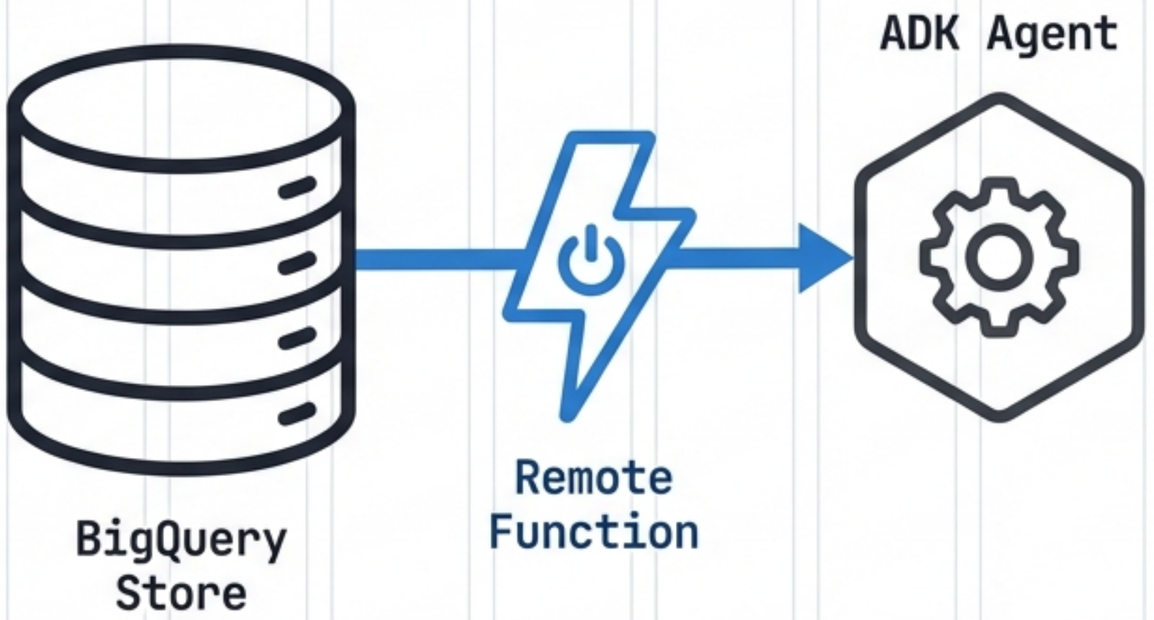
Intelligent Scale: Processing 10k Records with BigQuery & ADK

An architectural review of connecting SQL-based data to Multi-Agent Systems using Remote Functions.

SYSTEM CONTEXT	
SCALE:	10,000+ Customer Records
STACK:	BigQuery / Cloud Run / Google ADK
PATTERN:	Remote Function Integration



Executive Summary: The Strategy

The Challenge

Run complex, multi-agent analysis (Security, Billing, Retention) on **10,000+** raw customer rows without **ETL** latency or data movement.

The Solution

BigQuery Remote Functions act as a bridge to a **Cloud Run** service hosting Google's **Agent Development Kit (ADK)**. This brings compute directly to the data.

The Verdict

Achieves '**Write Once, Run Anywhere**' logic with high development velocity. Main trade-offs involve session management overhead vs. raw **batch API** costs.

Result: Complex reasoning accessible via standard **`SQL queries`**.

