

977

978 **Appendix**

979 In the appendix, we provide additional analysis, detailed results, and more discussions.

980

981

Table of Contents

982

A Confinement Rules	24
----------------------------	-----------

983

B Bounded Risk Window	25
------------------------------	-----------

984

C Ablation Study on ITBench	26
------------------------------------	-----------

985

D Tool Usage	26
---------------------	-----------

986

D.1 List of Tools	26
-----------------------------	----

987

D.2 Distribution of Tool Usage	27
--	----

988

E Oracle Usage	27
-----------------------	-----------

989

F Hyperparameter Sweep	29
-------------------------------	-----------

990

G Result Details	29
-------------------------	-----------

991

H Heuristic Evaluation of STRATUS’s Multi-agent Design	30
---	-----------

992

I An Example of STRATUS’s Trajectory	31
---	-----------

993

994

996 A Confinement Rules

997 To ensure safety and predictability, each STRATUS agent operates under sandboxed confinement rules.
 998 As detailed in Table 5, detection and diagnosis agents are restricted to a Read-Only role and are only
 999 permitted to *observe* the system using commands such as TelemetryCollection tools. These agents
 1000 are explicitly prohibited from executing any operation that alters the state of the system.

1001 To perform mitigation, STRATUS will need to use the Writer role, as changing the system state is
 1002 essential for mitigation. For state-changing tasks, STRATUS performs command-level validation to
 1003 verify that generated actions are syntactically correct and safe to run in production. Risky operations
 1004 such as deleting namespaces are prohibited.

1005 Certain kubectl commands, such as kubectl exec -it and kubectl edit, launch interactive shells
 1006 or editors. These commands are prevented because STRATUS cannot participate in interactive sessions,
 1007 which would cause execution to hang. Additionally, granting arbitrary shell access significantly
 1008 increases the potential for unintended and unsafe behaviors. Therefore, we explicitly disallow all
 1009 interactive commands.

Table 5: Agent confinement rules in STRATUS. shows the allowed sandboxing rules according to the agent’s role, while shows the blocked actions enforced by STRATUS.

Category	Subcategory	Confinement	Example(s)
Agent Role-Based	Read-Only Agents	Non-mutating commands only	kubectl get, kubectl describe kubectl logs, ls, cat TelemetryCollection
	Writer Agents	Sequential execution only	Only serialized state modifications are allowed
Command-Level Confinement (Blocked for all agents):			
Kubectl Subcommands	Destructive Operations	Namespace deletion	kubectl delete namespace my-ns
	Interactive Edit	debug, edit	kubectl debug pod/foo kubectl edit deployment/bar
	Input from stdin	-f - pattern	kubectl apply -f -
Kubectl Flags	Interactive Terminal	-stdin, -tty, exec -it	kubectl attach -tty pod/foo kubectl exec -it pod/bash
Shell Syntax	Pipelines	Pipe operator	kubectl get pods grep Running
	Compound Commands	&&, , ;	kubectl get pods && echo success
	Command Substitution	\$(...), backticks	kubectl delete pod \$(kubectl get pods -o name)
	Flow Control	if, for, while, until, case	if [...]; then ...; fi for pod in ...; do ...; done
	Shell Functions	Function definitions	myfunc() { ... }

Confinement Examples	
Example 1 (Input from stdin):	
1. STRATUS:	kubectl apply -f -
2. Confinement:	"Stdin redirection is not allowed."
Example 2 (Interactive Terminal):	
1. AOL-agent:	kubectl exec -it mongodb-rate-bfbcf4587-md6xl -n test-hotel-reservation -- mongo
2. AOL-agent:	Hang there until timeout.
1. STRATUS:	kubectl exec -it mongodb-rate-bfbcf4587-j7lsf -n test-hotel-reservation -- mongo
2. Confinement:	"Interactive flag detected: -it. Such commands are not supported."

1010 To further enforce safety, STRATUS uses static linting to validate command syntax before execution.
 1011 This validation includes rejecting shell pipes, compound commands (e.g., using &&, ||, or semicolons),
 1012 and other complex shell constructs. These are disallowed for two reasons: (1) there are exponentially
 1013 many arbitrary shell expressions that STRATUS may generate; comprehensively verifying the safety

1014 of all possible shell expressions is not realistic. On the contrary, individual kubectl invocations
 1015 can be easily checked. (2) we require the agent to operate incrementally, issuing and validating one
 1016 command at a time, to work effectively with our safety infrastructure and confinement layers.

1017 When supported, STRATUS employs Kubernetes’ dry-run mechanism to produce the effect of a
 1018 command without persisting changes. In this mode, the Kubernetes API server processes the request
 1019 and returns the result without modifying the system. This simulated output is presented to the agent,
 1020 which can then decide whether to proceed, adjust the command, or cancel the operation entirely.
 1021 This approach allows the mitigation agent to preview the impact of state-changing actions and apply
 1022 corrections before execution.

1023 The examples above show the effectiveness of command-level confinement in the mitigation agent of
 1024 STRATUS, particularly for preventing execution failures or system hangs due to unsafe operations.
 1025 In Example 1, the agent attempts to execute the command `kubectl apply -f -`, which will make
 1026 the agent hang due to missing input via stdin. However, it is prevented by STRATUS with feedback.
 1027 In Example 2, the agent issues an interactive command using the `-it` flag, attempting to launch a
 1028 terminal session within a running pod. Without confinement, this results in the execution hanging until
 1029 a timeout occurs. STRATUS’s confinement mechanism blocks such interactive terminal invocations,
 1030 returning a clear message to the agent: "Interactive flag detected: `-it`. Such commands are not
 1031 supported."

1032 In our evaluation, 10.5% of commands executed by STRATUS (GPT-4o) and 9.7% by STRATUS
 1033 (LLaMA3) would result in system hangs, in the absence of our confinement mechanisms.

1034 B Bounded Risk Window

1035 A key safety mechanism in STRATUS is the *bounded risk window*, which specifies the number
 1036 of actions that the mitigation agent (α_M) can execute in a single transaction. In this section, we
 1037 empirically investigate how the total step limit impacts STRATUS’s success rate on AIOpsLab
 1038 problems (detection, localization, root-cause analysis, and mitigation). For each problem, we run the
 1039 evaluated agents with total step limits setting to different values (3, 5, 10, 15, 20, and 30). Results are
 1040 presented in Figure 6.

1041 STRATUS (GPT-4o) improves from 0% to 53.53%, after switching the step limit from 3 to 10. This
 1042 shows that allowing STRATUS to take more steps can significantly boost its effectiveness. We also
 1043 find that among all evaluated agents, agent performance plateaus after 15 steps, indicating that taking
 1044 more steps after a certain threshold does not yield better performance significantly.

1045 The effectiveness gain can be observed on STRATUS with other LLM backends, but with lower
 1046 overall accuracy, relatively scaling with the LLM’s capability. Based on these observations, we set
 1047 the default bounded risk window to $K = 20$.

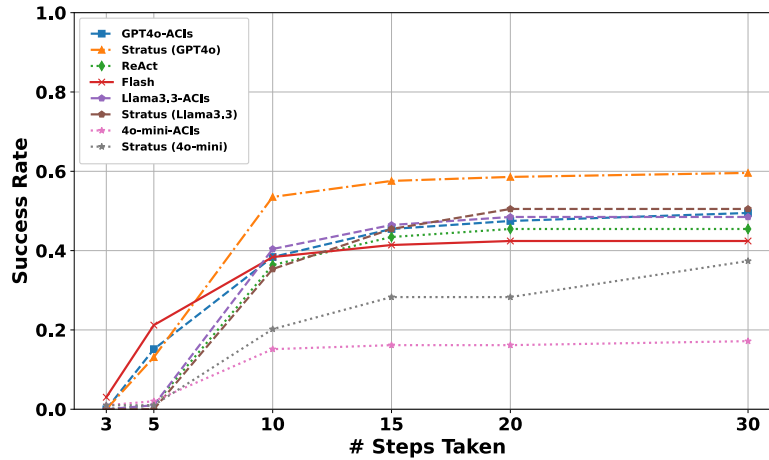


Figure 6: Impact of step limit on success rate of different agents in AIOpsLab.

C Ablation Study on ITBench

In §5, we presented an ablation study of STRATUS (GPT-4o) on AIOpsLab, demonstrating the effectiveness of its retry and undo mechanisms. We now examine STRATUS’s behavior under ablation on the ITBench benchmark, as shown in Table 6. We observe that (1) disabling retry substantially drops the success rate to 11.1%. Note that this configuration implicitly disables the undo agent, as there are no prior actions to undo. Without retries, the agent must determine the root cause and devise a mitigation plan correctly with one attempt, which is significantly challenging. (2) STRATUS achieves the same task-level success rate (solving 9/18 problems) when the undo agent is disabled. As mentioned in §5, STRATUS exploits the observation that in 8 out of 18 problems, restarting the target pods clears the incident alerts. Since ITBench evaluates correctness primarily based on the absence of immediate alerts, this allows the agent to resolve the incident without trying more destructive, state-changing actions, where our undo mechanism will be most effective.

To better understand this, we analyzed these 8 problems in detail. Among all mitigation actions issued by the agent, approximately 15.2% are pod deletion operations managed by Kubernetes Deployment, which can automatically recreate the pods. Furthermore, a significant portion (44.2%) of state-changing commands, e.g., changing a container image to an image with a hallucinated name, are easily detected and fixed by STRATUS in the subsequent run, since STRATUS has a reflection mechanism, which can help the agent to realize what went wrong in the previous run. The rest of the commands are a mix of direct restart or redeployment of the pods, which again lifts the underlying fault directly. In this regard, the undo mechanism does not improve the success rate in ITBench.

However, the absence of undo capabilities can still compromise the cloud over time. Latent or non-critical side effects from agent actions, such as misconfigurations or performance degradations that do not immediately trigger alerts, may go undetected. Without TNR, there is no guarantee ensuring that such side effects are identified and reverted.

Table 6: Ablation analysis of STRATUS (GPT-4o) on mitigation problems in ITBench.

Ablation	Succ. Rate	Time (s)	Cost (\$)
STRATUS (4o)	50%	1720.8	6.11
– No Retry	11.1%	267.2	0.72
– Naïve retry w/o undo	50%	1711.9	5.13

D Tool Usage

We develop a set of tools that enable agents to interact with the cloud environment through natural language. We describe the implemented tools in §D.1, and analyze their usage distribution across execution steps in §D.1.

D.1 List of Tools

In order to do complex reasoning and safe system manipulation, STRATUS has a comprehensive suite of tools, including observability tool and command-line tools. We describe some of the tools here.

Observability Tool. STRATUS has tools to observe the clouds, through collecting telemetry data (TelemetryCollection), or retrieving system states.

- **GetLogsTool:** Retrieves logs from specific component, e.g., application logs from a given pod.
- **GetTracesTool** and **ReadTracesTool:** Retrieve traces if the system is traced, e.g., traces from Jaeger. **ReadTracesTool** handles large trace files stored on disk.
- **GetBootstrapAnalysisTool** and **ReadBootstrapAnalysisTool:** Support trace-based bootstrapping by identifying likely fault locations. Instead of digesting the entire trace data, the agent will only read the traces with errors (e.g., 500 HTTP responses) as initial analysis results.

