



AI Investigation Agents for Debugging Live Incidents Using OpenSearch

Aman Kimothi, Jai Mashalkar & Ashish Gupta

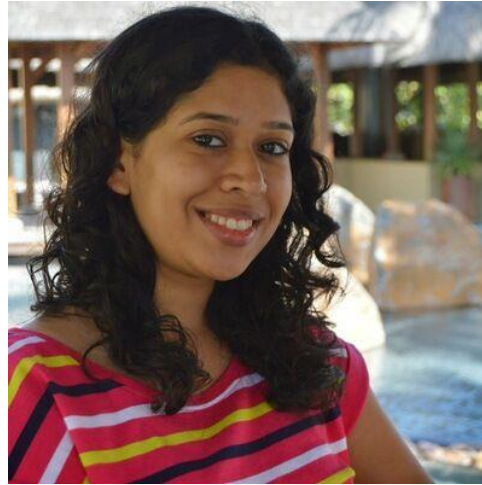
Oracle Cloud Infrastructure



OCI Search Service with OpenSearch



Aman Kimothi
Senior Software Engineer
[LinkedIn](#)



Jai Mashalkar
Senior Software Engineer
[LinkedIn](#)

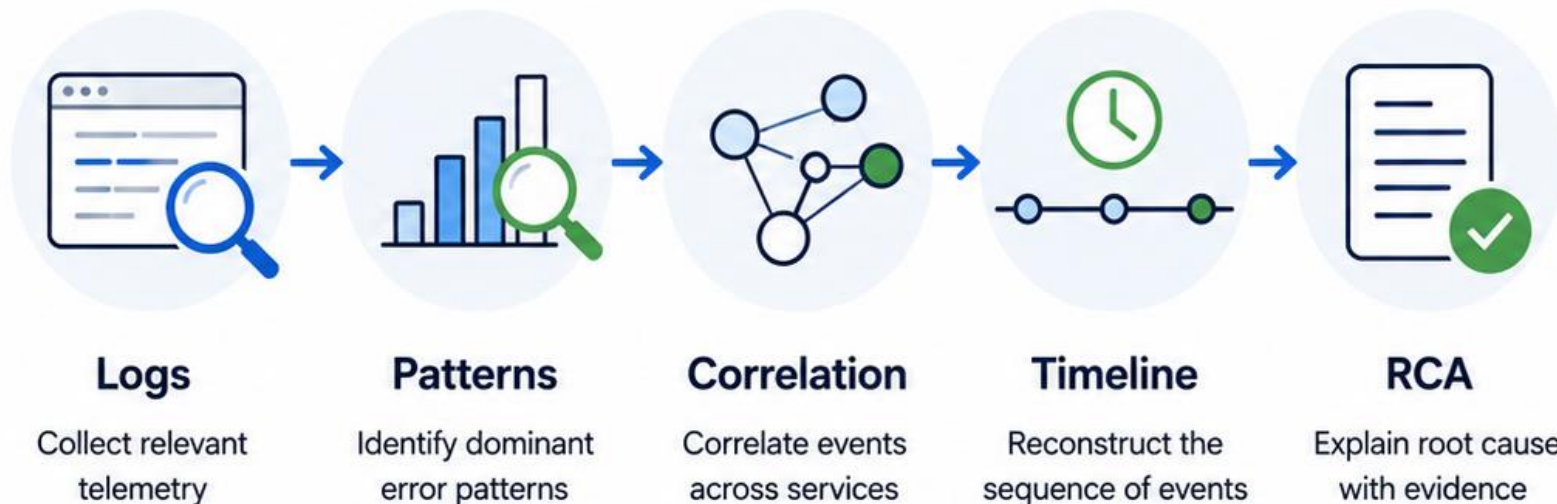


Ashish Gupta
Senior Engineering Manager
[LinkedIn](#)



