KETRIAGE: INCIDENT TRIAGE WITH EVOLVING HIERARCHICAL SYMBOLIC KNOWLEDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Cloud incidents can degrade user experience and cause financial losses, while identifying the responsible service through incident triage remains challenging due to large blast radii and complex fault propagation. Despite extensive efforts, existing approaches still face two fundamental challenges: the absence of integration mechanisms for domain knowledge and brittle adaptation to evolving systems. We present KETriage, a knowledge-enhanced and self-improving agentic framework that identifies the responsible service. To bridge the knowledge gap of LLM agents in incident triage, KETriage constructs a hierarchical symbolic knowledge base (HSKB) offline, a symbolic graph organized into ontology, instance, and case layers. During online inference, KETriage anchors incident terminologies to the HSKB ontology taxonomy and retrieves relevant knowledge segments to identify the responsible service. Given the inherent dynamic nature of cloud systems, KETriage further introduces post-resolution knowledge consolidation, enabling continual experience evolution without prohibitive retraining. We conducted comprehensive experiments using real-world incidents collected from an industrial cloud service system serving millions of users. The experimental results show that KETriage outperforms existing methods, achieving Acc@1 of 79.48% and 93.66% Acc@3 in incident triage.

1 INTRODUCTION

Cloud systems are vulnerable to incidents due to complex service dependencies and dynamic environments (Lee et al., 2023). Incident triage identifies the responsible service from numerous candidates (Yu et al., 2025), aiming to reduce the time to engagement (TTE) of relevant development teams (Beyer et al., 2016; Varanasi, 2025).

As illustrated in Figure 1, the service exhibiting the observed impact may differ from the responsible service, and an incident may be transferred across multiple services before the responsible service is identified (Wu et al., 2020). However, even for veteran on-call engineers (OCEs), incident triage remains a laborious and error-prone endeavor due to the large blast radius of incidents and intricate fault propagation (Deng et al., 2024; Xie et al., 2026; Wang et al., 2018; Liu et al., 2021). For example, a Google Cloud outage in 2025 disrupted more than 50 dependent services in 40+ regions, making the responsible service difficult to localize (Google Cloud, 2025). Misdi-

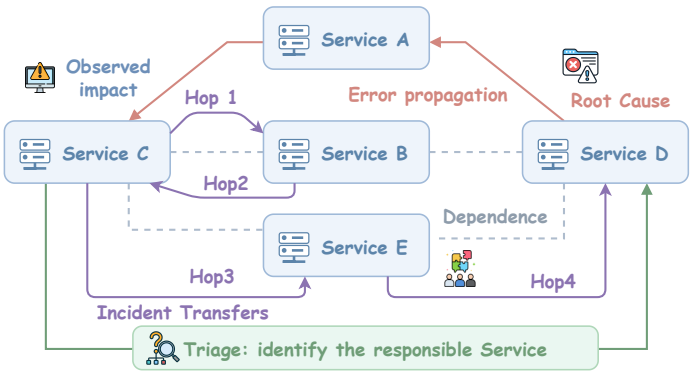


Figure 1: Illustration of incident triage. An observed impact may propagate across services, while triage aims to identify the responsible service.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rected incidents further increase TTE through redundant transfers, highlighting the need for automated and accurate incident triage (Chen et al., 2019a).

Significant efforts have been dedicated to incident triage through heuristic rules (Walek, 2017) and learning-based methods (Pham et al., 2020; Chen et al., 2019b; Han & Sun, 2020), striving to reduce manual effort. However, rule-based methods still require periodic manual maintenance and validation. Despite the success of learning-based approaches in accuracy, they lack interpretable inference chains that OCEs find convincing (Chen et al., 2024). OCEs often have to devote extra effort to validating triage results against multiple data sources. In addition, manual incident labeling remains an arduous task, which inevitably imposes additional burdens on OCEs.

Recent advances in large language models (LLMs) for software engineering (Xu et al., 2025) have motivated incident triage approaches that leverage LLMs' semantic understanding to provide transparent reasoning (Wang et al., 2024b; Yao et al., 2026; Liu et al., 2026; 2025; Ahmed et al., 2023; Chen et al., 2024). However, incident triage is essentially an evidence extraction and completion process (Luo et al., 2026). Specifically, the initial incident is often accompanied by many concurrent incidents and plausible dependencies (Chen et al., 2021). Key evidence is difficult to extract from such an overwhelming context (Liu et al., 2024). Furthermore, critical evidence such as service boundaries, cross-service dependencies, and failure patterns is scattered across numerous private domain documents. The reasoning process requires the continuous retrieval and integration of such evidence, which poses challenges for single-turn LLMs that cannot perform such iterative retrieval and integration.

To address this long-horizon task, existing methods employ agents that interact with software tools and domain resources (Jiang et al., 2020; Wang et al., 2024a; Yu et al., 2025). Despite their competitive performance, several challenges remain:

- **The absence of integration mechanisms for domain knowledge.** Incident triage requires knowledge of service functions, responsibility boundaries, and cross-service dependencies scattered across heterogeneous private documents. Although RAG can ingest these documents (Lewis et al., 2020; Guu et al., 2020; Edge et al., 2024), flat document organization lacks explicit relational links for multi-hop retrieval and causes granularity mismatch by mixing knowledge at different abstraction levels.
- **Brittle adaptation to evolving systems.** Software updates, incident mitigation, and deployment adjustments continually reshape service boundaries, dependencies, and fault patterns in cloud systems. However, current LLM-based agents struggle to accumulate reusable experience from resolved incidents, limiting the sustained growth of their cognitive and troubleshooting abilities.

These challenges motivate us to propose KETriage, a knowledge-enhanced and self-improving agentic framework for responsible-service localization in cloud incident triage. KETriage constructs a hierarchical symbolic knowledge base (HSKB) from resolved incidents, organizing domain concepts, operational knowledge, and historical evidence into ontology, instance, and case layers. During online triage, KETriage uses the HSKB to interpret incident terminology, retrieve relevant knowledge, and identify the responsible service with cited rationales. After an incident is independently resolved, KETriage consolidates reusable knowledge from the confirmed resolution records into the HSKB, enabling continual knowledge evolution without online retraining. The major contributions of this work are summarized as follows:

- We propose KETriage, a knowledge-enhanced and self-improving agentic framework that localizes the responsible service in cloud incident triage. It pairs HSKB, a symbolic graph over ontology, instance, and case layers, with post-resolution knowledge consolidation that continuously assimilates domain knowledge from resolved incidents.
- We anchor incident terminologies to the HSKB ontology taxonomy to resolve aliases and ambiguity. Moreover, we build a knowledge-augmented triage workflow that retrieves knowledge segments from the HSKB to pinpoint the responsible service.
- We evaluated KETriage on an industrial cloud platform that serves millions of users. The experimental results show that KETriage reaches 79.48% overall Acc@1 and 93.66% overall Acc@3, significantly reducing the time to engagement and providing compelling reasoning chains.

2 Related Works

Automated incident triage has been extensively studied to reduce the manual effort required to identify and route incidents to responsible teams. Rule-based approaches have been used for incident prioritization (Walek, 2017), while learning-based methods learn routing patterns from historical incidents and operational signals. DeepTriage combines multiple learning models to recommend appropriate transfer targets for cloud incidents (Pham et al., 2020). DeepCT formulates incident triage as a continuous process and progressively refines assignment decisions as new discussion information becomes available (Chen et al., 2019b). DeepRouting further learns routing patterns from ticket text and historical ticket-routing paths to support automated assignment (Han & Sun, 2020). These methods demonstrate the effectiveness of exploiting historical operational data.

