

A Cloud-Native AI Framework for Predictive Failure Detection and Self-Healing Distributed Infrastructure

Peter Mitrana

Department of Computer Science, University of Bucharest, Bucharest, Romania

Abstract

Cloud-native platforms already heal many failures on their own. Container orchestrators restart crashed containers, reschedule work away from failed machines, and continuously reconcile the running system with its declared state. What they cannot do is foresee failures that have not yet happened, or reason about problems that no simple rule covers, such as a slow dependency, a faulty configuration, or an unusual interaction between services. Two developments now address these gaps. Predictive anomaly detection learns the normal behavior of infrastructure and flags deviations early. Large language model (LLM) agents can investigate incidents by querying monitoring tools, reading logs and runbooks, forming hypotheses, and proposing or taking corrective actions. This article proposes a cloud-native AI framework that layers predictive detection and an AI operations agent on top of orchestrator self-healing. The agent acts through a tool gateway with tiered permissions; routine actions run automatically, consequential ones pass through a risk-governed workflow, and every step is recorded under audit-ready controls. Using a design-oriented approach grounded in eleven studies published between 2015 and 2025, the article maps the evidence base, defines layers of cloud-native self-healing, specifies agent tools and permissions, and maps agent risks to controls. It illustrates the design with a software-as-a-service payroll provider during its monthly payroll run. It argues that AI agents extend cloud-native self-healing safely only when their actions are bounded, verified, and accountable.

Keywords: cloud-native, self-healing infrastructure, predictive failure detection, anomaly detection, Kubernetes, container orchestration, AI agents, large language models, AIOps, incident management, tool use, prompt injection, risk-governed workflow, audit readiness

Introduction

Cloud-native infrastructure was designed with failure in mind. Google's Borg system showed how a cluster manager can run hundreds of thousands of jobs, restarting and rescheduling them automatically as machines fail (Verma et al., 2015). Container design patterns such as sidecars and adapters made it easier to build reliable distributed systems from reusable parts (Burns & Oppenheimer, 2016). Today's orchestrators continuously compare the actual state of the system with its declared state and correct any difference, which heals many common failures without human involvement.

This form of self-healing has two limits. First, it reacts: a container is restarted after it crashes, not before. Second, it follows fixed rules: it can restart a failed container but cannot work out why a service

is slow or which recent change caused an error. Such problems still require engineers to investigate.

Predictive detection addresses the first limit. Deep models of multivariate metrics can learn normal behavior and detect anomalies early (Su et al., 2019). The same principle underlies behavioral analytics in fraud detection, which learns each user's normal behavior and flags deviations (*Behavioral Analytics for Fraud Detection in Digital Payment Systems*, 2018); applied to infrastructure, each service and node has a behavioral baseline.

AI agents address the second limit. LLMs can interleave reasoning with actions, such as queries to external tools, and use the results to decide what to do next (Yao et al., 2023). They can also learn when and how to call tools (Schick et al., 2023). In cloud operations, LLMs have been evaluated for recommending root causes and mitigation steps for real incidents (Ahmed et al., 2023). Sannidhanam

(2025) examines autonomous AI agents for cloud infrastructure operations, in which agents monitor, diagnose, and remediate infrastructure issues.

Agents that act on production infrastructure bring new risks. Their reasoning can be wrong, and text they read, such as logs or tickets, may contain instructions that manipulate them, a form of indirect prompt injection (Greshake et al., 2023). Their actions must therefore be bounded and governed. Basireddy (2022) provides configurable enterprise workflow architecture for risk-based approval and escalation, and Basireddy (2023) proposes an audit-ready compliance platform architecture for enterprises using LLMs, which preserves the evidence of what an LLM saw, concluded, and did.

This article asks: *How can predictive detection and AI agents extend cloud-native self-healing in a way that remains safe and accountable?* It contributes an evidence map (Figures 1 and 2), a layered model of cloud-native self-healing, an integrated architecture, a specification of agent tools and permissions, a mapping of agent risks to controls, and an illustrative case.

