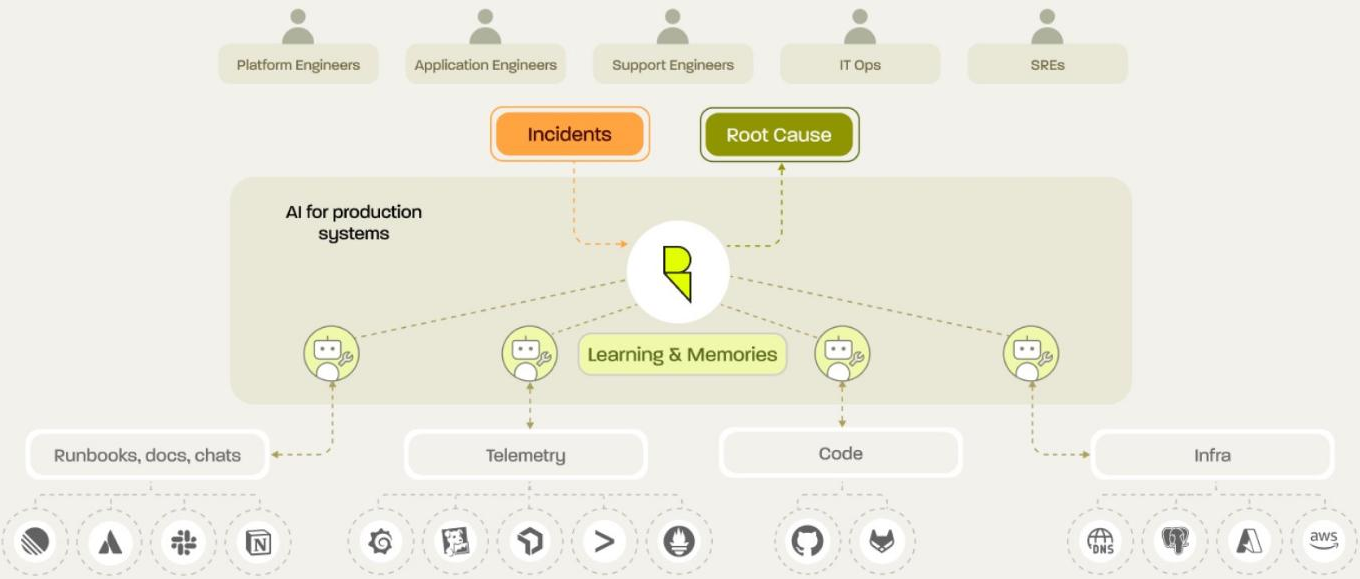
A PRACTICAL GUIDE

Adopting AI agents in engineering, beyond code generation

4 workflows where leading engineering organizations are deploying AI agents



Operating production with AI remains an unsolved problem

AI is changing software engineering faster than any previous technology shift. Models and coding agents have made writing code faster than ever. But software engineering is much more than writing code. While most AI investments are centered around code generation, 70% of engineering time goes into running production. Workflows like triaging alerts, investigating incidents, debugging systems, and hardening operations consume the majority of this time.

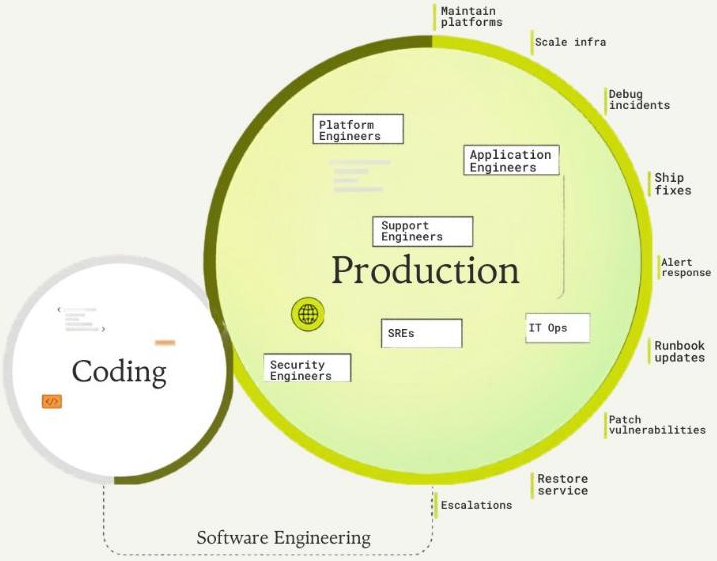
> **"*We're scaling code generation exponentially without scaling our ability to operate it.*"**

Production is an amalgamation of hundreds of tools and systems accumulated over generations of technology. Each component is built in isolation, optimizing for its own domain without integrating with the rest. But production workflows require reasoning across multiple of them, and this burden has always fallen on engineers.