More recently, LLMs have been increasingly adopted for cloud incident triage because of their strong capability to understand heterogeneous and unstructured operational evidence. COMET extracts semantic information from incident reports and incorporates domain knowledge to support interpretable triage (Wang et al., 2024b). TickIt applies LLMs to automated ticket escalation (Liu et al., 2025). Intelligent Triage extracts reusable triage reasoning from historical incident resolutions to construct interpretable triage workflows (Liu et al., 2026). Triangle introduces multiple LLM agents that reason over decentralized team knowledge and coordinate through negotiation for incident triage (Yu et al., 2025). Comfey adopts a decentralized agentic framework in which team-specific agents collaborate to route incidents across production cloud infrastructure (Yao et al., 2026). Collectively, these studies show a clear transition from fixed prediction models toward more flexible LLM- and agent-based triage systems. Cross-service incident triage must distinguish the impacted service from the responsible service when concurrent incidents and service dependencies suggest multiple plausible candidates. KETriage organizes operational knowledge into ontology, instance, and case layers while keeping current incident evidence separate, using shared semantics to support candidate construction and evidence-constrained attribution. Offline updates incorporate records from independently resolved incidents into subsequent knowledge revisions, excluding the agent's own predictions and keeping model parameters fixed.

3 Problem Formulation and Method

3.1 Problem Definition

Let $\mathcal{S}$ denote the set of services in a cloud system. An incident I_i manifests at an impacted service $s_i^{\text{imp}} \in \mathcal{S}$, and its ground-truth responsible service is $y_i \in \mathcal{S}$. For a self-inflicted incident, $y_i = s_i^{\text{imp}}$. For a cross-service incident, $y_i \neq s_i^{\text{imp}}$, as failures propagate across service boundaries. The input of incident I_i is represented as $x_i = (m_i, h_i, \mathcal{N}_i)$, where m_i contains the incident description and operational metadata, h_i contains alerts and unhealthy-service signals, and $\mathcal{N}_i = \{I_{i,j}\}_{j=1}^{n_i}$ with $n_i = |\mathcal{N}_i|$ is the set of concurrent incidents collected from the same operational window as I_i. These incidents can reveal upstream failures that contribute to the observed impact of I_i and provide candidate sources for responsible-service triage.

Given x_i, incident triage is formulated as a mapping $f : \mathcal{X} \to \mathcal{S}$ that predicts one responsible service:

$$\hat{y}_i = f(x_i), \qquad \hat{y}_i \in \mathcal{S}. \tag{1}$$

The prediction is correct when $\hat{y}_i = y_i$. The second objective is to reduce the time to engagement (TTE). Let t_i^{rep} denote the incident reporting time and t_i^{eng} the time at which the responsible service is engaged. We define $\text{TTE}_i = t_i^{\text{eng}} - t_i^{\text{rep}}$. Accurate responsible-service triage can direct an incident to the correct service earlier and thereby reduce TTE_i.

3.2 Hierarchical Symbolic Knowledge Base Construction

As illustrated in Figure 2, The HSKB is a symbolic knowledge base organized into ontology, instance, and case layers. After each round j, it is represented as:

$$\mathcal{K}_j = (\mathcal{O}_j, \mathcal{G}_j, \mathcal{H}_j, \pi_j), \tag{2}$$

where $\mathcal{O}_j$ defines the shared schema, G_j stores operational knowledge instantiated under this schema, $\mathcal{H}_j$ retains resolved incident cases, and π_j records the provenance linking knowledge claims

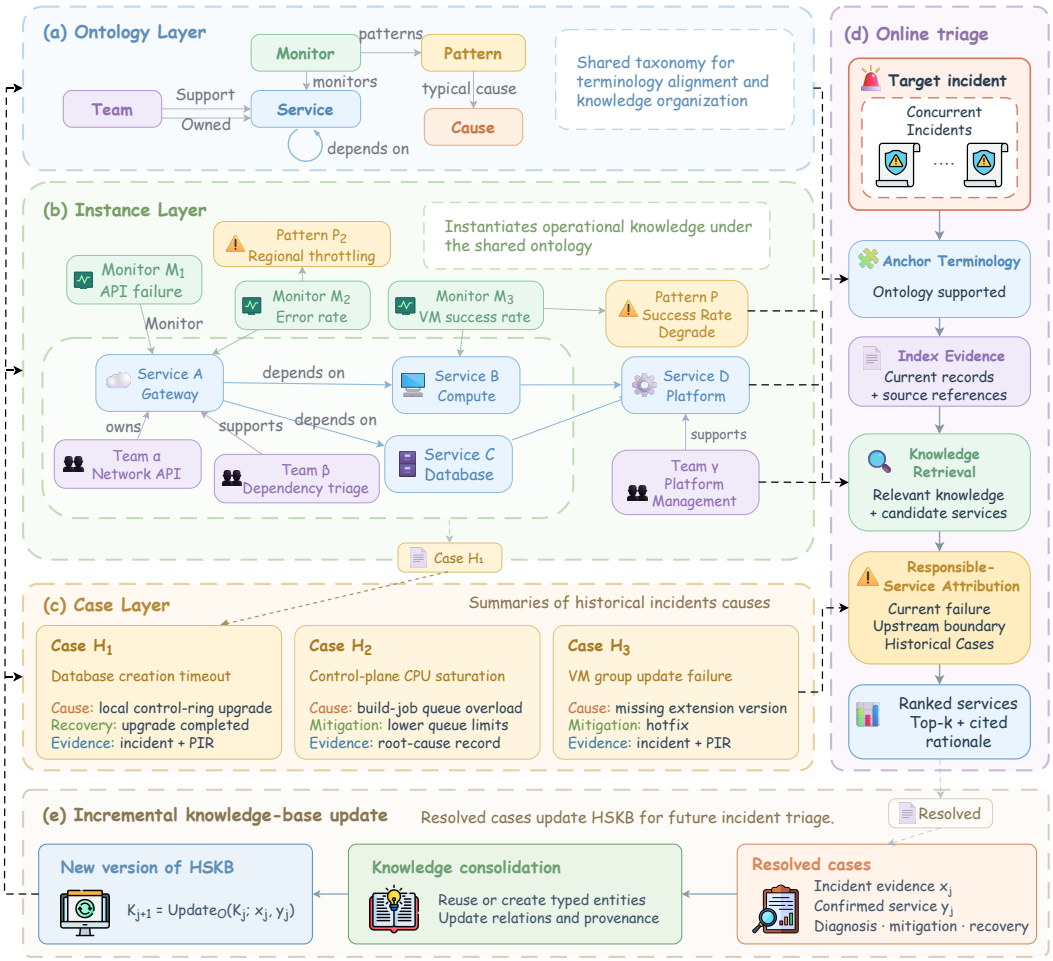


Figure 2: HSKB organization and online triage. (a) The ontology defines a taxonomy of top-level schemas. (b) The instance layer instantiates domain entities and their links, and the dashed outline highlights an illustrative knowledge segment. (c) Historical cases retain causes, mitigation or recovery, and source evidence. (d) Online triage anchors terminology to the taxonomy, indexes current evidence, retrieves knowledge and candidate services, filters concurrent incidents, and produces ranked attribution with cited rationales. (e) Resolved cases are consolidated into the HSKB.

to their supporting cases. Given a small expert-seeded schema, the HSKB can automatically expand by leveraging resolved incidents.

Ontology layer. The ontology layer defines the shared schema that governs the HSKB. We formalize it as:

$$\mathcal{O}_j = (\mathcal{T}, \mathcal{R}, \mathcal{P}_j, \mathcal{F}_j), \tag{3}$$

where $\mathcal{T}$ denotes object types, $\mathcal{R}$ is the links that connect them, $\mathcal{P}_j$ represents the properties of these types, and $\mathcal{F}_j$ contains the functions that operate over the properties and links. Note that $\mathcal{T}$ and $\mathcal{R}$ are supplied by experienced OCEs, while the $\mathcal{P}_j$ and $\mathcal{F}_j$ can be dynamically induced based on the instance layer and case layer at each round. As illustrated in Figure 2(a), the $\mathcal{T}$ includes Team, Service, Monitor, Pattern, and Cause. A team may own or support a service, a monitor observes a service and indicates a failure pattern, a pattern may have a typical cause, and a service may depend on another service.

