Pearson

Agentic AI in Platform Engineering

Leverage AI agents to reason through platform engineering challenges

Pearson Live Training | Monday, Jan 12th, 2026

Ajay Chankramath

CEO, Platformetrics LLC

About Ajay

- *Based out of Boulder, CO, in the foothills of Southern Rocky Mountains*
- *35+ years of experience in technology as a leader, architect, and developer*
- *Founder and CEO of Platformetrics, a unique advisory & consulting firm on platform engineering*
- *Previously held CTO, SVP, VP, Director roles at various companies such as Brillio, Thoughtworks, Oracle, Broadridge, & Xilinx (AMD)*
- *Author of multiple books on platform engineering*
- *Regular keynote speaker at technology conferences around the world*

https://platformetrics.com/about-ajay

Poll Question- Ice Breaker

Do you use AI or Agentic AI in your development teams?

- Yes
- No

Platform Engineering is Facing a n Explosion of Complexity

- The modern cloud-native stack is a sprawling ecosystem of tools and services. While powerful, this creates immense cognitive load on developers and platform teams.
- The core problem isn't a lack of tools, but a need for a new architecture of intelligence to manage the operational burden.

The Platform Engineering Paradox: AI Adoption is Soaring, But So Is Complexity

Despite massive AI adoption for development tasks, the underlying platforms are buckling under the strain of manual processes, team constraints, and cognitive overload.

The core problem isn't a lack of tools, but a lack of intelligence in the platform itself.

81%
Of engineers use AI in their daily work

~75%
Use AI for code generation

> 40%
Use it for Infrastructure as Code

57%
Have challenges with how to apply AI in platform engineering

71%
Believe their world will change drastically in the 12-24 months with AI

55%
Fear unsupervised AI is disastrous

AI adoption has crossed the chasm

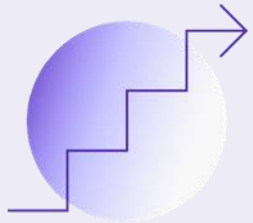
81%
Daily usage

8%
Intense usage

6.3%
Don’t use

Code Generation

Nearly 75% of the engineers use AI for code generation



Infra as Code

> 40% of the users use AI for Code generation

ChatOps

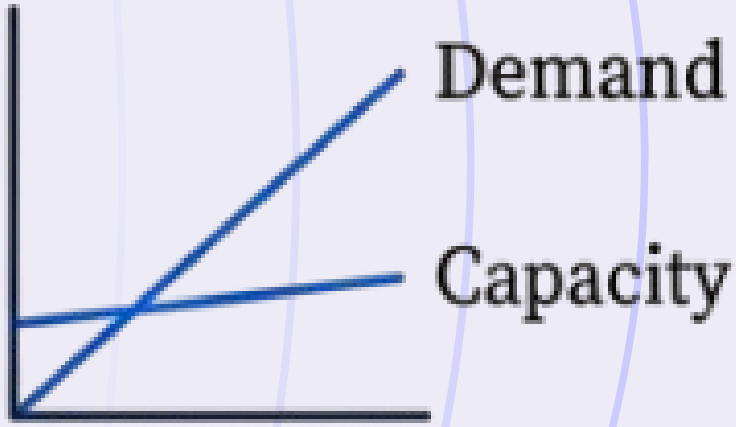
Variants of ChatOps – Natural Language Interface is the extremely pervasive

* Source: State of AI in Platform Engineering, platformengineering.org whitepaper based on actual interviews and research, Sep 2025

How are your teams doing?

The Capacity Gap

Requests consistently exceed capacity, leading to context switching, knowledge silos, and incident fatigue.



How are your teams doing?

The Toil Tax

- Onboarding new engineers (documentation hunts, tool setup).
- Updating configurations (endless YAML editing, PR cycles).
- Deploying new services (multiple 'terraform' runs, manual verification).
- Debugging production issues (log diving, metric correlation).

How are your teams doing?

Source of Cognitive Overload

- Logs across 12 services, metrics from 6 sources, 15+ tools in a typical workflow.

Course overview

Platform engineering pain points & AI solutions

SECTION 1

SECTION 2

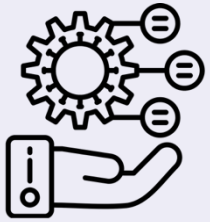
AI enhanced development lifecycle

Operational intelligence & multi-agent coordination

SECTION 3

SECTION 4

Implementation strategy & organizational readiness



From concepts to code: starter kits in action

SECTION 5

Pearson

Platform Pain Points and AI Solutions

Agentic AI in Platform Engineering

Poll Question- Lesson # 1

Which platform activity would benefit *most* from an AI agent that can reason across tools? (SELECT MULTIPLE)

- Incident triage across logs, metrics, traces, and deploy history
- Developer onboarding across repos, CI/CD, infra, and access
- Golden-path enforcement across CI/CD and IaC
- Diagnosing failed pipelines and PRs with full context
- Cost and reliability trade-offs across environments

Platform engineering pain points & AI solutions

- *Review manual processes, platform team constraints, and cognitive load*
- *Introduce Agentic AI: reasoning and adaptation beyond automation.*
- *Compare agent architectures: event-driven vs polling.*

Demo: *Agent diagnoses failed GitHub Actions workflow*

Take-home exercise: *Fork a repo, re-run the failing workflow, and use Claude to explain the error and suggest a fix*

Platform engineering reality

Manual Processes Everywhere end-to-end

- ***Onboard new engineers:*** *documentation hunts, knowledge transfer, tool setup*
- ***Update configurations:*** *YAML editing, PR reviews, testing cycles*
- ***Deploy new services:*** *multiple Terraform runs, kubectl applies, manual verification*
- ***Debug production issues:*** *log diving, metric correlation, context switching*
- ***Respond to incidents:*** *runbook execution, team coordination, postmortem documentation*

Platform Team Constraints

3x

Demand Growth

1:50

Platform to Dev Ratio

- *Requests exceed capacity*
- *Context switching overhead*
- *Knowledge silos form*
- *Incident fatigue increases*
- *Self-service remains elusive*

The Cognitive Load Problem

Decision Paralysis in Modern Infrastructure

Information Overload

- *Logs across 12 services*
- *Metrics from 6 sources*
- *Traces spanning regions*

Tool Fragmentation

- *15+ tools in workflow*
- *Different UIs and CLIs*
- *Context lost switching*

The Cognitive Load Problem - Data

Google SRE Research (2024)

Engineers spend **>50–60% of time gathering context** rather than executing fixes.

Microsoft DevOps Reports (2023)

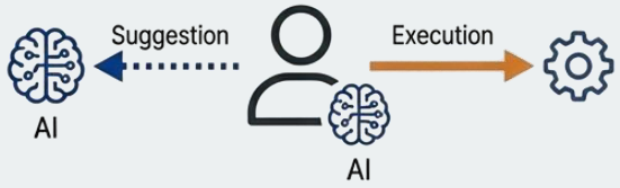
Context switching reduces developer productivity by **up to 40%**.

Gartner

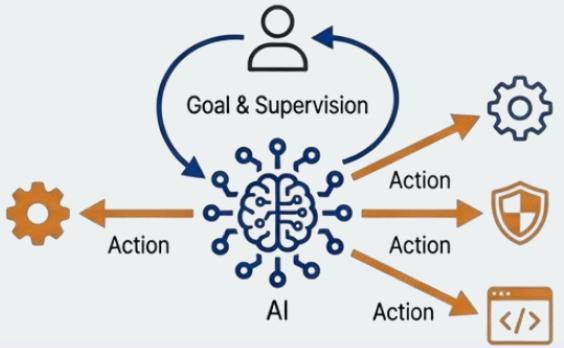
“Cognitive overload from fragmented tooling is the primary cause of platform inefficiency

The paradigm shift

Co-Pilot world



Agentic world



Not just a smarter tool, but a new paradigm

A 5-level framework for AI autonomy

L1: Rule-based automation

If X, then Y Logic, FAQ chatbots

L2:Co-Pilots & Routers

ML models classify, route and recommend. GitHub Co-Pilot

L3: Tool-using agents

Agent plans multi-step tasks, calls APIs, and uses tools to achieve goals. Agentic apps

L4: Multi-agent systems

A team of specialized agents coordinate and solve complex problems

L5: Fully Autonomous AGI

Agents that set their own goals and operate without human supervision. This is research, not reality.

Agentic AI: Beyond Automation

What Makes an AI Agent?

- ***Reasoning:*** *Analyzes context and patterns*
- ***Adaptation:*** *Learns from outcomes*
- ***Autonomy:*** *Makes informed decisions*
- ***Tool Use:*** *Calls APIs and executes actions*

The Key Difference

*Traditional automation follows rigid if-then rules. Agentic AI understands **intent**, evaluates **options**, and **adapts** its approach based on the specific situation.*

The 4 Pillars of an AI Agent

Reasoning

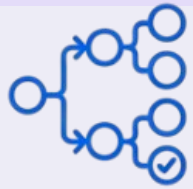
Analyzes context, understands system behavior, and connects patterns across disparate data sources (logs, metrics, traces).

Adaptation

Learns from the outcomes of its actions. It improves its decision-making over time, identifying flaky tests or optimizing processes based on historical data.