Related Work

Cloud-native orchestration

Verma et al. (2015) describe Borg, Google's cluster manager, which achieves high availability through automatic rescheduling, replication, and isolation. Burns and Oppenheimer (2016) describe design patterns for container-based distributed systems,

including single-container, single-node, and multi-node patterns, which make reliability features reusable.

Predictive anomaly detection

Su et al. (2019) propose OmniAnomaly, a stochastic recurrent network that detects anomalies in multivariate time series and indicates which metrics contributed. The 2018 study of behavioral analytics applies models of normal user behavior to fraud detection in digital payments.

LLM agents and tool use

Yao et al. (2023) propose ReAct, in which language models alternate between reasoning steps and actions, using observations from tools to guide further reasoning. Schick et al. (2023) propose Toolformer, which teaches language models to decide which tools to call and how. Ahmed et al. (2023) evaluate LLMs for recommending root causes and mitigation steps for cloud incidents at a large cloud provider. Sannidhanam (2025) examines autonomous AI agents for cloud infrastructure operations.

Agent security

Greshake et al. (2023) demonstrate indirect prompt injection, in which instructions embedded in content retrieved by an LLM-integrated application can take control of its behavior.

Governance and audit

Basireddy (2022) proposes configurable enterprise workflow architecture for risk management, with configurable approvals, escalation, and audit trails.

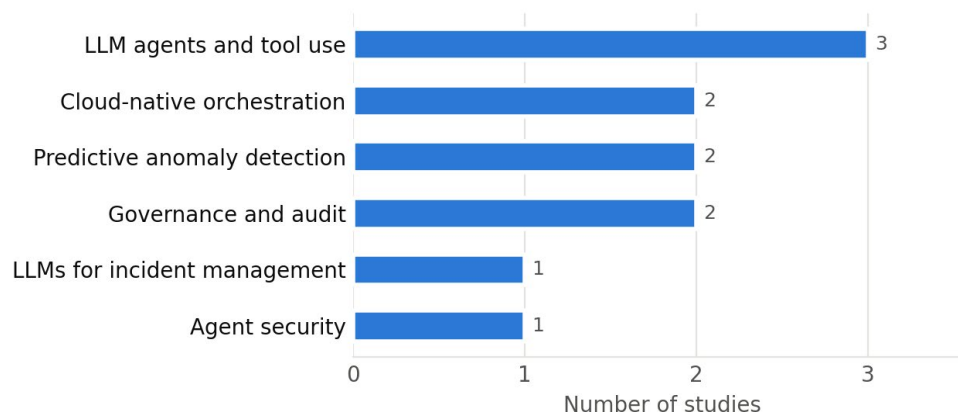


Figure 1: Number of reviewed studies primarily supporting each theme (n = 11)

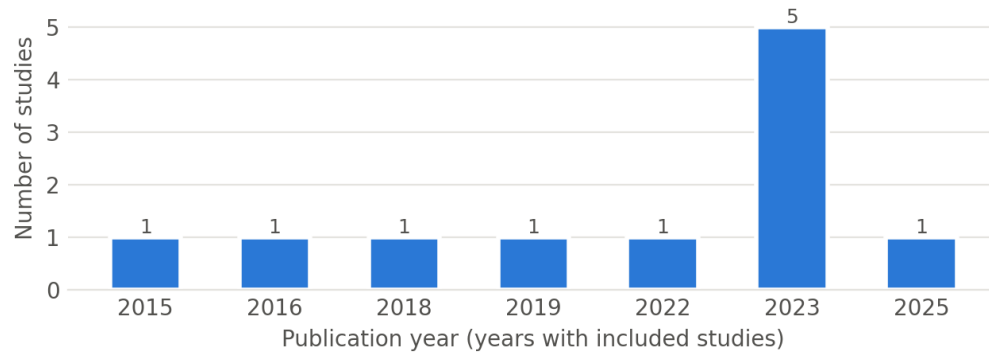


Figure 2: Reviewed studies by publication year (n = 11)

Basireddy (2023) proposes an audit-ready compliance platform architecture for regulated enterprises using LLMs.

Research gap

Cloud-native self-healing is reactive and rule-based. Predictive detection and LLM-based incident analysis are studied separately, and agent research rarely addresses the permissions and governance needed on production infrastructure. Few designs combine orchestrator self-healing, predictive detection, and governed AI agents in one framework. This article addresses that gap.