Instance layer. The instance layer stores concrete operational entities and their relations under the ontology. At round j, we build a knowledge graph $\mathcal{G}_j$:

$$G_j = (V_j, E_j, \tau_j), \qquad \tau_j : V_j \to \mathcal{T}, \tag{4}$$

where V_j contains operational entities, τ_j maps each entity to its object type, and E_j contains the typed relations admissible under the ontology, As shown in Figure 2(b), service instances are connected by depends-on relations, while monitors and teams are attached through typed relations such as monitors, owns, and supports.

Case layer. The case layer $\mathcal{H}_j$ preserves real resolved incidents as source-linked reasoning references for the triage agent. As illustrated in Figure 2(c), each case records how an observed symptom was connected to its responsible service and cause, together with the mitigation or recovery action and the supporting evidence. The provenance mapping π_j connects knowledge claims in the instance layer to the resolved cases that support them. These resolved cases provide concrete reasoning examples that help the agent interpret similar signals, compare candidate explanations, and construct a plausible attribution path during online triage.

3.3 Knowledge-Augmented Online Incident Triage

Given a target incident I_i, the online triage agent interacts with the fixed HSKB revision $\mathcal{K}_j$ to identify its responsible service. As shown in Figure 2(d), the workflow anchors incident terminologies to the ontology taxonomy, indexes current evidence, retrieves relevant knowledge segments, filters concurrent incidents, and produces a ranked attribution with cited rationales.

The agent indexes the target and concurrent incidents with their source references. The agent then consults HSKB service pages and aliases to interpret entity mentions as U_i, then retrieves a local knowledge segment $p_i = \mathrm{Retrieve}(\mathcal{K}_j, U_i)$. This context guides scope filtering, $\widetilde{\mathcal{N}}_i = \mathrm{Filter}(\mathcal{N}_i; m_i, h_i, p_i)$, which retains incidents plausibly connected to the target

The agent constructs the candidate service set $\mathcal{S}_i$ from the retained incidents and services suggested by current evidence and HSKB. For each external candidate s, it assesses $A_i(s) = F_i(s) \wedge B_i(s) \wedge D_i(s)$, where $F_i(s)$ requires evidence of current failure, $B_i(s)$ requires an external upstream relationship to the target, and $D_i(s)$ requires an incident-specific connection consistent with the known dependency. Historical cases help interpret this connection but cannot establish current responsibility on their own. The agent ranks candidates by evidential support and rechecks the leading candidates through focused HSKB queries, retaining the target-local alternative when no external explanation is sufficiently supported. It returns the ranked service list $\hat{\boldsymbol{\ell}}_i$ with Top-1 prediction $\hat{y}_i = \hat{\boldsymbol{\ell}}_{i,1}$ and a cited rationale $\hat{\mathbf{r}}_i$ that states any unresolved gaps.

3.4 Incremental Knowledge-Base Update

HSKB is updated offline after an incident is independently resolved. As shown in Figure 2(e), the update uses the incident evidence $x_i = (m_i, h_i, \mathcal{N}_i)$ together with the independently confirmed responsible service y_i and their associated resolution records; the online prediction $\hat{y}_i$ and rationale $\hat{r}_i$ are excluded. After sensitive information is redacted, this evidence is preserved as an immutable record and summarized as a source-linked case in $\mathcal{H}_{j+1}$. The consolidation process extracts reusable claims from this case and updates the corresponding entities and relations in G_j under the ontology $\mathcal{O}$. Existing monitor families, symptom patterns, and root-cause mechanisms are extended whenever possible; new entities are created only when no suitable identity exists. Incident-specific evidence remains in the case layer, while ambiguous or conflicting claims retain their respective sources through π_{j+1}. The update is summarized as

$$\mathcal{K}_{j+1} = \mathrm{Update}_{\mathcal{O}}(\mathcal{K}_j; x_i, y_i). \tag{5}$$

The new revision is available only to subsequent incidents, while I_i retains the fixed revision $\mathcal{K}_j$ used for its inference. Thus, HSKB evolves through source-linked symbolic memory without online retraining or self-reinforcement.

4 Experiments

4.1 Experimental Setup

Dataset and Execution We evaluate KETriage on 804 resolved incidents from an industrial cloud platform, created between June 16 and July 16, 2026 (UTC). The dataset comprises 609 self-inflicted

Table 1: Responsible-service prediction (%). All denotes the complete test set ($N = 804$). Self and Cross denote self-inflicted and cross-service incidents; Data and Control denote the operational-plane subsets. Columns @1 and @3 report Acc@1 and Acc@3; Avg and MRR report Avg@3 and MRR@3. Ablation abbreviations are defined in Section 4.3. Bold indicates the largest value in each column, including ties.

	All				Self		Cross		Data		Control	
Method	@1	@3	Avg	MRR	@1	@3	@1	@3	@1	@3	@1	@3
DeepTriage	77.36	86.19	82.26	81.28	**95.07**	98.69	22.05	47.18	80.90	88.04	63.12	78.75
DeepCT	70.77	85.07	78.86	77.20	83.74	93.60	30.26	58.46	76.24	90.99	48.75	61.25
COMET	65.05	84.33	75.75	73.61	79.31	93.43	20.51	55.90	72.52	88.98	35.00	65.62
TRIANGLE	72.39	88.68	80.85	79.33	89.49	97.04	18.97	62.56	78.57	93.79	47.50	68.12
KETriage	**79.48**	**93.66**	**87.89**	**86.05**	93.43	99.84	**35.90**	**74.36**	**83.39**	95.50	**63.75**	**86.25**
w/o CS	75.62	86.69	82.01	80.66	90.15	99.84	30.26	45.64	80.59	87.58	55.62	83.12
w/o HSKB	75.50	87.81	81.88	80.74	89.98	98.52	30.26	54.36	80.43	89.91	55.62	79.38
w/o EI	77.24	**93.66**	87.02	84.87	90.97	**100.00**	34.36	73.85	81.68	**95.81**	59.38	85.00
w/o HC	70.52	81.47	76.08	75.12	86.21	88.83	21.54	58.46	74.22	83.23	55.62	74.38

and 195 cross-service incidents. In self-inflicted incidents, the impacted service is also the responsible service. In cross-service incidents, the responsible service differs from the impacted service, as failures propagate across service boundaries. Along a separate dimension, the operational-plane labels identify 644 data-plane incidents and 160 control-plane incidents. Data-plane incidents concern workload execution and actual data processing or transfer, such as storage reads and writes. Control-plane incidents concern resource orchestration, configuration, management, and policy decisions, such as resource provisioning and configuration updates. Thus, Self/Cross distinguishes responsibility relative to the impacted service, whereas Data/Control distinguishes the operational function involved; each incident belongs to one category in each dimension. The HSKB used for evaluation has a declared historical cutoff of June 15, 2026, and remains fixed throughout each evaluation run. The experiments use `gpt-5.6-sol` with `codex` and medium reasoning effort. We evaluate one output per incident for each configuration across all 804 test incidents.

Baselines. We compare KETriage with four baselines:

- **DeepTriage** Pham et al. (2020): Combines boosted trees, similar-incident retrieval, and a CNN that processes incident text and context.
- **DeepCT** Chen et al. (2019b): Uses a GRU-based model with attention to incrementally process incident discussions along with incident textual and environment features. For our evaluation, we report its prediction after incorporating the last available discussion within the first 10 minutes.
- **COMET** Wang et al. (2024b): Extracts keywords and uses FastText embeddings to retrieve similar incidents.
- **TRIANGLE** Yu et al. (2025): Uses semantic distillation and multi-agent negotiation to identify the responsible service.