1089 • **OracleTools:** After each iteration of diagnosis or mitigation, oracles can be used to validate
1090 system health (e.g., alert clearance, workload throughput, cluster health). These oracles
1091 determine whether the agent should continue, retry, or terminate.

1092 Note that STRATUS disables direct metrics consumption, as raw numerical data can be difficult for
1093 the agent to interpret effectively. Instead, such information is surfaced in logs in a more pre-processed
1094 form, e.g., as filtered, aggregated, or human-readable summaries. These tools abstract away raw
1095 observability data, enabling the agent to reason over high-level, interpretable signals, and to determine
1096 its termination.

1097 **Command-line Tools.** On the action side, STRATUS has specialized tools to construct the command
1098 to change the system states and perform the faithful undo.

- 1099 • **NL2Kubect1:** Translates natural language instructions generated by the agent into Kubernetes
1100 commands. This decouples high-level reasoning from low-level execution syntax.
- 1101 • **LintingTool:** Acts as an internal tool to check the correctness of the command, e.g., the
1102 commands generated by NL2Kubect1; if the command is not correct or allowed, the tool will
1103 return a message to ask the agent to improve the command.
- 1104 • **RollbackTool:** Supports the undo agent to perform faithful undo actions.
- 1105 • **NotificationTool:** Whenever the agents feel confident that the task is finished, it can use
1106 the NotificationTool to notify human of task completion, or escalation. Notification can
1107 be made in different troubleshooting phases e.g., detection phase should notify human
1108 whether there is a nomaly in the system; while the agent should notify human whether the
1109 mitigation is successful or complete.

1110 D.2 Distribution of Tool Usage

1111 To analyze the tool invocation patterns in STRATUS, we log the tool usage of the STRATUS (GPT-4o)
1112 across all problems in the AIOpsLab. For each execution step (i.e., each iteration within the retry loop),
1113 we record which tools that interact with the clouds are invoked. In Figure 7, we show the distribution
1114 of external tool usage (including the TelemetryCollection, NL2Kubect1, NotificationTool, and
1115 RollbackTool) by the STRATUS (GPT-4o) at different steps in our AIOpsLab experiments.

1116 STRATUS exhibits distinct phases of tool usage across execution steps. In the initial stage (Iterations
1117 1–2), the RollbackTool is predominantly used by the undo agent, and bootstrapping tools are used
1118 to perform initial analysis, which are specified in the control flow. In early steps (Iterations 3–8),
1119 the agent primarily engages TelemetryCollection tools to gather observability data from logs and
1120 traces. As execution progresses (Iterations 9–23), NL2Kubect1Tool dominates, indicating the agent’s
1121 transition from information gathering using provided tools to more customized command generation
1122 for diagnosis and mitigation. Additionally, we observe a shift in preference from trace retrieval to log
1123 retrieval over time.

1124 E Oracle Usage

1125 In §4.2, we describe how STRATUS determines termination based on the oracles. Here, we use an
1126 example (Problem ID: wrong_bin_usage-mitigation-1 in AIOpslab) to illustrate the usage of our
1127 oracles in STRATUS. In this problem, a microservice is misconfigured to use an incorrect binary
1128 during container initialization. As a result, while the container is able to start and all pods appear
1129 healthy at the orchestration level, the application has functional failures. Specifically, the service
1130 responds with a high rate of HTTP errors (non-2xx/3xx responses), indicating that the application is
1131 not working as expected.

1132 As illustrated in Figure 8, during the initial validation phase, the system health oracle confirms that all
1133 pods are running normally. However, the workload oracle detects a number of erroneous responses
1134 (115), revealing that the issue has not been fully resolved. Based on this feedback, STRATUS further
1135 troubleshoots the issue, and later correctly identifies the root cause as an incorrect binary and proceeds
1136 to patch the deployment with the appropriate version. A second round of oracle validation shows
1137 that all checks pass: the cluster remains healthy and the workload oracle confirms the elimination

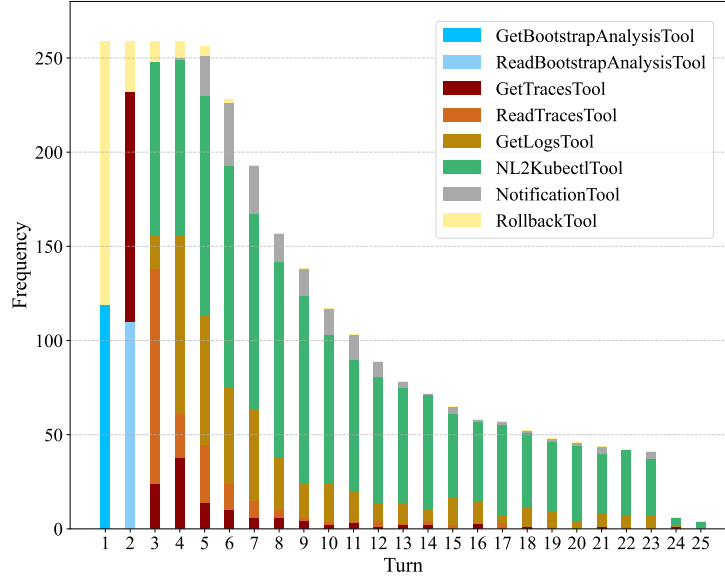


Figure 7: Distribution of tool usage by STRATUS (GPT-4o) across steps. The figure shows a histogram of tool invocations grouped by type and step index (Iteration).

1138 of HTTP errors (zero non-2xx/3xx responses). With successful validation, STRATUS notifies the
 1139 benchmark, marking the resolution of the issue.

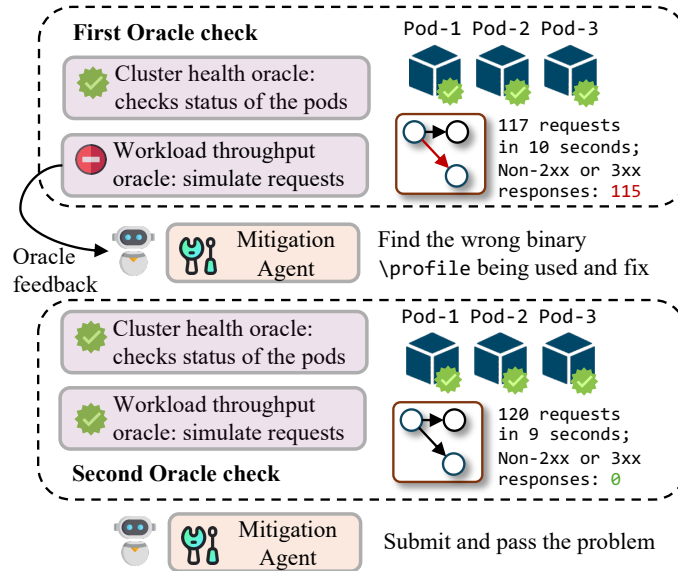


Figure 8: Oracle usage when STRATUS solving the wrong_bin_usage-mitigation-1 task. In the first validation round, the workload oracle detects 115 failed responses, indicating the issue is unresolved. STRATUS continues troubleshooting and patches the deployment with the correct binary. In the second round, both oracles confirm success.

1140 F Hyperparameter Sweep

1141 We perform a hyperparameter sweep over the temperature parameter in STRATUS (GPT-4o) using
 1142 all 13 mitigation tasks from AIOpsLab. Table 7 shows the results. We can see that the temperature
 1143 of 0.0 yields the best results. Setting temperature to 0 allows the LLM to output deterministically
 1144 with a given input. Setting a non-zero temperature allows “creativity” in LLM’s output. However,
 1145 in the SRE context, this may not be desirable. A non-zero temperature permits the LLM to output
 1146 non-deterministic results given the same input. For example, given a microservice application trace
 1147 that clearly indicates the root cause of a problem, we would like STRATUS to confidently take *one*
 1148 correct mitigation action. In addition, setting overly high temperature can directly affect the tool’s
 1149 ability in generating the correct commands, which harms STRATUS’s performance critically.

Table 7: Effect of Temperature on Mitigation Success Rate.

Temperature	Succ.	Time (s)	Step	\$
0.0	69.2%	811.9	46.3	0.877
0.2	51.3%	961.9	71.6	0.647
0.4	46.2%	950.58	67.9	0.493
0.6	56.4%	836.8	55.1	0.388
0.8	33.3%	1209.51	77.2	0.533

1150 G Result Details

1151 We present detailed evaluation results for STRATUS (GPT-4o) in AIOpsLab. Table 8 shows the results
 1152 of detection, localization, and RCA tasks. Table 9 shows the mitigation results.

Table 8: Detailed results of Detection, Localization, and RCA for STRATUS (GPT-4o) in AIOpsLab. We use the Pass@3 metric to calculate localization accuracy. Non-existent tasks are denoted as -.

Problem ID	Detection			Localization				RCA		
	Time (s)	Pass	\$	Time (s)	Pass@3	Succ.	\$	Time (s)	Succ.	\$
k8s_target_port-misconfig-1	39.04	T	0.100	29.90	T	100	0.093	49.33	0%	0.073
k8s_target_port-misconfig-2	49.46	T	0.176	27.97	T	100	0.088	28.97	50%	0.089
k8s_target_port-misconfig-3	42.46	T	0.141	41.19	T	100	0.147	41.00	50%	0.075
auth_miss_mongodb	42.26	T	0.117	28.70	T	50	0.115	43.30	0%	0.083
revoke_auth_mongodb-1	42.98	T	0.066	58.26	T	100	0.050	37.05	50%	0.042
revoke_auth_mongodb-2	45.04	T	0.089	33.27	T	100	0.041	45.45	50%	0.056
user_unregistered_mongodb-1	35.63	T	0.043	34.47	T	100	0.041	34.10	50%	0.042
user_unregistered_mongodb-2	38.21	T	0.044	34.26	T	100	0.042	35.72	50%	0.042
misconfig_app_hotel_res	45.74	T	0.095	36.91	T	100	0.029	45.97	100%	0.058
scale_pod_zero_social_net	72.04	T	0.113	35.15	T	100	0.110	42.39	0%	0.076
assign_to_non_existent_node_social_net	28.62	T	0.045	28.91	T	100	0.045	28.54	0%	0.046
container_kill	43.77	T	0.042	39.72	T	100	0.063	-	-	-
pod_failure_hotel_res	68.94	T	0.050	42.97	T	100	0.064	-	-	-
pod_kill_hotel_res	55.75	T	0.072	275.83	F	0	0.120	-	-	-
network_loss_hotel_res	43.17	T	0.061	58.97	F	0	0.070	-	-	-
network_delay_hotel_res	38.83	T	0.082	35.87	F	0	0.081	-	-	-
noop_detection_hotel_reservation	38.62	F	0.101	-	-	-	-	-	-	-
noop_detection_social_network	38.10	F	0.052	-	-	-	-	-	-	-
astronomy_shop_ad_service_failure	87.84	T	0.094	67.51	F	0	0.067	-	-	-
astronomy_shop_ad_service_high_cpu	40.44	T	0.060	67.16	T	50	0.071	-	-	-
astronomy_shop_ad_service_manual_gc	42.17	T	0.086	64.73	F	0	0.055	-	-	-
astronomy_shop_cart_service_failure	38.37	T	0.092	60.96	T	100	0.111	-	-	-
astronomy_shop_image_slow_load	51.59	T	0.072	76.70	T	33.33	0.209	-	-	-
astronomy_shop_kafka_queue_problems	66.00	T	0.126	65.48	F	0	0.059	-	-	-
astronomy_shop_payment_service_failure	65.14	T	0.054	67.03	F	0	0.085	-	-	-
astronomy_shop_payment_service_unreachable	52.18	T	0.065	66.44	F	0	0.045	-	-	-
astronomy_shop_product_catalog_service_failure	36.90	T	0.084	62.56	F	0	0.046	-	-	-
astronomy_shop_recommendation_service_cache_failure	78.68	T	0.101	54.93	F	0	0.090	-	-	-
astronomy_shop_loadgenerator_flood_homepage	29.59	T	1.183	35.45	F	0	1.345	-	-	-
redploy_without_PV	40.72	T	0.044	-	-	-	-	42.74	0%	0.070
wrong_bin_usage	40.32	T	0.170	298.09	F	0	0.149	40.86	50%	0.137
noop_detection_astronomy_shop	68.54	F	0.056	-	-	-	-	-	-	-
Average	48.35	29/32	0.118	65.34	16/28	51.19	0.126	39.65	34.62%	0.068