Research Methodology

The study follows a design-oriented research approach. The problem, the limits of reactive, rule-based self-healing and the risks of autonomous agents, was identified from the literature. Requirements were derived from the reviewed studies: early detection,

investigation of novel problems, bounded actions, protection against manipulated inputs, human approval for consequential actions, and complete records. The framework was designed from eleven studies published between 2015 and 2025 and demonstrated through an illustrative case.

To make the evidence base transparent, each study was assigned to the theme it primarily supports. Figure 1 shows the distribution across themes, and Figure 2 shows the studies by publication year. These figures describe the literature used in this article. They are not performance measurements. The framework has not been evaluated empirically, which is left to future work.

LLM agents and tool use have the most studies, with three. Cloud-native orchestration, predictive anomaly detection, and governance and audit have two each. LLMs for incident management and agent security have one each. Most studies were published in 2023, reflecting how recent agent research is.

Table 1: Layers of cloud-native self-healing

Layer	Mechanism	What it heals	Limitation	Supporting studies
Orchestrator	Health checks, restart, reschedule	Crashed containers, failed machines	Reactive; fixed rules	Verma et al. (2015)
Platform patterns	Sidecars, replicas, reconciliation	Local faults via reusable components	Cannot reason about causes	Burns & Oppenheimer (2016)
Predictive detection	Behavioral baselines, anomaly models	Emerging problems before failure	Detects but does not diagnose	Su et al. (2019); Behavioral analytics (2018)
AI operations agent	Reasoning with tools; runbooks	Novel problems needing investigation	Can be wrong or manipulated	Yao et al. (2023); Ahmed et al. (2023); Sannidhanam (2025)
Governed human decision	Approval and escalation	Consequential or uncertain actions	Slower	Basireddy (2022)

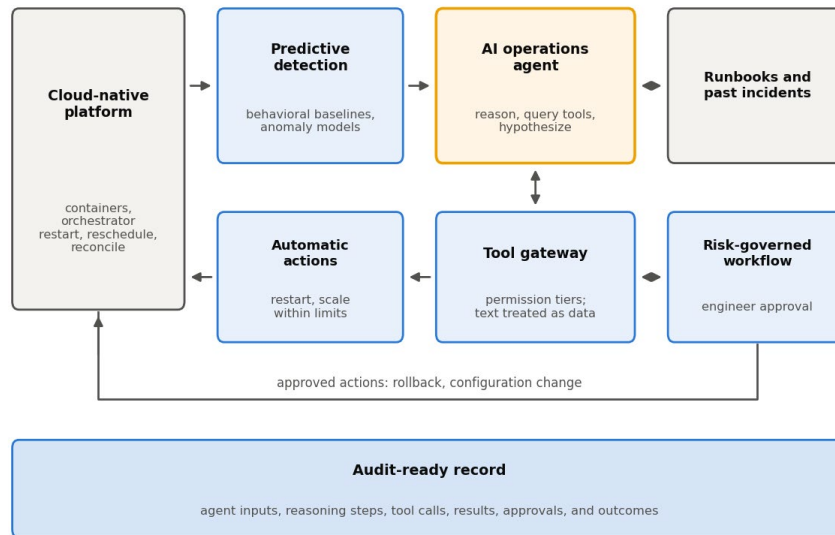


Figure 3: shows the framework.

Layers of Cloud-Native Self-Healing

The framework treats self-healing as a set of layers, each handling what the layer below cannot (Table 1).

The layers are complementary. Most failures are still handled by the orchestrator in seconds. Predictive detection catches problems earlier. The agent handles problems that need investigation, and people decide where the stakes are high.

Integrated Framework

Architecture

Figure 3. Cloud-native AI framework for predictive failure detection and self-healing

- Cloud-native platform. Services run in containers on an orchestrator that restarts, reschedules, and reconciles them (Verma et al., 2015; Burns & Oppenheimer, 2016).
- Predictive detection. Behavioral baselines are learned for each service and node, and multivariate anomaly models flag deviations with the metrics that contributed (Su et al., 2019; Behavioral analytics, 2018).
- AI operations agent. When predictive detection raises an issue that the orchestrator cannot resolve, the agent investigates. It reasons step by step, calling tools to query metrics, read logs, inspect recent changes, and consult runbooks,

and forms a hypothesis about the cause and a mitigation (Yao et al., 2023; Schick et al., 2023; Ahmed et al., 2023; Sannidhanam, 2025).