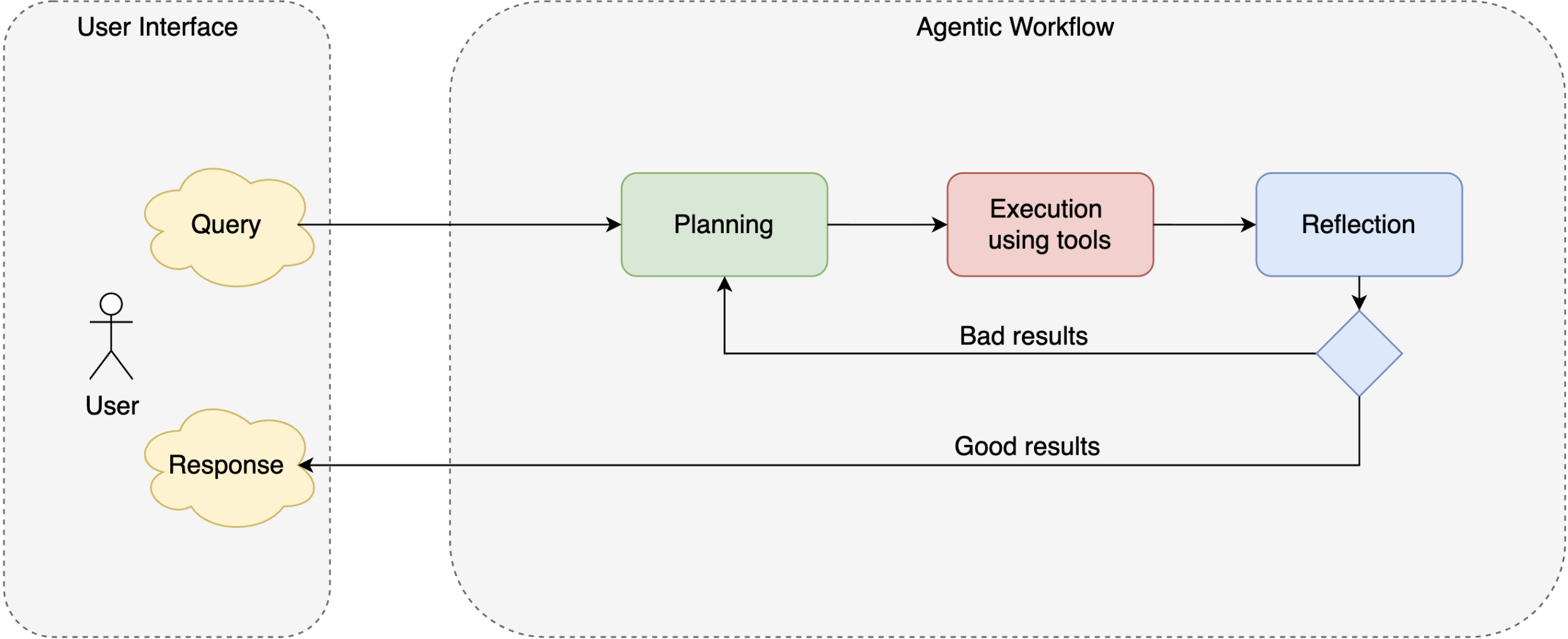
Problem Statement

Debugging production incidents in distributed systems is hard.