When a service goes down, engineers need to understand which of hundreds of microservices is failing, trace dependencies across systems they may not own, and determine if it's an infrastructure or application problem. This requires collecting information from multiple systems and coordinating across teams while customers are impacted and the business is losing money.

◆ THE CORE GAP

70% of engineering time goes into running production. Most AI investment targets the other 30%.



◆ ONLY A FEW ENGINEERS

In every org, a small number of engineers carry the context to reason across the whole system. They get pulled into every production workflow.

For most teams, only a few engineers carry the context to reason across the whole system, and they are constantly pulled in for every production workflow. AI agents are now capable of operating production across systems, building context from every interaction, and getting smarter about your specific environment over time.

This paper is a practical framework for applying AI to four workflows: alert triage, incident investigation, production debugging, and operational reviews.

4 production workflows you can operate with AI agents

The following four workflows cover where most production engineering time goes. For each one, we've mapped what agents should handle and how engineers should work with them. Every team's situation is different, and where you stand on the continuum will depend on your current maturity and where the pain is most acute.

01 Alert Triage

Automating context assembly before an engineer opens a page. Classification, blast radius, and pattern matching against past incidents.

02 Incident Investigation

Parallel hypothesis pursuit across domains. Correlated signals, causal path tracing, root cause with grounded evidence across all systems simultaneously.

03 Production Debugging

Querying the right systems in sequence, interpreting results in context, building institutional memory from every session so knowledge is available to every engineer.

04 Operational Reviews

Post-mortems, runbook updates, and pattern detection as default outputs of every investigation — not scheduled tasks assigned after the fact.

This paper is organized around these four workflows. For each one, the framework covers what agents should handle autonomously, how engineers work alongside them, and what good adoption looks like over time. The final section covers where most AI approaches break before they scale, and the infrastructure required to make any of this work.

01 AI that participates in every on-call rotation

Automating context assembly before an engineer opens a page

On-call engineers are the first line of defense between a production issue and customer impact. When an alert fires, the engineer has to figure out if this is real, how bad it is, and what to do about it. Answering those three questions requires pulling context from five different systems before the investigation can even begin.

Current production environments generate hundreds of alerts every day. The vast majority are symptoms of other issues, known flakiness, or misconfigured thresholds. Buried in that noise are signals that matter, and missing the wrong alert has consequences. So engineers respond to every page while spending significant time gathering context to make a triage decision.

THE COLD-START PROBLEM

Every alert is a cold start. The on-call engineer may not own the service or have a runbook for this exact pattern. Context assembly before page-open changes the nature of on-call work entirely.

Every alert is a cold start. The engineer on call may not own the service or have a runbook for the pattern that appeared last time. Assembling context manually under pressure is exactly the kind of work AI agents are well suited for.

WHAT AI AGENTS HANDLE	HOW ENGINEERS WORK WITH AGENTS
Metrics, logs, and traces pulled the moment an alert fires, before an engineer has looked at it	**Validate the classification** and feed that judgment back explicitly — every correction is how the system learns what good looks like in your environment
Recent deployments and config changes checked against the affected service to surface what changed and when	**Apply business context** the agent doesn't have: severity given what's happening across the org, priorities not visible in the data
Pattern matching against past incidents to identify whether this has been seen before and how it resolved	
Classification as noise or action required, with the supporting evidence behind that call	**Define guardrails** governing how the agent operates: which alert types it handles autonomously, at what confidence threshold, and when it escalates
Blast radius identification when action is needed, isolating the issue to the responsible domain	

Stratos Pavlakis
CTO, Blueground

4x
faster time to triage

"With Resolve AI, there is no going back now, on-call and time-sensitive troubleshooting will be forever AI-first."

02 AI agents that debug alongside you in every incident war room

Pursues multiple hypotheses in parallel, converging on root cause with evidence

When an incident is declared, engineers need to pull context from observability tools, recent deployments, service dependencies, past incidents, and more. In large organizations, investigations require cross-domain expertise across application, networking, databases, and security. That's why incidents typically pull engineers from multiple teams into the same bridge.