- Tool gateway. The agent cannot act directly on infrastructure. Every tool call passes through a gateway that checks the agent's permissions (Table 2) and treats text read from logs, tickets, and other sources as data, not instructions (Greshake et al., 2023).
- Risk-governed workflow. Actions above the automatic tier are sent to an engineer with the agent's reasoning and evidence (Basireddy, 2022).
- Audit-ready record. Every agent input, reasoning step, tool call, result, and decision is recorded (Basireddy, 2023).

Agent tools and permissions

Table 2 shows how tools are tiered by risk.

Tiering follows a simple principle: the agent may observe freely, may take small reversible actions automatically, must ask before consequential actions, and cannot take destructive ones at all.

Risks and Controls

Agent risks and controls

Table 3 maps the main risks of AI agents in infrastructure operations to controls in the framework.

Table 2: Agent tools and permission tiers

<i>Tool</i>	<i>Purpose</i>	<i>Permission</i>	<i>Guardrail</i>
Query metrics, logs, traces	Investigate	Automatic	Read-only; results treated as data
Search runbooks and past incidents	Find known fixes	Automatic	Approved sources only
Restart or reschedule a container	Clear local faults	Automatic	Rate-limited; health verified after
Scale a deployment	Relieve load	Automatic within limits	Minimum and maximum set per service
Roll back a release	Undo a faulty change	Engineer approval	Previous version must be healthy
Change configuration or network rules	Fix misconfiguration	Engineer approval	Change reviewed; reversible
Delete data or resources	Not allowed	Forbidden	Not exposed to the agent

Agent loop

Figure 4 shows the loop through which the agent works.

Predictive detection detects an emerging issue. The agent investigates by querying tools and hypothesizes a cause and mitigation. It then acts within its permissions; this step is highlighted because it is where the tool gateway and workflow bound the agent's power. The agent verifies that the action helped, and the whole episode is recorded, informing runbooks and future investigations (Basireddy, 2023).

Illustrative Case: A Payroll Platform on Pay Day

Consider a software-as-a-service payroll provider that calculates and pays salaries for thousands of employers. On the last working day of each month, load rises sharply as payroll runs are submitted, and any failure can delay people's pay. The case is illustrative and reports no measured results.

Orchestrator layer

During the morning, two calculation containers crash because of a known memory issue. The orchestrator restarts them within seconds (Verma et al., 2015). No other layer is involved.

Predictive detection

At midday, the payment file service shows a pattern of rising latency and queue depth that departs from its behavioral baseline for pay day (Behavioral analytics, 2018). The anomaly model flags it and indicates that database connection wait time contributes most (Su et al., 2019).

Agent investigation

The agent queries metrics and finds that connection waits began shortly after a configuration change that reduced the database connection pool size. Logs show many "pool exhausted" errors. A runbook describes this pattern (Yao et al., 2023; Ahmed et al., 2023; Sannidhanam, 2025). One log line, written by a misbehaving test tool, contains text resembling an

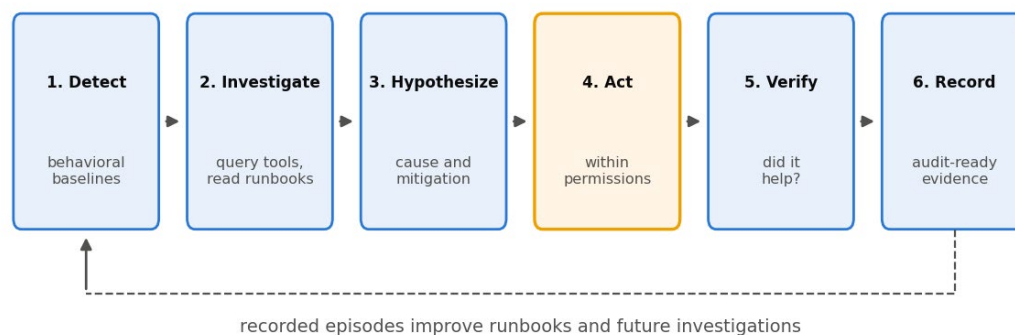
**Figure 4:** Agent loop for predictive self-healing