Tool Use

Interacts with the real world by calling APIs, executing shell commands, and managing infrastructure tools such as kubectl and Terraform.



Autonomy

Makes informed, goal-oriented decisions without constant human intervention, operating within predefined guardrails

Traditional Automation vs Agentic AI

Traditional Automation

- Predefined workflows
- Brittle on edge cases
- Requires constant updates
- No context awareness
- Sequential execution

VS

Agentic AI

- Goal-oriented reasoning
- Handles ambiguity
- Self-improving
- Understands intent
- Dynamic adaptation

Agent Architectures

Two Fundamental Approaches to Triggering Agent Actions

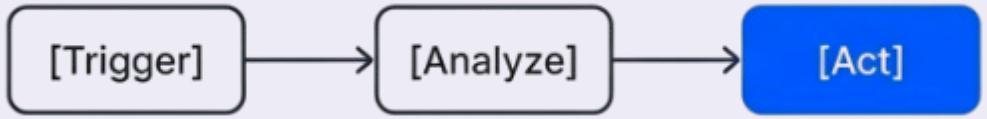
Event-Driven

Reacts to specific triggers

Polling

Regularly checks conditions

Event-Driven Agents



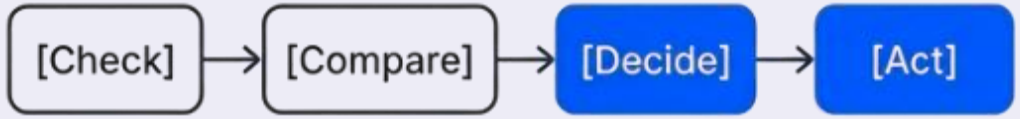
- *GitHub Actions workflow failure*
- *Alert fires from monitoring*
- *Deployment starts/completes*
- *PR opened or merged*
- *Resource utilization threshold*

Best For

- *Real-time responses*
- *Incident detection*
- *CI/CD pipeline integration*
- *Cost-efficient operations*

Minimal overhead, immediate action

Polling Agents



- *Drift detection in configurations*
- *SLO budget tracking*
- *Security compliance checks*
- *Resource optimization scans*
- *License expiration monitoring*

Best For

- *Proactive monitoring*
- *State validation*
- *Trend analysis*
- *Scheduled maintenance*

Catches issues before they become incidents

Choosing Your Architecture

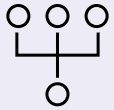
Choosing the appropriate agentic architecture is critical for the success of your AI journey.

Aspect	Event-Driven	Polling
Trigger	*Immediate on event*	*Scheduled intervals*
Latency	*Near real-time*	*Depends on interval*
Resource Use	*Minimal (idle when quiet)*	*Constant overhead*
Missed Events	*Low risk*	*Possible between checks*
Complexity	*Requires webhooks/events*	*Simpler to implement*
Best Use Case	*Incident response*	*Drift detection*

Live Demo: Event-Driven Diagnostic Agent (CI/CD Failure Analysis)

Trigger

CI/CD workflow fails → agent invoked automatically



Inputs

Raw error logs + workflow context (branch, runtime, environment)

Agent reasoning

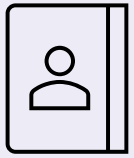
Identifies *root cause* (not just symptom).
Explains *why* the failure occurred

Output

- Step-by-step fix
- Prevention strategies to avoid recurrence

Human role

- The engineer reviews the diagnosis and applies the fix.



Pearson

Take-home exercise: Reproduce & Diagnose a CI/CD Failure

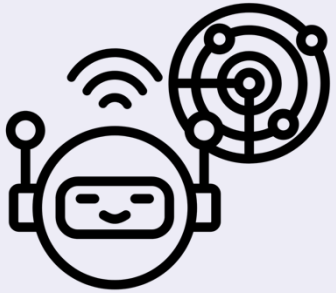
1. Fork the demo repository
2. Re-run the failing GitHub Actions workflow
3. Capture the raw error output
4. Ask Claude to:
    - Identify the root cause
    - Suggest a fix
    - Recommend prevention steps
5. Compare AI diagnosis with your own intuition

Experience how agentic reasoning compresses 30+ minutes of log analysis into seconds

Section 1 Recap

What We Covered:

- *Platform engineering pain points and cognitive load*
- *Agentic AI fundamentals: reasoning and adaptation*
- *Event-driven vs polling agent architectures*
- *Hands-on agent diagnosis of workflow failures*



AI Enhanced Development Lifecycle

Agentic AI in Platform Engineering

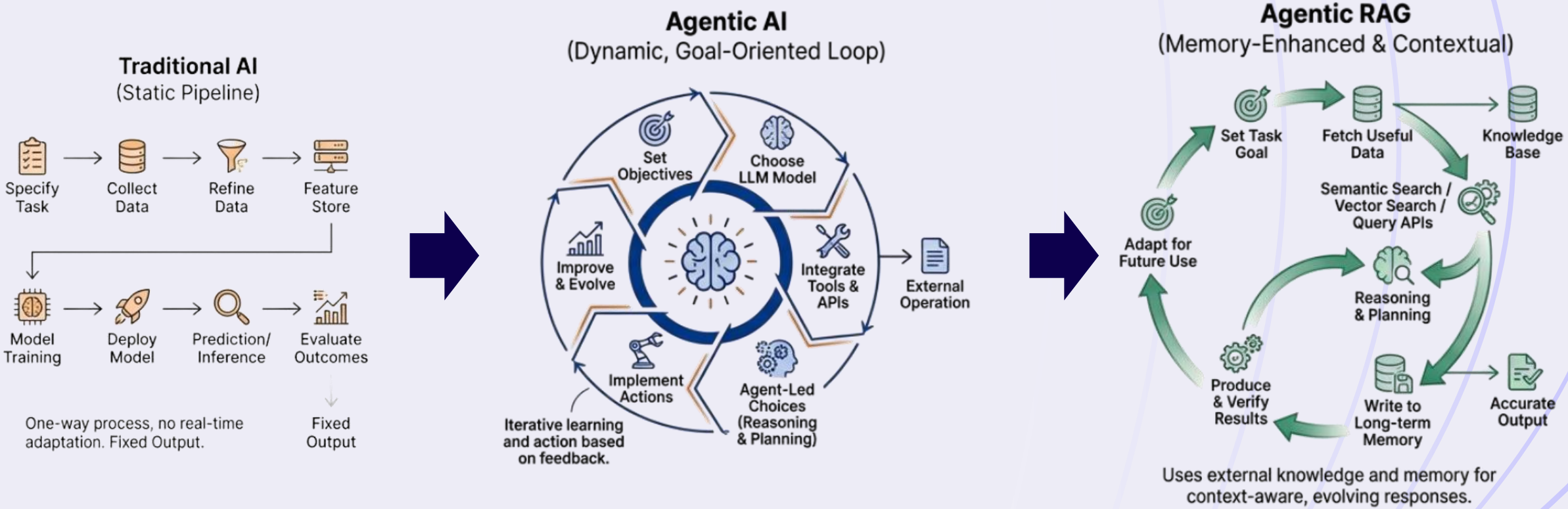
Poll Question- Lesson # 2

What currently prevents your platform from trusting AI-assisted release decisions?
(SELECT MULTIPLE)

- Quality signals exist, but are fragmented across tools
- Thresholds are static and don’t reflect real risk
- Humans don’t trust AI reasoning yet
- No audit trail or explainability for decisions
- Nothing — we already allow AI to influence releases

Architecture of Action

The fundamental architecture of AI is evolving from a static, one-way process to a dynamic, iterative loop that enables goal-oriented action.



AI-enhanced development cycle

- *Explore intelligent CI/CD pipelines: self-optimization and automated rollback decisions*
- *Understand AI-driven remediation patterns and conversational operations interfaces*

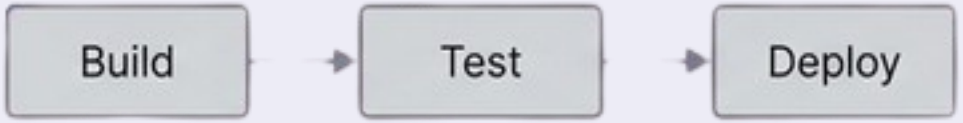
Demo: *Agent evaluates release readiness using quality gates (test coverage, performance, security)*

Take-home exercise: *Edit quality gates in repo, rerun the pipeline, and use Claude to generate a release/rollback rationale*

Traditional CI/CD Pipeline

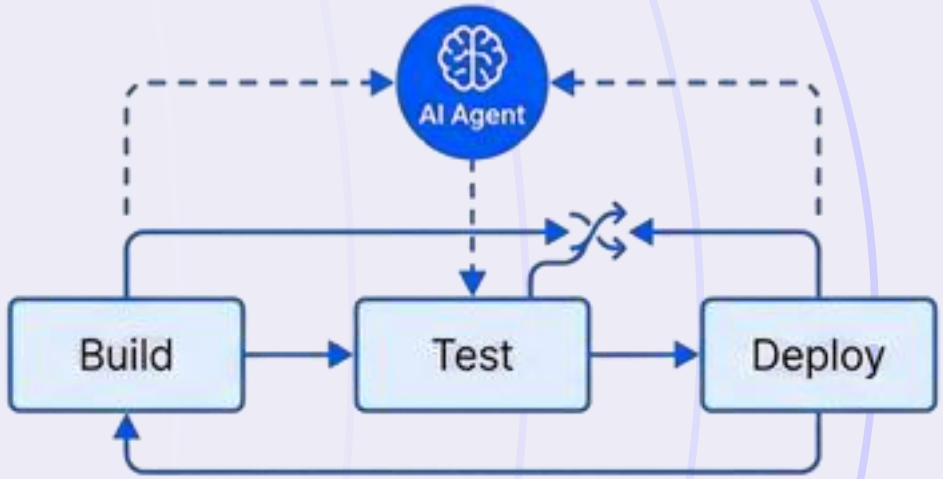
The Familiar Pattern:

- *Fixed pipeline stages: build, test, deploy*
- *Binary pass/fail gates with static thresholds*
- *Manual intervention required for failures*
- *Rollback decisions made by humans reviewing metrics*



Intelligent CI/CD Pipelines

- *Contextual Evaluation*
  - *Agents analyze test results, performance metrics, and security scans beyond simple pass/fail.*
- *Self-Optimization:*
  - *The pipeline learns and improves itself.*
  - *Identifies and quarantines flaky tests.*
  - *Optimizes test execution order based on failure patterns.*
  - *Dynamically adjusts timeouts and retry policies.*
- *Automated Decision-Making*
  - *Recommends actions with confidence scores and a clear rationale.*

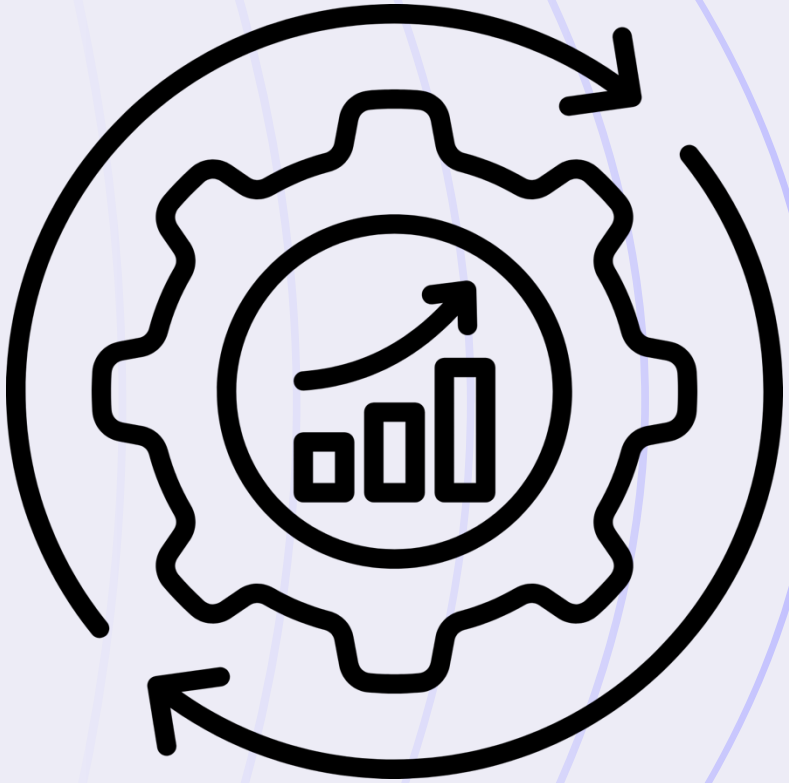


AI doesn't remove human judgment—it augments it with evidence.

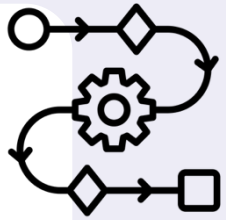
Self-Optimization Concept

Continuous Improvement:

- *Identifies flaky tests and suggests fixes or quarantine*
- *Optimizes test execution order based on failure patterns*
- *Adjusts timeout values and retry policies dynamically*
- *Recommends infrastructure changes when bottlenecks detected*



Automated Rollback Decisions

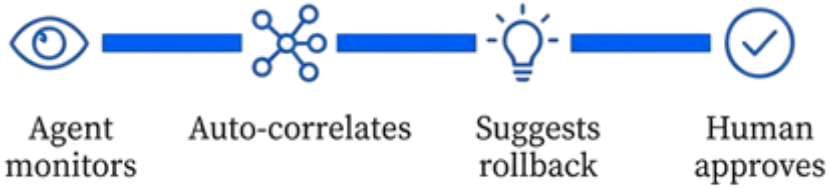


Traditional Approach

- Alert fires
- Engineer investigates
- Manual correlation
- Rollback decision
- Time: 15-30 minutes

AI-Enhanced Approach

- Agent monitors deployment
- Correlates metrics automatically
- Evaluates error patterns
- Suggests rollback with rationale
- Time: 30-60 seconds



Speed of rollback matters more than perfection of deployment.

AI-Driven Remediation Patterns

Diagnostic

Find the why?

- Automated root cause analysis for failed workflows.
- Correlation of log patterns across multiple services.
- Dynamic dependency mapping during an incident.

Corrective

Fix the “now”

- Safe configuration adjustments (e.g., increasing memory).
- Automated service restarts.
- Intelligent traffic rerouting to healthy instances.

Preventive

Prevent the “later”

- Detection of recurring error patterns before they become critical.
- Policy recommendations (e.g., 'This service needs a new rate limit').
- Automated infrastructure improvements during idle periods.

Conversational Operations Interfaces

Ask Questions

- Why did deployment fail?
- What's the error rate trend?

Get Explanations

- Explain this metric spike
- Compare to last release

Request Actions

- Roll back this deployment
- Scale up service X

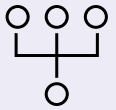
Review History

- Show recent rollbacks
- What changed yesterday?

Live Demo: Release readiness agent

Trigger

Pull request opened



Signals

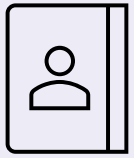
Test coverage | Performance regression | Security vulnerabilities | Code complexity

Agent behavior

- Weighs risks vs thresholds
  Produces a deploy/hold/rollback recommendation
- Explains trade-offs with confidence score

Human role

- Final approval remains with the engineers



Take-home exercise: Teach an agent how your team thinks about risk

- Edit quality-gate thresholds in the repo configuration
- Re-trigger the pipeline
- Use Claude to:
  - Explain the decision
  - Justify risk trade-offs
  - Suggest mitigation actions
- Observe how *small changes* shift the recommendation

Understand how judgment, not automation, is encoded into agents

Section 2 Recap

What We Covered:

- *Intelligent CI/CD pipelines with self-optimization*
- *Automated rollback decision-making*
- *AI-driven remediation patterns*
- *Quality gate evaluation with agent analysis*

Operational Intelligence & Multi-agent Coordination

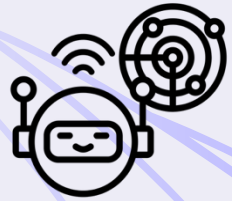
Agentic AI in Platform Engineering

Poll Question- Lesson # 3

Based on your understanding of this topic so far, where would multiple specialized agents most likely conflict in your platform today?
(SELECT MULTIPLE)

- Cost optimization vs production reliability
- Security enforcement vs developer velocity
- Deployment speed vs SLO error budgets
- Auto-remediation vs human incident command
- We don’t have agents yet, so we haven’t hit this

Operational Intelligence & Multi-Agent Coordination



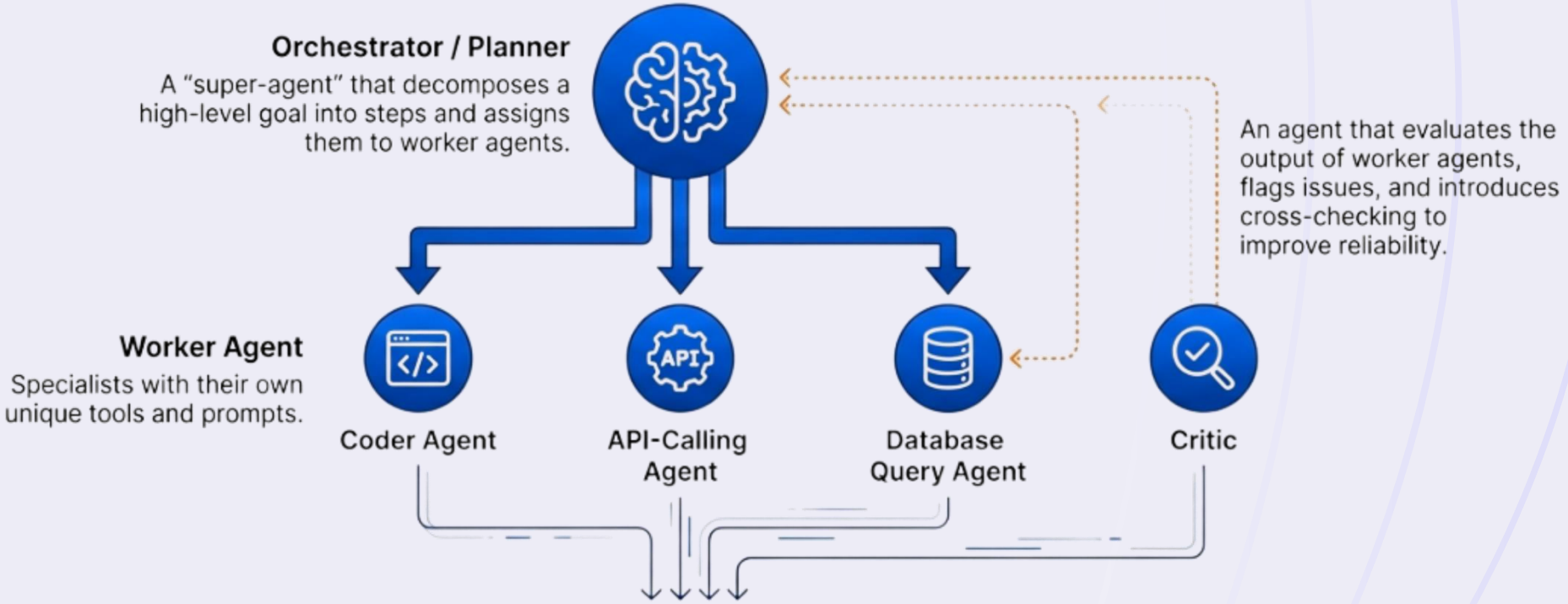
- *Explore AI-enhanced observability: anomaly detection and predictive analytics*
- *Understand agent communication patterns, conflict resolution, and failure handling*