Pursuing each hypothesis requires correlating information from systems like AWS, Datadog, Grafana, and Git repos. These systems specialize in their own domains and were never designed to work together. Someone has to be the connective tissue between them, and that someone is usually your most experienced engineer.

SEQUENTIAL VS. PARALLEL

A senior engineer can pursue one hypothesis at a time. An agent pursues all of them simultaneously — the way a full team would if they were all available and already had context.

Humans can only pursue one hypothesis at a time. A senior engineer investigating a latency spike decides: is this a database problem or an application problem? They pick one, rule it out, then try the other. Meanwhile the incident is ongoing and the blast radius is growing. AI agents can pursue multiple hypotheses in parallel — the way a full team would, if all of them were available and already had context.

WHAT AI AGENTS HANDLE	HOW ENGINEERS WORK WITH AGENTS
Correlated signals pulled across observability, infrastructure, and deployment systems simultaneously rather than sequentially	**Validate the root cause hypothesis** and confirm it fits what you know about the system; when it doesn't, correct it and capture why
Causal path tracing across services, dependencies, and recent changes to build a coherent picture of what happened	**Add context the agent doesn't have:** recent team changes, known system quirks, architectural decisions made outside documented channels
Parallel hypotheses formed and pursued simultaneously, the way a full team would if they were all available and had context	**Define escalation criteria:** when the agent investigates autonomously versus when a human needs to be in the loop based on severity and system criticality
Pattern matching against past incidents so prior investigations become active precedents rather than forgotten knowledge	
Grounded root cause with cited evidence across systems, not just a hypothesis	

87%

03 AI that makes every engineer fluent in production debugging

Any engineer can ask any question about their production systems while debugging issues or building new

Every engineer has questions about production, either while investigating an issue or responding to a shoulder tap from a teammate. Getting to an answer requires three distinct skills that are rarely distributed evenly across a team: knowing which system to check, knowing how to operate each system like an expert, and understanding the quirks of your specific environment.

The first two are teachable. Documentation describes which systems are relevant, and tools have query interfaces that can be learned. The third is institutional knowledge — it isn't written down anywhere. Senior engineers accumulate it over months or years, and it leaves with them when they move on.

AI agents can handle the first two directly. For the third, an agent that sits in the execution path of every investigation accumulates institutional knowledge by learning from every interaction. The more investigations run through the system, the better it understands your specific environment.

INSTITUTIONAL MEMORY

An agent that sits in the execution path of every investigation accumulates context over time. The more investigations run through the system, the better it understands your environment.

WHAT AI AGENTS HANDLE	HOW ENGINEERS WORK WITH AGENTS
Right systems, right sequence to answer a production question, without the engineer needing to know where to look	**Validate edge cases** for questions that touch systems with known quirks the agent may not have learned yet; capture those quirks explicitly
Contextual interpretation of what a metric means for this specific service under these conditions, not just the raw number	**Apply business context:** why a service is configured a certain way, what tradeoffs were made, constraints not visible in the data
Causal path tracing across services, dependencies, and recent changes to explain why something is happening	
Pattern matching against past investigations so institutional knowledge is available to every engineer, not just the ones who've been around long enough	**Set the bar** for what a complete answer looks like: the evidence required, the confidence level, and when the agent should say it doesn't know rather than guess

Angelo Marletta
Software Engineer, Coinbase

73%
faster time to root cause

"Resolve AI proved it could deliver real results in production. It identified dependencies, surfaced accurate root causes 73% faster than our teams, all while integrating cleanly into our existing stack."

At Resolve AI, we're investing in proactive optimization as the next outcome for production debugging. The goal is for Resolve to monitor

04 AI that handles routine operations and learns from every interaction

Generates postmortems, Git PRs, code fixes, or scripts that for your environment

Production knowledge is created in context. The learnings from high-stakes scenarios like incidents mostly stay in the heads of the engineers who were in the room, until they move teams or leave the company. This plays out as three distinct problems: inconsistent reporting during and after incidents, no system to capture and propagate the judgment an experienced engineer develops over time, and knowledge that exists in the organization but isn't accessible in the moment it's needed.