Metrics. We measure whether the responsible service appears among the first one, two, or three predictions. For an evaluation subset D, let H_k be the number of incidents whose responsible service appears in the first k positions. We report $\mathrm{Acc@}k = 100H_k/|D|$ for $k \in \{1, 2, 3\}$, the average accuracy $\mathrm{Avg@3} = (\mathrm{Acc@1} + \mathrm{Acc@2} + \mathrm{Acc@3})/3$, and the mean reciprocal rank within the first three positions, $\mathrm{MRR@3} = 100[H_1 + (H_2 - H_1)/2 + (H_3 - H_2)/3]/|D|$. Acc@3 measures how often the candidate list includes the responsible service, while MRR@3 gives more credit when it appears earlier in the ranking.

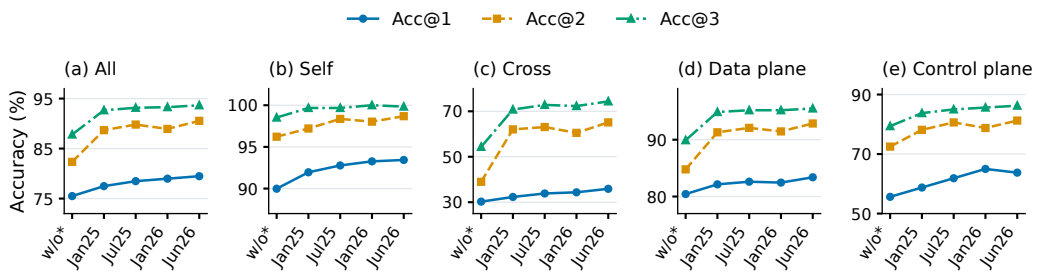


Figure 3: HSKB evolution on fixed cohorts: (a) All ($N = 804$), (b) Self ($N = 609$), (c) Cross ($N = 195$), (d) Data plane ($N = 644$), and (e) Control plane ($N = 160$). The horizontal axis shows w/o HSKB followed by four knowledge revisions, connected for each Acc@1–3 curve. Panels use different vertical scales.

4.2 Overall Comparison

Table 1 shows that KETriage achieves 79.48% overall Acc@1, exceeding the strongest learned baseline, DeepTriage, by 2.12 percentage points. Its advantage becomes larger when the ranked output is considered: KETriage reaches 93.66% Acc@3 and includes the responsible service among the first three candidates in 753 of 804 incidents. It exceeds TRIANGLE, the strongest learned baseline in overall shortlist coverage, by 4.98 percentage points. KETriage also obtains the highest overall MRR@3 of 86.05%, showing that its coverage does not come from merely appending plausible services to the list; the responsible service is generally placed near the beginning of the ranking. These results provide complementary evidence for both first-choice routing and shortlist-assisted investigation.

On cross-service incidents, KETriage achieves 35.90% Acc@1 and 74.36% Acc@3, compared with 62.56% Acc@3 for the strongest learned baseline, TRIANGLE. These results demonstrate the effectiveness of our knowledge-base-guided approach in identifying responsible upstream services and retaining them within a compact shortlist.

4.3 Component Ablations

Table 1 compares the complete system with four ablations, where "w/o" denotes removal of a component: CS denotes Concurrent Scope; HSKB denotes the Hierarchical Symbolic Knowledge Base; EI denotes Evidence Indexing; and HC denotes HSKB Construction. The w/o HSKB variant removes access to the knowledge base, whereas w/o HC retains access to the raw evidence without the constructed knowledge base.

The effects differ across components. Removing CS reduces cross-service Acc@3 from 74.36% to 45.64%, showing that neighboring failures contribute candidate upstream services that are often absent from the target record. Removing HSKB lowers cross-service Acc@3 to 54.36%, indicating that current failures alone are insufficient when their dependency direction or ownership boundary is unclear. Removing EI has a smaller effect on shortlist coverage, but lowers overall Acc@1 from 79.48% to 77.24% and MRR@3 from 86.05% to 84.87%; organizing current records and source references therefore primarily helps the agent assess and prioritize candidates. The largest degradation occurs without HC: overall Acc@1 and Acc@3 fall by 8.96 and 12.19 percentage points, respectively, and cross-service Acc@3 falls to 58.46%. Direct access to the same raw evidence does not recover the terminology alignment, relational organization, and reusable historical context supplied by the constructed HSKB.

Table 2: Online execution cost.

Method	Commands	Time (s)
KETriage	**22.92**	**190.08**
w/o HC	33.21	679.79

Table 2 further shows the effect of HSKB construction on online execution. Under the same model, reasoning effort, and 804-incident test set, the complete system reduces the average command count

from 33.21 to 22.92, a reduction of 31.0%, and shortens the mean execution time from 679.79 to 190.08 seconds, a reduction of 72.0%. The ontology, instance, and case layers provide explicit identities and traversal paths, allowing the agent to retrieve relevant evidence directly instead of repeatedly searching and reconstructing relationships from raw records. HSKB construction therefore guides the agent through a shorter evidence-acquisition process and reduces the latency of online triage.

4.4 Knowledge Evolution

Figure 3 compares w/o HSKB with four dated revisions of the historical knowledge base. The dated runs use the same 804 test incidents, model, reasoning effort, and recorded incident scopes; each HSKB revision contains historical incidents up to its cutoff and remains fixed during evaluation.



Figure 4: Paired Top-1 changes.

Knowledge growth does not improve every prediction or cutoff monotonically. Figure 4 shows 46 Top-1 corrections and 14 regressions overall, yielding a net gain of 32 correct predictions. Similar non-monotonic behavior appears across subgroups: data-plane Acc@1 dips from 82.61% to 82.45% in January 2026, while control-plane Acc@1 peaks at 65.00% before declining to 63.75%. Intermediate Acc@2/3 regressions are also observed. These results show that knowledge consolidation improves overall attribution but can alter previously correct predictions, motivating regression testing after each HSKB update.

4.5 TTE

We evaluate how responsible-service ranking can accelerate engagement by replaying historical ownership trajectories. This replay uses 126 of the 195 cross-service incidents in our test set for which historical ownership records identify the first assignment to the responsible service. Let $B_i > 0$ be the historical TTE and S_i the replay start time, both measured from incident creation. Replay begins at the first recorded ownership observation. Let $r_{i,m}$ be the position of the responsible service in method m's ranking, with $r_{i,m} = \infty$ if the service is absent. The potential relative TTE reduction is

$$R_{i,m,k} = \begin{cases} \dfrac{\max\{B_i - S_i,\, 0\}}{B_i}, & \text{if } r_{i,m} \le k, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

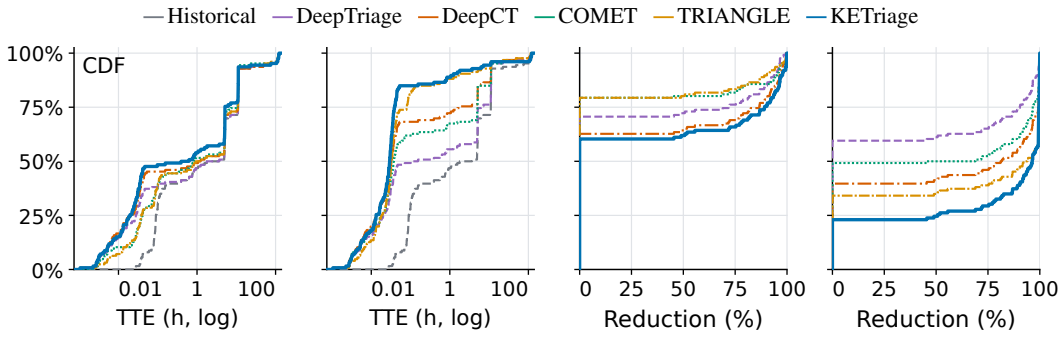
Historical handling continues in parallel: incidents whose responsible service is outside the top-k, or whose replay starts at or after the historical endpoint, retain the original TTE and contribute zero reduction.

This ranking-only replay assumes immediate engagement at replay start when the responsible service is covered. For Acc@3, it additionally assumes oracle selection of the correct service within the shortlist. The resulting reduction is therefore a potential-savings upper bound, not a measured production improvement. All methods use the same historical endpoints and replay start times.

