



STRUCTURE OVER PROMPTS

Building SRE Agents on Gemini Enterprise Agent Platform

Olga Mirensky

Senior Staff SRE, Ping Identity

DevFest – Sydney – 10 Oct 2026



AGENDA

Introduction – GEAP and Ping Identity

What is Gemini Enterprise Agent Platform. What is Ping Identity. What challenges we are working with and how Agentic SRE Platform helps.

Google AI SRE (Site Reliability Engineering)

Google's blueprint for Agentic SRE

ADK and Agent Runtime

Technical showcase of key ADK capabilities and optimisations. How agent runtime implements these constructs

Demo

Sample e2e-agent working on an OSS repo, analysing test failures, storing them in BigQuery enabling environment-wide intelligence. From agent Playground to wider GEAP ecosystem.

Introducing Gemini Enterprise Agent Platform, powering the next wave of agents

April 23, 2026

What's new: Today, we're launching Gemini Enterprise Agent Platform — our new, comprehensive platform to build, scale, govern, and optimize agents. It's the evolution of Vertex AI, bringing the model selection, model building, and agent building capabilities that customers love, together with new features for agent integration, DevOps, orchestration, and security.

✦ Gemini Enterprise Agent Platform

Build

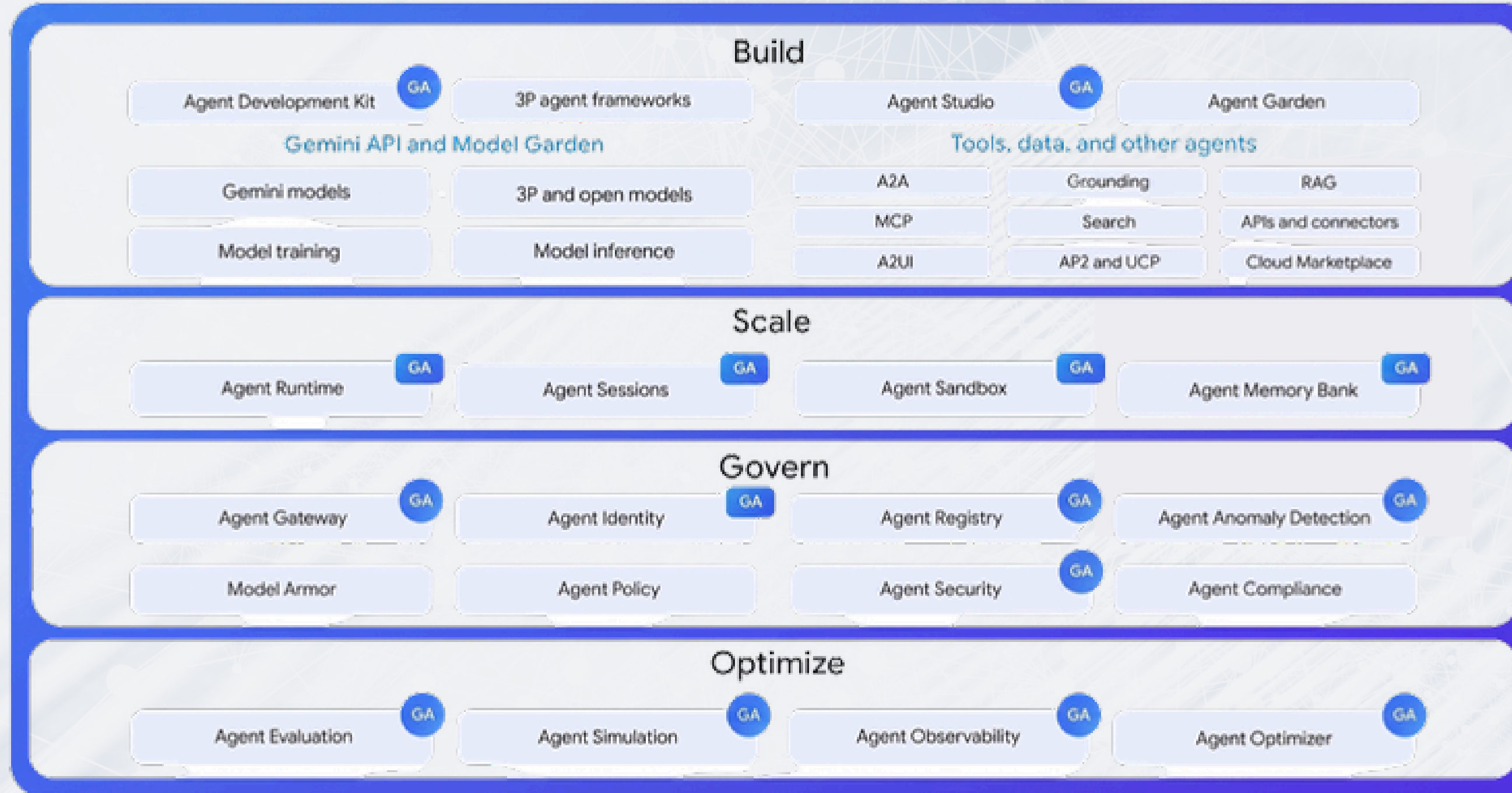
Scale

Govern

Optimize

<https://cloud.google.com/blog/products/ai-machine-learning/introducing-gemini-enterprise-agent-platform>