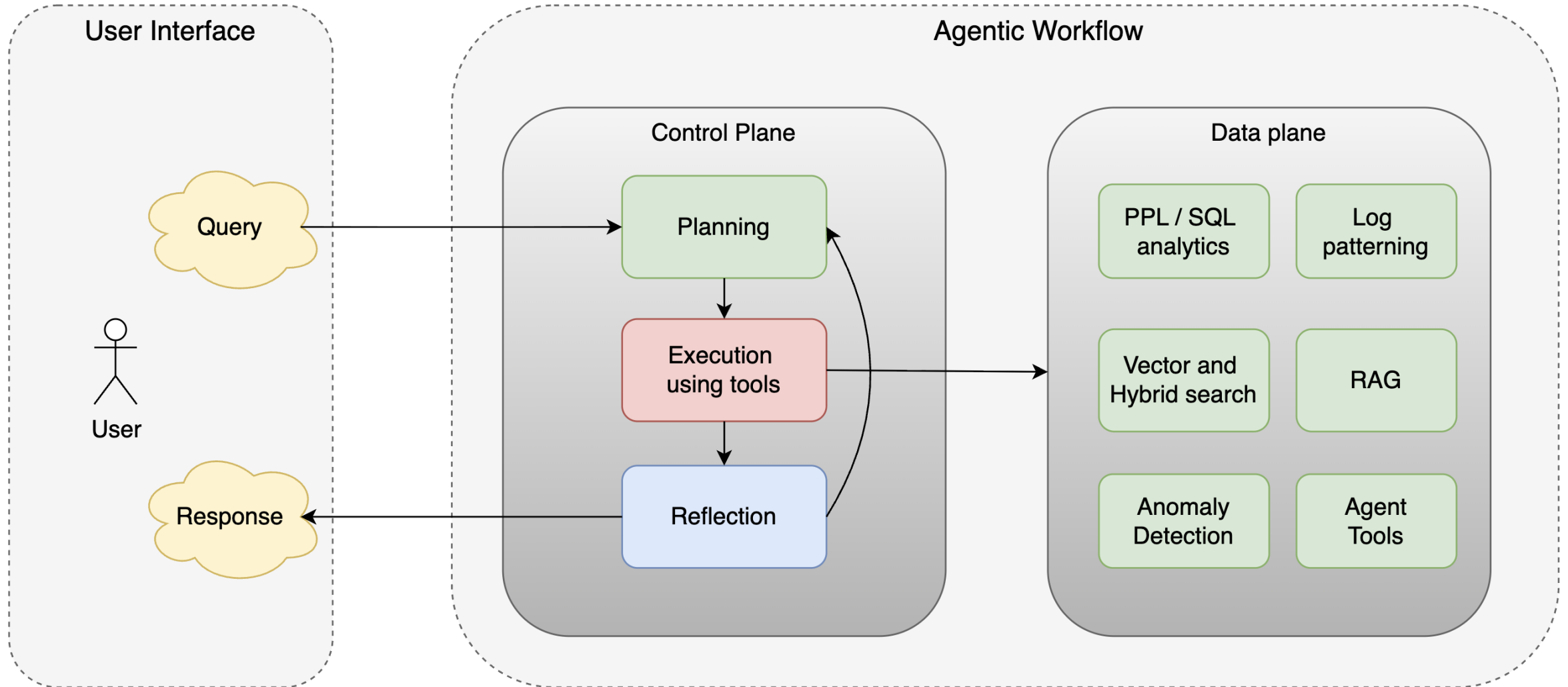


From noise to **understanding**. From searching to **solving**.