Demo: *Multi-agent simulation (cost optimizer + incident responder) running across GitHub Actions jobs*

Take-home exercise: *View logs/output in a provided playground, no setup required*

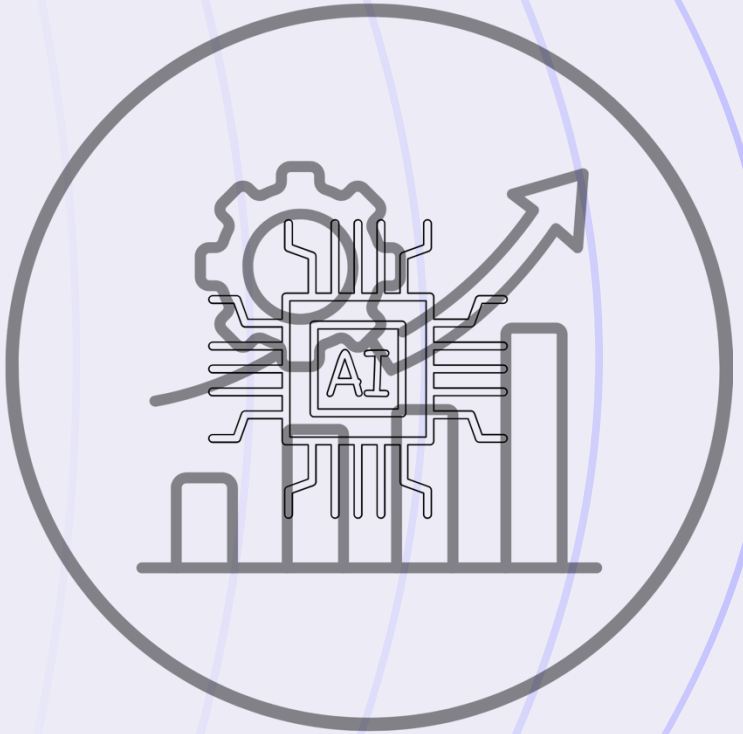
The premise of multi-agent orchestration

Instead of a single generalist agent, the future is a team of specialized agents collaborating to solve complex problems. Many predict this as the #1 predicted AI trend for 2026.



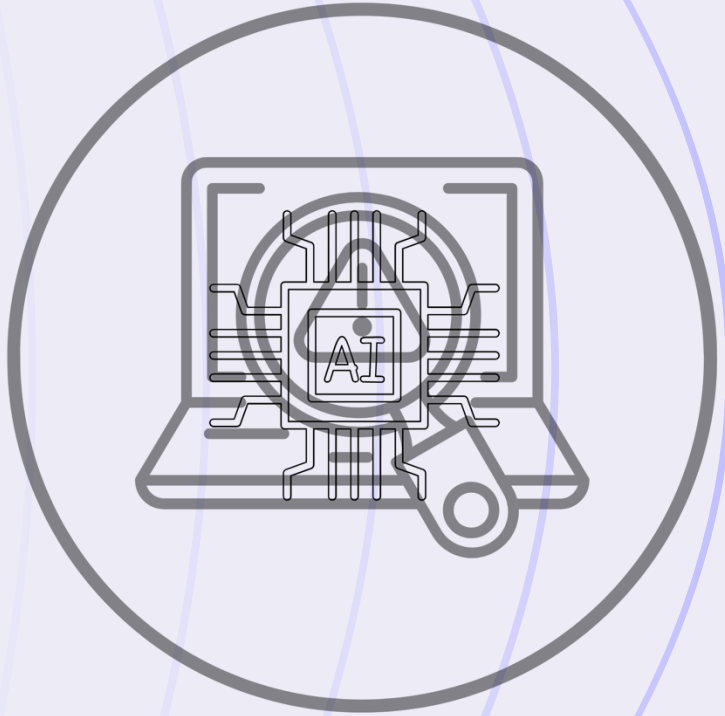
AI-Enhanced Observability

- *Detects anomalies without predefined thresholds*
- *Understands system behavior patterns over time*
- *Correlates metrics across services automatically*
- *Predicts issues before they impact users*



Anomaly Detection: Intelligent Pattern Recognition

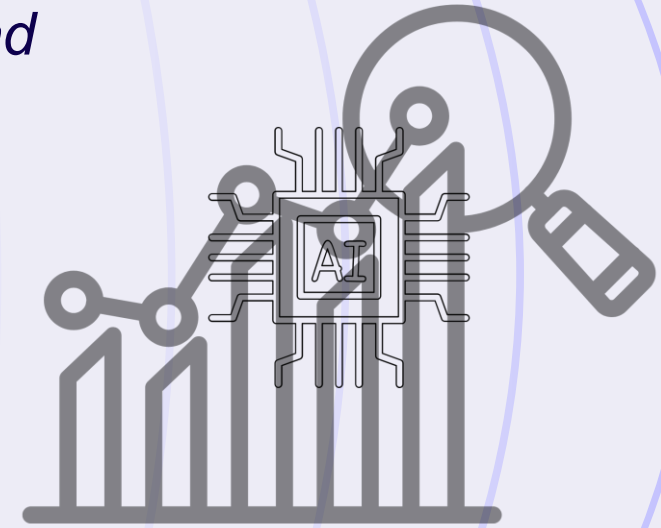
- *Baseline learning: Understands normal system behavior*
- *Deviation detection: Spots unusual patterns automatically*
- *Contextual analysis: Distinguishes real issues from noise*
- *Adaptive thresholds: Adjusts to changing conditions*



Anomalies are contextual, not numerical

Predictive Analytics: Proactive Problem Prevention

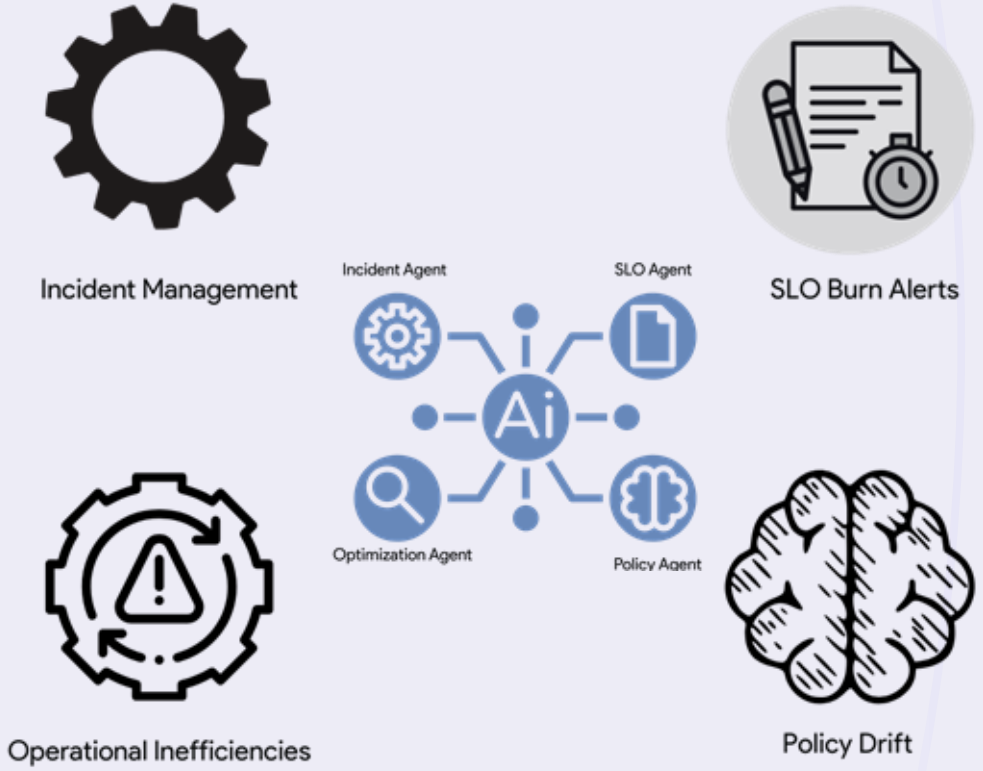
- *Resource exhaustion: Predicts disk/memory issues hours ahead*
- *Performance degradation: Forecasts latency increases*
- *Capacity planning: Recommends scaling actions*
- *Incident prevention: Suggests preemptive actions*