Figure 5 shows that KETriage achieves the largest mean relative TTE reduction under both cutoffs. With first-choice routing, its mean reduction is 34.76%, compared with 32.12% for DeepCT, the strongest baseline. Using the first three candidates increases KETriage's mean reduction to 71.60%, compared with 61.86% for TRIANGLE. These results show how KETriage's strong candidate coverage expands the opportunity for earlier engagement beyond its first prediction.

4.6 Case Study

Figure 6 illustrates how KETriage identifies an upstream responsible service for a VM provisioning incident. Compute B experiences degraded VM PUT reliability in Region R1, while several depen-



(a) TTE @1 (b) TTE @3 (c) Reduction @1 (d) Reduction @3

Figure 5: Empirical CDFs on 126 cross-service incidents with observable first responsible-service ownership. (a,b) Absolute TTE; leftward is better. (c,d) Relative TTE reduction; rightward is better.

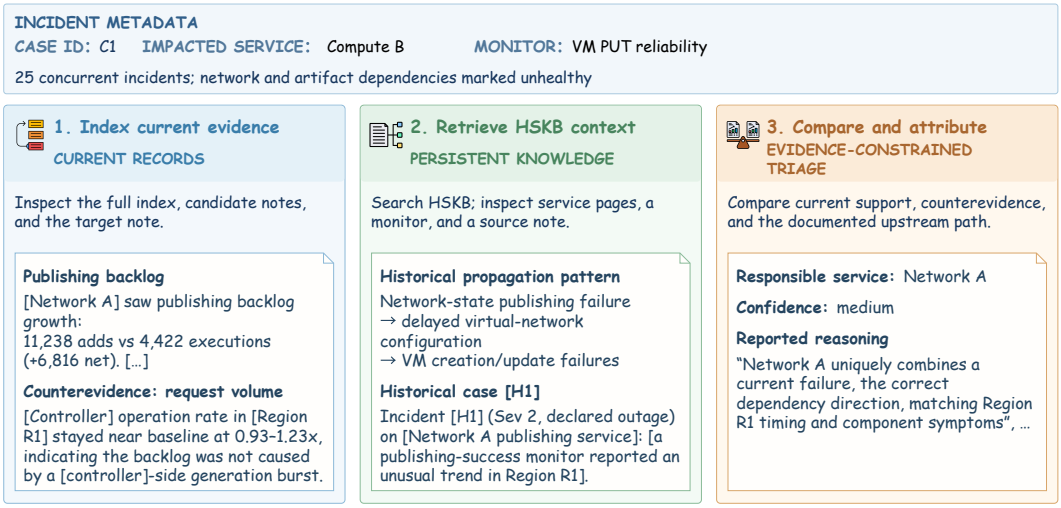


Figure 6: An example of KETriage's execution, from evidence indexing and HSKB retrieval to responsible-service attribution. Service and region names are anonymized.

dencies report unhealthy signals. Among 25 concurrent incidents, five Network A records describe a consistent regional failure involving publication timeouts and network-state backlogs.

KETriage combines these current signals with HSKB knowledge documenting how network-state publication failures can delay network-interface programming and disrupt VM provisioning. This dependency context connects Network A's failure to Compute B's symptoms and helps distinguish it from other unhealthy services lacking a supported propagation path. The agent selects Network A with medium confidence, matching the dataset label. The example demonstrates how current evidence and historical knowledge jointly support cross-service attribution, while missing operation-level traces and recovery correlation remain explicit evidence gaps.

5 Conclusion

We presented KETriage, a knowledge-enhanced and self-improving agentic framework for responsible-service prediction. KETriage constructs HSKB organized into ontology, instance, and case layers. During online triage, the agent uses HSKB to align incident terminology and retrieves relevant knowledge to support attribution from current evidence. Post-resolution consolidation incorporates resolved experience into subsequent knowledge revisions without retraining the LLM. Experiments on real-world industrial cloud incidents show that KETriage outperforms existing methods.

AI USE STATEMENT

We used generative AI tools, including OpenAI Codex, to assist with related works search, manuscript drafting, and revision. Generative language models also form part of KETriage and the evaluated LLM-based methods, as described in the experimental settings. The authors manually reviewed the AI-assisted text, analyses, code, and figures included in this work. We take responsibility for the final content, including all claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies responsible-service prediction using industrial cloud incident records. Such records may contain sensitive operational information, including internal service relationships, resource identifiers, and incident discussions. Their processing and any release of data or artifacts require appropriate authorization and protection of confidential information. The case-study presentation uses anonymized identifiers to communicate attribution mechanisms. KETriage is intended to support operational triage, with source-linked rationales enabling human assessment of its recommendations. Deployment should preserve human oversight and established escalation procedures, particularly when an incorrect assignment could delay incident handling.

REFERENCES

Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1737–1749, Melbourne, Australia, May 2023. IEEE. ISBN 978-1-6654-5701-9. doi: 10.1109/ICSE48619.2023.00149.

Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy (eds.). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly, Beijing Boston Farnham Sebastopol Tokyo, first edition, 2016. ISBN 978-1-4919-2912-4 978-1-4919-5118-7.

Junjie Chen, Xiaoting He, Qingwei Lin, Yong Xu, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. An Empirical Investigation of Incident Triage for Online Service Systems. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 111–120, Montreal, QC, Canada, May 2019a. IEEE. ISBN 978-1-7281-1760-7. doi: 10.1109/ICSE-SEIP.2019.00020.

Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. Continuous Incident Triage for Large-Scale Online Service Systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 364–375, San Diego, CA, USA, November 2019b. IEEE. ISBN 978-1-7281-2508-4. doi: 10.1109/ASE.2019.00042.

Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pp. 674–688. ACM, 2024. doi: 10.1145/3627703.3629553.

Zhuangbin Chen, Jinyang Liu, Yuxin Su, Hongyu Zhang, Xuemin Wen, Xiao Ling, Yongqiang Yang, and Michael R. Lyu. Graph-based Incident Aggregation for Large-Scale Online Service Systems. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 430–442, Melbourne, Australia, November 2021. IEEE. ISBN 978-1-6654-0337-5. doi: 10.1109/ASE51524.2021.9678746.

Shuiguang Deng, Hailiang Zhao, Binbin Huang, Cheng Zhang, Feiyi Chen, Yinuo Deng, Jianwei Yin, Schahram Dustdar, and Albert Y. Zomaya. Cloud-Native Computing: A Survey From the Perspective of Services. *Proceedings of the IEEE*, 112(1):12–46, January 2024. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2024.3353855.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130, 2024. URL https://arxiv.org/abs/2404.16130.

Google Cloud. Multiple GCP products are experiencing service issues. https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW, June 2025. Accessed: 2026-09-25.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938. PMLR, 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

Jianglei Han and Aixin Sun. DeepRouting: A Deep Neural Network Approach for Ticket Routing in Expert Network. In *2020 IEEE International Conference on Services Computing (SCC)*, pp. 386–393, Beijing, China, November 2020. IEEE. ISBN 978-1-7281-8789-1. doi: 10.1109/SCC49832.2020.00057.

Jiajun Jiang, Weihai Lu, Junjie Chen, Qingwei Lin, Pu Zhao, Yu Kang, Hongyu Zhang, Yingfei Xiong, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1410–1420, Virtual Event USA, November 2020. ACM. doi: 10.1145/3368089.3417054.

Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R. Lyu. Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-source Data. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1750–1762, Melbourne, Australia, May 2023. IEEE. ISBN 978-1-6654-5701-9. doi: 10.1109/ICSE48619.2023.00150.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. MicroHECL: High-efficient root cause localization in large-scale microservice systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 338–347. IEEE, 2021. doi: 10.1109/ICSE-SEIP52600.2021.00043.

Fengrui Liu, Xiao He, Tieying Zhang, Jianjun Chen, Yi Li, Lihua Yi, Haipeng Zhang, Gang Wu, and Rui Shi. TickIt: Leveraging Large Language Models for Automated Ticket Escalation. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 343–354, Clarion Hotel Trondheim Trondheim Norway, June 2025. ACM. ISBN 979-8-4007-1276-0. doi: 10.1145/3696630.3728558.