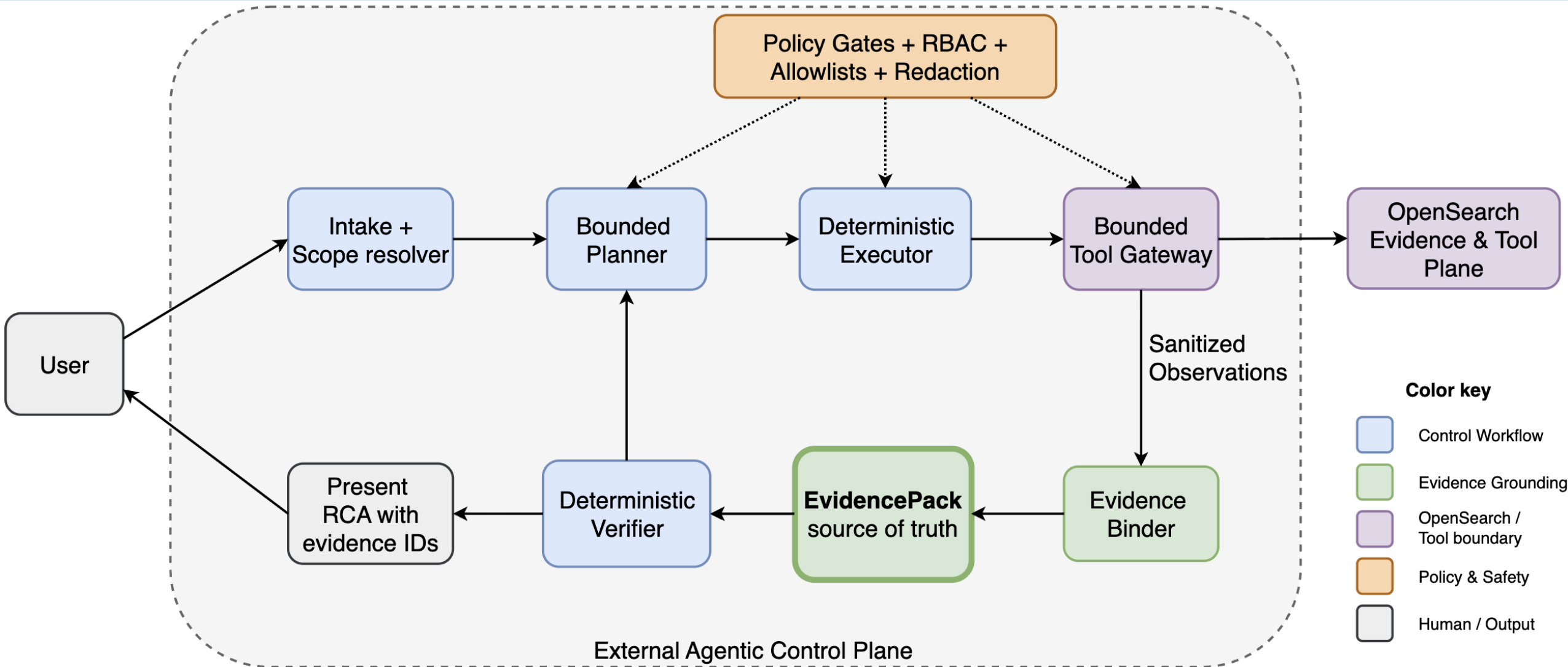
Agentic Workflow



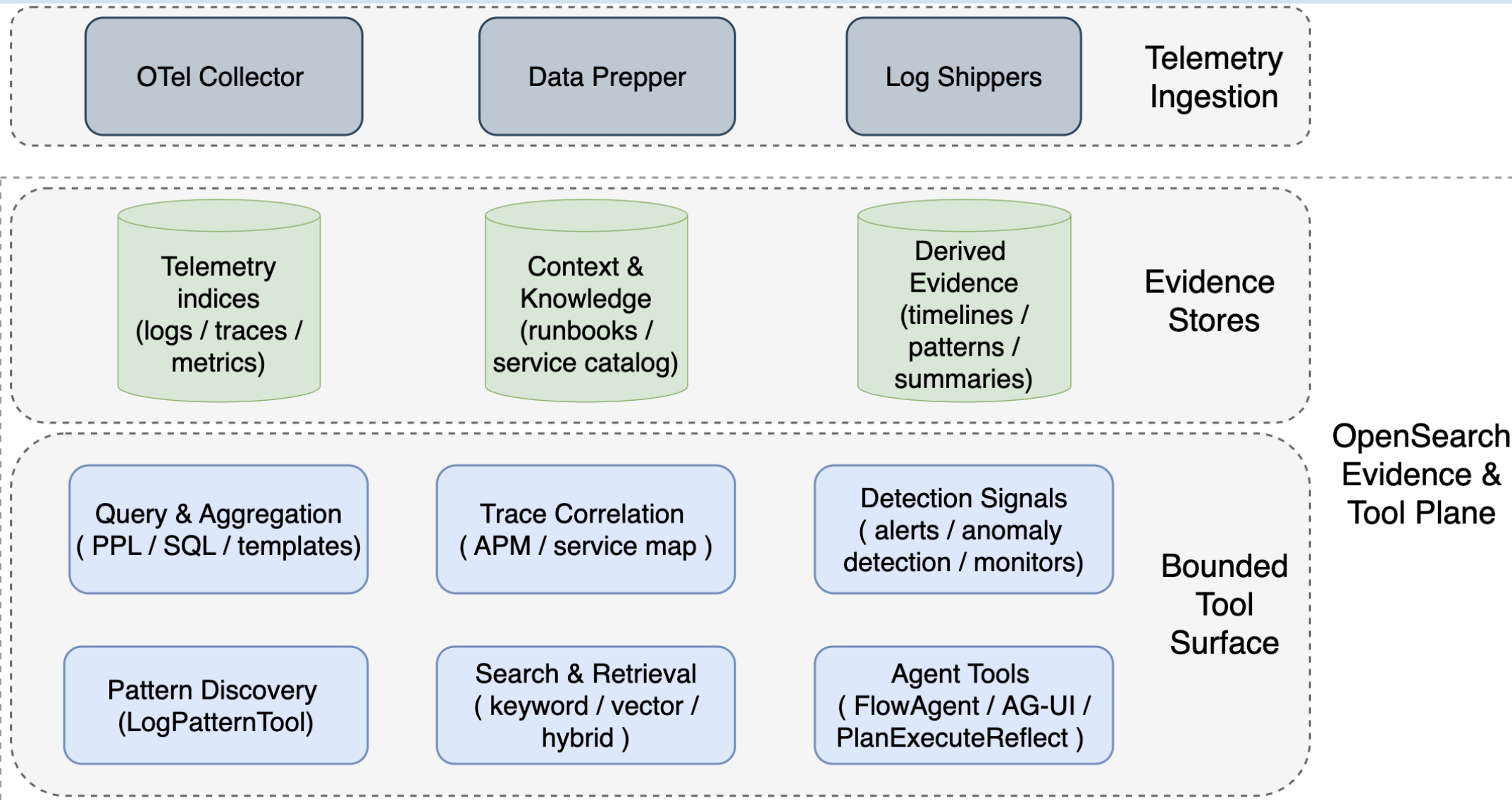
Investigation agent using OpenSearch



Implementation: External Agentic Control Plane



OpenSearch Evidence & Tool Plane



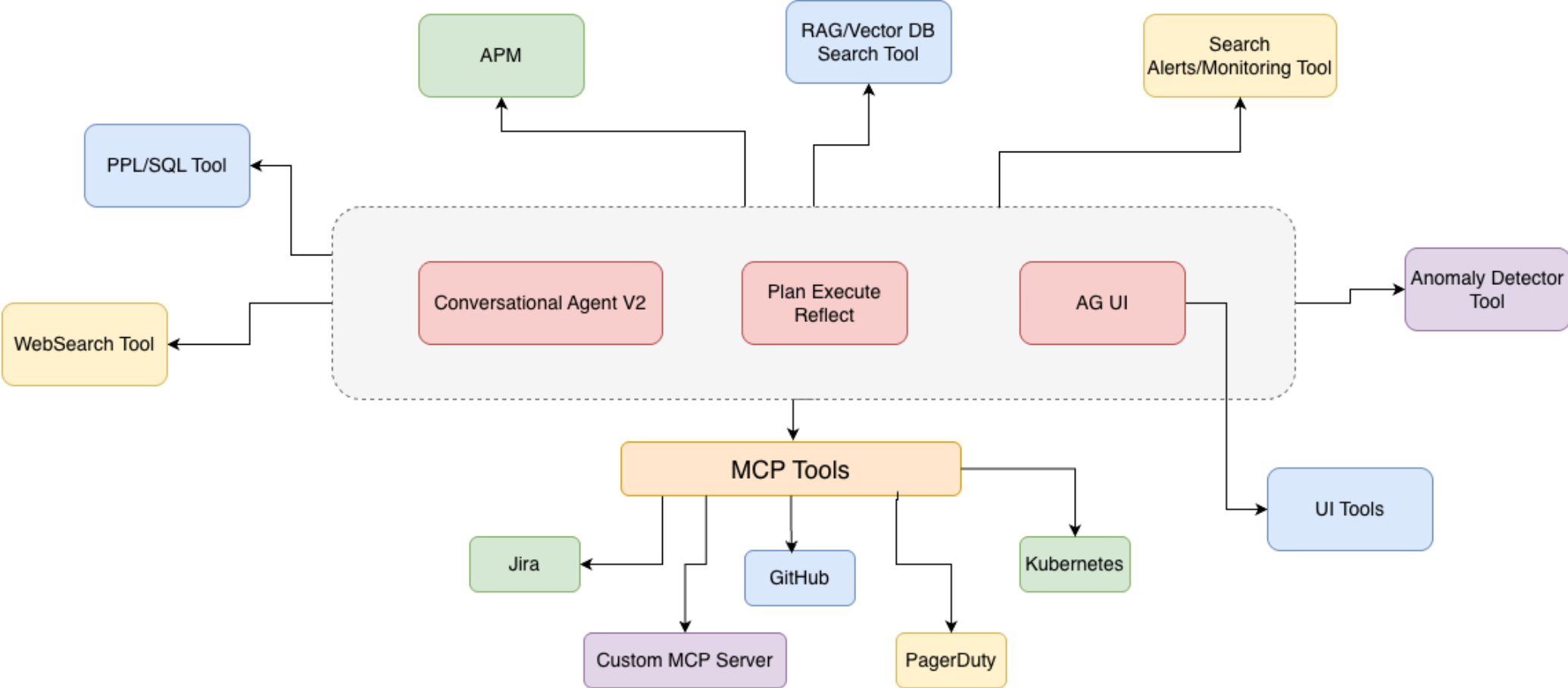
- Use strictly **bounded** time windows and service scoping
- Use typed **search templates** rather than arbitrary planner-generated DSL/PPL queries
- Use **PIT** (Point in time) for multi-step consistency
- Keep **raw telemetry** separate from derived investigation artefacts.
- Use query **quotas**, **retries**, and **circuit breakers** at the orchestration layer