The through-line across all three is the same: knowledge is created in moments of high context and lost because there's no system to capture and surface it. AI agents that sit in the execution path of every investigation change this. They capture what happened, why, and what was learned automatically as a default output of the work itself.

KNOWLEDGE IN CONTEXT

AI agents that sit in the execution path of every investigation capture what happened, why, and what was learned — automatically, as a default output of the work itself.

WHAT AI AGENTS HANDLE	HOW ENGINEERS WORK WITH AGENTS
Post-mortem drafts generated from investigation traces and resolution timelines — documentation as a default output of every incident	**Validate post-mortem drafts** for accuracy; ensure the reasoning behind decisions is captured, not just what happened
Runbook updates based on what the investigation actually revealed about system behavior, not how it was originally designed to behave	**Review runbook updates** to confirm they reflect current system behavior and are specific enough to be useful under pressure
Gap identification between documented system behavior and what investigations are finding in practice	**Define standards:** templates, what gets formally codified versus noted for context, and how documentation feeds back into active investigations so the knowledge base compounds
Pattern surfacing across incidents so repeat issues become visible before they become habits	

Chris Umbel
AIOps Lead & SRE, Zscaler

30%
fewer engineers per incident

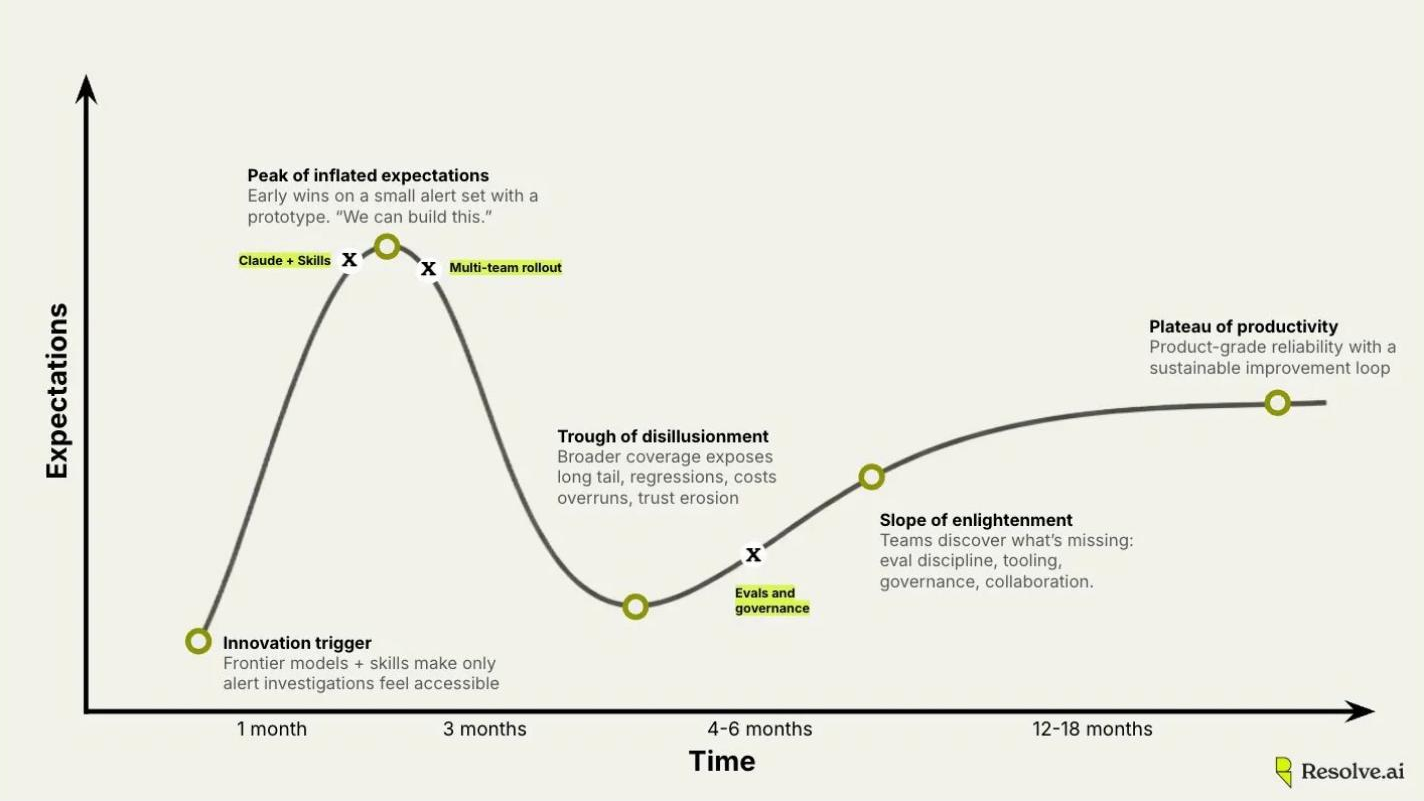
"Incident response at our scale isn't about collecting more signals. It's about understanding why something is failing, quickly enough to limit customer impact."