Jianing Liu, Hao Ren, Yu Kang, Minghua Ma, Fangkai Yang, Yong Xu, Xin Gao, Meng Zhang, Hongbin Wang, Xuedong Gao, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, Dongmei Zhang, Qi Zhang, Chetan Bansal, and Yangfan Zhou. Intelligent Triage: Interpretable Incident Triage Workflow using LLM Extracted Triage Reasoning. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice*, pp. 451–462, Rio de Janeiro Brazil, April 2026. ACM. ISBN 979-8-4007-2426-8. doi: 10.1145/3786583.3786887.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, February 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00638.

Yu Luo, Rongchen Gao, Lu Teng, Xidao Wen, Jiamin Jiang, Qingliang Zhang, Yongqian Sun, Shenglin Zhang, Jiasong Feng, Tong Liu, Wenjie Zhang, and Dan Pei. Graph of states: Solving abductive tasks with large language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=F6PoIbPAe2.

Phuong Pham, Vivek Jain, Lukas Dauterman, Justin Ormont, and Navendu Jain. DeepTriage: Automated Transfer Assistance for Incidents in Cloud Services. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3281–3289, Virtual Event CA USA, August 2020. ACM. ISBN 978-1-4503-7998-4. doi: 10.1145/3394486.3403380.

Sai Raghavendra Varanasi. A Survey on Automated Incident Management Practices in Site Reliability Engineering for Cloud-Native Environments. In *2025 International Conference on Electronics and Computing, Communication Networking Automation Technologies (ICEC2NT)*, pp. 1–7, Pune, India, September 2025. IEEE. ISBN 979-8-3315-2345-9. doi: 10.1109/ICEC2NT65402.2025.11380120.

Bogdan Walek. Intelligent System for Ordering Incidents in Helpdesk System. In *2017 21st International Computer Science and Engineering Conference (ICSEC)*, pp. 1–5, Bangkok, November 2017. IEEE. ISBN 978-1-5386-0787-9. doi: 10.1109/ICSEC.2017.8444003.

Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. CloudRanger: Root cause identification for cloud native systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 492–502. IEEE, 2018. doi: 10.1109/CCGRID.2018.00076.

Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. RCAgent: Cloud Root Cause Analysis by Autonomous Agents with Tool-Augmented Large Language Models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4966–4974, Boise ID USA, October 2024a. ACM. doi: 10.1145/3627673.3680016.

Zexin Wang, Jianhui Li, Minghua Ma, Ze Li, Yu Kang, Chaoyun Zhang, Chetan Bansal, Murali Chintalapati, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang, Changhua Pei, and Gaogang Xie. Large Language Models Can Provide Accurate and Interpretable Incident Triage. In *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 523–534, Tsukuba, Japan, October 2024b. IEEE. ISBN 979-8-3503-5388-4. doi: 10.1109/ISSRE62328.2024.00056.

Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. MicroRCA: Root Cause Localization of Performance Issues in Microservices. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, Budapest, Hungary, April 2020. IEEE. ISBN 978-1-7281-4973-8. doi: 10.1109/NOMS47738.2020.9110353.

Shuaiyu Xie, Jian Wang, Hanbin He, Zhihao Wang, Yuqi Zhao, Neng Zhang, and Bing Li. TVDiag: A Task-oriented and View-invariant Failure Diagnosis Framework for Microservice-based Systems with Multimodal Data. *ACM Transactions on Software Engineering and Methodology*, 35 (2):1–39, February 2026. ISSN 1049-331X, 1557-7392. doi: 10.1145/3734868.

Junjielong Xu, Qinan Zhang, Zhiqing Zhong, Shilin He, Chaoyun Zhang, Qingwei Lin, Dan Pei, Pinjia He, Dongmei Zhang, and Qi Zhang. OpenRCA: Can large language models locate the root cause of software failures? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M4qNIzQYpd.

Yuhan Yao, Yuxuan Jiang, Minghua Ma, Madhura Vaidya, Jieren Deng, Yigong Hu, Chetan Bansal, Ze Li, and Murali Chintalapati. An Agentic Framework for Triaging Incidents in Production Cloud Infrastructure. In *Proceedings of the 34th ACM International Conference on the Foundations of Software Engineering*, pp. 557–567, Concordia University Montreal QC Canada, July 2026. ACM. ISBN 979-8-4007-2636-1. doi: 10.1145/3803437.3805228.

Zhaoyang Yu, Aoyang Fang, Minghua Ma, Jaskaran Singh Walia, Chaoyun Zhang, Shu Chi, Ze Li, Murali Chintalapati, Xuchao Zhang, Rujia Wang, Chetan Bansal, Saravan Rajmohan, Qingwei

Lin, Shenglin Zhang, Dan Pei, and Pinjia He. Triangle: Empowering Incident Triage with Multi-Agent. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 674–686, Seoul, Korea, Republic of, November 2025. IEEE. ISBN 979-8-3503-5733-2. doi: 10.1109/ASE63991.2025.00062.

A End-to-End Cross-Service Case Studies

This appendix examines four anonymized cross-service incidents to show how HSKB and incident alerts jointly support responsible-service attribution. The cases span network-to-compute propagation, cluster-to-database dependencies, facility-to-network impact, and platform-to-endpoint failures. In each case, the service reporting the pattern is not sufficient to identify the responsible upstream entity. The central question is therefore not merely which services are failing at the same time, but which candidate has a supported dependency relation that explains the target impact.

A.1 C1: Network Failure Behind Compute-Provisioning Errors

Context and ambiguity. Case `C1` begins with failed compute-resource create and update operations, without an upstream cause in the target record. Concurrent network-publication timeouts and controller backlog make Network Service `A` a plausible candidate. However, shared time and region do not distinguish an upstream dependency failure from unrelated degradation; a target-local explanation remains an alternative.

Evidence-to-attribution reasoning. The decisive bridge is the publication-to-programming dependency in `K1--K2`: network-state updates are required for interface programming, which supports compute provisioning. This relation gives direction to the current publication and backlog observations. Historical records support the propagation pattern, but do not establish that every intermediate interface state occurred in this incident. Figure 7 separates these roles in the F/B/D checks.

Outcome and interpretation. The complete system ranks Network Service `A` first; w/o HSKB selects the target-local explanation while retaining the network service as an alternative. Knowledge therefore contributes a reason to prefer an already visible candidate, rather than a new failure observation. The contrast concerns service-level attribution, not confirmation of the full historical mechanism.

C1: Network publication failure behind compute-provisioning errors

1. Target incident
Impacted service: Compute Service B. **Region:** R1.
Symptom: elevated failures in compute-resource create and update operations. The target record does not identify an upstream cause.

2. Concurrent observations (current evidence)
S1: Network Service A, message-publication timeouts.
S2: Network Service A, publication-queue growth and retry accumulation.
S3: Network Service A, network-controller programming backlog.
Scope match: all three incidents occur in R1 and overlap the target window.

3. Retrieved operational knowledge (not current evidence)
K1: the publication component of Network Service A distributes network-state updates required by interface programming.
K2: delayed interface programming can block compute-resource provisioning.
K3 [H1]: publication failure and programming backlog previously accompanied degradation of the same provisioning indicator.
K4 [H2]: a historical regional incident links publication timeouts to stalled interface state and downstream compute-operation failures.

4. Attribution checks and supported propagation path
F: S1−−S3 establish a current failure in Network Service A.
B: K1−−K2 place that service outside Compute Service B and upstream of provisioning.
D: K1−−K4 connect the observed publication and programming symptoms to the target operation, rather than relying on regional co-occurrence alone.
Path: publication failure → delayed interface programming → compute-provisioning failure.

5. Output comparison
Complete system, Top-1: Network Service A.
w/o HSKB, Top-1: target-local (Compute Service B); Network Service A remains an alternative, but its exact rank is not specified in this trace summary.