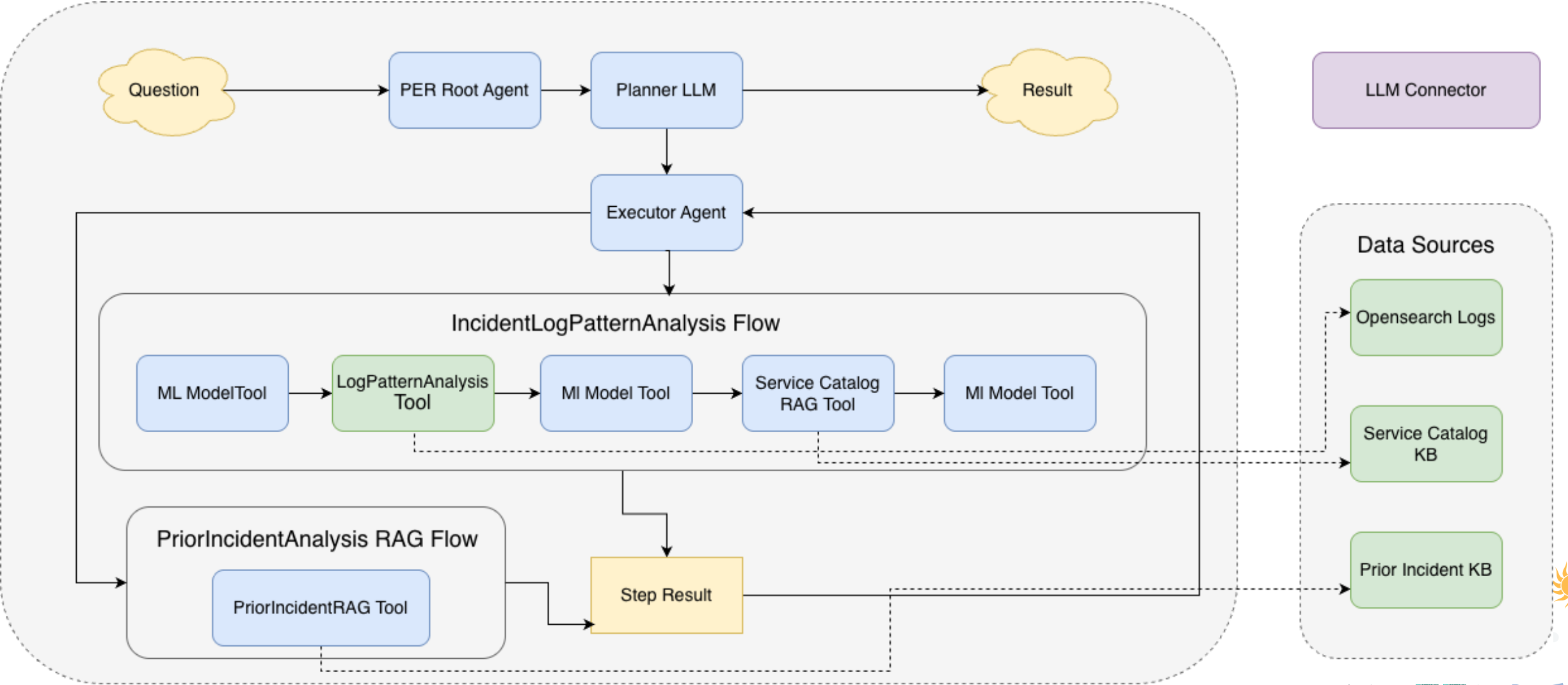
Demo for external agentic orchestration with OpenSearch