Table 9: Detailed results of Mitigation tasks for STRATUS (GPT-4o) in AIOpsLab.

Problem ID	Time (s)	Step	Succ.	\$
k8s_target_port-misconfig-mitigation-1	124.59	13	T	0.116
k8s_target_port-misconfig-mitigation-2	210.08	23	T	0.338
k8s_target_port-misconfig-mitigation-3	148.49	11	T	0.084
assign_to_non_existent_node_social_net-mitigation-1	115.42	12	T	0.051
scale_pod_zero_social_net-mitigation-1	139.03	88	T	0.232
user_unregistered_mongodb-mitigation-1	2,007.58	123	T	0.330
user_unregistered_mongodb-mitigation-2	2,090.04	46	F	0.955
revoke_auth_mongodb-mitigation-1	1,405.48	121	F	0.436
revoke_auth_mongodb-mitigation-2	1,321.47	108	F	0.644
auth_miss_mongodb-mitigation-1	136.60	20	T	0.148
misconfig_app_hotel_res-mitigation-1	1,367.99	10	F	0.805
redploy_without_PV-mitigation-1	839.00	152	T	6.704
wrong_bin_usage-mitigation-1	649.44	51	T	0.562
Average	811.94	59.85	9/13	0.877

H Heuristic Evaluation of STRATUS’s Multi-agent Design

STRATUS adopts a multi-agent design. Recent work [53] categorizes common failure modes of multi-agent systems, including *Specification Issues*, *Inter-Agent Misalignment*, and *Task Verification*. We present a heuristic evaluation of the STRATUS design and how the design addresses common failure modes. Table 10 provides a summary.

Table 10: Failure modes from [53] and STRATUS techniques that mitigate them. Abbreviations: DCF=*Deterministic Control Flow*, CRB=*Confinement (role-based)*, CCL=*Confinement (command-level)*, ORC=*Oracles*, STS=*Specialized Toolset*, UDO=*Undo*, LTR=*Linter*, BTS=*Bootstrapping*, ATD=*Agent Thought Dropout*, RWR=*Reflection with Retries*. Each row shows how the technique conceptually addresses the failure.

Failure Mode	STRATUS Technique(s)	Description and Examples
Specification Issues (System Design)		
Disobey Task Specification	CRB, CCL	Role-based confinement assigns specific tasks for different troubleshooting stages; command-level confinement will validate each action;
Disobey Role Specification	CRB, STS	Each agent has clear roles (diagnosis, mitigation, etc) and is given tools corresponding to the agent type. This allows for agent-specific confinement where each agent is only given the necessary tools to complete the task it’s been assigned.
Step Repetition	ATD, BTS	Agent thought dropout prevents agents from being trapped in previous misleading thoughts; bootstrapping gives focused start.
Loss of Conversation History	RWR	Persistent agent thoughts retain context across retries.
Unaware of Termination Conditions	ORC	Oracles identify task completion and provide extra context.
Inter-Agent Misalignment (Agent Coordination)		
Conversation Reset	DCF, UDO	Dataflow direction between agents is the same as control flow; undo restores state on incorrect or destructive actions.
Fail to Ask for Clarification	RWR	A new retry round can be initiated if the agent fails the task.
Task Derailment	BTS, ATD, UDO	Bootstrapping orients initial step (i.e., where the agent starts); contaminated thought can be dropped; undo can be used to restore the state.
Information Withholding	DCF	Agents will be forced to give a final answer (a part of CrewAI’s design) to determine its success or to notify of failure.
Ignored Other Agent’s Input	DCF, RWR	Agents consume its predecessor’s output (including output of the previous round in retrying) in the dataflow until the task is completed.
Reasoning-Action Mismatch	–	Probably related more to the model’s own capabilities.
Task Verification (Quality Control)		
Premature Termination	ORC, DCF	Task ends only after all agents finish and oracles confirm completion.
No or Incomplete Verification	ORC, CRB, CCL	Task completion and single action are both verified.
Incorrect Verification	ORC	Multiple oracles provide validation to reduce false positives.

Specification Issues. A common failure pattern arises from incomplete or ambiguous role/task definitions, or insufficient system design encoding in agent prompts [53]. STRATUS avoids this failure pattern in multiple aspects (see Table 10). Role-based confinement (CRB) is used to confine the agent’s behavior according to the role and task specifications; command-level confinement (CCL)

will validate each action. Also, STRATUS has specialized toolset (STS) for different agents, to retrieve the relevant information from the system, which can facilitate their task completion.

Inter-Agent Misalignment. The Inter-Agent Misalignment category includes coordination failures among agents, such as one agent’s incorrect output misleading downstream agents [53]. Take our context as the example. If the detection agent identifies the wrong error location, the mitigation agent may inadvertently repair the wrong service. STRATUS addresses such issues by employing deterministic control flow (DCF), which coordinates the agents to work together and communicate clearly with each other. STRATUS also uses reflection with retries (RWR), which enables the system to reassess and relaunch a task (within the bounded risk window K). In addition, the undo capability (UDO) allows the system to revert to a safe state after erroneous executions. Furthermore, agent thought dropout (ATD) prevents potentially corrupted or misleading context propagated from previous agents.

Task Verification. Inadequate verification that fails to detect or correct errors or premature termination of tasks can make multi-agent systems fail [53]. STRATUS deploys three oracles (ORC) including alert oracle, user request oracle, and system health oracle, which periodically assess system states and validate whether termination conditions have been met, to ensure the agent is terminating appropriately. Complementing these, STRATUS has confinement (CRB, CCL, LTR), e.g., command dry-run and syntax check, to ensure that commands are semantically and syntactically checked prior to execution.

I An Example of STRATUS’s Trajectory

We show the trajectory of STRATUS when solving a mitigation task (Problem ID: redeploy_without_PV-mitigation-1) in AIOpsLab. In this problem, the storageclass of the persistent volumes (PVs) is mistakenly removed during redeployment. As a result, the persistent volumes claims (PVCs) will not be able to find the corresponding storageclass in the new deployment, and most of the pods will be in an erroneous state.

We show the key actions taken by STRATUS in “Gold Action.” Statements marked as “Note” are not part of the trajectory but are added for clarity.

Phase: Initialization	Run #1
<pre>[21:34:50] All pods in namespace 'test-hotel-reservation' are ready. == Fault Injection == [21:34:51] Waiting for namespace 'test-hotel-reservation' to be deleted... [21:35:18] Namespace 'test-hotel-reservation' has been deleted. Namespace 'test-hotel-reservation' deleted successfully. Deleting namespace test-hotel-reservation without deleting the PV. Redeploying test-hotel-reservation. Namespace test-hotel-reservation not found. Creating namespace. Namespace test-hotel-reservation created successfully: namespace/test-hotel-reservation created</pre>	

Task: Bootstrapping	Diagnosis Agent
<p>Analyze the traces to localize which part of the system, e.g., a service, a component, a pod, a node, etc., has ↳ anomalies. Always use the <code>get_traces</code> tool first; then use a <code>read_traces</code> tool to help you with the localization. ↳ Note that it may not be helpful to use the tools too many times. You are working on the namespace ↳ 'test-hotel-reservation', please use this namespace when using the tools.</p>	

Thought & Action	Diagnosis Agent
<p>To analyze the traces and localize anomalies in the 'test-hotel-reservation' namespace, I will first use the <code>get_traces</code> tool to fetch the traces for a specified duration. Then, I will use the <code>read_traces</code> tool to read and analyze the traces.</p> <hr/> <p>Call <code>get_traces</code> with <code>{"namespace": "test-hotel-reservation", "duration": 30}</code></p>	

Tool Usage: <code>get_traces</code>	Diagnosis Agent
<p>Traces data exported to: <code>/eval/xagent/trace_output/traces_1747186602.csv</code> Please take the next action</p>	

