AAIF MEETUP - Nov 3rd, 2026, Sydney

Platform Engineering for AI Agents

What the platform enforces, and what you still decide

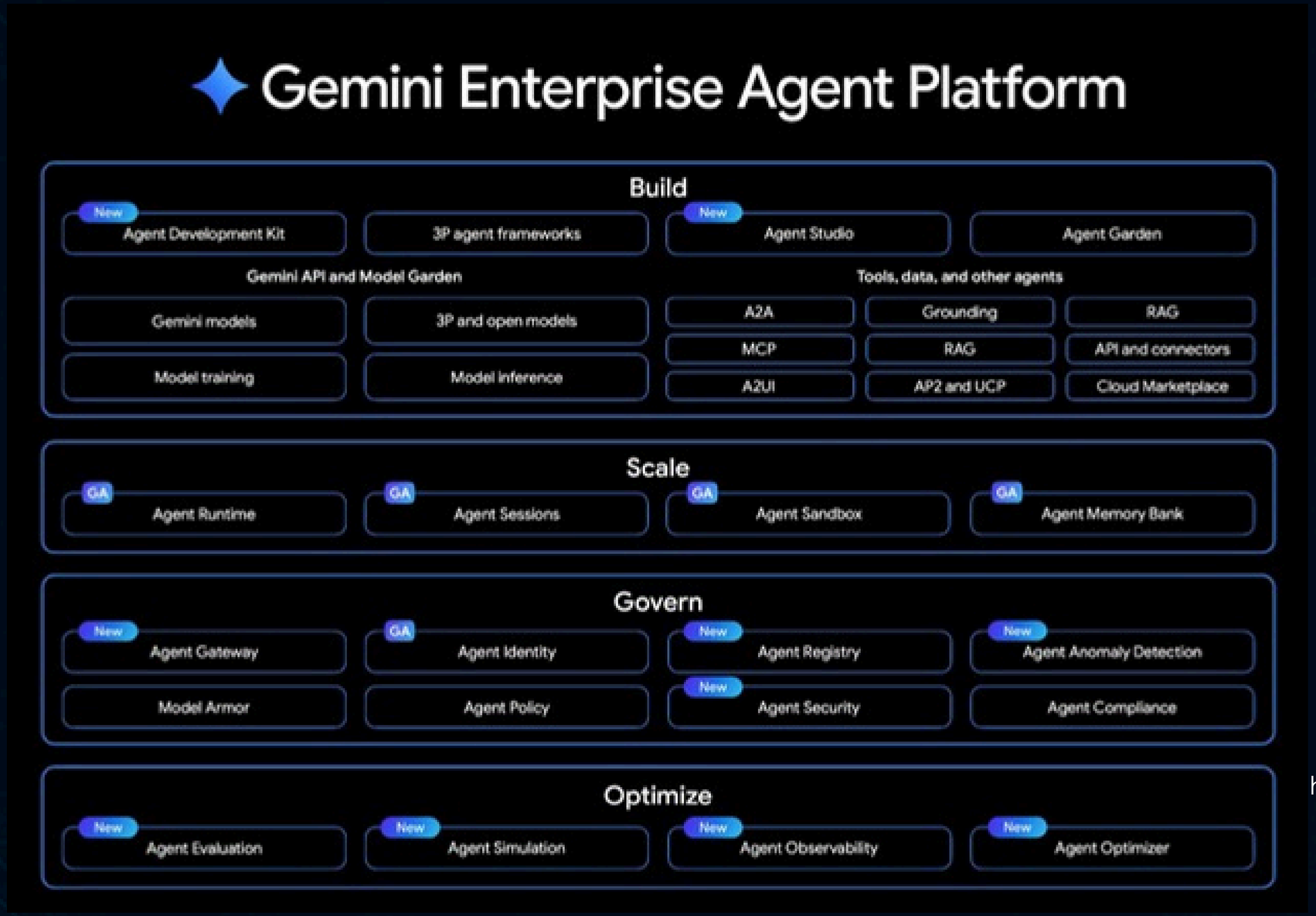
Olga Mirensky
Senior Staff SRE, Ping Identity