6. Knowledge contribution and evidence boundary
Contribution: supplies dependency direction for a candidate already visible in the current scope.
Boundary: the current records establish publication failure, programming backlog, and provisioning impact; history supports their connection but does not independently confirm every intermediate interface state.

Figure 7: Case C1: dependency knowledge supports a network-to-compute attribution over the target-local alternative.

A.2 C2: Cluster-Placement Failure Behind Database Timeouts

Context and ambiguity. Case C2 concerns logical-database creation timeouts alongside two external signals: placement-constraint failures in Cluster Platform B and control-plane latency in Compute Service C. Both overlap the target deployment and window. The ambiguity is which service-layer failure explains the database operation, rather than which incident most closely resembles a generic timeout.

Evidence-to-attribution reasoning. The hosting relation in K5 connects database control rings to Cluster Platform B; K6 explains how unresolved placement constraints can prevent management-replica activation. Together, these relations connect a specific current failure to the target operation. Historical evidence supports this path over generic compute latency, without making replica unavailability a directly observed current state. Figure 8 records the candidate comparison and its evidence.

Outcome and interpretation. The complete system selects Cluster Platform B, whereas w/o HSKB ranks Compute Service C first and omits the platform from its Top-3. This is recovery in the ranked attribution, not discovery of a missing scope incident: S4 was already available. Knowledge supplies the hosting context needed to recognize its relevance to database availability.

C2: Cluster-placement failure behind database-creation timeouts

1. Target incident
Impacted service: Database Service U. **Region:** R2.
Symptom: logical-database create operations exceed the timeout threshold.

2. Concurrent observations (current evidence)
S4: Cluster Platform B, repeated placement-constraint failures.
S5: Compute Service C, elevated control-plane operation latency.
Scope match: both incidents overlap the target window and regional deployment.

3. Retrieved operational knowledge (not current evidence)
K5: database control rings are hosted on Cluster Platform B.
K6: unresolved placement constraints prevent activation of database-management replicas.
K7 [H3--H5]: historical incidents link replica-placement failures to logical-database operation timeouts.
K8 [H6]: historical evidence distinguishes platform-placement failure from generic compute latency.

4. Attribution checks and supported propagation path
F: S4 establishes a current placement failure in Cluster Platform B.
B: K5 places the platform outside Database Service U and upstream of its control rings.
D: K6--K8 connect placement failure to the target timeout; this path is more specific than the correlated compute-latency signal in S5.
Path: placement-constraint failure $\rightarrow$ unavailable management replica $\rightarrow$ database-creation timeout.

5. Output comparison
Complete system, Top-1: Cluster Platform B.
w/o HSKB, Top-1: Compute Service C; Cluster Platform B is absent from its Top-3 output.

6. Knowledge contribution and evidence boundary
Contribution: recovers the platform in the ranked attribution by connecting its observed placement failure to the database hosting relation. The platform incident itself is already present in the scope.
Boundary: management-replica unavailability is a knowledge-supported intermediate state, not a separately reported observation in the displayed current evidence.

Figure 8: Case C2: hosting and placement knowledge favors Cluster Platform B over a less specific compute-service symptom.

A.3 C3: Datacenter Event Behind Physical-Network Loss

Context and ambiguity. Case C3 reports abrupt physical-fabric reachability loss across multiple hosts. Although the visible symptoms belong to Network Service A, the concurrent scope also includes a facility event at the affected location. The attribution problem is to distinguish the owner of the upstream physical disruption from the service through which users observe its impact.

Evidence-to-attribution reasoning. The facility incident supplies the current upstream signal, while the location match connects it to the link-loss observations. Knowledge then contributes two different relations: K10 identifies the facility owner, and K9 and K11 support facility-to-network propagation. Neither ownership alone nor a historical physical-failure pattern establishes present responsibility. Figure 9 combines them with the current multi-host and downstream impact to support the external attribution.

Outcome and interpretation. The complete system selects Datacenter Operations Group C; w/o HSKB selects the impacted network service and omits the group from its Top-3. Knowledge thus shifts attribution across an operational ownership boundary, not merely between software services. The result identifies the responsible group without claiming that a particular power fault or rack-device failure has been confirmed.

C3: Datacenter facility event behind physical-network loss

1. Target incident
Impacted service: Network Service A. **Region:** R3.
Symptom: abrupt loss of physical-fabric reachability across multiple hosts.

2. Concurrent observations (current evidence)
S6: Network Service A, multiple server-to-rack link failures.
S7: Datacenter Operations Group C, a facility event affecting the same location.
S8: downstream services, simultaneous connectivity alarms.
Scope match: the facility event affects the target location, alongside concurrent link loss and downstream connectivity impact.

3. Retrieved operational knowledge (not current evidence)
K9: correlated server-to-rack link loss across independent hosts is consistent with an upstream rack or facility event.
K10: Datacenter Operations Group C owns the facility and power boundary for the affected location.
K11 [H7−−H9]: historical incidents connect facility disruption to rack-device loss and regional network black-holing.

4. Attribution checks and supported propagation path
F: S7 establishes a current facility event at the affected location.
B: K10 identifies an external facility owner, distinct from the impacted Network Service A.
D: K9−−K11 connect the facility boundary to physical-network loss, corroborated by the multi-host and downstream symptoms in S6 and S8.
Path: facility disruption → rack-device or link loss → physical-network reachability loss.

5. Output comparison
Complete system, Top-1: Datacenter Operations Group C.
w/o HSKB, Top-1: target-local (Network Service A); Datacenter Operations Group C is absent from its Top-3 output.

6. Knowledge contribution and evidence boundary
Contribution: resolves the responsibility boundary from the impacted network service to the upstream infrastructure-operations group.
Boundary: the current facility event and location match support this attribution; historical propagation patterns do not establish a specific power fault or rack-device failure in the current incident.

Figure 9: Case C3: facility ownership and physical-dependency knowledge distinguish the responsible operations group from the impacted network service.

A.4 C4: SERVING-PLATFORM FAILURE BEHIND AI ENDPOINT ERRORS

Context and ambiguity. Case C4 concerns widespread endpoint 5XX errors and timeouts alongside six regional incidents from Model-Serving Platform D. The platform is already prominent in the scope, but incident count and temporal overlap do not establish that it serves the affected endpoints. The remaining question is whether its failure explains the target signature rather than accompanying it.

Evidence-to-attribution reasoning. The serving relation in K12 places the platform on part of the inference and ingress path for AI Service V. Knowledge in K13--K14 then connects platform disruption to downstream errors and timeouts. The current platform incidents establish present failure; the historical ingress-side cause supports a possible propagation mechanism, not its exact recurrence. Figure 10 preserves that distinction in the final attribution.

Outcome and interpretation. The complete system ranks Model-Serving Platform D first, while w/o HSKB selects the target-local explanation. Unlike candidate recovery in C2, the main contribution is to connect a prominent candidate to the serving path. The rationale deliberately stops at platform-level responsibility because the displayed current evidence does not establish the historical component-level cause.

C4: Serving-platform failure behind AI endpoint errors

1. Target incident
Impacted service: AI Service V. **Region:** R1.
Symptom: elevated HTTP 5XX and timeout rates across model endpoints.

2. Concurrent observations (current evidence)
S9--S14: six incidents from Model-Serving Platform D, sharing a regional ingress and infrastructure-failure signature.
Scope match: the platform incidents occur in the target region and overlap the endpoint-error window.

3. Retrieved operational knowledge (not current evidence)
K12: Model-Serving Platform D hosts part of the inference and ingress path for AI Service V.
K13: platform-ingress failure can produce downstream endpoint 5XX errors and timeouts.
K14 [H10]: a historical incident documents this platform-to-endpoint propagation pattern and a confirmed ingress-side cause.