◆ Gemini Enterprise Agent Platform



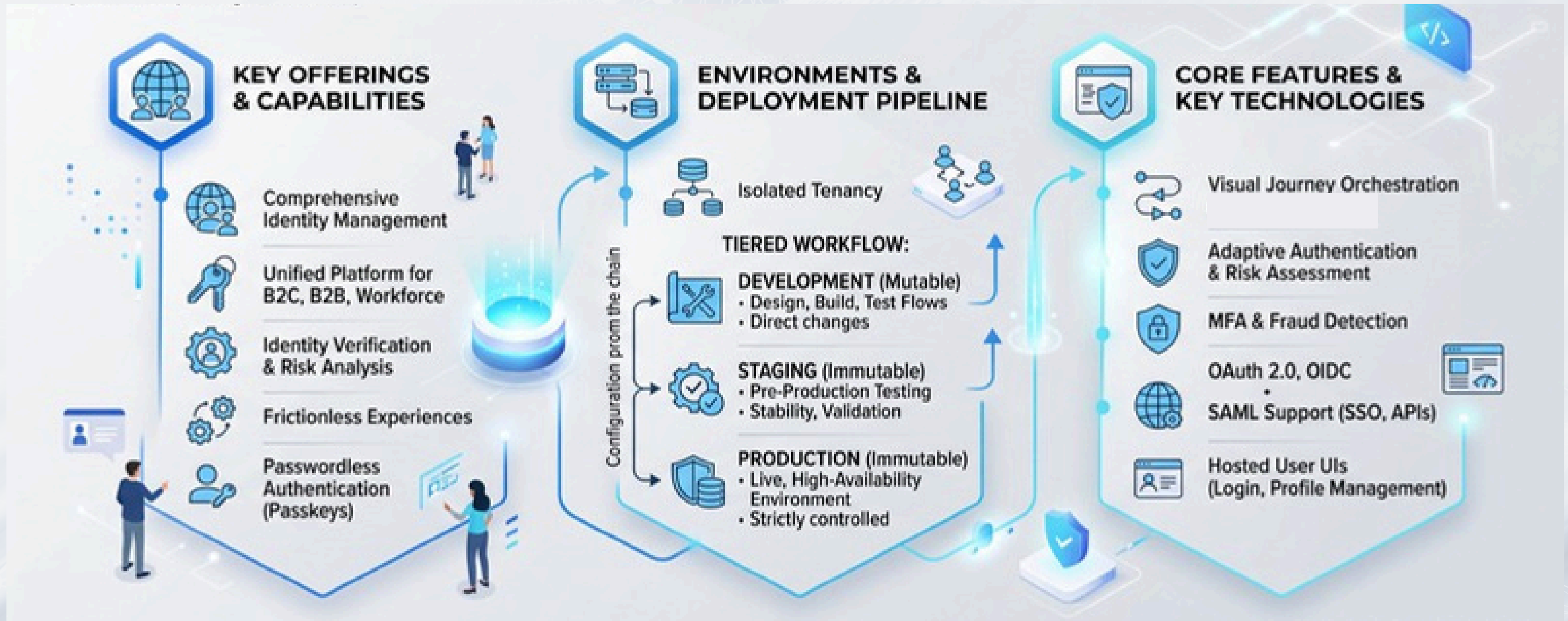
GA GA at announcement
Apr, 2026

GA GA as of Sept 2026

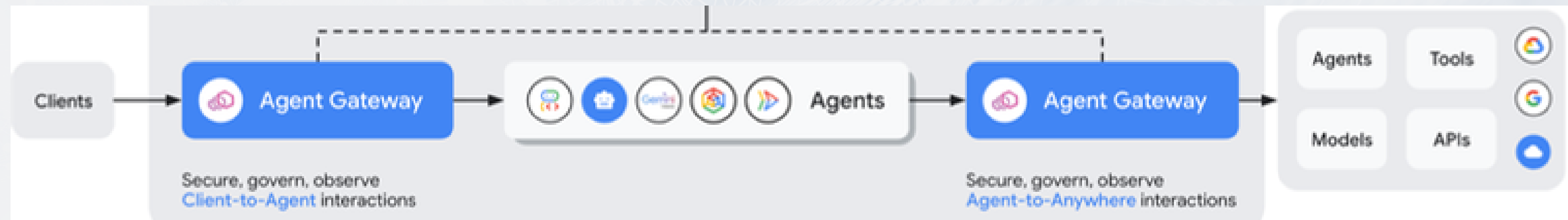
<https://cloud.google.com/blog/products/ai-machine-learning/introducing-gemini-enterprise-agent-platform>

<https://cloud.google.com/blog/products/ai-machine-learning/whats-new-in-gemini-enterprise-agent-platform>

PingOne Advanced Identity Cloud

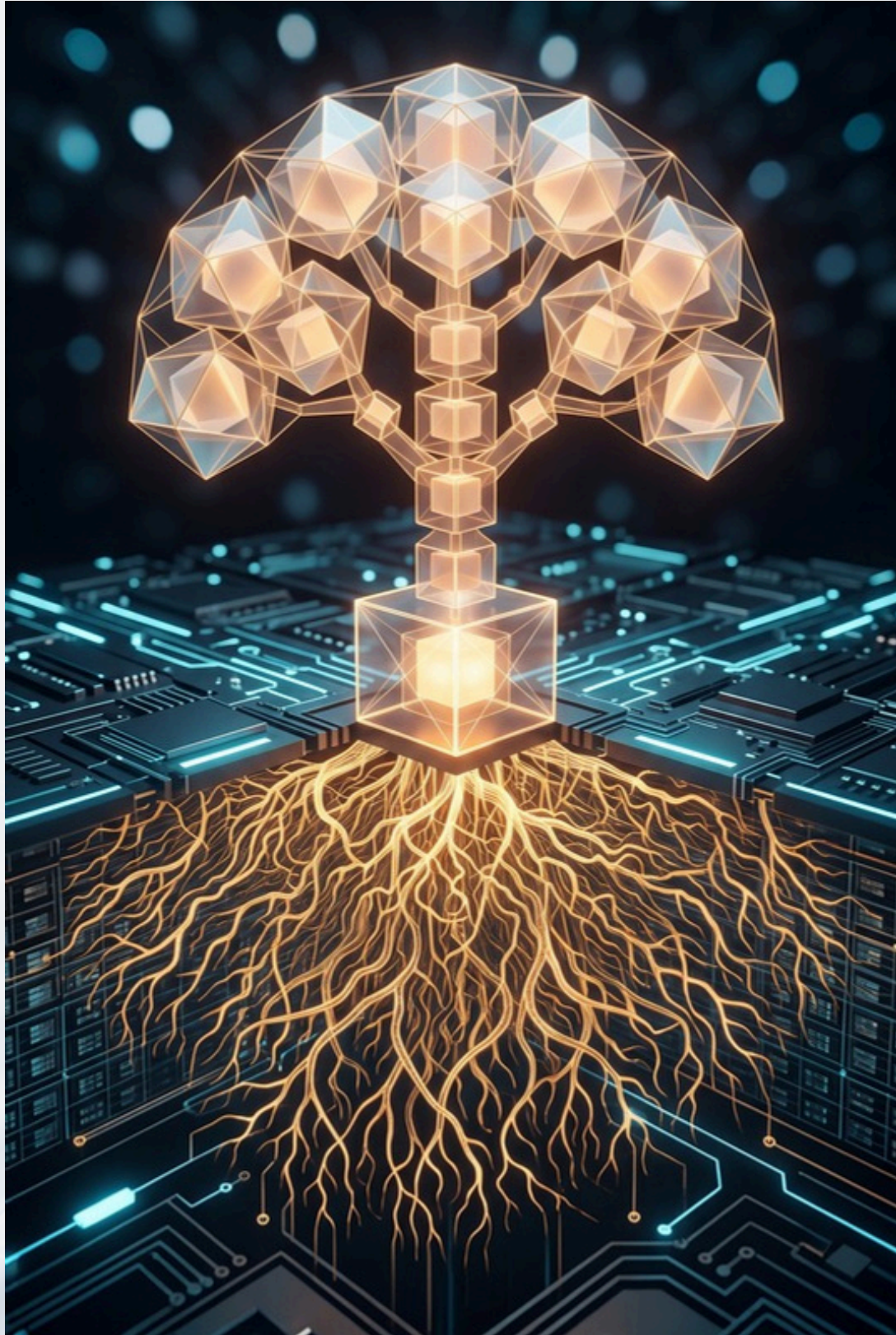


Ping Identity – Agent Gateway Partner



Ping Identity integrates with Agent Gateway to deliver real-time, fine-grained authorization for agent and tool traffic. This integration ensures every request is continuously verified based on user, agent, context, and policy—moving beyond static credentials. Together, they provide centralized, consistent governance and visibility across all agent interactions without changing application code.