At Resolve AI, we're investing in making operational reviews continuous rather than episodic. The goal is to shift from documentation that happens after the fact to production intelligence that runs in the background all the time — Resolve continuously analyzes production signals to surface reliability patterns, flag SLO risks, and identify optimization opportunities before they become incidents.

Where most AI approaches for production fall short

Most engineering teams are using one of two approaches to apply AI to production workflows. Both make reasonable bets given what's visible. Both break before they scale.

VENDOR AI TOOLS	IN-HOUSE AI BUILDS
The appeal Already paying for observability tools or cloud vendors. They're part of the stack and most have some AI capability built in.	**The appeal** Full control. Off-the-shelf models connected to your stack via MCP. Works well for a subset of alerts managed by the team that built it.
Where it breaks Each tool is optimized for its own domain. AI features simplify retrieving information from that system — they don't cross domain boundaries.	**Where it breaks** As coverage expands, accuracy degrades, costs creep up as context windows expand, and trust erodes faster than it was built. Many internal projects reach saturation in 4–6 months.
The ceiling Datadog knows metrics. PagerDuty knows alerts. Neither can determine how a specific deployment interacted with a specific infrastructure state to produce a failure.	**The real problem** The problem is rarely the model or the agent architecture. It's everything that needs to exist around them: evals, governance, multi-team collaboration, continuous learning.
The core problem Production workflows are cross-domain by nature. A tool that only sees its own data gives you part of the answer.	**The maintenance burden** Unlike a standard software project, this isn't built once. Models change monthly. Prompting patterns that work today need to be rewritten with the next release.



Productionizing AI agents requires building and maintaining substantial engineering layers that most teams underestimate: integrations across dozens of tools, deep investigative reasoning that understands time and causality, continuous learning, evaluation infrastructure, collaboration primitives for multi-engineer workflows, and enterprise-grade security and governance. None of these are features you bolt on at the end.

Architecture of AI systems that operate production

The four workflows in this framework are only achievable if the right infrastructure exists underneath them. Five layers need to work together.

5-LAYER ARCHITECTURE STACK	
Context	Live model of your production environment: service dependencies, infrastructure ownership, deployment propagation. Updates continuously. Captures institutional knowledge from runbooks, past incidents, Slack threads, and senior engineers' heads.
Model	Domain-specific models post-trained on code and operational context. General-purpose foundation models aren't trained on production data — effective production reasoning requires models that understand failure modes, signal relationships, and operational patterns.
Agent	Specialized agents for different domains (application, infrastructure, database, network) coordinated by a planner that pursues multiple hypotheses simultaneously and converges on root cause through evidence.
Action	Automated remediation within defined guardrails, with approval workflows and rollback controls. Requires the first three layers to be reliable before it can be trusted.
Interface	A single interface to all production work. Investigations start automatically, run where engineers already work (Slack, Teams), and surface outputs engineers can act on with confidence.

Resolve AI connects and works with your entire tech stack across 50+ tools, APIs, alert systems, and runbooks. It builds a context graph that captures institutional and tribal knowledge and keeps it current. It spawns parallel agents to pursue multiple hypotheses simultaneously and converges on root cause with cited, grounded reasoning — using models post-trained on production context that general-purpose foundation models can't access. This is the infrastructure that makes the framework in this paper work and it's available today, without a two-year build.

Closing the human gap in working with AI agents

The gap between a working prototype and something engineers actually trust and adopt is not model quality. It is whether the system becomes the default operating experience: the thing engineers reach for first because it works where they work, shows its reasoning, and gets smarter over time. Five things determine whether that happens.

1
WORKFLOW AUTOMATION
Investigations start automatically and run in Slack or Teams. Follow-ups in the same thread.

2
GROUNDED IN EVIDENCE
Every output backed by evidence, reasoning path, and confidence level. Engineers know when to act.