Traditional vs AI-Enhanced

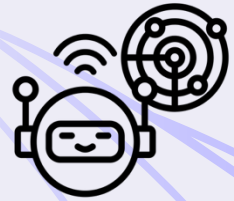
Traditional

- Dynamic baselines
- Predictive insights



AI-Enhanced

- Static thresholds
- Reactive alerts



Multi-Agent Coordination

Working together toward shared goals

- *Specialized agents with different responsibilities*
- *Communication protocols and conflict resolution*

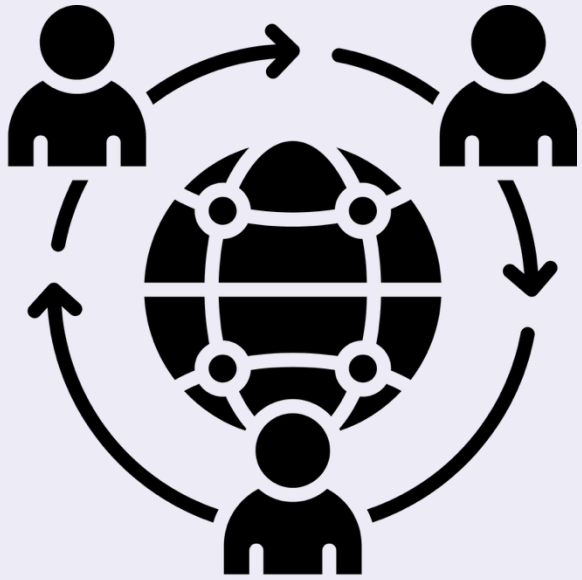
Autonomy without coordination is chaos

*Multi-agent systems require **governance**, not just intelligence.*

Agent Communication Patterns

Direct

Shared state event notifications



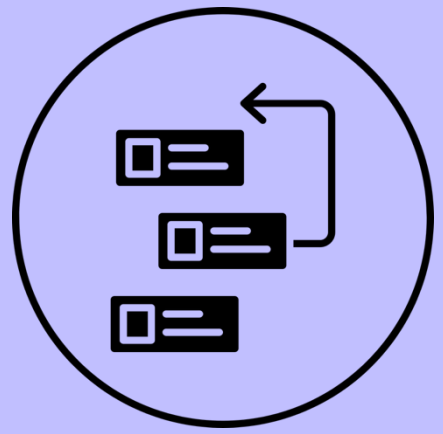
Broadcast

Point-to-point request-response

Conflict Resolution Strategies

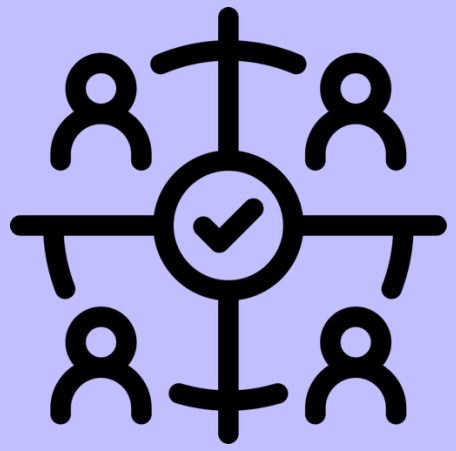
Priority

- Hierarchy-based
- Security overrides cost



Consensus

- Agreement required
- Majority vote

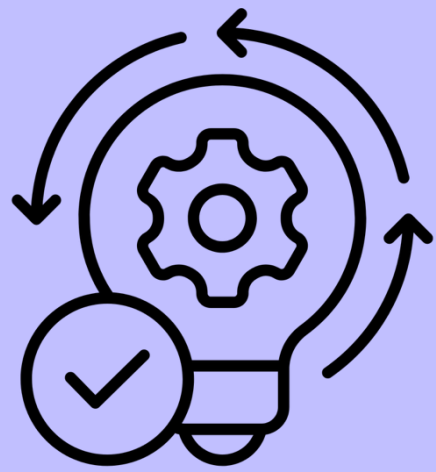


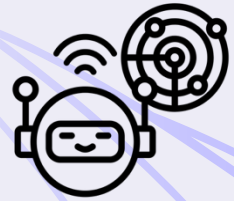
Arbitration

- Escalate to human
- Request decision

Deferral

- Wait and retry
- Queue action





Failure Handling

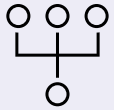
Resilience Mechanisms:

- *Graceful degradation: Continue with reduced functionality*
- *Circuit breakers: Prevent cascading failures*
- *Retry with backoff: Smart persistence*
- *Failover to backup: Automatic agent replacement*

Live Demo: Multi-agent coordination & conflict resolution

Agents

Cost optimization agent
Incident response agent



Actions

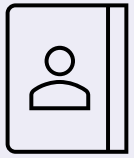
Analyzes the same system
Optimizes for a different goal

Conflict

Cost savings vs production reliability

Resolution

- Agents negotiate
- Higher-priority risk defers optimization



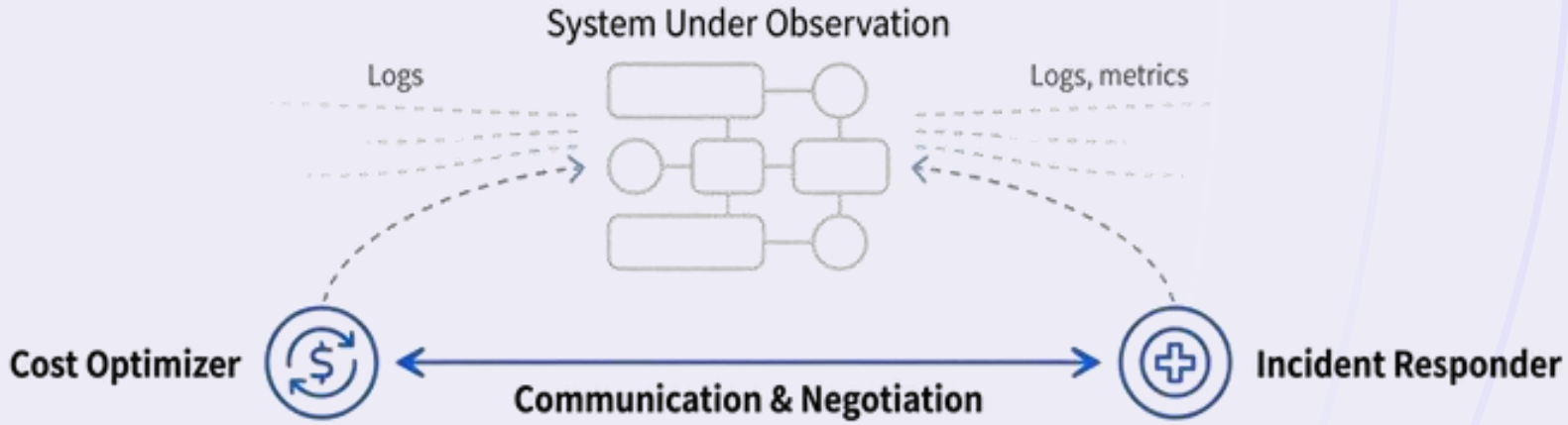
Take-home exercise: Cost optimizer + Incident responder

Scenario: *A Cost Optimizer agent and an Incident Responder agent work together.*

Action 1: *Cost Optimizer identifies an oversized service and suggests scaling it down by 30%.*

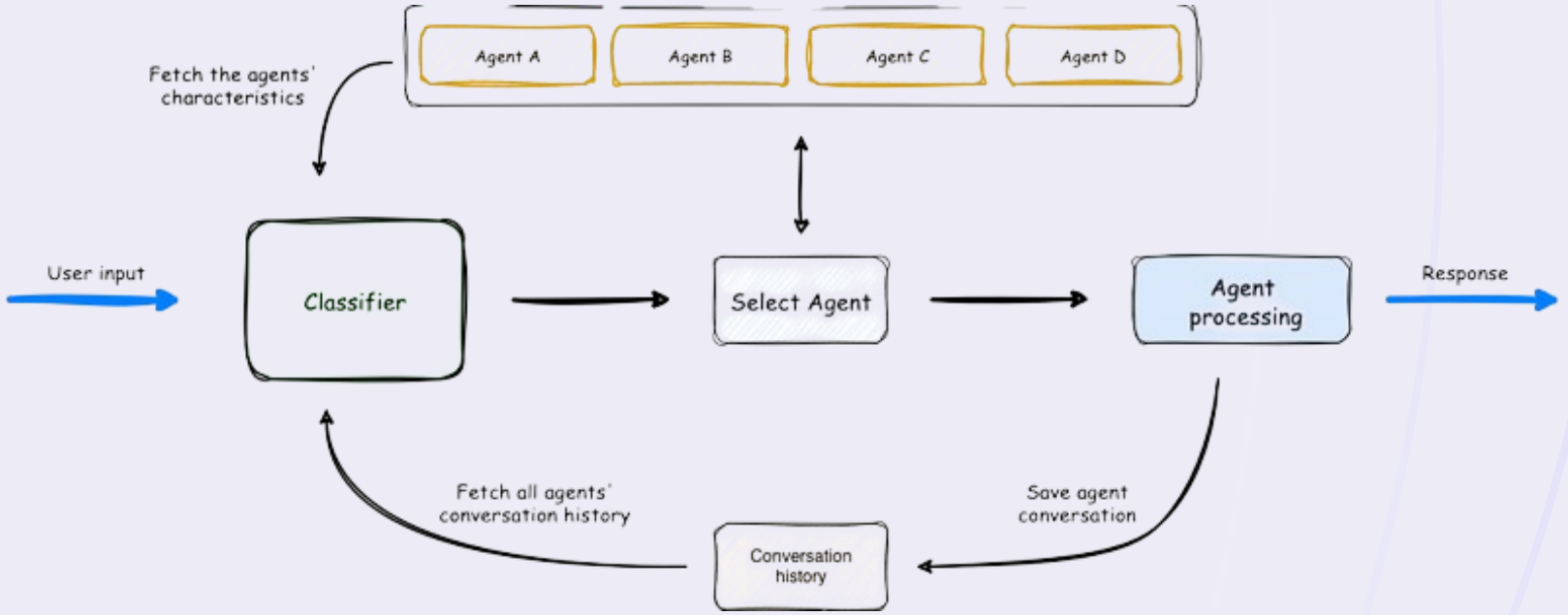
Action 2: *Incident Responder detects a 5% increase in error rate for a related service.*