<https://docs.cloud.google.com/gemini-enterprise-agent-platform/govern/gateways/agent-gateway-overview>



AGENTIC SRE

SRE – Site Reliability Engineering

“ *Hope is not a strategy.* ”

Traditional SRE saying

<https://sre.google/sre-book/introduction/>

an SRE team is responsible for the availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning of the service they support.

Google AI SRE

SRE AI Autonomy Levels



	Monitor	Investigate	Mitigate	Actuate	Self-Direct
L0 – Manual	Automation	Human	Human	Human	Human
L1 – Assisted	Automation	Automation	Human	Human	Human
L2 – Partial	Automation	Automation	Human	Automation	Human
L3 – High	Automation	Automation	Automation	Automation	Human
L4 – Full	Automation	Automation	Automation	Automation	Automation

Google AI SRE whitepaper

The Safety Trifecta Google

Transparent AI reasoning - - - real-time risk assessments - - - progressive authorization

Strict architectural guardrails

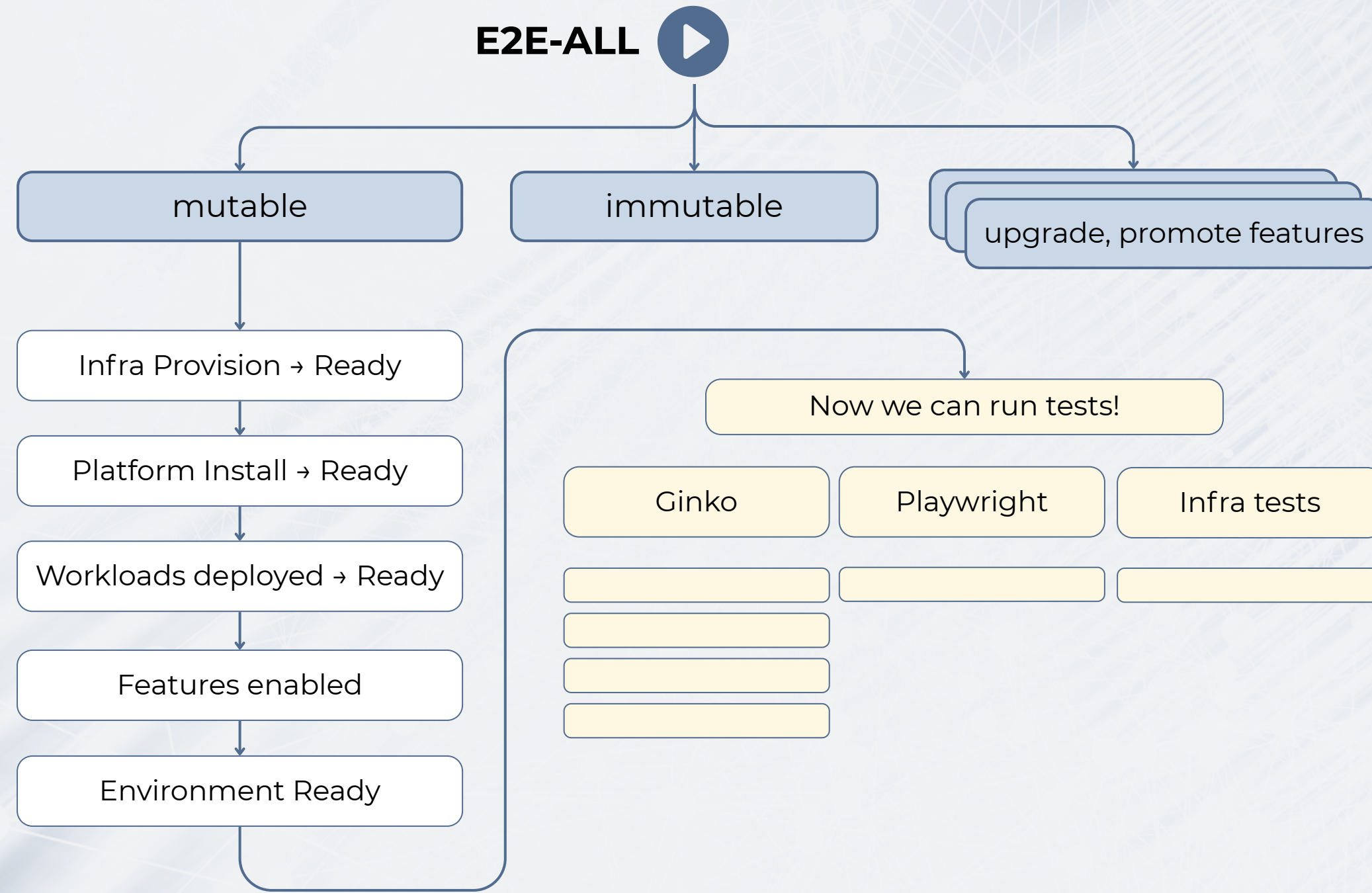
AI agents must operate under rigid technical constraints including least-privilege access, automated circuit breakers, zero-trust actuation tools.

Human Trajectories

Generating a human trajectory involves understanding and tracking the sequential actions and decisions made by engineers during an incident or a task. Creating and maintaining “Golden Trajectories” to feed evaluation pipelines.



First SRE Agent: e2e-agent



Multiple stages across multiple tests can fail

Wide range of error types – from infra to UI

Change rate is different for PR or release

Finding the middle ground

Deep analysis, correct for almost all failures

Analyse any e2e failure with high degree of depth and correctness

Challenging agent structure, token management and identifying rightfully deep sessions from chasing-the-tail.

Unpredictable execution time makes it harder for users to know when to expect the result.

Not required in every analysis.

Strictly bound time and token execution

User-facing focus:

Provide reliable, sufficiently correct analysis for most e2e failures, grounded in evidence and source code, bounded by explicit time and token budgets and ready to guide following processes to final investigation and action.