$whoami

Olga Mirensky
Senior Staff SRE @ Ping Identity

Platform and SRE roles across Enterprise, FSI and Tech

Building Agentic platforms on Gemini Enterprise Agent Platform for 1.5 yrs



From April'26 announcement
(New are now GA)

https://cloud.google.com/blog/topics/google-cloud-next/welcome-to-google-cloud-next26

Why not just prompt?

```
$ cat root_agent.yaml

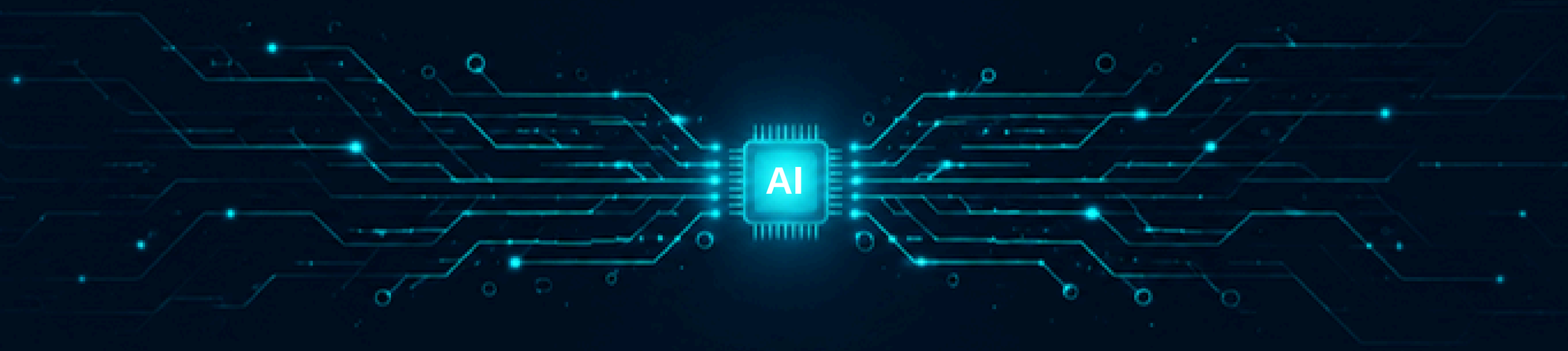
name: test_fixer
model: gemini-3.7-flash
instruction: “Fix all test failures, make no mistakes.”
```

```
$ agents-cli deploy --project "$PROJECT_ID" --region "$REGION"
```

Shared Responsibility Model

LAYER	MANAGED PLATFORM	IMPLEMENTATION EXAMPLE
Sandbox	Agent Sandbox, Agent Runtime	Custom tools, no shell, in-memory
Identity	Agent Identity, IAM	Agent Identity, Least Privilegde, IAM bindings
Network	Agent Gateway	No free-form external access, network security
Runtime policy	Agent Gateway Policy Engine	In code: Fixed call graph, tool-callback deny
Audit	Agent Observability, traces	Config, routing, storage