1193	<div>Thought & Action</div> <div>Diagnosis Agent</div> <div>Call read_traces with {"file_path": "/eval/xagent/trace_output/traces_1747186602.csv"}</div>
1194	<div>Tool Usage: read_traces</div> <div>Diagnosis Agent</div> <div> <p>The following list of services are the last service the request reached before returning an error, these services</p> <p>↪ may not be the exact place of the root cause, so you may need to dig deeper. These services correspond to a</p> <p>↪ list of operations happened there, which is the service relying on and could be where the root cause comes</p> <p>↪ from. Here is the list:</p> <p>1: {service: "search", operation: "geo"}</p> <p>2: {service: "frontend", operation: "recommendation"}</p> </div>
1195	<div>Task Bootstrapping Finished</div> <div>Diagnosis Agent</div> <div> <ul style="list-style-type: none"> List of entities with anomalies: <ul style="list-style-type: none"> 1. Service: "search", Operation: "geo" 2. Service: "frontend", Operation: "recommendation" Possible root cause entities for further analysis: <ul style="list-style-type: none"> Service: "search", Operation: "geo" Service: "frontend", Operation: "recommendation" </div>
1196	<div>Task: Mitigation</div> <div>Mitigation Agent</div> <div>The problem description is ''You are an expert DevOps engineer assigned to mitigate anomalies in a deployed ↪ service...</div>
1197	<div>Thought & Action</div> <div>Mitigation Agent</div> <div> <p>To begin the mitigation process, I need to gather more information about the services with anomalies. The anomalies are present in the "search" service with the "geo" operation and the "frontend" service with the "recommendation" operation. I will start by fetching the logs for these services to understand the issues better. First, I will fetch the logs for the "search" service in the "test-hotel-reservation" namespace.</p> <hr/> <p>Call get_logs with {"namespace": "test-hotel-reservation", "service": "search"}</p> </div>
1198	<div>Tool Usage: get_logs</div> <div>Mitigation Agent</div> <div> <pre> {"level":"info","time":"2025-05-14T01:36:03Z","message":"TLS disabled."} {"level":"info","time":"2025-05-14T01:36:03Z","message":"Set global log level: info"} {"level":"info","time":"2025-05-14T01:36:03Z","message":"Tune: setGCPPercent to 100"} 2025-05-14T01:36:03Z INF cmd/search/main.go:24 > Reading config... 2025-05-14T01:36:03Z INF cmd/search/main.go:40 > Read target port: 8082 2025-05-14T01:36:03Z INF cmd/search/main.go:41 > Read consul address: consul:8500 2025-05-14T01:36:03Z INF cmd/search/main.go:42 > Read jaeger address: jaeger:6831 2025-05-14T01:36:03Z INF cmd/search/main.go:51 > Initializing jaeger agent [service name: search host: ↪ jaeger:6831]... 2025-05-14T01:36:03Z INF tracing/tracer.go:26 > Jaeger client: adjusted sample ratio 1.000000 2025-05-14T01:36:03Z INF cmd/search/main.go:56 > Jaeger agent initialized 2025-05-14T01:36:03Z INF cmd/search/main.go:58 > Initializing consul agent [host: consul:8500]... 2025-05-14T01:36:03Z INF cmd/search/main.go:63 > Consul agent initialized 2025-05-14T01:36:03Z INF cmd/search/main.go:74 > Starting server... 2025-05-14T01:36:03Z INF registry/registry.go:91 > Trying to register service [name: srv-search, id: ↪ 254379a6-bcf4-478c-a828-268a443478fb, address: 10.244.2.21:8082] 2025-05-14T01:36:03Z INF services/search/server.go:100 > Successfully registered in consul Please take the next action </pre> </div>
1199	<div>Thought & Action</div> <div>Mitigation Agent</div> <div> <p>The logs for the "search" service do not indicate any errors or anomalies. The service seems to be initializing and registering successfully. Next, I will fetch the logs for the "frontend" service in the "test-hotel-reservation" namespace to investigate the "recommendation" operation anomaly.</p> <hr/> <p>Call get_logs with {"namespace": "test-hotel-reservation", "service": "frontend"}</p> </div>

1200

Tool Usage: get_logs	Mitigation Agent
<pre>{"level":"info","time":"2025-05-14T01:35:46Z","message":"TLS disabled."} {"level":"info","time":"2025-05-14T01:35:46Z","message":"Set global log level: info"} {"level":"info","time":"2025-05-14T01:35:46Z","message":"Tune: setGCPercent to 100"} 2025-05-14T01:35:46Z INF cmd/frontend/main.go:23 > Reading config... ... 2025/05/14 01:35:46 grpc/lb/consul: error retrieving instances from Consul: Get ↳ "http://consul:8500/v1/health/service/srv-search?passing=1": dial tcp 10.96.73.138:8500: connect: connection ↳ refused ... Please take the next action</pre>	

1201

Thought & Action	Mitigation Agent
<p>The logs for the "frontend" service indicate a connection issue with the Consul service. The error message "dial tcp 10.96.73.138:8500: connect: connection refused" suggests that the frontend service is unable to connect to the Consul service, which is likely causing the anomaly in the "recommendation" operation. To address this issue, I will check the status of the Consul service in the "test-hotel-reservation" namespace to ensure it is running and accessible.</p> <hr/>	
Call NL2Kubectl Tool with {"nl_query": "get pods -n test-hotel-reservation"}	

1202

Tool Usage: NL2Kubectl Tool			Mitigation Agent	
NAME	READY	STATUS	RESTARTS	AGE
consul-b84877fb4-v9sgv	1/1	Running	0	74s
frontend-69d95967fb-cgrjc	1/1	Running	0	73s
geo-99d8c58c-5wrbn	0/1	Error	2 (32s ago)	73s
jaeger-67445b6dd6-7kbpd	1/1	Running	0	71s
memcached-profile-66bb9c8b9d-86bpf	1/1	Running	0	70s
memcached-rate-68585796cb-gtpg6	1/1	Running	0	67s
memcached-reserve-646c7dcd6b-vwz8t	1/1	Running	0	62s
mongodb-geo-5ff578bcb8-bg6cr	0/1	Pending	0	71s
mongodb-profile-758cb77f9f-thlpl	0/1	Pending	0	70s
mongodb-rate-56cc8659c9-7dwkw	0/1	Pending	0	66s
mongodb-recommendation-55699465f7-f9hng	0/1	Pending	0	64s
mongodb-reservation-5994859869-zvmt	0/1	Pending	0	60s
mongodb-user-969c6c449-86tnh	0/1	Pending	0	58s
profile-676fcc86d5-v7c55	1/1	Running	2 (26s ago)	69s
rate-c9bc58c85-klcq7	1/1	Running	2 (24s ago)	65s
recommendation-8675c6c9ff-hqbl8	1/1	Running	2 (22s ago)	64s
reservation-7699b8fdd4-756lh	0/1	Error	1 (31s ago)	60s
search-65df96d8df-6lnw2	1/1	Running	0	59s
user-8477d787d8-d8d45	0/1	Error	1 (29s ago)	57s

1203

Thought & Action	Mitigation Agent
<p>The Consul pod is running, which suggests that the issue might not be with the Consul service itself but rather with network connectivity or configuration. The error message "connection refused" typically indicates that the service is not reachable at the specified address or port. Given that the Consul pod is running, I will now check the service configuration for Consul to ensure it is correctly set up to expose the necessary ports.</p> <hr/>	
Call NL2Kubectl Tool with {"nl_query": "describe service consul -n test-hotel-reservation"}	

1204

Tool Usage: NL2Kubectl Tool	Mitigation Agent
<pre>Name: consul ... Port: 8300 8300/TCP TargetPort: 8300/TCP Endpoints: 10.244.2.17:8300 Port: 8400 8400/TCP TargetPort: 8400/TCP Endpoints: 10.244.2.17:8400 Port: 8500 8500/TCP TargetPort: 8500/TCP Endpoints: 10.244.2.17:8500 Port: 8600 8600/UDP TargetPort: 53/UDP Endpoints: 10.244.2.17:53 ...</pre>	