Trust earned through consistency

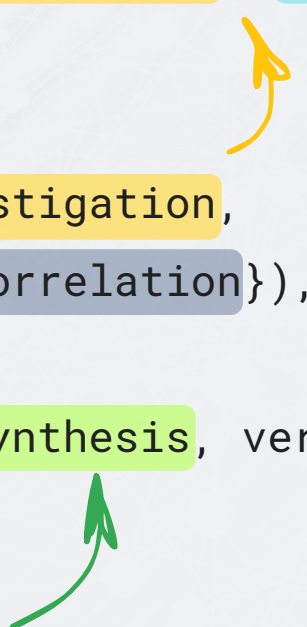


Agent Development Kit

Control flow

Graph workflow

```
root_agent = Workflow(  
    name="e2e-agent",  
    edges=[  
        ("START", intake, investigation, investigation_router),  
        (investigation_router,  
         {"continue": investigation,  
          "done": change_correlation}),  
        (change_correlation, synthesis, verifier, verifier_router),  
        (verifier_router,  
         {"repair": synthesis,  
          "done": done}),  
    ]  
)
```



```
investigator_agent = Agent(  
    name="investigator",  
    instruction=[...]  
    tools=[...],  
    before/after_agent_callback,  
    before/after_model_callback,  
    before/after_tool_callback,  
)
```

Dynamic workflows / nodes

Runner loop

github: [google/adk-recipes/core](https://github.com/google/adk-recipes/core)

Verifier

Rubber-stamp and a quality win

Overreach

log-only mechanism
claims; correlation !=
causation

Broken references

dangling references
and gap IDs; gaps
contradicting findings

Unverified claims

mechanisms with no
source code read

Report shape

no findings,
placeholder summary,
malformed verdict

+ a few
more.

Synthesis: “Engine apply failure `serverside-applied ... signal: killed` the engine’s own deadline killing the apply... `client rate-limiter... context deadline exceeded` - indicating API server saturation”

f_82bd4sd

certainty: proven

ev_28f7aa3

kind: log

Verifier

Rubber-stamp and a quality win

Overreach

Broken references

Unverified claims

Report shape

+more

Synthesis: “Engine apply failure ``serverside-applied ... signal: killed`` the engine’s own deadline killing the apply.... ``client rate-limiter... context deadline exceeded`` - indicating API server saturation”



``signal: killed`` is external to the process -> increased timeout does not help

Recommendation: increase timeout

f_82bd4sd

certainty: proven

ev_28f7aa3

kind: log

client rate limiter - mechanism to prevent API server overload, not a symptom that the server is actually overloaded.

Verifier

Rubber-stamp and a quality win

Overreach

Broken references

Unverified claims

Report shape

+more

Synthesis: “Engine apply failure `serverside-applied ... signal: killed` the engine’s own deadline killing the apply... `client rate-limiter... context deadline exceeded` - indicating API server saturation”

Recommendation: increase timeout

f_82bd4sd

certainty: proven

ev_28f7aa3

kind: log

Verifier: “the logs show the symptoms; the causal mechanism is inferred - no evidence of kind code to back it”

“no open gap recorded to confirm this claim.”

Synthesis:

Reworded analysis

Changed recommendation

f_82bd4sd

certainty: probable

User / integrations

ADK: HITL Human-in-the-loop

Nested

```
hitl_workflow = Workflow(  
    name="sre_investigation"  
    edges=[  
        ("START", inner_workflow, dormancy_gate)  
        (dormancy_gate,  
         {"ask_deep_dive": targeted_deep_dive,  
          "close": session_complete }),....]
```

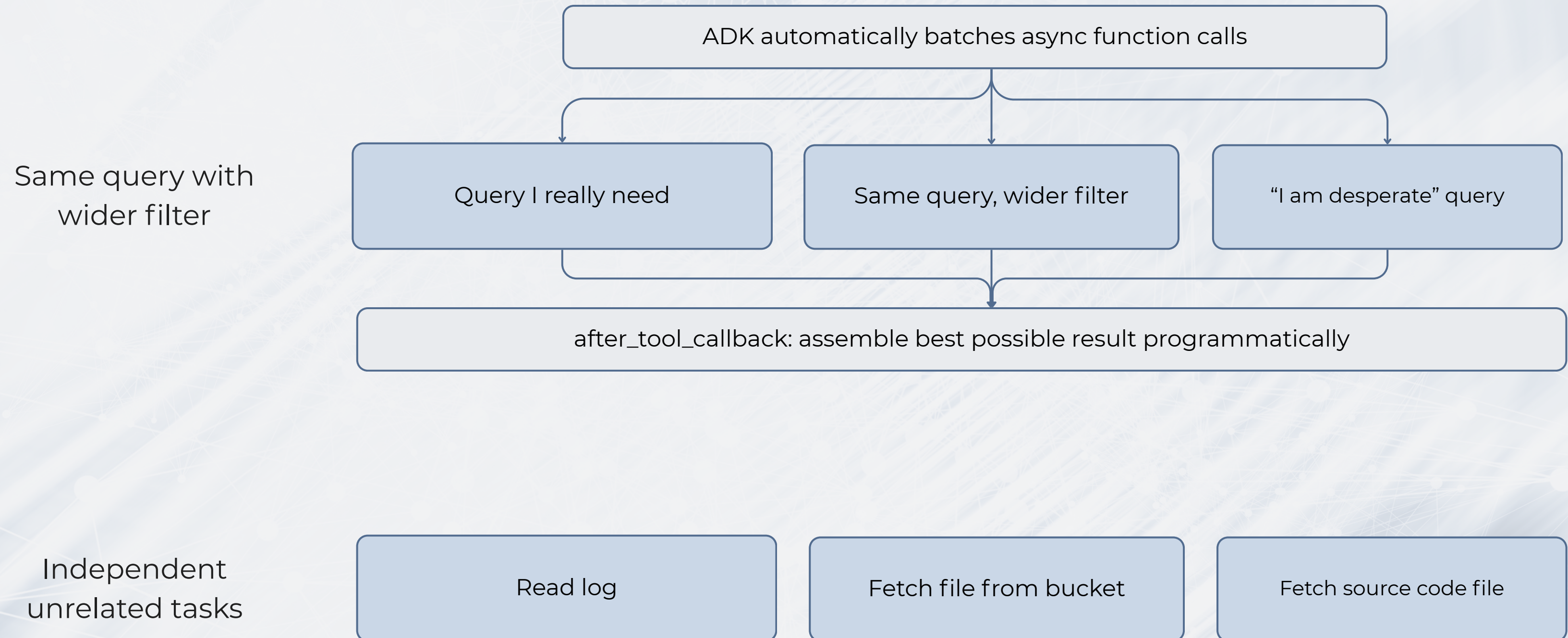
OR

Linear

```
hitl_workflow = Workflow(  
    name="sre_investigation",  
    edges=[  
        ("START", ..., router),  
        (router, {  
            "await_human": dormancy_gate,  
            "done": final_output,  
        }),  
        (dormancy_gate, {"invalid_feedback": dormancy_gate,  
                          "ask_deep_dive": targeted_deep_dive}),  
        (... , done)]
```

```
async def dormancy_gate(ctx: Context, node_input: dict):  
    yield RequestInput(  
        interrupt_id=interrupt_id,  
        message="Text for the user",  
        payload=node_input,  
        response_schema=DeepDiveFeedback  
    )
```


Parallel tool calls



Agent execution example

Agent Runtime Playground - ADK web view. Shows session events, tools execution, agent state and a lot more