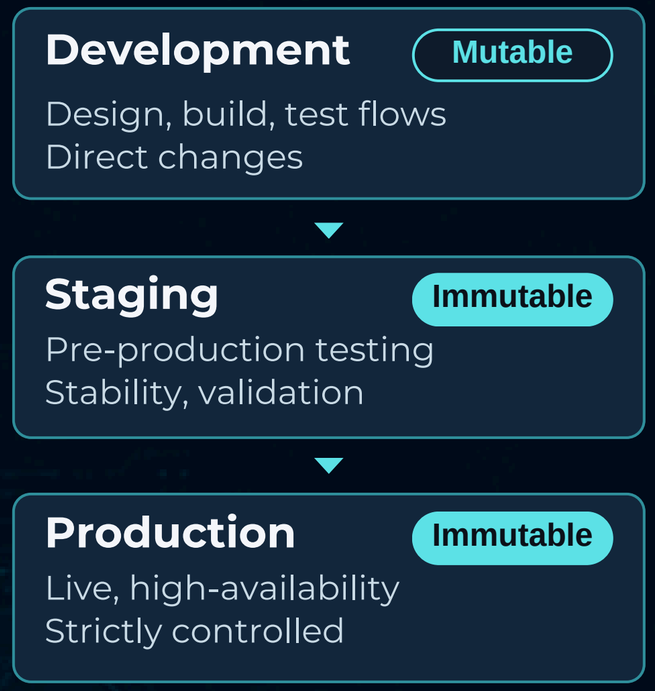
Diagnostic SRE Agents



PingOne Advanced Identity Cloud

Enterprise SSO | Adaptive MFA | Fine Grained Access | Fraud Detection | Agent Identity

Isolated tenancy



E2E example

Use e2e as example for this talk

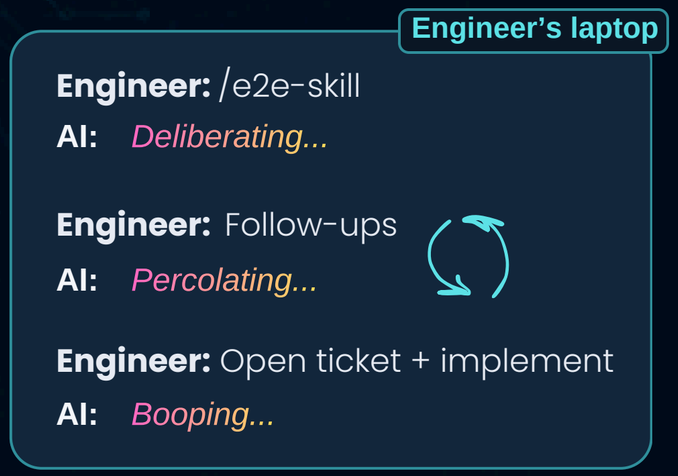
Full e2e matrix contains multiple tracks

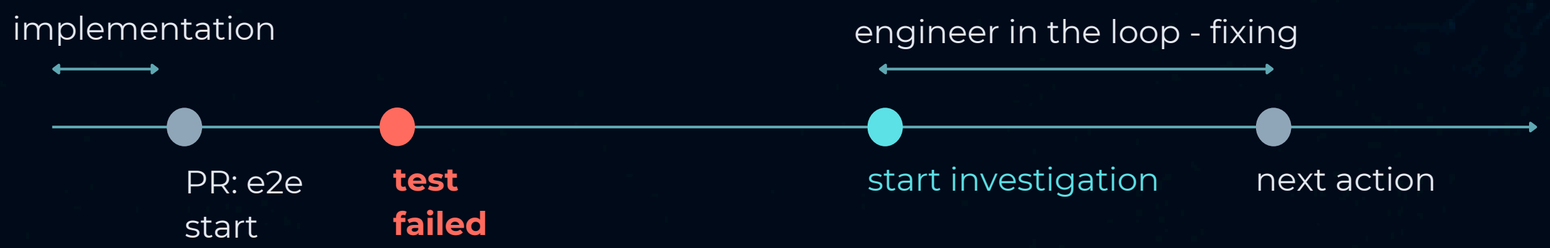
One suite can have more than one test failure