The screenshot displays a code editor interface with a file explorer on the left, a central diagram, and a terminal at the bottom. The file explorer shows a project structure with folders like 'recordings', 'scenarios', 'screenshots', 'scripts', 'docs', 'evals', 'infra', 'services', 'src', and 'incident_agent'. The central diagram, titled 'microservice.png', illustrates a system architecture. It features a 'frontend' box on the left that connects to two boxes: 'recommendation-service' (top) and 'checkout-service' (bottom). The 'checkout-service' box further connects to three boxes: 'inventory-service', 'shipping-service', and 'payment-service'. The 'payment-service' box connects to a 'payment-db' box at the bottom. The terminal window shows the command prompt in a virtual environment: `(.venv) ~/scm-workspace/opensearchcon/ai-incident-investigation-agents git:[phase-6]`. The bottom status bar indicates the current file path: `opensearchcon > ai-incident-investigation-agents > demo > outputs > runs > 20260612-101115`, along with system information like 'up-to-date', '1789 of 4500M', and 'OCA'.

Agentic Tooling in OpenSearch



Agentic Flow using Plan Execute Reflect Agent



- **Extensible Plug and Play Architecture:** A baseline framework that can be extended through plug-and-play knowledge sources for service dependencies, ownership, error patterns, and log locations, reducing the complexity of the prompts.
- **Continuous Operational Learning:** Operators can refine the system over time by curating resolved incidents, runbooks, and lessons learned into the incident and service knowledge bases.
- **Hybrid Agent Orchestration:** PER agent handles adaptive, evidence-driven investigation, while flow agents execute deterministic analysis paths such as structured log search.



Demo for Plan Execute Reflect Agent

- COLLECTIONS
- POST execute log pattern analysis Copy 4
- POST execute log pattern analysis Copy 4
- POST execute log pattern analysis 3 services C...
- POST execute log pattern analysis 3 services C...
- POST execute log pattern analysis Copy 5
- POST execute log pattern analysis Copy 5
- POST execute rag
- GET get agent Copy 2
- Plan Execute Reflect Agent
 - POST root agent
 - POST executor agent
 - POST flow agent - IncidentLogAnalysisFlow
 - POST execute
 - GET Get Task
 - GET get memory
 - GET get memory traces
 - POST register-model Copy
 - PUT Add doc
 - PUT Add doc Copy
 - PUT Add doc Copy 2
 - PUT Search
 - PUT set skip unavailable
 - GET Role get all access mapping
 - PUT Role Copy
 - PUT Role Copy
 - PUT user
 - GET remote info
 - PUT Add doc Copy 4
 - GET Analyze API
 - GET Clear cache API
- local testina
- ENVIRONMENTS
- SPECS
- FLOWS