Portal for agent developers, not end-user UI!

```
thinking_planner = BuiltInPlanner(  
    thinking_config=genai_types.ThinkingConfig(  
        include_thoughts=True,  
        thinking_budget=1024),  
)
```

```
investigator_agent = Agent(  
    name="investigator",  
    model=...,  
    instruction=...,  
    tools=[...],  
    output_schema=...,  
    planner=thinking_planner,  
    ...  
)
```



```
id: "4748638785896448888"
invocationId:
  "e-778b771f-248e-43ed-aa5f-f757d9588a19"
author: "investigator"
context:
  role: "user"
  parts: [
    0:
      functionResponse:
        name: "read_log_window"
        response: ...
    1:
      functionResponse:
        name: "changed_symbols"
        response:
          path:
            "cd/ateapi/internal/controlapi/functionaltest/actor_test.go"
          sha:
            "88f8259a1293f3a39a18559f4db81b11aab778e7"
          status: "modified"
          generated: false
          changed_symbols: [
            0:
              kind: "func"
              name: "buildDeploymentApplyConfig"
              start_line: 78
              end_line: 201
              changed_lines: [ ... ]
            changed_lines_outside_symbols: [
            ]
          ]
    2:
      functionResponse: ...
    3:
      functionResponse: ...
    4:
      functionResponse: ...
    5:
      functionResponse: ...
  ]
status: "Completed"
```

#14



My Diagnostic Plan

My immediate thought is to zero in on the problematic area. I need to examine the log window specifically around line 2839 to see what was happening investigate the symbols that have been modified across relevant files and then drill down into the details of the failing test case within `cd/ateapi/internal/controlapi/functionaltest/actor_test.go`.

To achieve this, my plan is to execute the following actions:

- 1. I'll start by reading the log window, specifically requesting the lines from 2830 to 2860 for job ID 99120183352. This will give me the immediate context.
- 2. Next, I'll look at the changed symbols within `cd/atecontroller/internal/controllers/workerpool_apply.go`. This file seems like a prime candidate.
- 3. Following that, I'll do the same for the corresponding test file, `cd/atecontroller/internal/controllers/workerpool_apply_test.go`, to understand catching a regression.
- 4. I'll also broaden my scope to check changed symbols in `cd/ateos-gvisor/main.go`, as changes here could have downstream effects.
- 5. Similarly, I'll examine `cd/ateos-microvm/main.go` for any recent modifications that might be contributing to the issue.
- 6. Finally, to get a complete picture of the failing test, I'll generate an outline of the `cd/ateapi/internal/controlapi/functionaltest/actor_test.go` `88f8259a1293f3a39a18559f4db81b11aab778e7`, to ensure I'm looking at the exact version that failed.

outline_file

#15



- ✓ read_log_window
- ✓ changed_symbols
- ✓ changed_symbols
- ✓ changed_symbols
- ✓ changed_symbols
- ✓ outline_file

#16



Understanding the Behavior of TestResumeActor_RelocatesAfter

My objective is to thoroughly understand the implications of the achieve this, I need to examine the specific test case itself and a worker being suspended and then resumed from a paused state.

First, I'll dive into the test function `TestResumeActor_RelocatesAfter` being validated.

Following that, I need to see how the recent changes, presumably `cd/atecontroller/internal/controllers/workerpool_apply` transitions that might involve relocation.

Then, I'll investigate the `ateos-gvisor` and `ateos-microvm` components looking at the relevant code snippets in `cd/ateos-gvisor/main` immediate handlers or responders to the events orchestrated by

By examining these specific sections, I aim to connect the test's

FUNCTION RESPONSE

```
path: "cd/ateapi/internal/controlapi/functionaltest/actor_test.go"
sha: "88f8259a1293f3a39a18559f4db81b11aab778e7"
total_lines: 3251
rune: "parsed"
generated: false
symbol_count: 82
outline:
  "cd/ateapi/internal/controlapi/functionaltest/actor_test.go @ 88f8259a1293f3a39a18559f4db81b11aab778e7 (parsed)"
imports imports L17-45
test TestCreateActor_SuccessIt *testing.T) L52-181
test TestCreateActor_WithExternalVolumesIt *testing.T) L183-185
test TestCreateActor_TemplateNotFoundIt *testing.T) L168-181
test TestCreateActor_RejectsTemplateMatchExpressionsIt *testing.T) L183-185
test TestCreateActor_SubstrateTemplateRefIt *testing.T) L231-237
test TestCreateActor_DuplicateIt *testing.T) L276-298
test TestCreateActor_StampsFullSpanIdentityIt *testing.T) L392-393
test TestCreateActor_RejectsDifferentTemplateForDataSnapshotIt *testing.T) L392-393
test TestCreateActor_RejectsDifferentTemplateForDataSnapshotIt *testing.T) L392-393
```