Failures can happen at different stages



Accelerating diagnostics to drive SDLC velocity





before

Accelerating diagnostics to drive SDLC velocity

Engineer's laptop

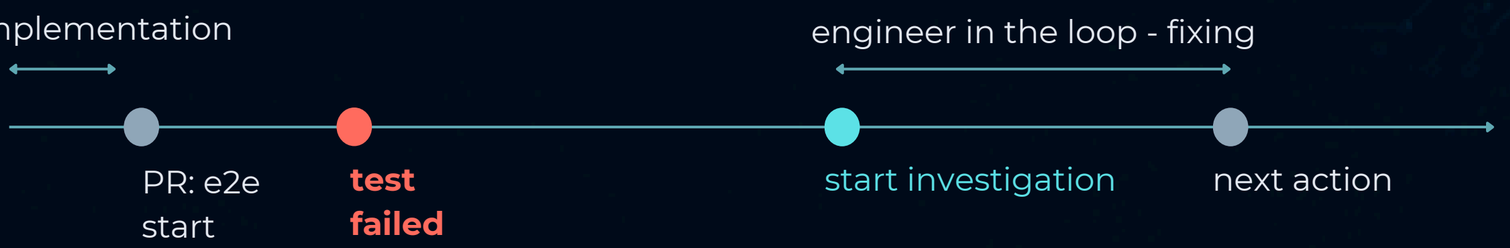
Engineer: /e2e-skill
AI: *Deliberating...*

Engineer: Follow-ups
AI: *Percolating...*

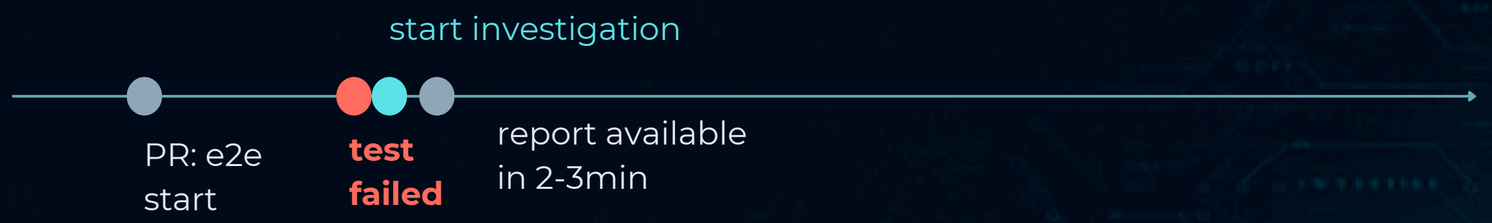
Engineer: Open ticket + implement
AI: *Booping...*



before



after



Cloud hosted workflows benefits

Immediate trigger on an event or schedule

Full coverage

Building shared context



Same process. Same risk profile?

Human oversight & accountability

- No human oversight (for fully autonomoous agents)
- Delegation when needed

Scale

- Run-away agentic loops
- Semantic loops, hallucinations

Network

- Cloud Environment vs corporate laptop

State

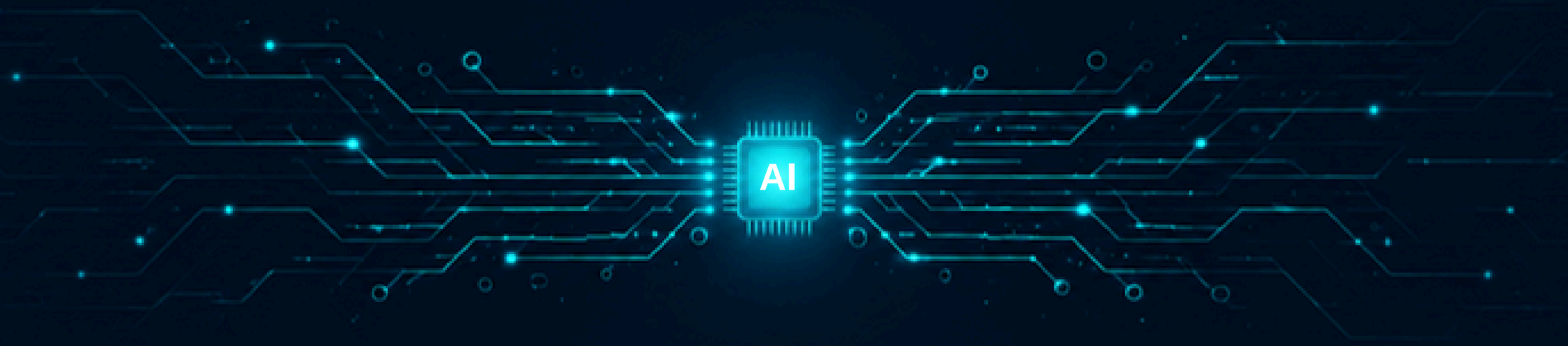
- Produced and consumed by agents
- Tampered state can affect other agents
- Which can be further exacerbated by scale and speed