Moving Beyond Regex: The Need for Intelligent Analysis

The Data Landscape

`loyalty_program (RECORD)`

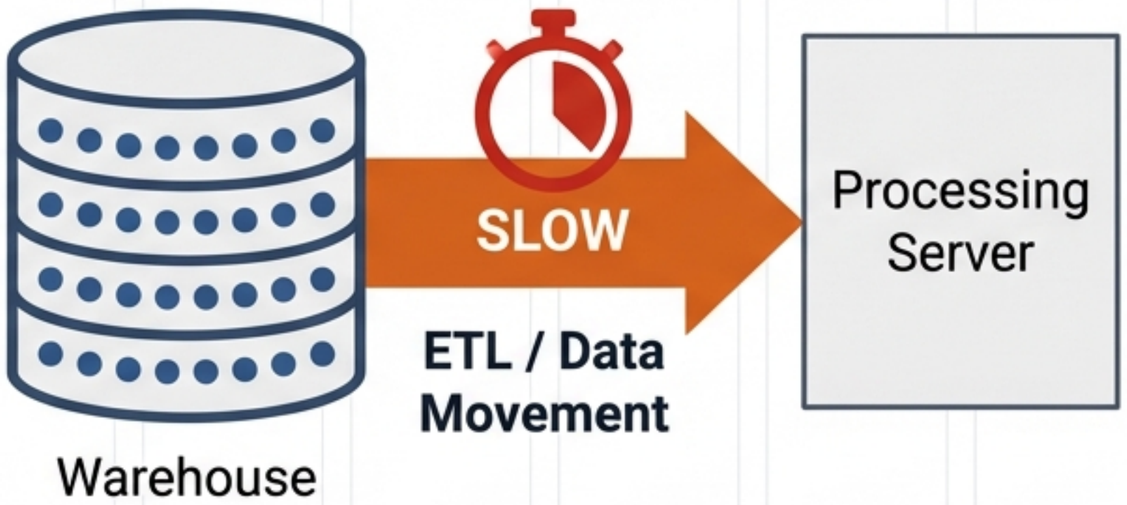
`usage_stats (RECORD)`

`security_profile (RECORD)`

`billing_summary (RECORD)`

Problem: Standard SQL or Regex cannot reliably distinguish between nuanced states, such as a ‘Billing Retry’ vs. a ‘Churn Risk’.

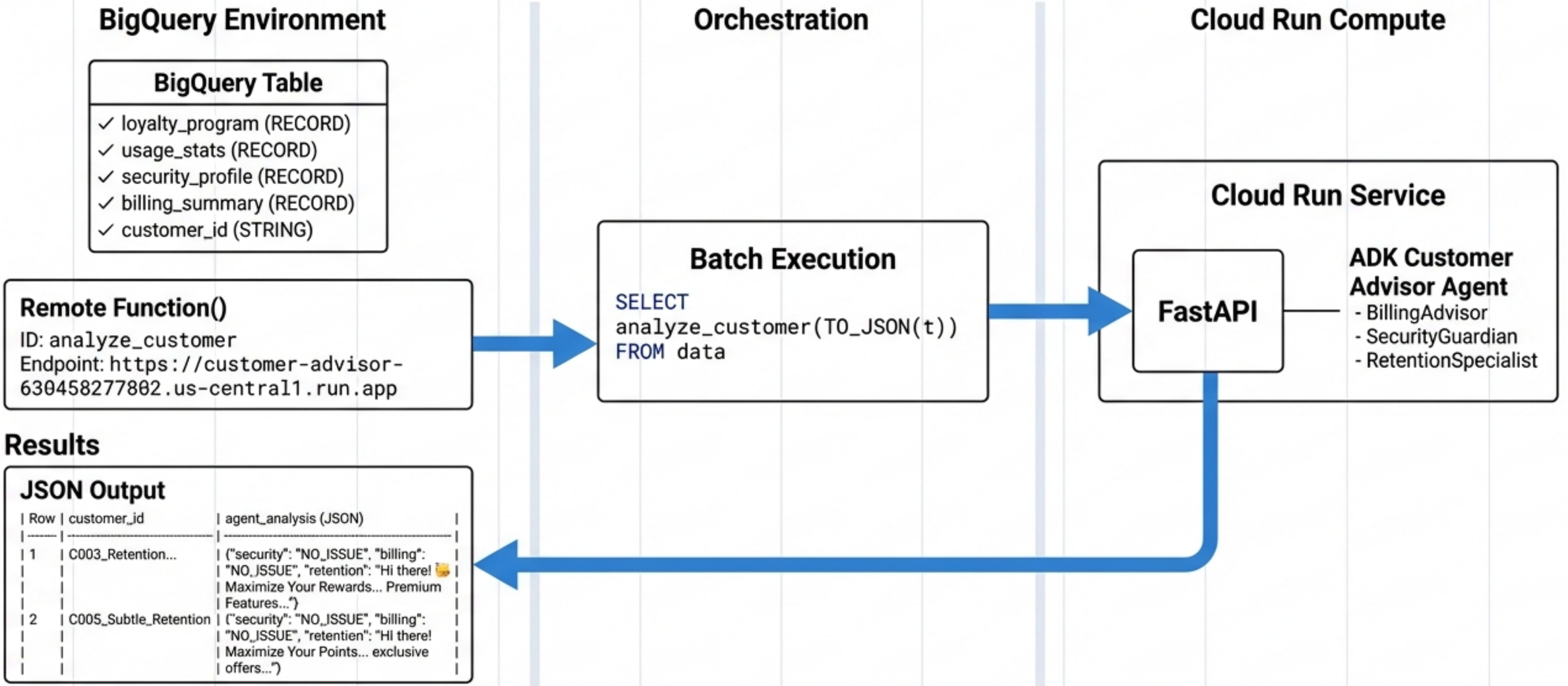
The Bottleneck



Constraint: Moving 10k records out of the Warehouse for processing adds latency and security risks. We need to bring the compute to the data.