1205	<table border="1"> <thead> <tr> <th data-bbox="284 186 1104 231">Thought & Action</th><th data-bbox="1104 186 1339 231">Mitigation Agent</th></tr> </thead> <tbody> <tr> <td data-bbox="284 231 1104 378"> <p>The Consul service is correctly configured to expose port 8500, which is the port the frontend service is trying to connect to. The IP address and port are correctly set, and the endpoints are available. This suggests that the issue might be related to network policies or firewall rules that are preventing the frontend service from connecting to the Consul service.</p> <p>To further investigate, I will check if there are any network policies in place that might be affecting connectivity between the frontend and Consul services.</p> </td><td data-bbox="1104 231 1339 378"></td></tr> <tr> <td data-bbox="284 378 1104 430"> <p>Call NL2Kubectl Tool with {"nl_query": "get networkpolicy -n test-hotel-reservation"}</p> </td><td data-bbox="1104 378 1339 430"></td></tr> </tbody> </table>	Thought & Action	Mitigation Agent	<p>The Consul service is correctly configured to expose port 8500, which is the port the frontend service is trying to connect to. The IP address and port are correctly set, and the endpoints are available. This suggests that the issue might be related to network policies or firewall rules that are preventing the frontend service from connecting to the Consul service.</p> <p>To further investigate, I will check if there are any network policies in place that might be affecting connectivity between the frontend and Consul services.</p>		<p>Call NL2Kubectl Tool with {"nl_query": "get networkpolicy -n test-hotel-reservation"}</p>	
Thought & Action	Mitigation Agent						
<p>The Consul service is correctly configured to expose port 8500, which is the port the frontend service is trying to connect to. The IP address and port are correctly set, and the endpoints are available. This suggests that the issue might be related to network policies or firewall rules that are preventing the frontend service from connecting to the Consul service.</p> <p>To further investigate, I will check if there are any network policies in place that might be affecting connectivity between the frontend and Consul services.</p>							
<p>Call NL2Kubectl Tool with {"nl_query": "get networkpolicy -n test-hotel-reservation"}</p>							
1206	<table border="1"> <thead> <tr> <th data-bbox="284 430 1104 483">Tool Usage: NL2Kubectl Tool</th><th data-bbox="1104 430 1339 483">Mitigation Agent</th></tr> </thead> <tbody> <tr> <td data-bbox="284 483 1104 577"></td><td data-bbox="1104 483 1339 577"></td></tr> </tbody> </table>	Tool Usage: NL2Kubectl Tool	Mitigation Agent				
Tool Usage: NL2Kubectl Tool	Mitigation Agent						
1207	<table border="1"> <thead> <tr> <th data-bbox="284 577 1104 619">Thought & Action</th><th data-bbox="1104 577 1339 619">Mitigation Agent</th></tr> </thead> <tbody> <tr> <td data-bbox="284 619 1104 945"> <p>There are no network policies listed in the "test-hotel-reservation" namespace, which means there are no restrictions on network traffic between the services. This suggests that the issue might be related to the configuration of the frontend service itself or a misconfiguration in the Consul service.</p> <p>Given the information gathered, the root cause of the anomaly in the "frontend" service's "recommendation" operation appears to be a connectivity issue with the Consul service. The error message "connection refused" indicates that the frontend service is unable to reach the Consul service at the specified address and port.</p> <p>Mitigation Plan:</p> <ol style="list-style-type: none"> 1. Verify the network configuration and ensure that the frontend service can reach the Consul service on port 8500. 2. Check the firewall settings and ensure that there are no rules blocking traffic between the frontend and Consul services. 3. Review the configuration of the frontend service to ensure it is correctly set up to connect to the Consul service. 4. If necessary, restart the Consul service to ensure it is running correctly and accepting connections. <p>Thought: I have identified the root cause of the anomaly and outlined a mitigation plan. I will now submit the solution.</p> </td><td data-bbox="1104 619 1339 945"></td></tr> <tr> <td data-bbox="284 945 1104 1018"> <p>Call submit with {}</p> </td><td data-bbox="1104 945 1339 1018"></td></tr> </tbody> </table>	Thought & Action	Mitigation Agent	<p>There are no network policies listed in the "test-hotel-reservation" namespace, which means there are no restrictions on network traffic between the services. This suggests that the issue might be related to the configuration of the frontend service itself or a misconfiguration in the Consul service.</p> <p>Given the information gathered, the root cause of the anomaly in the "frontend" service's "recommendation" operation appears to be a connectivity issue with the Consul service. The error message "connection refused" indicates that the frontend service is unable to reach the Consul service at the specified address and port.</p> <p>Mitigation Plan:</p> <ol style="list-style-type: none"> 1. Verify the network configuration and ensure that the frontend service can reach the Consul service on port 8500. 2. Check the firewall settings and ensure that there are no rules blocking traffic between the frontend and Consul services. 3. Review the configuration of the frontend service to ensure it is correctly set up to connect to the Consul service. 4. If necessary, restart the Consul service to ensure it is running correctly and accepting connections. <p>Thought: I have identified the root cause of the anomaly and outlined a mitigation plan. I will now submit the solution.</p>		<p>Call submit with {}</p>	
Thought & Action	Mitigation Agent						
<p>There are no network policies listed in the "test-hotel-reservation" namespace, which means there are no restrictions on network traffic between the services. This suggests that the issue might be related to the configuration of the frontend service itself or a misconfiguration in the Consul service.</p> <p>Given the information gathered, the root cause of the anomaly in the "frontend" service's "recommendation" operation appears to be a connectivity issue with the Consul service. The error message "connection refused" indicates that the frontend service is unable to reach the Consul service at the specified address and port.</p> <p>Mitigation Plan:</p> <ol style="list-style-type: none"> 1. Verify the network configuration and ensure that the frontend service can reach the Consul service on port 8500. 2. Check the firewall settings and ensure that there are no rules blocking traffic between the frontend and Consul services. 3. Review the configuration of the frontend service to ensure it is correctly set up to connect to the Consul service. 4. If necessary, restart the Consul service to ensure it is running correctly and accepting connections. <p>Thought: I have identified the root cause of the anomaly and outlined a mitigation plan. I will now submit the solution.</p>							
<p>Call submit with {}</p>							
1208	<table border="1"> <thead> <tr> <th data-bbox="284 1018 1104 1060">Phase: Submission Validation</th><th data-bbox="1104 1018 1339 1060">Run #1</th></tr> </thead> <tbody> <tr> <td data-bbox="284 1060 1104 1249"> <p>Submission triggered. Validating...</p> <p>Waiting 15 seconds for changes to take effect...</p> <p>Testing workload generator...</p> <p>...</p> <p>117 requests in 10.00s, 30.62KB read</p> <p>Non-2xx or 3xx responses: 117</p> <p>Requests/sec: 11.70</p> <p>Transfer/sec: 3.06KB</p> </td><td data-bbox="1104 1060 1339 1249"></td></tr> </tbody> </table>	Phase: Submission Validation	Run #1	<p>Submission triggered. Validating...</p> <p>Waiting 15 seconds for changes to take effect...</p> <p>Testing workload generator...</p> <p>...</p> <p>117 requests in 10.00s, 30.62KB read</p> <p>Non-2xx or 3xx responses: 117</p> <p>Requests/sec: 11.70</p> <p>Transfer/sec: 3.06KB</p>			
Phase: Submission Validation	Run #1						
<p>Submission triggered. Validating...</p> <p>Waiting 15 seconds for changes to take effect...</p> <p>Testing workload generator...</p> <p>...</p> <p>117 requests in 10.00s, 30.62KB read</p> <p>Non-2xx or 3xx responses: 117</p> <p>Requests/sec: 11.70</p> <p>Transfer/sec: 3.06KB</p>							
1209	<table border="1"> <thead> <tr> <th data-bbox="284 1249 1104 1302">Tool Usage: submit</th><th data-bbox="1104 1249 1339 1302">Mitigation Agent</th></tr> </thead> <tbody> <tr> <td data-bbox="284 1302 1104 1396"> <p>The system is not in a valid state. Please resolve all issues before submitting.</p> </td><td data-bbox="1104 1302 1339 1396"></td></tr> </tbody> </table>	Tool Usage: submit	Mitigation Agent	<p>The system is not in a valid state. Please resolve all issues before submitting.</p>			
Tool Usage: submit	Mitigation Agent						
<p>The system is not in a valid state. Please resolve all issues before submitting.</p>							
1210	<table border="1"> <thead> <tr> <th data-bbox="284 1396 1104 1438">Thought & Action</th><th data-bbox="1104 1396 1339 1438">Mitigation Agent</th></tr> </thead> <tbody> <tr> <td data-bbox="284 1438 1104 1554"> <p>The submission failed because the system is not in a valid state, indicating unresolved issues. I need to further investigate and resolve any remaining issues before submitting the solution.</p> <p>The "geo" operation in the "search" service also has an anomaly. I will check the logs for the "geo" pod to gather more information about the error status.</p> </td><td data-bbox="1104 1438 1339 1554"></td></tr> <tr> <td data-bbox="284 1554 1104 1627"> <p>Call get_logs with {"namespace": "test-hotel-reservation", "service": "geo"}</p> </td><td data-bbox="1104 1554 1339 1627"></td></tr> </tbody> </table>	Thought & Action	Mitigation Agent	<p>The submission failed because the system is not in a valid state, indicating unresolved issues. I need to further investigate and resolve any remaining issues before submitting the solution.</p> <p>The "geo" operation in the "search" service also has an anomaly. I will check the logs for the "geo" pod to gather more information about the error status.</p>		<p>Call get_logs with {"namespace": "test-hotel-reservation", "service": "geo"}</p>	
Thought & Action	Mitigation Agent						
<p>The submission failed because the system is not in a valid state, indicating unresolved issues. I need to further investigate and resolve any remaining issues before submitting the solution.</p> <p>The "geo" operation in the "search" service also has an anomaly. I will check the logs for the "geo" pod to gather more information about the error status.</p>							
<p>Call get_logs with {"namespace": "test-hotel-reservation", "service": "geo"}</p>							
1211	<table border="1"> <thead> <tr> <th data-bbox="284 1627 1104 1669">Tool Usage: get_logs</th><th data-bbox="1104 1627 1339 1669">Mitigation Agent</th></tr> </thead> <tbody> <tr> <td data-bbox="284 1669 1104 1879"> <p>...</p> <p>2025-05-14T01:37:15Z INF cmd/geo/main.go:37 > Initializing DB connection...</p> <p>2025-05-14T01:37:27Z PNC cmd/geo/db.go:26 > no reachable servers</p> <p>panic: no reachable servers</p> <p>goroutine 1 [running]:</p> <p>github.com/rs/zerolog.(*Logger).Panic.func1({0x96e211, 0x0})</p> <p>...</p> <p>Please take the next action</p> </td><td data-bbox="1104 1669 1339 1879"></td></tr> </tbody> </table>	Tool Usage: get_logs	Mitigation Agent	<p>...</p> <p>2025-05-14T01:37:15Z INF cmd/geo/main.go:37 > Initializing DB connection...</p> <p>2025-05-14T01:37:27Z PNC cmd/geo/db.go:26 > no reachable servers</p> <p>panic: no reachable servers</p> <p>goroutine 1 [running]:</p> <p>github.com/rs/zerolog.(*Logger).Panic.func1({0x96e211, 0x0})</p> <p>...</p> <p>Please take the next action</p>			
Tool Usage: get_logs	Mitigation Agent						
<p>...</p> <p>2025-05-14T01:37:15Z INF cmd/geo/main.go:37 > Initializing DB connection...</p> <p>2025-05-14T01:37:27Z PNC cmd/geo/db.go:26 > no reachable servers</p> <p>panic: no reachable servers</p> <p>goroutine 1 [running]:</p> <p>github.com/rs/zerolog.(*Logger).Panic.func1({0x96e211, 0x0})</p> <p>...</p> <p>Please take the next action</p>							

1212

Thought & Action	Mitigation Agent
<p>The logs for the "geo" service indicate a panic due to "no reachable servers," which suggests a connectivity issue with the MongoDB database. The error occurs during the initialization of the database connection, indicating that the "geo" service is unable to connect to the MongoDB instance at "mongodb-geo:27017."</p> <p>To address this issue, I will check the status of the MongoDB pod for the "geo" service to ensure it is running and accessible.</p>	
<hr/> <p>Call NL2Kubectl Tool with {"nl_query": "describe pod mongodb-geo-5ff578bcb8-bg6cr -n test-hotel-reservation"}</p>	

1213

Tool Usage: NL2Kubectl Tool	Mitigation Agent
<pre>Name: mongodb-geo-5ff578bcb8-bg6cr Namespace: test-hotel-reservation Priority: 0 Service Account: default Node: <none> Labels: io.kompose.service=mongodb-geo pod-template-hash=5ff578bcb8 ... Events: Type Reason Age From Message ---- - Warning FailedScheduling 2m default-scheduler 0/3 nodes are available: pod has unbound immediate ↳ PersistentVolumeClaims. 0/3 nodes are available: 3 Preemption is not helpful for scheduling.</pre>	

1214

Thought & Action	Mitigation Agent
<p>The "mongodb-geo" pod is in a "Pending" state due to a failed scheduling event. The message "pod has unbound immediate PersistentVolumeClaims" indicates that the pod is unable to start because it cannot bind to the required PersistentVolumeClaim (PVC) named "geo-pvc."</p> <p>To resolve this issue, I need to check the status of the "geo-pvc" to ensure it is correctly configured and available.</p>	
<hr/> <p>Call NL2Kubectl Tool with {"nl_query": "describe pvc geo-pvc -n test-hotel-reservation"}</p>	

1215

Tool Usage: NL2Kubectl Tool	Mitigation Agent
<pre>Name: geo-pvc Namespace: test-hotel-reservation StorageClass: geo-storage Status: Pending Volume: Labels: <none> Annotations: <none> Finalizers: [kubernetes.io/pvc-protection] Capacity: Access Modes: VolumeMode: Filesystem Used By: mongodb-geo-5ff578bcb8-bg6cr Events: Type Reason Age From Message ---- - Warning ProvisioningFailed 4s (x9 over 2m4s) persistentvolume-controller storageclass.storage.k8s.io ↳ "geo-storage" not found</pre>	

1216

Thought & Action	Mitigation Agent
<p>The "geo-pvc" is in a "Pending" state because the storage class "geo-storage" is not found. This is preventing the PersistentVolumeClaim from being bound, which in turn is causing the "mongodb-geo" pod to remain in a "Pending" state.</p> <p>To resolve this issue, I need to create or correct the storage class "geo-storage" so that the PVC can be bound and the pod can start.</p> <p>Mitigation Plan:</p> <ol style="list-style-type: none"> 1. Create a storage class named "geo-storage" with the appropriate configuration to support the "geo-pvc" PersistentVolumeClaim. 2. Ensure that the storage class is available and correctly configured to allow the PVC to bind. 3. Once the storage class is created, the "geo-pvc" should bind successfully, allowing the "mongodb-geo" pod to start. <p>I will now proceed to create the storage class "geo-storage."</p>	
<hr/> <p>Call NL2Kubectl Tool with {"nl_query": "create storageclass geo-storage --provisioner=kubernetes.io/aws-efs --parameters type=gp2"}</p>	

1217

Tool Usage: NL2Kubectl Tool

Mitigation Agent

Dry-run failed. Potentially it's an invalid command. stderr: error: unknown flag: --provisioner
See 'kubectl create --help' for usage.