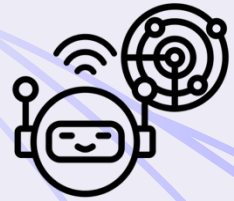
Conflict Resolution: *The agents communicate. Because the Incident Responder's priority is higher, they reach a consensus: defer cost optimization until the error rate is resolved.*



Bonus exercise: Review Agent Squad

Flexible, lightweight open-source framework for orchestrating multiple AI agents to handle complex conversations -- https://github.com/awslabs/agent-squad

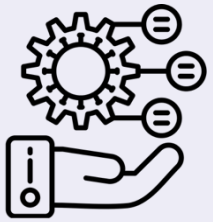




Section 3 Recap

What We Covered:

- *AI-enhanced observability with anomaly detection*
- *Predictive analytics for proactive problem prevention*
- *Multi-agent coordination and conflict resolution*
- *Live simulation of cost optimizer and incident responder*



Implementation Strategy & Organizational Readiness

Agentic AI in Platform Engineering

Poll Question- Lesson # 4

What is the *first* platform action you would *not* allow an AI agent to take autonomously?
(SELECT ONE)

- Rolling back a production deployment
- Modifying infrastructure configuration
- Enforcing or changing security policies
- Scaling resources up/down for cost optimization
- None — autonomy is acceptable with guardrails

Implementation strategy & organizational readiness

- *Assess organizational readiness: team skills and infra requirements*
- *Review phased rollout strategies: monitoring-first evolving to autonomous actions*
- *Consider risk mitigation: guardrails, ethics, and human oversight frameworks*

Demo: *Agent-enabled developer portal hosted via GitHub Pages with AI-driven onboarding/self-service endpoint*

Take-home exercise: *Trigger the endpoint with curl/browser and use Claude to turn the suggestion into a task list*

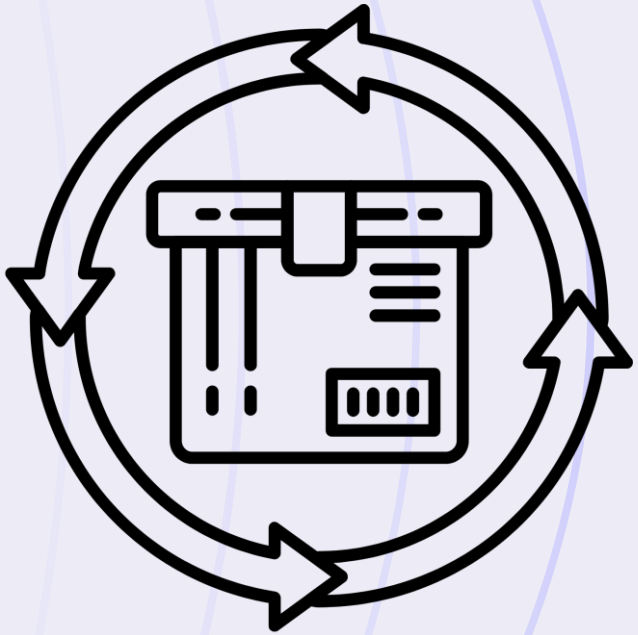
Role evolution of developers

Manual, direct work focusing on execution

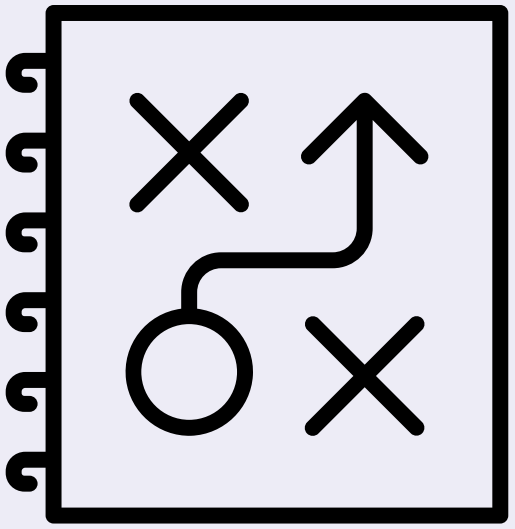
Deep operational experience and system knowledge, but applies it differently.

Organizational Readiness: Platform-as-a-Product

- *Treat platform as internal product with real customers*
- *Define clear ownership and product accountability*
- *Establish feedback loops with developer users*
- *Measure adoption and developer satisfaction*



3 cardinal sins of platforms in the AI world



1. *Desiring and request trust before autonomy*
2. *Building platforms for technology instead of users*
3. *AI adoption as a technical rollout, before the organizational strategy*

Team Skills Assessment

- *Platform engineering: API design, self-service tooling*
- *AI/ML integration: Prompt engineering, model selection*
- *Observability: Metrics, logging, distributed tracing*
- *Product thinking: User research, value measurement*

Infrastructure Requirements

Foundation

- LLM API access
- Webhook support

AI Enhancement

- API-first design
- Software-defined

Start Simple, Evolve Continuously

- *A complex system that works evolves from a simple system that works*
- *Begin with single team, expand as adoption grows*
- *Prove value before investing in comprehensive features*

Phased Rollout Strategy

Monitor

Observe, analyze & build trust

Start with AI-assisted diagnostics. Agents analyze failures and provide root-cause explanations but take no action. This proves their value and accuracy safely.

Recommend

Suggest actions, with a human in the loop

Agents propose solutions (e.g., "Roll back this deployment," "Apply this configuration fix"). A human must approve the action before it is executed.

Act

Execute autonomously with guardrails

Grant agents the authority to perform specific, low-risk actions on their own (e.g., restarting a service, scaling a replica set). AI actions are fully audited.

Coordinate

Enable multi-agent systems and optimization

Deploy specialized agents that collaborate, negotiate, and resolve conflicts to manage the platform proactively.

Risk Mitigation: Guardrails

- ***Scope limits:*** *Define what agents can and cannot do*
- ***Cost controls:*** *Budget caps and approval thresholds*
- ***Blast radius:*** *Limit impact of automated actions*
- ***Rollback capability:*** *Always maintain escape hatch*

Human Oversight Framework

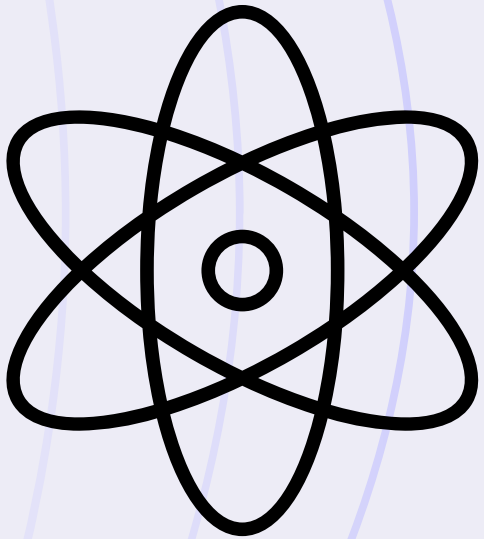
- ***Escalation paths:*** *When to involve humans*
- ***Audit logs:*** *Complete traceability of all actions*
- ***Review cycles:*** *Regular assessment of agent decisions*
- ***Override authority:*** *Clear ownership and accountability*

Ethics and Responsibility

- ***Transparency**: Agents disclose automated vs human actions*
- ***Fairness:** Avoid bias in recommendations and decisions*
- ***Privacy:** Protect sensitive data in agent interactions*
- ***Accountability:** Clear ownership of outcomes*

Guardrails, ethics and human oversight

- *EU AI Act & NIST AI RMF* emphasize auditability and override mechanisms.
- *OpenAI* safety guidance: human-in-the-loop is mandatory for high-impact actions.
- *Gartner* recommends **progressive autonomy models**.