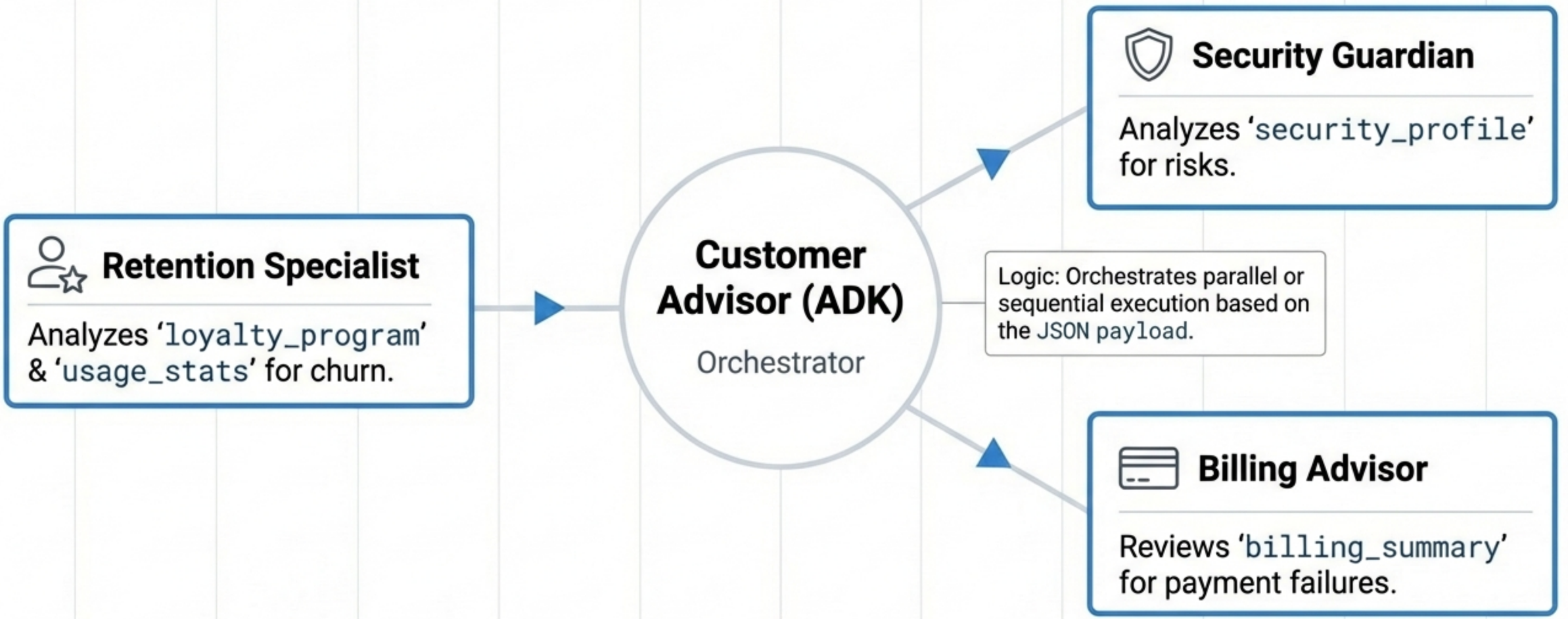
Engineering Editorial

The Integration Pattern: Remote Functions



Engineering Editorial

The Brain: Multi-Agent Orchestration



Engineering Editorial

Invoking Intelligence via SQL

SQL Query

```sql
SELECT
  customer_id,
  remote_function_dataset.
    analyze_customer(TO_JSON(t)) as
    agent_analysis
FROM
  dataset.customer_scenarios AS t;
```