1218

Thought & Action

Mitigation Agent

It seems the command to create the storage class was incorrect. I will correct the command and attempt to create the storage class "geo-storage" using a valid configuration.

Call NL2Kubectl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: geo-storage\nprovisioner: kubernetes.io/aws-efs\nparameters:\n type: gp2\nEOF"}

1219

Tool Usage: NL2Kubectl Tool

Mitigation Agent

storageclass.storage.k8s.io/geo-storage created

1220

Thought & Action

Mitigation Agent

The storage class "geo-storage" has been successfully created. I will now check the status of the "geo-pvc" to ensure it is bound and the "mongodb-geo" pod can start.

Call NL2Kubectl Tool with {"nl_query": "describe pvc geo-pvc -n test-hotel-reservation"}

1221

Tool Usage: NL2Kubectl Tool

Mitigation Agent

Name: geo-pvc
Namespace: test-hotel-reservation
StorageClass: geo-storage
Status: Pending
Volume:
Labels: <none>
Annotations: <none>
Finalizers: [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode: Filesystem
Used By: mongodb-geo-5ff578bcb8-bg6cr
Events:

Type	Reason	Age	From	Message
Warning	ProvisioningFailed	14s (x9 over 2m14s)	persistentvolume-controller	storageclass.storage.k8s.io ↳ "geo-storage" not found

1222

Thought & Action

Mitigation Agent

The "geo-pvc" is still in a "Pending" state, and the error message indicates that the storage class "geo-storage" is not found. This suggests that there might be an issue with the creation of the storage class or its recognition by the persistent volume controller.
To resolve this, I will verify the existence of the storage class "geo-storage" and ensure it is correctly configured.

Call NL2Kubectl Tool with {"nl_query": "get storageclass geo-storage"}

1223

Tool Usage: NL2Kubectl Tool

Mitigation Agent

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	AGE
geo-storage	kubernetes.io/aws-efs	Delete	Immediate	false	6s

1224

Thought & Action

Mitigation Agent

The storage class "geo-storage" exists and is correctly configured with the provisioner "kubernetes.io/aws-efs." However, the persistent volume controller is still unable to find it, which suggests a potential issue with the cluster configuration or the controller itself.
Given the information gathered, the root cause of the anomaly in the "geo" operation of the "search" service is the inability to bind the "geo-pvc" due to the persistent volume controller not recognizing the storage class "geo-storage."
Mitigation Plan:
1. Verify the cluster configuration to ensure the persistent volume controller is correctly set up to recognize storage classes.
2. Check the logs of the persistent volume controller for any errors or misconfigurations.

1225

```
3. If necessary, restart the persistent volume controller to ensure it is running correctly and can recognize the
storage class.
Thought: I have identified the root cause of the anomaly and outlined a mitigation plan. I will now submit the
solution.

Call submit with {}
```

1226

Phase: Submission Validation

Run #1

```
Submission triggered. Validating...
Waiting 15 seconds for changes to take effect...
Testing workload generator...
...
117 requests in 10.00s, 30.62KB read
Non-2xx or 3xx responses: 117
Requests/sec: 11.70
Transfer/sec: 3.06KB
```

1227

Tool Usage: submit

Mitigation Agent

The system is not in a valid state. Please resolve all issues before submitting.

1228

```
...
A series of observation and submission.
...
```

1229

Task Mitigation Finished

Mitigation Agent

```
Root Cause Analysis:
1. The "frontend" service's "recommendation" operation anomaly is due to a connectivity issue with the Consul
  ↳ service. The error message "connection refused" indicates that the frontend service is unable to reach the
  ↳ Consul service at the specified address and port.
2. The "geo" operation in the "search" service is experiencing a connectivity issue with the MongoDB database. The
  ↳ error "no reachable servers" suggests that the "geo" service is unable to connect to the MongoDB instance due
  ↳ to the "geo-pvc" being in a "Pending" state.
3. The "geo-pvc" is unable to bind because the persistent volume controller is not recognizing the storage class
  ↳ "geo-storage," despite it being correctly configured.

Mitigation Plan:
1. For the "frontend" service:
  - Verify the network configuration and ensure that the frontend service can reach the Consul service on port
    ↳ 8500.
  - Check firewall settings to ensure there are no rules blocking traffic between the frontend and Consul services.
  - Review the configuration of the frontend service to ensure it is correctly set up to connect to the Consul
    ↳ service.
  - Restart the Consul service if necessary to ensure it is running correctly and accepting connections.

2. For the "geo" operation:
  - Verify the cluster configuration to ensure the persistent volume controller is correctly set up to recognize
    ↳ storage classes.
  - Check the cluster setup to ensure all necessary components are installed and configured.
  - Restart the persistent volume controller if necessary to ensure it is running correctly and can recognize the
    ↳ storage class "geo-storage."

These steps should help resolve the anomalies in the "frontend" and "search" services.
```

1230

Phase: Validation & Retry

Run #1

```
-----
117 requests in 10.00s, 44.57KB read
Non-2xx or 3xx responses: 63
Requests/sec: 11.70
Transfer/sec: 4.46KB

Adding issues [' Non-2xx or 3xx responses: 117']
Issues found with pod geo-99d8c58c-5wrbn:
- Container hotel-reserv-geo is in CrashLoopBackOff
...
Issues found with pod user-8477d787d8-d8d45:
- Container hotel-reserv-user is in CrashLoopBackOff
Found 12 issues in the cluster.
Adding issues ['Container hotel-reserv-geo is in CrashLoopBackOff', ...]
Validation result: {'success': False, 'issues': [' Non-2xx or 3xx responses: 117', 'Container hotel-reserv-geo is
  ↳ in CrashLoopBackOff', ...]}
Output written to: eval/05-13_21-28-46-redeploy_without_PV-mitigation-1/lumyn_output/agent_output_0.json
Analyzing previous run results for improvement...
Reflection generated successfully.
```

Note (not part of the trajectory)

A storageclass has been created but not deleted in the first run. However it's created using the wrong provisioner. The Persistent Volume Controller will report the following error:

- Waiting for a volume to be created either by the external provisioner 'ebs.csi.aws.com' or manually by the system administrator. If volume creation is delayed, please verify that the provisioner is running and correctly registered.

Any overwrite to such existing storageclass will result in errors like:

The StorageClass "<storage-class-name>" is invalid:

* provisioner: Forbidden: updates to provisioner are forbidden.

* volumeBindingMode: Invalid value: "WaitForFirstConsumer": field is immutable

Rollback prevents such overwrites and errors in the next run.

1231

Task: Rollback

Rollback Agent

You are given a system and a rollback tool. The system is either in a initial faulty state, or in the state you
↳ left it incompletely repaired after the previous round of mitigation. You need to rollback the system to the
↳ initial state since the operations you performed in the previous round may negatively impact your current round
↳ of mitigation.
You should run the rollback tool as many times as you can, since this rollback tool undos the last operation
↳ performed on the system. You should run the rollback tool until it returns that there are no more operations to
↳ undo.

1232

Thought & Action

Rollback Agent

To rollback the system to its initial state, I need to use the rollback tool repeatedly until it indicates that there are no more operations to undo.

Call rollback_tool with {"argument": "rollback the last operation"}

1233

Tool Usage: rollback_tool

Rollback Agent

Rollbacked the previous command: kubectl apply -f - <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: geo-storage
provisioner: kubernetes.io/aws-ebs
parameters:
 type: gp2
EOF, using rollback:kubectl delete storageclass.storage.k8s.io geo-storage -n test-hotel-reservation

1234

Thought & Action

Rollback Agent

Thought: I need to provide a different argument to the rollback tool to continue the rollback process.

Call rollback_tool with {"argument": "undo the previous operation"}

1235

Tool Usage: rollback_tool

Rollback Agent

No more actions to rollback.

1236

Task Rollback Finished

Rollback Agent

None.

1237

Task: Bootstrapping

Diagnosis Agent

Analyze the traces to localize which part of the system, e.g., a service, a component, a pod, a node, etc., has
↳ anomalies. Always use the get_traces tool first; then use a read_traces tool to help you with the localization.
↳ Note that it may not be helpful to use the tools too many times. You are working on the namespace
↳ 'test-hotel-reservation', please use this namespace when using the tools.

1238

1239	<div>Thought & Action</div> <div>Diagnosis Agent</div> <div> <p>I need to fetch traces from the 'test-hotel-reservation' namespace to analyze and identify any anomalies in the system. I'll start by using the get_traces tool to retrieve the necessary data.</p> <hr/> <p>Call get_traces with {"namespace": "test-hotel-reservation", "duration": 60}</p> </div>
1240	<div>Tool Usage: get_traces</div> <div>Diagnosis Agent</div> <div> <p>Traces data exported to: /eval/xagent/trace_output/traces_1747186964.csv Please take the next action</p> </div>
1241	<div>Thought & Action</div> <div>Diagnosis Agent</div> <div> <p>Thought: I have successfully fetched the traces data. Now, I need to read the traces from the file to analyze and identify any anomalies in the system.</p> <hr/> <p>Call read_traces with {"file_path": "/eval/xagent/trace_output/traces_1747186964.csv"}</p> </div>
1242	<div>Tool Usage: read_traces</div> <div>Diagnosis Agent</div> <div> <p>The following list of services are the last service the request reached before returning an error, these services ↳ may not be the exact place of the root cause, so you may need to dig deeper. These services correspond to a ↳ list of operations happened there, which is the service relying on and could be where the root cause comes ↳ from. Here is the list: 1: {service: "frontend", operation: "on/GetRecommendations"} 2: {service: "search", operation: "geo"} 3: {service: "frontend", operation: "user"}</p> </div>
1243	<div>Task Bootstrapping Finished</div> <div>Diagnosis Agent</div> <div> <ul style="list-style-type: none"> List of entities with anomalies: <ol style="list-style-type: none"> Service: "frontend", Operation: "on/GetRecommendations" Service: "search", Operation: "geo" Service: "frontend", Operation: "user" Possible root cause entities: <ul style="list-style-type: none"> The anomalies may originate from the "frontend" service, particularly in the "on/GetRecommendations" and "user" operations, as well as the "search" service in the "geo" operation. Further analysis can start from these root cause entities. </div>
1244	<div>Task: Mitigation</div> <div>Mitigation Agent</div> <div> <p>The problem description is ''You are an expert DevOps engineer assigned to mitigate anomalies in a deployed ↳ service...</p> <p>Reflection on the previous run: Root Cause: 1. The anomaly in the "frontend" service's "recommendation" operation is due to a connectivity issue with the ↳ Consul service, indicated by the error message "connection refused." 2. The anomaly in the "geo" operation of the "search" service is due to the inability to bind the "geo-pvc" because ↳ the persistent volume controller is not recognizing the storage class "geo-storage."</p> <p>Mitigation Plan: 1. For the "frontend" service: <ul style="list-style-type: none"> Verify the network configuration and ensure that the frontend service can reach the Consul service on port ↳ 8500. Check the firewall settings and ensure that there are no rules blocking traffic between the frontend and ↳ Consul services. Review the configuration of the frontend service to ensure it is correctly set up to connect to the Consul ↳ service. If necessary, restart the Consul service to ensure it is running correctly and accepting connections. For the "geo" operation in the "search" service: <ul style="list-style-type: none"> Verify the cluster configuration to ensure the persistent volume controller is correctly set up to recognize ↳ storage classes. Check the cluster setup to ensure all necessary components are installed and configured. If necessary, restart the persistent volume controller to ensure it is running correctly and can recognize the ↳ storage class.. <p>The following issues were found in the cluster: - Non-2xx or 3xx responses: 117 - Container hotel-reserv-geo is in CrashLoopBackOff - Pod mongodb-geo-5ff578bcb8-bg6cr is in Pending state - Pod mongodb-profile-758cb77f9f-thlpl is in Pending state - ...</p> </p></div>