Remember

- *Guardrails are not a constraint on intelligence—they are an enabler of trust.*
- *Every autonomous action must have a human escape hatch.*
- *Explainability is the price of autonomy.*

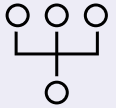
Measuring Success: Platform Product Metrics

- *Adoption rate: Developer teams actively using agents*
- *Time savings: Reduction in manual intervention*
- *Developer satisfaction: NPS from platform users*
- *Incident reduction: Fewer production issues*

Live Demo: Context-aware developer portal agent

Trigger

Developer asks a natural-language question

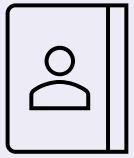


Inputs

Agent loads context: experience, team, tech stack

Agent responses

- Tailored explanation
- Relevant docs
- Code examples
- Task-oriented next steps



Mini Exercise: Experience personalized self-service

1. Trigger the portal endpoint (browser or curl)
2. Review the AI-generated onboarding guidance
3. Ask Claude to convert suggestions into a task list

Reflect on how would this change onboarding in your org?

From Concepts to Code

Agentic AI in Platform Engineering

Poll Question- Lesson # 5

If you built *one* agent in the next 30 days, what platform problem would it solve?
(SELECT ONE)

- CI/CD failure diagnosis with actionable fixes
- AI-driven release readiness and quality gates
- Incident correlation and response recommendations
- Continuous cost–performance optimization
- Personalized developer onboarding via platform APIs

From concepts to code: starter kits in action

- *Review key architectural patterns and implementation priorities*
- *Agent templates and starter repos*

Demo: *Showcase starter kit of open-source GitHub Action templates*

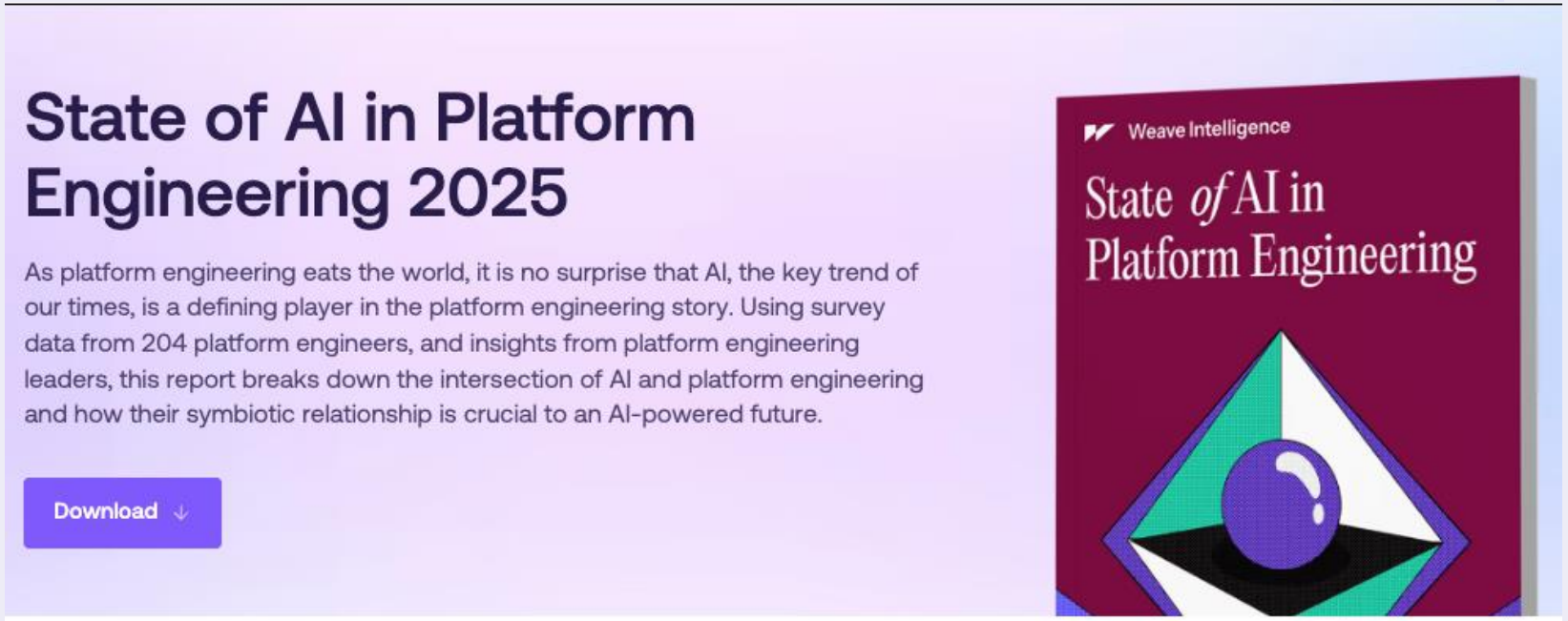
Take-home exercise: *Fork one template repo and use Claude to scaffold a README with setup and troubleshooting steps*

Key Architectural Patterns: Recap

- ***Event-driven agents:*** *Webhook triggers for real-time response*
- ***Polling agents:*** *Scheduled checks for drift and compliance*
- ***Multi-agent coordination:*** *Specialized agents working together*
- ***Quality gates:*** *AI-powered release readiness evaluation*

State of AI in platform engineering

Engineering platforms *consistently enable ideal AI adoption* within organizations, but only *when intelligence is embedded into the platform itself*, not bolted on afterward.



https://platformengineering.org/reports/state-of-ai-in-platform-engineering-2025

Starter kits and practical adoption

- CNCF end-user surveys show adoption accelerates with **reference implementations**.
- GitHub: repos with runnable examples see significantly higher reuse.
- Thoughtworks Tech Radar promotes “paved paths + examples” over frameworks.

Remember:

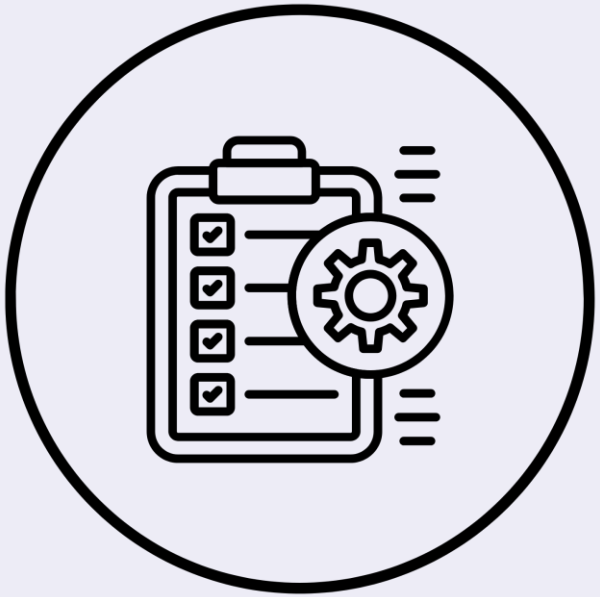
- *Abstractions don’t teach—examples do.*
- *The fastest way to adoption is something developers can fork.*
- *Platforms become real when they compile.*

CLOUD NATIVE COMPUTING FOUNDATION

GitHub

thoughtworks

Implementation Priorities



- *Start with diagnostics: Workflow failure analysis and root cause*
- *Add quality gates: Test coverage, security scans, performance*
- *Enable recommendations: Suggest actions, require approval*
- *Scale to autonomous: Multi-agent with conflict resolution*

Required evolution during the AI hype

- AI/Agentic automation
- Fragmented infrastructure
- Automating chaos
- Uncontrolled resource consumption

- Golden paths
- Control planes
- Domain APIs
- Supervised AI

Your action plan

Audit Your Runbooks
Go through every runbook and categorize them as 'mechanical' or 'judgment-based! The mechanical ones (likely 60-70%) are your prime candidates for 'Delegate'.

Pilot on Low-Risk, High-Volume Tasks
Don't start with production database failover. Start with log analysis for incident triage or cost optimization recommendations for dev environments. Build trust gradually.

Build Your 'Guardrail Muscle'
Keep a log of every time an AI agent suggests an action and you decide to override it. Reviewing this log reveals the patterns where human judgment is most critical.

Invest in AI Orchestration Skills
This is the new core competency. Learn prompt engineering for operations, how to structure tasks for agents, and how to debug when they make mistakes.

Track the 'Last Mile Gap'
For every task an AI handles, document the exact point where it hands off to a human. This map defines your team's unique, irreplaceable expertise.