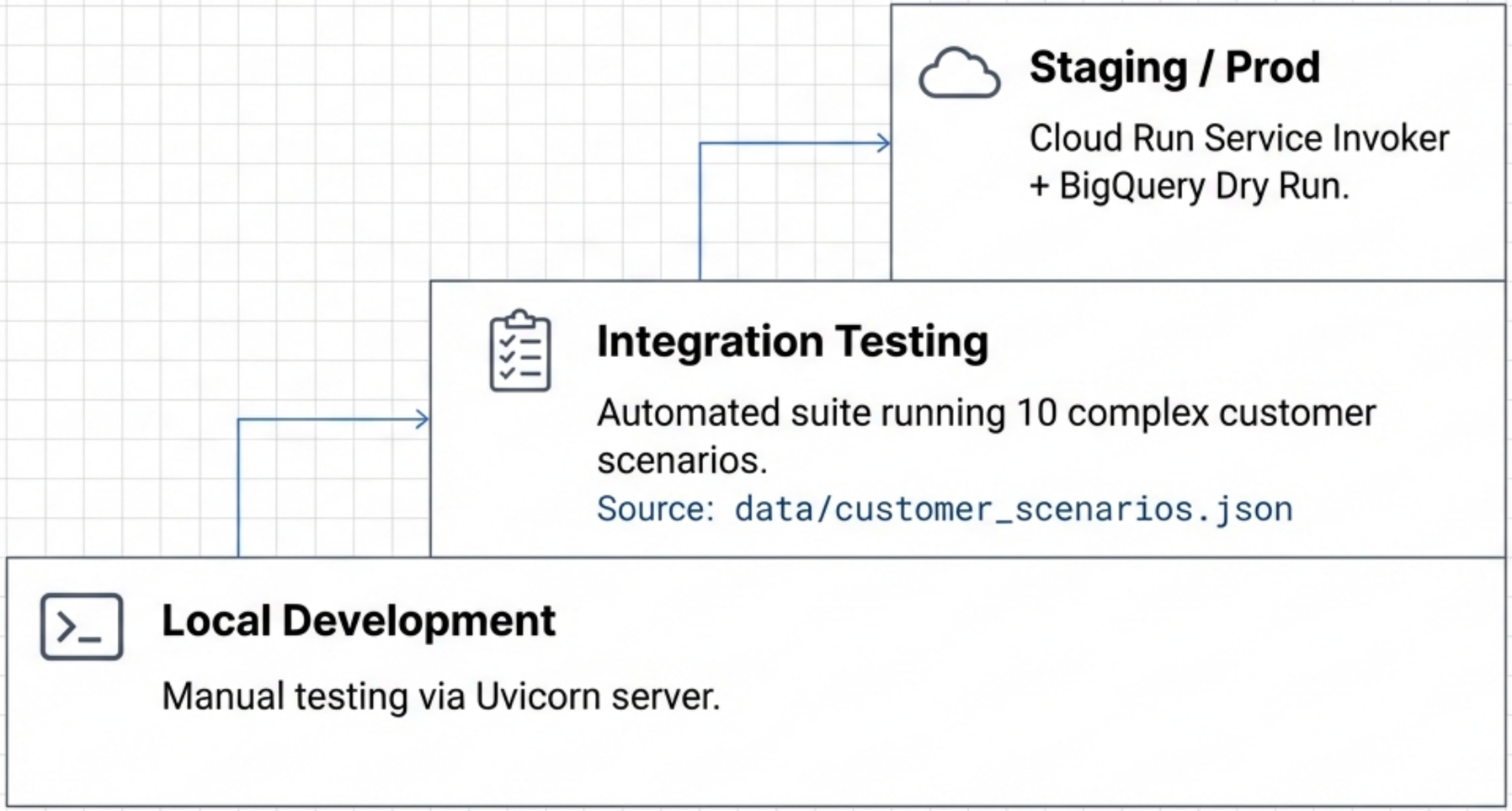
Encapsulated Complexity

Structured Response

```
{
  'security': 'NO_ISSUE',
  'billing': 'NO_ISSUE',
  'retention': 'Hi there! Maximize
Your Rewards. We notice you have
unused points...'
}
```

Engineering Editorial

Validating the Logic: Testing Strategy



Validation Coverage:

- ✓ Security Logic
- ✓ Billing Logic
- Retention Logic
- Response Formatting

Engineering Editorial

Path to Production: Deployrment & Configuration

01 → 02 → 03 → 04

Deploy Agent

Push ADK container to Cloud Run. Ensure ‘Service Invoker’ role is active.

Connect BQ

Create Cloud Resource Connection in region us-central1.

Define Function $f(x)$

Map the Remote Function in BigQuery to the Cloud Run endpoint.

Load Data

Import scenarios to `rocketech-de-pgcp-sandbox`.

Accelerator: Using the ‘Agent Starter Pack’ dramatically speeds up the prototype-to-production lifecycle.

Engineering Editorial

Architectural Verdict: ADK vs. Native Implementation

The Wins (Why ADK)

Decoupling: "Write Once, Use Anywhere'. Same agent works in Gemini Enterprise or Event-Driven flows.

Sophistication: Built-in Tracing, Tool calling, and Multi-Agent orchestration.

Velocity: Faster dev cycles via the Agent Starter Pack.

Net Positive Verdict

The Challenges

State Management: ADK is session-based; adapting to stateless BQ functions requires memory memory optimization (`InMemorySessionService`).

Batching: Modular sub-agents make standard batch inference harder/more costly than raw Vertex AI Batch API.

Engineering Editorial

Final Recommendation

Primary Use Case: Adopt this architecture for logic-heavy reasoning over warehouse data where development velocity and agent reusability are prioritized over raw batch throughput costs.

Alternative: Evaluate Vertex AI Batch API if volume > 1M records and logic is simple.

Action Items

1. Development: Use **Gemini CLI** for AI-assisted development (context pre-configured).

2. Testing: Run the integration test suite to validate the 10 scenarios.