	Thought & Action	Mitigation Agent
1245	<p>The previous run has already identified the root causes and provided a mitigation plan. I will follow the plan to resolve the issues. The plan involves two main tasks: fixing the connectivity issue with the Consul service for the "frontend" service and addressing the persistent volume controller issue for the "geo" operation in the "search" service.</p> <p>First, I will verify the network configuration and ensure that the frontend service can reach the Consul service on port 8500. I will check the firewall settings and review the configuration of the frontend service. If necessary, I will restart the Consul service.</p> <p>Next, I will verify the cluster configuration to ensure the persistent volume controller is correctly set up to recognize storage classes. I will check the cluster setup and restart the persistent volume controller if necessary.</p> <p>Let's start by checking the network configuration for the "frontend" service to ensure it can reach the Consul service.</p> <hr/> <p>Call NL2Kubectl Tool with {"nl_query": "describe service frontend -n test-hotel-reservation"}</p>	

	Tool Usage: NL2Kubectl Tool	Mitigation Agent
1246	<pre> Name: frontend Namespace: test-hotel-reservation Labels: io.kompose.service=frontend Annotations: kompose.cmd: kompose convert kompose.version: 1.22.0 (955b78124) Selector: io.kompose.service=frontend Type: ClusterIP IP Family Policy: SingleStack IP Families: IPv4 IP: 10.96.54.225 IPs: 10.96.54.225 Port: 5000 TargetPort: 5000/TCP ... </pre>	

	Thought & Action	Mitigation Agent
1247	<p>The "frontend" service is configured with a ClusterIP and is listening on port 5000. However, the issue is related to connectivity with the Consul service on port 8500. I need to check if the Consul service is running and accessible on the expected port.</p> <p>Next, I will describe the Consul service to verify its configuration and ensure it is running correctly.</p> <hr/> <p>Call NL2Kubectl Tool with {"nl_query": "describe service consul -n test-hotel-reservation"}</p>	

	Tool Usage: NL2Kubectl Tool	Mitigation Agent
1248	<pre> Name: consul ... Port: 8300 TargetPort: 8300/TCP Endpoints: 10.244.2.17:8300 Port: 8400 TargetPort: 8400/TCP Endpoints: 10.244.2.17:8400 Port: 8500 TargetPort: 8500/TCP Endpoints: 10.244.2.17:8500 Port: 8600 TargetPort: 53/UDP Endpoints: 10.244.2.17:53 ... </pre>	

	Thought & Action	Mitigation Agent
1249	<p>The Consul service is correctly configured with a ClusterIP and is listening on port 8500, which matches the expected configuration. The endpoints are also correctly set up. This suggests that the issue might be related to firewall settings or the frontend service configuration.</p> <p>I will now check the firewall settings to ensure there are no rules blocking traffic between the frontend and Consul services.</p> <hr/> <p>Call NL2Kubectl Tool with {"nl_query": "get networkpolicy -n test-hotel-reservation"}</p>	

	Tool Usage: NL2Kubectl Tool	Mitigation Agent
1250		

1251

Thought & Action

Mitigation Agent

There are no network policies defined in the "test-hotel-reservation" namespace, which means there are no firewall rules blocking traffic between the frontend and Consul services. The issue might be related to the frontend service configuration.

I will now check the configuration of the frontend service to ensure it is correctly set up to connect to the Consul service.

Call NL2Kubectl Tool with {"nl_query": "describe deployment frontend -n test-hotel-reservation"}

1252

Tool Usage: NL2Kubectl Tool

Mitigation Agent

Name:frontendNamespace:test-hotel-reservationCreationTimestamp:Tue, 13 May 2025 21:35:37 -0400Labels:io.kompose.service=frontendAnnotations:deployment.kubernetes.io/revision: 1kompose.cmd: kompose convertkompose.version: 1.22.0 (955b78124)io.kompose.service=frontendSelector:io.kompose.service=frontendReplicas:1 desired | 1 updated | 1 total | 1 available | 0 unavailableStrategyType:RollingUpdate...

1253

Thought & Action

Mitigation Agent

The frontend deployment is configured correctly, and there are no apparent issues with the environment variables or command setup. The deployment is running and available, which suggests that the connectivity issue might be transient or related to the Consul service itself.

I will proceed to restart the Consul service to ensure it is running correctly and accepting connections.

Call NL2Kubectl Tool with {"nl_query": "rollout restart deployment consul -n test-hotel-reservation"}

1254

Tool Usage: NL2Kubectl Tool

Mitigation Agent

deployment.apps/consul restarted

1255

Thought & Action

Mitigation Agent

The Consul deployment has been restarted. This should help resolve any transient connectivity issues between the frontend service and the Consul service.

Next, I will address the issue with the "geo" operation in the "search" service, which is related to the persistent volume controller not recognizing the storage class "geo-storage." I will start by checking the persistent volume claims to verify their status.

Call NL2Kubectl Tool with {"nl_query": "get pvc -n test-hotel-reservation"}

1256

Tool Usage: NL2Kubectl Tool

Mitigation Agent

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS
↪ AGE						
geo-pvc	Pending				geo-storage	<unset>
↪ 7m23s						
profile-pvc	Pending				profile-storage	<unset>
↪ 7m20s						
rate-pvc	Pending				rate-storage	<unset>
↪ 7m16s						
recommendation-pvc	Pending				recommendation-storage	<unset>
↪ 7m14s						
reservation-pvc	Pending				reservation-storage	<unset>
↪ 7m11s						
user-pvc	Pending				user-storage	<unset>
↪ 7m8s						

1257

Thought & Action

Mitigation Agent

The persistent volume claims (PVCs) are all in a "Pending" state, indicating that they are not being bound to any persistent volumes. This is likely due to the storage classes not being recognized by the persistent volume controller.

I will now check the storage classes to ensure they are correctly configured and recognized by the cluster.

Call NL2Kubectl Tool with {"nl_query": "get storageclass"}

1258

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION
↪ AGE openebs-device	openebs.io/local	Delete	WaitForFirstConsumer	false
↪ 13m openebs-hostpath (default)	openebs.io/local	Delete	WaitForFirstConsumer	false
↪ 13m standard (default)	rancher.io/local-path	Delete	WaitForFirstConsumer	false
↪ 14m				

1259

Thought & Action

Mitigation Agent

The storage classes "geo-storage", "profile-storage", "rate-storage", "recommendation-storage", "reservation-storage", and "user-storage" are not listed among the available storage classes. This indicates that these storage classes are not defined in the cluster, which is why the PVCs are in a "Pending" state. To resolve this issue, I need to create the missing storage classes. I will start by creating the "geo-storage" class.

Call NL2Kubecttl Tool with {"nl_query": "create storageclass geo-storage --provisioner=rancher.io/local-path --reclaim-policy=Delete --volume-binding-mode=WaitForFirstConsumer"}

1260

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

Dry-run failed. Potentially it's an invalid command. stderr: error: unknown flag: --provisioner
See 'kubecttl create --help' for usage.

1261

Gold Action 1

Mitigation Agent

It seems the command to create a storage class was incorrect. I will use the correct syntax to create the "geo-storage" storage class using a YAML file.

Call NL2Kubecttl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: geo-storage\nprovisioner: rancher.io/local-path\nreclaimPolicy: Delete\nvolumeBindingMode: WaitForFirstConsumer\nEOF"}

1262

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

storageclass.storage.k8s.io/geo-storage created

1263

Gold Action 2

Mitigation Agent

The "geo-storage" storage class has been successfully created. I will proceed to create the other missing storage classes: "profile-storage", "rate-storage", "recommendation-storage", "reservation-storage", and "user-storage".

Call NL2Kubecttl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: profile-storage\nprovisioner: rancher.io/local-path\nreclaimPolicy: Delete\nvolumeBindingMode: WaitForFirstConsumer\nEOF"}

1264

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

storageclass.storage.k8s.io/profile-storage created

1265

Gold Action 3

Mitigation Agent

The "profile-storage" storage class has been successfully created. I will continue to create the remaining storage classes.

Call NL2Kubecttl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: rate-storage\nprovisioner: rancher.io/local-path\nreclaimPolicy: Delete\nvolumeBindingMode: WaitForFirstConsumer\nEOF"}

1266

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

storageclass.storage.k8s.io/rate-storage created

1267

Gold Action 4

Mitigation Agent

The "rate-storage" storage class has been successfully created. I will proceed to create the "recommendation-storage" storage class.

Call NL2Kubectl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: recommendation-storage\nprovisioner: rancher.io/local-path\nreclaimPolicy: Delete\nvolumeBindingMode: WaitForFirstConsumer\nEOF"}

1268

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

storageclass.storage.k8s.io/recommendation-storage created

1269

Gold Action 5

Mitigation Agent

The "recommendation-storage" storage class has been successfully created. I will proceed to create the "reservation-storage" storage class.

Call NL2Kubectl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: reservation-storage\nprovisioner: rancher.io/local-path\nreclaimPolicy: Delete\nvolumeBindingMode: WaitForFirstConsumer\nEOF"}

1270

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

storageclass.storage.k8s.io/reservation-storage created

1271

Gold Action 6

Mitigation Agent

The "reservation-storage" storage class has been successfully created. I will proceed to create the "user-storage" storage class.

Call NL2Kubectl Tool with {"nl_query": "apply -f - «EOF\napiVersion: storage.k8s.io/v1\nkind: StorageClass\nmetadata:\n name: user-storage\nprovisioner: rancher.io/local-path\nreclaimPolicy: Delete\nvolumeBindingMode: WaitForFirstConsumer\nEOF"}

1272

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

storageclass.storage.k8s.io/user-storage created

1273

Thought & Action

Mitigation Agent

The "user-storage" storage class has been successfully created. All the necessary storage classes are now in place. This should resolve the issue with the persistent volume claims being in a "Pending" state. I will now check the status of the persistent volume claims to ensure they are bound correctly.