4. Attribution checks and supported propagation path
F: S9--S14 establish a current regional failure in Model-Serving Platform D.
B: K12 places the external platform on the serving path of AI Service V.
D: K13--K14 explain how a platform failure can produce the target error signature, consistent with the overlapping current incidents.
Path: serving-platform failure $\rightarrow$ disruption of the inference or ingress path $\rightarrow$ AI endpoint 5XX errors and timeouts.

5. Output comparison
Complete system, Top-1: Model-Serving Platform D.
w/o HSKB, Top-1: target-local (AI Service V); the platform's exact rank is not specified in this trace summary.

6. Knowledge contribution and evidence boundary
Contribution: connects an already prominent platform candidate to the affected serving path and downstream error signature.
Boundary: attribution stops at the platform level. The current evidence does not confirm recurrence of the component-level ingress cause recorded in H10.

Figure 10: Case C4: serving-path knowledge supports platform-level attribution without importing a historical component-level cause into the current incident.

B CASE STUDIES OF KNOWLEDGE-UPDATE GAINS

We follow two incidents across four HSKB revisions: 2025-01, 2025-07, 2026-01, and 2026-06. Each case uses the same incident input across revisions. The version dates identify historical cutoffs in the knowledge archive. The cases show which historical incidents the agent uses, how those incidents clarify a service dependency, and how the responsible service's rank changes. Cases, services, regions, and evidence records use pseudonyms; historical months and revision dates are retained.

B.1 IDENTIFYING THE CAPACITY SERVICE BEHIND VM ALLOCATION FAILURES

Current incident and earlier prediction. Case B1 reports VM provisioning failures in Region R4. VM Control Plane L is the impacted service, and current record S1 reports zonal allocation failures in the same region. Dependency diagnostics mark Capacity Service K and Compute Management Service M unhealthy. With the 2025-01 knowledge revision, the agent attributes the incident to a failure within the impacted service and ranks Capacity Service K second. It recognizes the allocation symptoms but leaves Capacity Service K's role in the provisioning path unresolved.

Historical incidents added to the knowledge base. H1, from April 2025, describes failed Analytics Service J cluster creation in Region R5. The incident identifies Capacity Service K as responsible and records recovery after Capacity Service K increased compute capacity. H2, from May 2025, links Region R1 Analytics Service J cluster-launch failures to parent incident P1 for VM allocation. Both records first appear in the 2025-05 archive and are incorporated into the Analytics Service J service page. H1 explains Capacity Service K's capacity role, while H2 connects VM allocation failures to downstream cluster-launch failures. Together, they describe how a capacity problem can affect VM allocation and then the services that depend on it.

Updated prediction and gain. The 2025-07 output cites H1 and H2 when assessing the current allocation event. It uses the capacity history to interpret Capacity Service K as an upstream service and promotes it from second to first. The local explanation moves to second, and Compute Management Service M remains third. Capacity Service K stays first in the two later revisions, giving a rank sequence of 2/1/1/1. The useful addition is the link between capacity responsibility and allocation symptoms: the agent can now connect a named unhealthy dependency to the operation that is failing. For the 33 test incidents attributed to Capacity Service K and affecting VM Control Plane L, correct first-choice predictions increase from 31 to 32 between 2025-01 and 2025-07, with two corrections and one regression.

B.2 CONNECTING MODEL-SERVING FAILURES TO API ERRORS

Current incident and earlier prediction. Case B2 reports AI Service H HTTP 5XX errors for model Q1 in Region R1 and Region R4. Its input contains several Serving Platform I-owned alerts: the current records grouped as `G1`. These records describe a Region R1 serving-availability event across several other deployments. Related AI Service H alerts show failures on the API side. In the first three revisions, the agent favors an explanation within AI Service H and ranks Serving Platform I second. It treats the platform alerts as a concurrent event with an unresolved connection to the target model.

Historical incident added to the knowledge base. H3, from April 2026, describes historical model Q0 errors caused by requests reaching backend serving endpoints with no active deployment. Responders identify upstream resets and route the investigation to the serving platform. Removing the failing endpoints in two serving regions from rotation restores service. H3 first appears in the 2026-05 archive, where the AI Service H service page incorporates this endpoint-lifecycle failure. The record provides a concrete path from a serving-platform problem to customer-facing API errors, together with the action that restored service.

Updated prediction and gain. The latest output cites H3 and identifies Serving Platform I as upstream of the affected API. It groups the current platform alerts into one regional event and connects them to the API failures through their location and HTTP 500-class symptoms. The platform

moves to first, the local explanation moves to second, and Analytics Service J remains third. Its rank sequence is 2/2/2/1. The gain comes from making better use of a platform event that was already visible: the historical record explains how a failure at that boundary can produce the target's symptoms. Across 55 incidents attributed to Serving Platform I and affecting AI Service H, correct first-choice predictions increase from 1 to 8 between 2026-01 and 2026-06. Four of the seven corrected outputs cite H3.

B.3 SUMMARY

The two cases show how historical knowledge helps the agent interpret information in the incident input. B1 connects allocation failures to capacity responsibility. B2 connects serving-endpoint failures to API errors. In each case, the responsible service moves from second to first as the output gives a clearer account of its upstream role.

C Error Analysis

Method. We analyze 165 incorrect Top-1 predictions: 125 cross-service and 40 self-inflicted incidents. To assess input coverage, we search all scalar fields of the saved $x_i = (m_i, h_i, N_i)$, including descriptions, health/module signals, and concurrent records. Whole-token phrase matching normalizes case and punctuation and uses responsible-service label variants, evaluator aliases, and aliases recorded during HSKB lookup. This measures service mentions in the saved input, not the novelty of a service or failure type.

Each error enters the first applicable category. Rules 1–4 apply only to cross-service errors; all remaining errors, including self-inflicted errors, enter rule 5.

1. **Absent from saved input.** No mention of the responsible service or its matched aliases is found anywhere in x_i.
2. **Absent from saved candidates.** The service appears in x_i but is absent from the union of saved candidates, focused rechecks, the final Top-3 list, the selected candidate, and the provisional/final Top-1 review. Top-3 omission alone does not satisfy this rule.
3. **Boundary unresolved or incorrect.** At least one matching assessment does not classify the service as a distinct upstream service. We use focused rechecks when available and otherwise use candidate assessments.
4. **Reported evidence shortfall.** After the boundary check, matching focused rechecks exist and all report only partial support or no matching current evidence. These are agent-reported assessments of evidence sufficiency.
5. **Residual selection error.** None of the preceding rules applies. This includes self-inflicted incidents assigned externally and does not establish that the correct service was fully assessed or had sufficient evidence.

These mutually exclusive categories describe saved inputs and outputs, not verified physical root causes. Missing candidate records do not prove that a service was never considered during execution.

Results. The responsible service is unmentioned in 21 of the 195 cross-service inputs (10.77%). All 21 are Top-1 errors, accounting for 16.80% of the 125 cross-service errors; one nevertheless recovers the service at rank 3. Another 29 errors mention the service in the input but omit it from all saved candidate and decision fields. Table 3 follows the classification order.

Table 3: Mutually exclusive partition of all 165 incorrect Top-1 predictions, in rule order. Shares use 165 errors as the denominator; Cross and Self denote cross-service and self-inflicted incidents.

Error category	Count	Share	Cross	Self
Absent from saved input	21	12.73%	21	0
Absent from saved candidates	29	17.58%	29	0
Boundary unresolved or incorrect	24	14.55%	24	0
Reported evidence shortfall	50	30.30%	50	0
Residual selection error	41	24.85%	1	40
Total	165	100.00%	125	40

All 50 errors with reported evidence shortfalls retain the service at rank 2 or 3. The 41 residual errors include all 40 self-inflicted errors. Overall, 114/165 errors (69.09%) retain the service at rank 2 or 3; 51 omit it from Top-3. Improving these predictions may require better ownership interpretation and current-evidence assessment as well as candidate ranking.

Counting scripts and incident-level assignments preserve the rules. Missing mentions may reflect source-data gaps or input construction; the exports cannot distinguish these causes or establish that all relevant evidence is absent.