTONOMY ON GREEN PATH

Expand policy bounds for services with sufficient memory in the runtime model. The model knows which services have predictable failure modes — those get autonomy first. Untested services stay constrained.

STEP 4 · CONTINUOUS CALIBRATION

Every executed action calibrates predictions. Treat it as a quarterly engineering review — same discipline as a load test or security audit. Runtime models that don't recalibrate decay.

NOFire.ai

Make autonomy safe in your own stack.

A 30-minute call with a founder. We map your runtime model, replay an incident, and leave you with a written gap analysis — where your stack is, what the context boundary looks like today, and what changes first.

Book a demo → nofire.ai/book-a-demo