Call NL2Kubectl Tool with {"nl_query": "get pvc -n test-hotel-reservation"}

1274

Tool Usage: NL2Kubecttl Tool

Mitigation Agent

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS
↪ VOLUMEATTRIBUTESCLASS	AGE				
geo-pvc	Bound	pvc-2937753a-cc60-4904-9371-d89b0db18ecd	1Gi	RWO	geo-storage
↪ <unset>	7m53s				
profile-pvc	Bound	pvc-a4f1d897-89ca-4cdf-8f94-0958942aa243	1Gi	RWO	profile-storage
↪ <unset>	7m50s				
rate-pvc	Pending				rate-storage
↪ <unset>	7m46s				
recommendation-pvc	Pending				
↪ recommendation-storage	<unset>	7m44s			
reservation-pvc	Pending				
↪ reservation-storage	<unset>	7m41s			
user-pvc	Pending				user-storage
↪ <unset>	7m38s				

Thought & Action	Mitigation Agent
<p>The "geo-pvc" and "profile-pvc" are now bound, indicating that the storage classes are being recognized. However, the other PVCs are still in a "Pending" state. This might be due to resource constraints or scheduling issues. I will check the events for one of the pending PVCs to understand why it is not being bound.</p> <hr/> <p>Call NL2Kubect1 Tool with {"nl_query": "describe pvc rate-pvc -n test-hotel-reservation"}</p>	

Tool Usage: NL2Kubectrl Tool

Mitigation Agent

Name:rate-pvc

Namespace:test-hotel-reservation

StorageClass:rate-storage

Status:Pending

Volume:

Labels:<none>

Annotations:volume.beta.kubernetes.io/storage-provisioner: rancher.io/local-path
volume.kubernetes.io/selected-node: kind-worker2
volume.kubernetes.io/storage-provisioner: rancher.io/local-path

Finalizers:[kubernetes.io/pvc-protection]

Capacity:

Access Modes:

VolumeMode:Filesystem

Used By:mongodb-rate-56cc8659c9-7dwkw

Events:

TypeReasonAgeFrom

↩Message---------

↩-----

WarningProvisioningFailed100s (x26 over 7m48s)persistentvolume-controller

↩storageclass.storage.k8s.io "rate-storage" not found

NormalWaitForPodScheduled10spersistentvolume-controller

↩waiting for pod mongodb-rate-56cc8659c9-7dwkw to be scheduled

NormalExternalProvisioning5spersistentvolume-controller

↩Waiting for a volume to be created either by the external provisioner 'rancher.io/local-path' or manually by the system administrator. If volume creation is delayed, please verify that the provisioner is running and correctly registered.

NormalProvisioning5s

↩rancher.io/local-path_local-path-provisioner-58cc7856b6-n29rr_bf374c2e-efd5-4d8b-a53f-0ac317a46e1a External provisioner is provisioning volume for claim "test-hotel-reservation/rate-pvc"

Thought & Action	Mitigation Agent
<p>The "rate-pvc" is pending due to the storage class "rate-storage" not being found initially, but it seems the provisioner is now attempting to provision the volume. This indicates that the issue might resolve itself as the provisioner completes its task.</p> <p>I will check the status of the persistent volume claims again to see if they have been bound.</p>	
<p>Call NL2Kubect1 Tool with {"nl_query": "get pvc -n test-hotel-reservation"}</p>	

Tool Usage: NL2Kubectrl Tool				Mitigation Agent		
NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	
↳ VOLUMEATTRIBUTESCLASS	AGE					
geo-pvc	Bound	pvc-2937753a-cc60-4904-9371-d89b0db18ecd	1Gi	RWO	geo-storage	
↳ <unset>		7m57s				
profile-pvc	Bound	pvc-a4f1d897-89ca-4cdf-8f94-0958942aa243	1Gi	RWO	profile-storage	
↳ <unset>		7m54s				
rate-pvc	Pending				rate-storage	
↳ <unset>		7m50s				
recommendation-pvc	Pending					
↳ recommendation-storage	<unset>	7m48s				
reservation-pvc	Bound	pvc-b7dfe091-9243-41ba-8175-92d350c81f95	1Gi	RWO		
↳ reservation-storage	<unset>	7m45s				
user-pvc	Pending				user-storage	
↳ <unset>		7m42s				

Thought & Action	Mitigation Agent
<p>The "reservation-pvc" has been successfully bound, but "rate-pvc", "recommendation-pvc", and "user-pvc" are still pending. The provisioner is working on these claims, and they may resolve shortly.</p> <p>I will now check the logs of the "hotel-reserv-geo" container to see if there are any errors related to the CrashLoopBackOff state.</p>	
<p>Call <code>get_logs</code> with <code>{ "namespace": "test-hotel-reservation", "service": "hotel-reserv-geo" }</code></p>	

1280

Tool Usage: get_logs

Mitigation Agent

Error: Your service/namespace does not exist. Use kubectl to check.
Please take the next action

1281

Thought & Action

Mitigation Agent

It seems there was an error retrieving logs for the "hotel-reserv-geo" container, possibly due to a misconfiguration or the service not existing. I will check the pods in the "test-hotel-reservation" namespace to verify their status and identify any issues.

Call NL2Kubectl Tool with {"nl_query": "get pods -n test-hotel-reservation"}

1282

Tool Usage: NL2Kubectl Tool

Mitigation Agent

NAME	READY	STATUS	RESTARTS	AGE
consul-65844d4555-5vpkm	1/1	Running	0	40s
frontend-69d95967fb-cgrjc	1/1	Running	0	8m3s
geo-99d8c58c-5wrbn	0/1	CrashLoopBackOff	6 (41s ago)	8m3s
jaeger-67445b6dd6-7kbpd	1/1	Running	0	8m1s
memcached-profile-66bb9c8b9d-86bpf	1/1	Running	0	8m
memcached-rate-68585796cb-gtpp6	1/1	Running	0	7m57s
memcached-reserve-646c7dcd6b-vwz8t	1/1	Running	0	7m52s
mongodb-geo-5ff578bcb8-bg6cr	1/1	Running	0	8m1s
mongodb-profile-758cb77f9f-thlpl	1/1	Running	0	8m
mongodb-rate-56cc8659c9-7dwkw	0/1	Pending	0	7m56s
mongodb-recommendation-55699465f7-f9hng	0/1	Pending	0	7m54s
mongodb-reservation-5994859869-zvmt	1/1	Running	0	7m50s
mongodb-user-969c6c449-86tnh	0/1	Pending	0	7m48s
profile-676fcc86d5-v7c55	0/1	Error	6 (3m16s ago)	7m59s
rate-c9bc58c85-klcq7	0/1	CrashLoopBackOff	6 (25s ago)	7m55s
recommendation-8675c6c9ff-hqb18	0/1	CrashLoopBackOff	6 (26s ago)	7m54s
reservation-7699b8fdd4-756lh	0/1	CrashLoopBackOff	5 (2m58s ago)	7m50s
search-65df96d8df-6lnw2	1/1	Running	0	7m49s
user-8477d787d8-d8d45	0/1	Error	6 (3m1s ago)	7m47s

1283

Thought & Action

Mitigation Agent

The "geo" pod is in a CrashLoopBackOff state, and several other pods are pending or experiencing errors. This suggests there may be underlying issues with the application configuration or resource constraints. I will check the logs of the "geo" pod to identify any specific errors causing the CrashLoopBackOff state.

Call get_logs with {"namespace": "test-hotel-reservation", "service": "geo"}

1284

Tool Usage: get_logs

Mitigation Agent

```
{   "level": "info",   "time": "2025-05-14T01:42:48Z",   "message": "TLS disabled." } {"level": "info", "time": "2025-05-14T01:42:48Z", "message": "Set global log level: info"} {"level": "info", "time": "2025-05-14T01:42:48Z", "message": "Tune: setGCPPercent to 100"} 2025-05-14T01:42:48Z INF cmd/geo/main.go:23 > Reading config... 2025-05-14T01:42:48Z INF cmd/geo/main.go:36 > Read database URL: mongodb-geo:27017 2025-05-14T01:42:48Z INF cmd/geo/main.go:37 > Initializing DB connection... 2025-05-14T01:43:00Z PNC cmd/geo/db.go:26 > no reachable servers panic: no reachable servers  goroutine 1 [running]: github.com/rs/zerolog.(*Logger).Panic.func1({0x96e211, 0x0}) /go/src/github.com/harlow/go-micro-services/vendor/github.com/rs/zerolog/log.go:405 +0x2d github.com/rs/zerolog.(*Event).msg({0xc00019e150, {0x96e211, 0x14}}) /go/src/github.com/harlow/go-micro-services/vendor/github.com/rs/zerolog/event.go:158 +0x2b8 github.com/rs/zerolog.(*Event).Msg(...) /go/src/github.com/harlow/go-micro-services/vendor/github.com/rs/zerolog/event.go:110 main.initializeDatabase({0xc000028288, 0xc0000fddd0}) /go/src/github.com/harlow/go-micro-services/cmd/geo/db.go:26 +0x167 main.main() /go/src/github.com/harlow/go-micro-services/cmd/geo/main.go:38 +0x877  Please take the next action
```

1285

Task Mitigation Finished

Mitigation Agent

Root Cause:
1. The "frontend" service's "recommendation" operation is experiencing connectivity issues with the Consul service, indicated by the error message "connection refused." The Consul service was restarted to resolve any transient connectivity issues.

1286

```
2. The "geo" operation in the "search" service is unable to bind the "geo-pvc" due to the persistent volume
↳ controller not recognizing the storage class "geo-storage." The missing storage classes were created to resolve
↳ this issue.
3. The "geo" pod is in a CrashLoopBackOff state due to a database connection error, specifically "no reachable
↳ servers." This indicates a problem with the MongoDB connection.

Mitigation Plan:
1. For the "frontend" service:
  - Ensure the frontend service is correctly configured to connect to the Consul service on port 8500.
  - Verify there are no network policies or firewall rules blocking traffic between the frontend and Consul
  ↳ services.
  - Monitor the Consul service to ensure it remains operational and accepting connections.

2. For the "geo" operation in the "search" service:
  - Ensure the persistent volume claims are bound correctly now that the storage classes have been created.
  - Investigate the MongoDB connection issue causing the "geo" pod to crash. Verify the MongoDB service is running
  ↳ and accessible at the expected endpoint.

3. For the pending PVCs:
  - Monitor the provisioner to ensure it completes the task of binding the remaining PVCs.
  - Check for any resource constraints or scheduling issues that may be preventing the PVCs from being bound.

Conclusion:
The primary issues have been addressed by restarting the Consul service and creating the missing storage classes.
↳ The remaining tasks involve monitoring the system to ensure the changes take effect and investigating the
↳ MongoDB connection issue for the "geo" pod.
```

1287

Note (not part of the trajectory)

Although not all the pods are ready in the last observation, this doesn't means the agent failed this attempt. The operations it performed are correct, but the system is still in the process of stabilizing. Kubernetes will attempt to restart faulty pods periodically. After a few minutes of waiting, all pods will be ready and the workloads will be able to run without issues.

1288

Phase: Validation & Retry

Run #2

```
Waiting 300.0 seconds for changes to take effect...
...
-----
117 requests in 10.00s, 48.95KB read
Requests/sec: 11.70
Transfer/sec: 4.89KB
```

1289

Phase: AIOpsLab Validation

Run #2

```
== Evaluation ==
Results:
{'success': True}
```

1290