3
HISTORICAL CONTEXT
Persistent memory across incidents. An investigation from six months ago is relevant now.

4
COLLABORATIVE UX
Shared investigations, multi-user workflows, handoffs without losing context.

5
ENTERPRISE READY
Permissions parity, full auditability, safety controls built in, not bolted on.

How Resolve AI helps engineers operate production systems

Architecture derived from first principles to solve the production problem in its entirety

We built Resolve AI as the manifestation of this opportunity. Our architecture has been derived from first principles to solve the production problem in its entirety. Here's how it works:

1

Understands production and operates all your tools

Resolve AI connects to your entire production stack: code, infrastructure, telemetry, knowledge sources, and a long tail of in-house and MCP tools. It operates each tool with expert-level proficiency, formulating precise queries that respect rate limits, interpreting results efficiently, and managing context without overwhelming the model.

2

Absorbs context and continually learns

Resolve AI gathers knowledge and context from undocumented sources: runbooks, documentation, past incidents, and Slack threads. It maps both the infrastructure dependencies and the teams that own them. When an engineer corrects its reasoning, that correction becomes part of the agent's knowledge. Over time, Resolve AI becomes the single and up-to-date source of truth for all production-related context.

3

Reasons with investigative expertise across boundaries

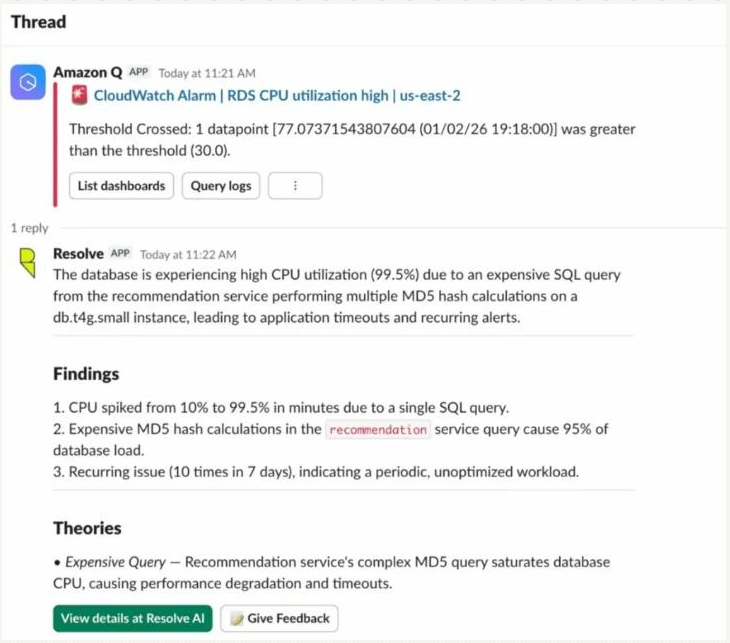
Resolve AI brings deep investigative expertise about how to debug and operate production systems together with the specific expertise of multiple teams in every execution. During investigations, the planner creates a plan spanning multiple systems and teams, spawning agents that pursue several hypotheses in parallel. As evidence emerges, the planner continuously refines the plan: ruling out hypotheses that don't fit, doubling down on those that do.

Here's how Resolve AI helps you in production workflows:

01 ON-CALL ROTATION

Participates in every on-call rotation

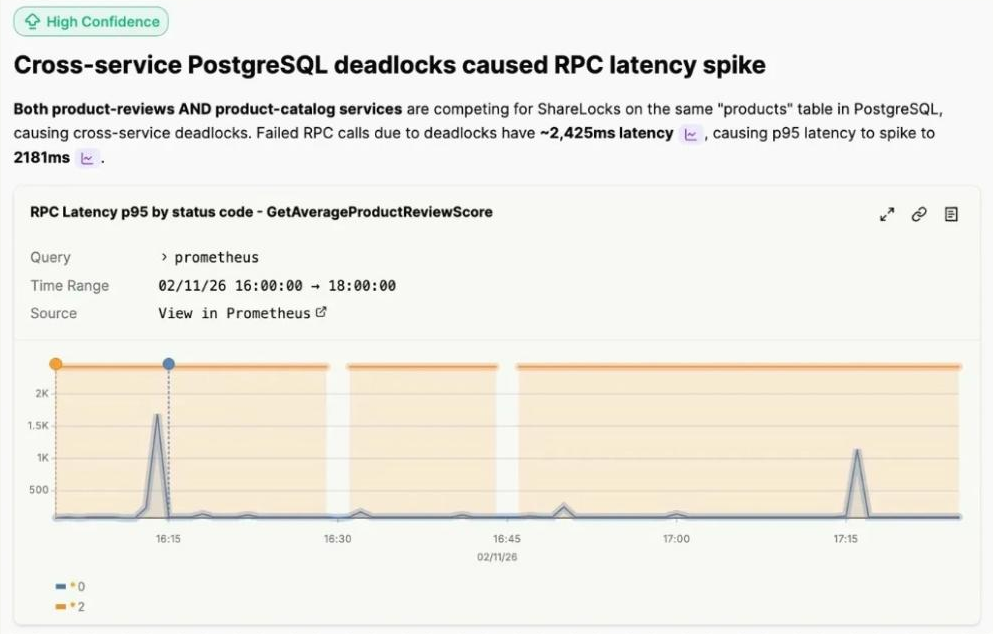
Autonomously investigates incidents and builds initial theories before you start.



02 INCIDENT INVESTIGATION

Gets you to root cause and fix

Pursues multiple hypotheses in parallel and validates each against real evidence. Creates code snippets, drafts PRs, and generates post-mortems.



03 PRODUCTION DEBUGGING

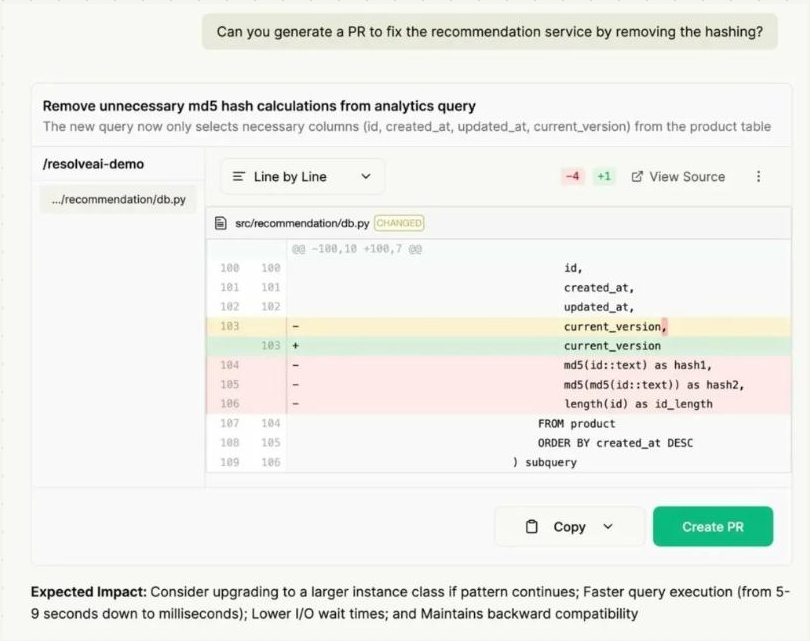
Debug any production issue

Anyone can ask questions during incidents or development and get precise answers about your production systems.

04 OPERATIONAL HARDENING

Execute remediation actions

Generates post-mortems, customer-facing reports, Git PRs, kubectl commands, and scripts that work with your production systems.



HOW YOU CAN GET STARTED

We've been working with engineering teams at Coinbase, DoorDash, Zscaler, Salesforce, and others to establish what good looks like for AI agents in production workflows. The broader goal is helping engineers operate production systems as fluently as they write code.

A few ideas central to our technical worldview and how we are investing in these outcomes at Resolve AI:

Models + agents

Purpose-built for production. Every improvement compounds across every workflow.

Closed-loop automation

We are investing in root cause analysis, remediation, prevention, and containment end-to-end.

Proactive operations

We are investing in shifting workflows from reactive to anticipatory; spotting issues before they become incidents.

AI code generation will produce software at unprecedented scale. The bottleneck is now how you operate and run the software in production.

Success will be determined by whether you can operate what you build with the same speed and reliability you use to build it. If you want to think through how AI agents can change how your team works with production, we'd love to have a conversation.

REACH OUT FOR A CONVERSATION **TRY IT FOR YOURSELF**

Resolve.ai