From local process to cloud based workflow



Structure over prompts



Fixed wiring over discovery

```
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}),
    ]
)
```

The call graph is written in code

The e2e agent is invoked programmatically

no extra auth surface, nothing to discover

adding an agent is a code change.

Workflows

```
hitl_workflow = Workflow(
    name=“investigation”
    edges=[
      ("START", inner_workflow, dormancy_gate)
      (dormancy_gate,
        {“ask_deep_dive”: deep_dive,
         “close”: session_complete }),....]
```

OR

```
hitl_workflow = Workflow(
    name="investigation",
    edges=[
        ("START", ..., router),
        (router, {
            "await_input": dormancy_gate,
            "done": final_output,
        }),
        (dormancy_gate, {"invalid_feedback": dormancy_gate,
                         "ask_deep_dive": deep_dive}),..]
```

```
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
 )
```

Agent Platform handles scale to zero, resume and session state

BLAST RADIUS

A limit is an outcome, not a crash

nodes.py (simplified)

```
while budget["current_turn"] < budget["max_turns"]:
    budget["current_turn"] += 1
    try:
        raw = await ctx.run_node(investigator_agent,
                                 node_input=brief)
    except ClientError as exc:
        if exc.code != 429: raise
        quota_exhausted = True      # routed exit
        break
    except LlmCallsLimitExceededError:
        llm_calls_exhausted = True  # whole-run breaker
        break
    ...

# exit_reason: quota_exhausted | llm_call_limit_exceeded
#              | budget_exceeded | investigation_complete
```

Two levels of limit

Turns per investigation, plus a whole-run LLM-call breaker.

Every exit is recorded

The run still reaches the verifier, the human and the dataset.

Still to build

A kill switch: stop a running agent from outside.

From a reference implementation: github.com/olga-mir/sre-agents

IDENTITY + AUDIT

The model proposes. Code writes.

state.py, nodes.py (simplified)

```python
Verdict = Literal["INTRODUCED_BY_COMMIT", "PRE_EXISTING",
                  "INFRA_SUSPECTED", "FLAKE_SUSPECTED",
                  "INDETERMINATE"]

class CodeFinding(BaseModel):
    verdict: Verdict = "INDETERMINATE"
    confidence: float = Field(default=0.0,
        description="0.0 to 1.0. Never above 1.0.")
    evidence: list[EvidenceRef] = Field(default_factory=list)

# the model never sees BigQuery
record_outcome_node = FunctionNode(
    func=record_outcome,          # deterministic writer
    retry_config=RetryConfig(max_attempts=2),
)
```

Typed output

A schema the verifier and the dataset can rely on.

A node that is code

The write is a FunctionNode, not a tool the model can call.

Credential enforces it

Only the writer's identity can write to the dataset.

From a reference implementation: github.com/olga-mir/sre-agents

DATA PROVENANCE & SEMANTIC INTEGRITY

Verifier

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

...

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

DATA PROVENANCE & SEMANTIC INTEGRITY

Verifier

Overreach | Broken references | Unverified claims | Report shape | ...