POST {{host}}/_plugins/_ml/agents/_register

Overview Params Authorization Headers 9 Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "incident-investigator-per-db-throttle-delegated-flow",
3   "type": "plan_execute_and_reflect",
4   "description": "Plans a three-hop incident investigation and delegates execution to the fixed log analysis flow",
5   "llm": {
6     "model_id": "Z0OotZ48uGImM3cG3rHz",
7     "parameters": {
8       "prompt": "${parameters.question}"
9     }
10  },
11  "memory": {
12    "type": "conversation_index"
13  },
14  "parameters": {
15    "llm_response_filter": "$.chatResponse.text",
16    "executor_agent_id": "_4MUp4BuGImM3cGuLFI",
17    "executor_max_iterations": "6",
18    "max_steps": "6",
19    "executor_system_prompt": "You are an OpenSearch ReAct executor. Return only one valid JSON object.\n\nTools:\n- IncidentLogAnalysisFlow\n- PriorIncidentRAGFlow\n\nChoose exactly one tool per invocation:\n- If the step contains 'Use PriorIncidentRAGFlow', 'previous incidents', 'remediation', 'resolution', 'validation checks', 'future improvements', or 'ROUTING_DECISION: stop', call PriorIncidentRAGFlow.\n- Otherwise, if the step contains 'Use IncidentLogAnalysisFlow', 'analyze logs', 'investigate', 'NEXT_HOP_SERVICE', or 'root cause', call IncidentLogAnalysisFlow.\n\nPriorIncidentRAGFlow call:\n{thought: 'Search previous incidents for remediation guidance.', action: 'PriorIncidentRAGFlow', action_input: '{question}'}\n\nIncidentLogAnalysisFlow call:\n{thought: 'Run incident log analysis.', action: 'IncidentLogAnalysisFlow', action_input: '{question}'}\n\nAfter tool output, call no more tools. Return:\n{thought: 'I have the tool output.', final_answer: '<tool output summary preserving markers>', response: '<same final answer>'}\n\nPreserve markers if present: ANALYZED_SERVICE, NEXT_HOP_SERVICE, FINAL_ROOT_CAUSE_SERVICE, ROUTING_DECISION, PRIOR_INCIDENT_RAG, RAG_ROUTING_DECISION.\n\nDo not ask for logs or index. Do not return steps/result.",
20    "planner_prompt": "Services: checkout-service -> payment-service -> payment-db. payment-db is terminal.\n\nCreate exactly one initial step.\n\nStart service selection:\n- Explicit payment-db/database/throttling/overload request: start payment-db.\n- Explicit payment-service request: start payment-service.\n- Otherwise start checkout-service.\n\nReturn JSON only: {steps: [ 'Use IncidentLogAnalysisFlow to analyze <service> logs for error/warning patterns and decide if failure is local or caused by the next internal dependency.', 'result: {} ]}\n\nUse only service names: checkout-service, payment-service, payment-db.",
21    "reflect_prompt": "Review latest executor result and return JSON only with top-level steps and result.\n\nRules:\n- If result has ROUTING_DECISION: continue and NEXT_HOP_SERVICE is not none, return one step: 'Use IncidentLogAnalysisFlow to analyze <NEXT_HOP_SERVICE> logs for the same incident window and decide if failure is local or caused by the next internal dependency.'\n- Set result to '{}'.\n- If result has ROUTING_DECISION: stop and does not have PRIOR_INCIDENT_RAG: done, return one step: 'Use PriorIncidentRAGFlow to search previous incidents and recommend immediate resolution steps, validation checks, and future improvements. Current incident analysis: <latest result>'. Set result to '{}'.\n- If result has PRIOR_INCIDENT_RAG: done, return {steps: [], result: '<final answer combining root cause and prior-incident recommendations>'}.\n\nNever repeat an already analyzed service. Never call PriorIncidentRAGFlow before ROUTING_DECISION: stop."
22  }
23 }
```

Response History

Send + Get a successful response

Send + Visualize response

Dimension	OpenSearch Agentic Tools	External Agentic Tools
Scope of solution	Best for focused, end-to-end investigation inside OpenSearch using native tools, logs, alerts, search, and ML Commons agents.	Better for broader workflows spanning many enterprise systems, custom APIs, ticketing, deployment systems, and external state.
Scale	Suitable for smaller flows with limited parallel execution and checkpointing needs.	Better manageability for large-scale, parallel, long-running, and resumable workflows.
Guardrails	Guardrails depend on tool permissions, prompts, connector controls, and OpenSearch security boundaries.	Stronger control over action approval, policy enforcement, retries, checkpoints, and audit gates.
Complexity	Planner is prompt-driven and simple to set up, making it fast for POCs and bounded investigation flows.	Supports elaborate rules, explicit workflow logic, state machines, and custom orchestration patterns.



Questions & Answers



OpenSearchCon

INDIA