AI is at the core of Agentic platform but it is

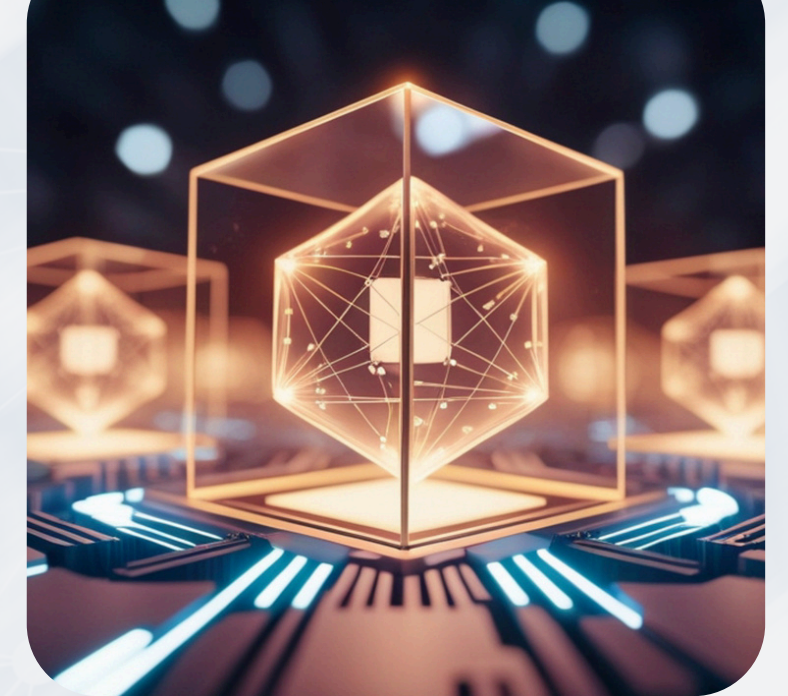
PRODUCT VISION & ARCHITECTURE

that define success



Product

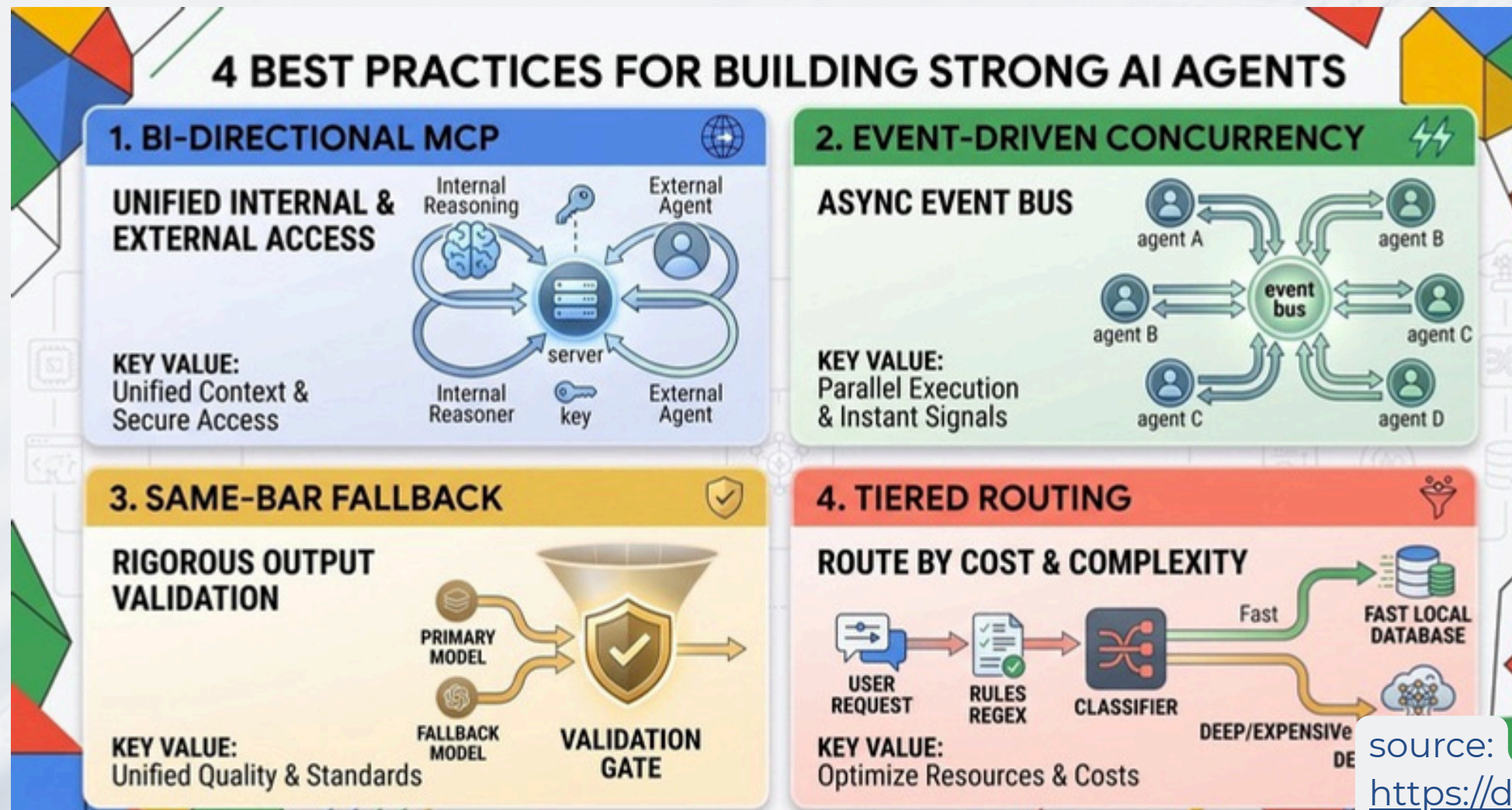
Who are we building for?
How do we measure?



	Users of e2e	Environment Owners
What do they care about	My test run; Recommendations	Entire environment
Analysis depth	Just tell me if it is in my PR if so, where?	New issues, tracking → depth and accuracy is crucial
Coverage	single test run scope	All failures, attributed by release; defects lineage

Architecture

“Across all these four patterns none of them truly require bigger teams or newer models. They represent sound engineering practices that are frequently overlooked.”

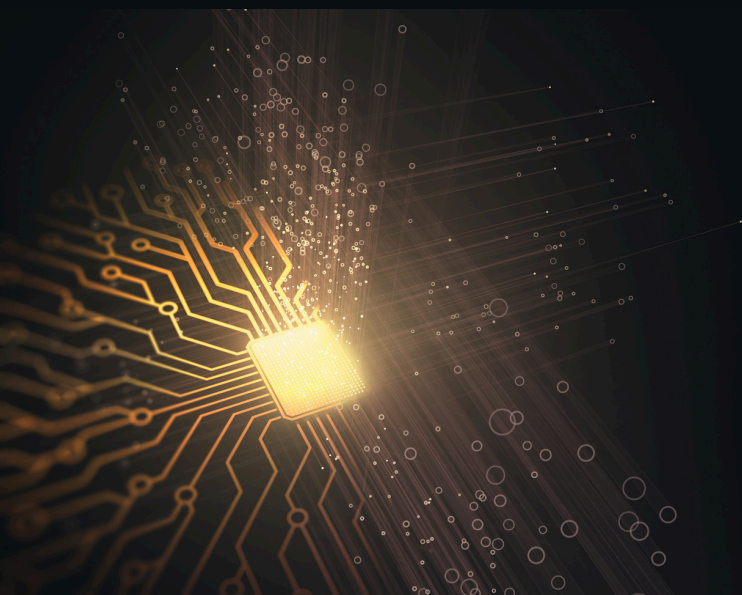
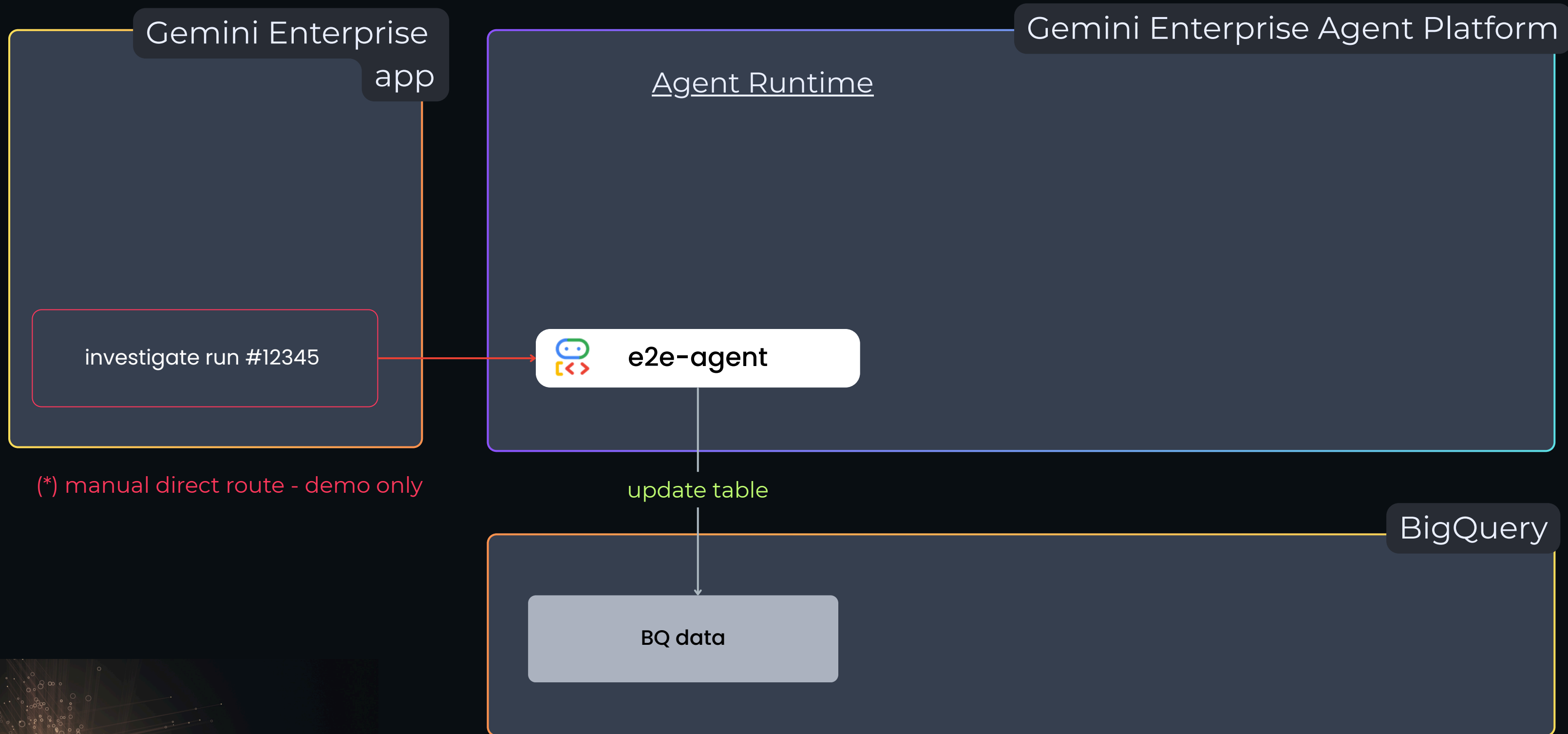