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



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

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

DATA PROVENANCE & SEMANTIC INTEGRITY

Verifier

Overreach | Broken references | Unverified claims | Report shape | ...

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

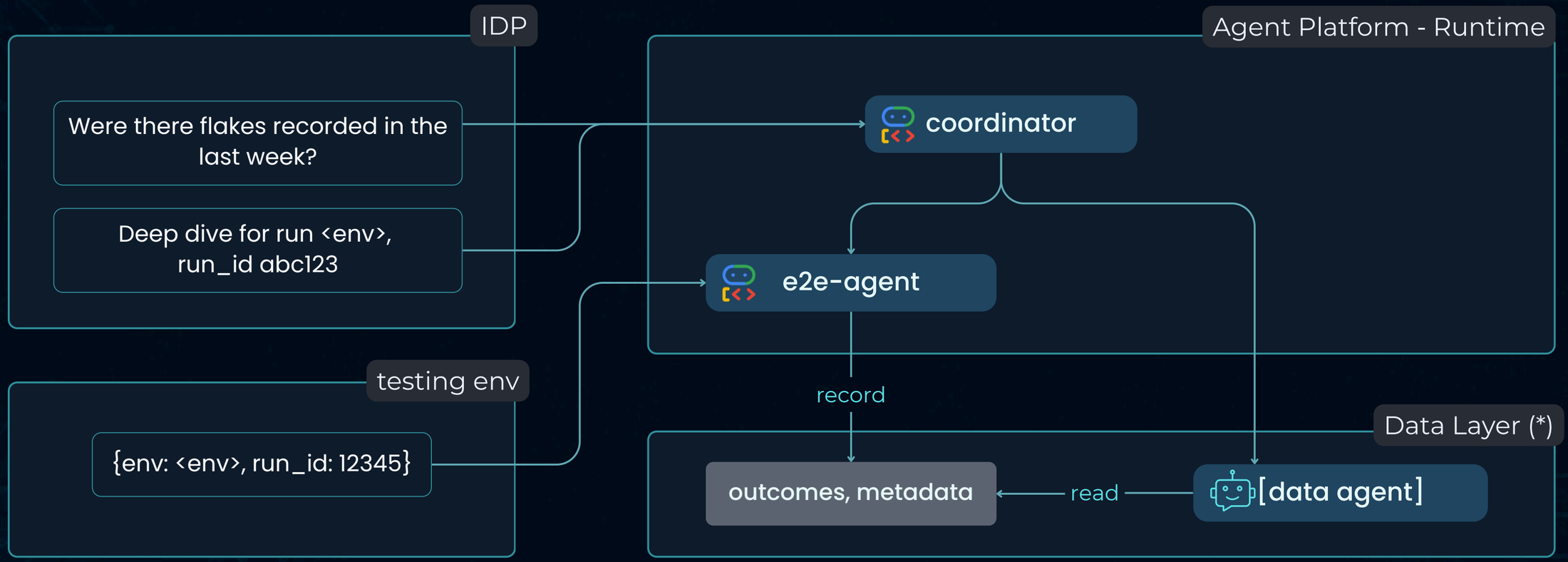
Changed recommendation

f_82bd4sd | certainty: **probable**

User / integrations


DEMO

Demo architecture



demo implementation: github.com/olga-mir/sre-agents

(*) - implemented in public demo with BigQuery Data Agents

Takeaways

Enforcement is provided

The platform gives you the enforcement points. The policy is yours, derived from your architecture.

Structure over prompts

Fixed wiring, tool-level denies, budgets with routed exits. A kill switch is next.

Effectiveness over perfection

Triage every failure, dig deep where it pays. Deciding what to build is the hard part.



References

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/

Google's Approach for Secure AI Agents

https://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf

Secure AI Framework - SAIF

https://saif.google/focus-on-agents

Connect

olga-mir.github.io

github.com/olga-mir/sre-agents

Olga Mirensky

github.com/olga-mir