Table 3: Risks of AI agents in operations and framework controls

<i>Risk</i>	<i>Example</i>	<i>Control</i>	<i>Supporting studies</i>
Wrong diagnosis	Agent blames a healthy database	Evidence required for each hypothesis; verification after action	Ahmed et al. (2023); Yao et al. (2023)
Prompt injection	A log line contains instructions to delete resources	Retrieved text treated as data; destructive tools not exposed	Greshake et al. (2023)
Excessive action	Agent restarts many services in a loop	Rate limits; permission tiers	Basireddy (2022)
Missing accountability	No record of why an action was taken	Audit-ready record of reasoning and actions	Basireddy (2023)
Over-trust	Engineers approve without reading	Reasoning and evidence shown with every approval request	Basireddy (2022); Sannidhanam (2025)

instruction to “clear all queues”; the gateway treats it as data and the agent does not act on it (Greshake et al., 2023).

Governed action

The agent scales the payment file service within its limits, an automatic action, to reduce queue growth. It then proposes reverting the configuration change, which requires approval. The engineer sees the agent’s reasoning, the evidence, and the proposed change, and approves (Basireddy, 2022). Latency returns to normal and the agent verifies recovery.

Record

The entire episode, including the ignored log line, is recorded for audit and review (Basireddy, 2023). The runbook is updated with a check on pool size during pay-day preparation.

The case shows the layers working together: the orchestrator handling routine crashes, prediction catching an emerging problem, the agent diagnosing it, and governance keeping a consequential change under human control.

Discussion

Implications

For research, the article positions AI agents as a layer above cloud-native self-healing and predictive detection, and specifies the permissions, guardrails, and records needed to deploy them safely. For practitioners, the lessons are to let the orchestrator handle what it can; to add behavioral baselines for early detection; to let agents observe freely but act only through a gateway with tiered permissions; to

treat all text agents read as data; to require approval with evidence for consequential actions; and to record everything the agent does.

Conclusion

Cloud-native platforms heal many failures automatically, but only reactively and only by rule. This article proposed a framework that adds predictive detection based on behavioral baselines and an AI operations agent that investigates and remediates problems through a permission-tiered tool gateway, with consequential actions governed by workflow and every step recorded for audit. The payroll case showed the layers cooperating on pay day. The central conclusion is that AI agents extend cloud-native self-healing safely only when their actions are bounded, verified, and accountable.

References

1. Ahmed, T., Ghosh, S., Bansal, C., Zimmermann, T., Zhang, X., & Rajmohan, S. (2023). Recommending root-cause and mitigation steps for cloud incidents using large language models. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering* (pp. 1737–1749).
2. Basireddy, S. R. (2022). Digitizing risk management through configurable enterprise workflow architecture. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, 14(4), 231–239.
3. Basireddy, S. (2023). Audit-ready compliance platform architecture for large language model regulated enterprises. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, 15(01), 226–232. <https://doi.org/10.18090/samriddhi.v15i01.40>
4. Sannidhanam, A.H., Alladi, S. (2017). Behavioral Analytics for Fraud Detection in Digital Payment Systems. *International Journal of Engineering and Science Research*, 8(3), 01-10. <https://ijesr.org/index.php/ijesr/article/view/1849>
5. Burns, B., & Oppenheimer, D. (2016). Design patterns for

- container-based distributed systems. In *Proceedings of the 8th USENIX Workshop on Hot Topics in Cloud Computing*.
6. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security* (pp. 79–90).
 7. Sannidhanam, A. H. (2025). Autonomous AI agents for cloud infrastructure operations. *Journal of Contemporary Science and Technology Management*, 1(01), 83–100. <https://jcstm.info/index.php/jcstm/article/view/20>
 8. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems* 36.
 9. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., & Pei, D. (2019). Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2828–2837).
 10. Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems*, Article 18.
 11. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*.