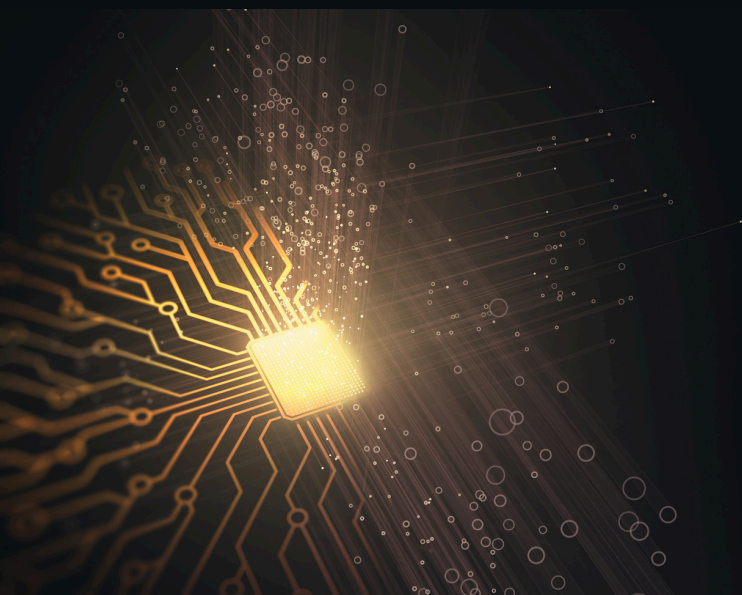
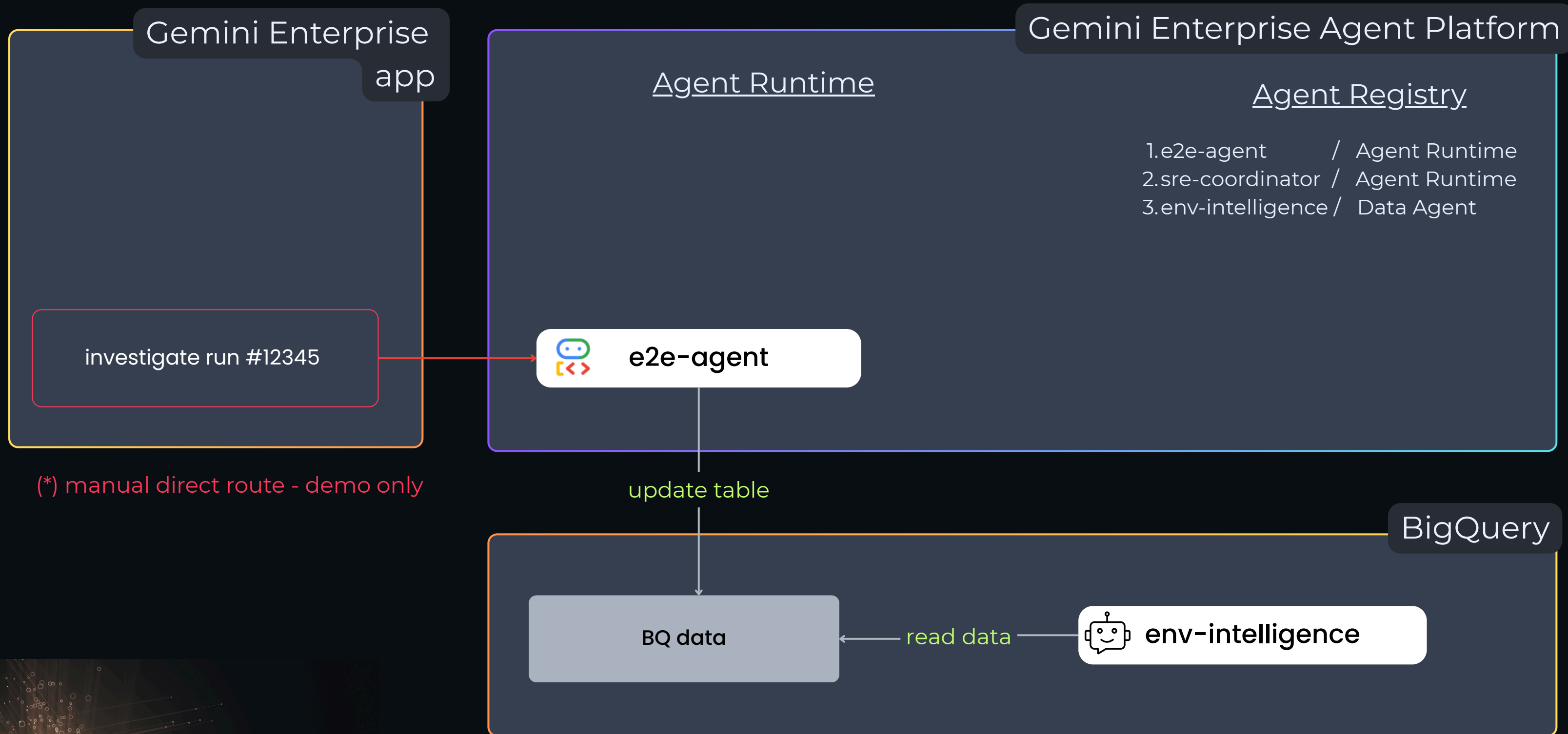
source:

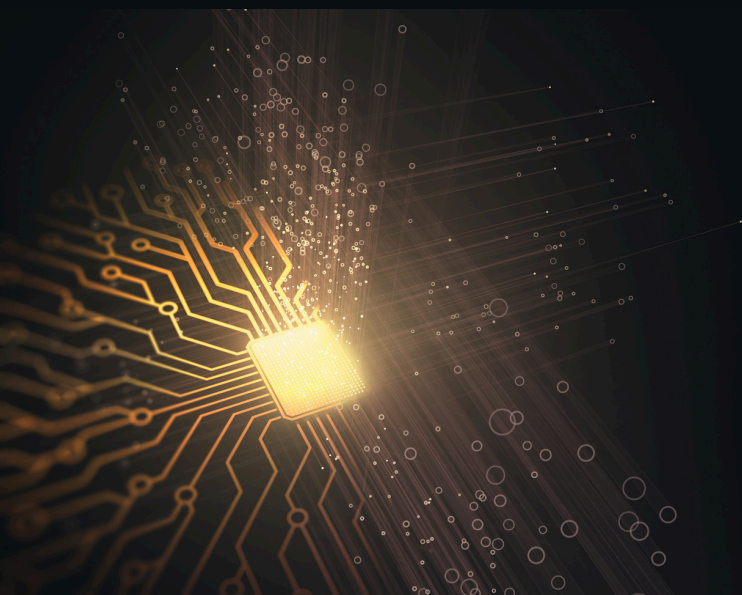
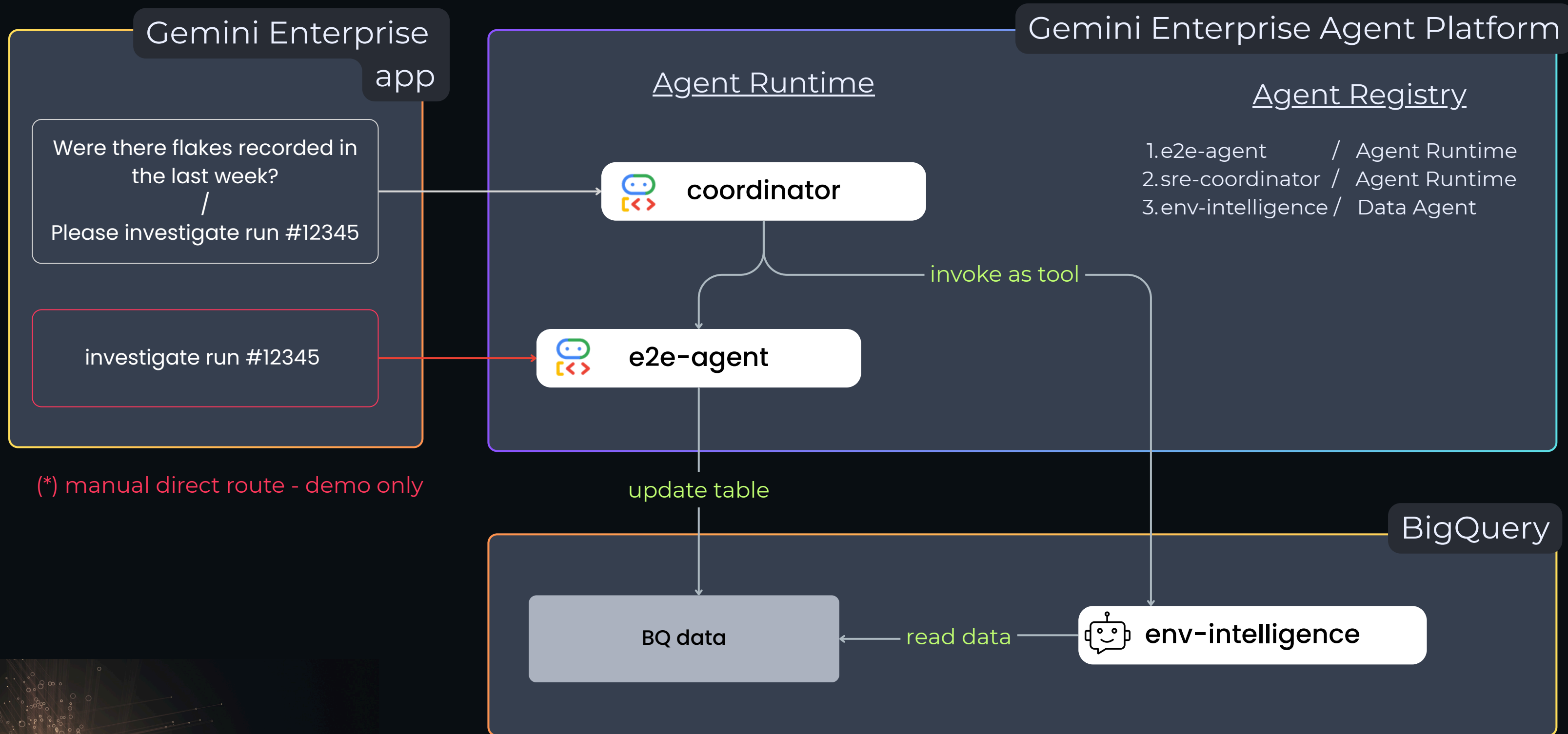
<https://developers.googleblog.com/4-engineering-patterns-behind-the-strongest-ai-agents-challenge-submissions/>



DEMO







RECAP

Autonomy framework and gating conditions

Know which level your agent is at, and state what it must prove before it moves up.

Product and architecture

The model is the core, but knowing who you build for and applying sound engineering patterns is what decides success.

Structure over prompts

Budgets, routing and verification live in the graph and in code, not in instructions to the model.



REFERENCES / CONNECT

AI in SRE: How Google is Engineering the Future of Reliable Operations [whitepaper]

<https://sre.google/resources/practices-and-processes/ai-engineering-reliable-operations/>

Google for Startups AI Agents Challenge – winners, Sept. 2, 2026

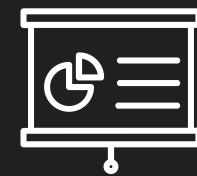
<https://developers.googleblog.com/4-engineering-patterns-behind-the-strongest-ai-agents-challenge-submissions/>

ADK Coding with AI

<https://adk.dev/tutorials/coding-with-ai/>

ADK blueprints

<https://github.com/google/adk-recipes/tree/main/core>



olga-mir.github.io/



github.com/olga-mir/sre-agents



Olga Mirensky



github.com/olga-mir