Course Resources

Templates

- Complete working examples
- Ready-to-fork repos
- Typescript & Python

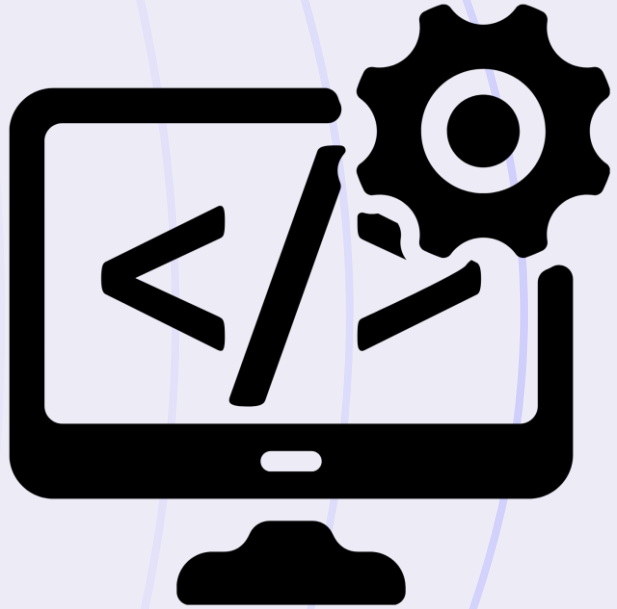
Starter Kits

- GitHub Actions workflows
- Agent configurations

https://github.com/achankra/agentic-aiplatformengineering/blob/main/README.md

Agent Template Library

- *Diagnostic Agent: Analyzes CI/CD failures and suggests fixes*
- *Quality Gate Agent: Evaluates release readiness criteria*
- *Cost Optimizer: Identifies resource savings opportunities*
- *Incident Responder: Monitors health and escalates issues*



Starter Repository Features

- *Working GitHub Actions workflows with Claude API integration*
- *Configuration files with customizable parameters*
- *Example prompts and response handling*
- *Setup instructions and troubleshooting guides*

Open-source resources – Agentic AI frameworks

Repository	Description	What you should try
crewAIInc/crewAI	Multi-agent framework for autonomous AI teams	Cost optimizer + reliability agents that debate
langchain-ai/langgraph	Graph-based agent orchestration with state	Workflow diagnostic agent with decision branches
langchain-ai/open-agent-platform	No-code agent builder with supervisor patterns	Multi-agent coordination visually

Open-source resources – SRE & Incident response agents

Repository	Description	What you should try
fuzzylabs/sre-agent	Open-source AI SRE with K8s + GitHub MCP	Diagnosis of pod failures, auto-creating GitHub issues
VoltAgent/awesome-claude-code-subagents	100+ specialized Claude subagents (SRE, DevOps, Platform)	Incident-responder, terraform-engineer agents
HAIstings (original: stacklok/codegate)	LangGraph agent for GitOps vulnerability prioritization	Prioritize CVEs automatically by infrastructure context

Open-source resources – Multi-agent examples

Repository	Description	What you should try
crewAIInc/crewAI-examples	Official CrewAI examples: code review, lead scoring, content	Show Game Builder Crew (designer + coder + tester agents)
microsoft/autogen	Microsoft's multi-agent conversation framework	Agents debate and validate each other's findings
Significant-Gravitas/AutoGPT	Autonomous goal-driven agent with task decomposition	Show agent breaking down complex infra tasks

Take-home exercise

- Explore starter repositories
- Identify:
  - Triggers
  - Prompts
  - Guardrails
- See where:
  - Reasoning happens
  - Humans stay in the loop

Look for patterns over tools

Concluding thoughts from the industry

“The real power of AI agents is not automation—it’s delegated judgment with constraints.”
— Anthropic, *Claude Systems & Agent Design guidance, 2024–2025*

“AI should accelerate human decision-making, not replace accountability.”
— NIST, *AI Risk Management Framework*

“Cognitive overload is now the primary scalability limit in software delivery.”
— Gartner, *Platform Engineering & DevOps productivity reports*

“A platform without intelligence becomes the bottleneck in an AI-powered organization.”
— Thoughtworks, *Technology Radar & Platform Engineering insights, 2024*

Next steps

Continue Your Journey:

- *Explore all starter templates in the repository*
- *Join the Platformetrics community for updates*
- *Share your implementations and learnings*
- *Reference: Effective Platform Engineering book*
  - *(CHANKRAMATH45 for 45% off)*

Thank You!

Access all materials at

https://agentic-pe.platformetrics.com

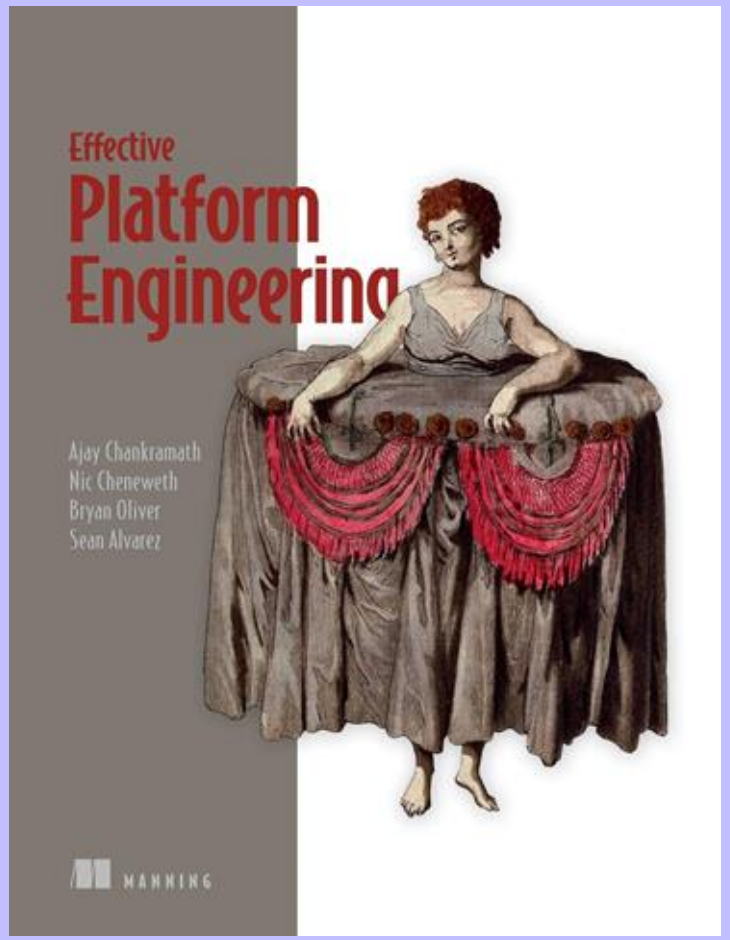
https://github.com/achankra/agentic-aiplatformengineering/blob/main/README.md

Questions?

Reach out via GitHub Discussions

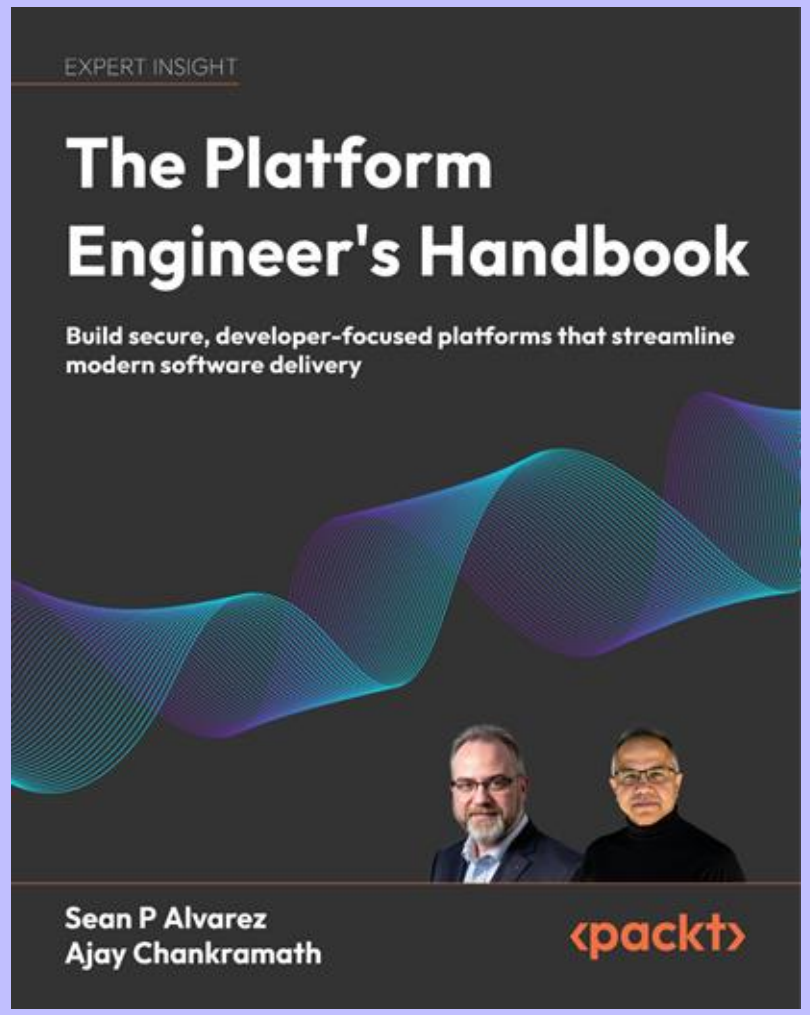
Engineering Platforms consistently enable ideal AI adoption within organizations. Learning how to do it well is critical for the success of your AI journey

45% | CHANKRAMATH45
40% | laplatform40



Pearson

Drill down another level to go into the basics of building your engineering platform with the companion book!



Domain-driven platform engineering explores a field of modern platform engineering hitherto untouched!

With Eamonn Ryan

Pearson

Stay in touch!

ajay@platformetrics.com

www.platformetrics.com

https://www.linkedin.com/in/chankramath/

https://cal.com/platformetrics

https://effectiveplatformengineering.com

@ajchan.bsky.social

Pearson