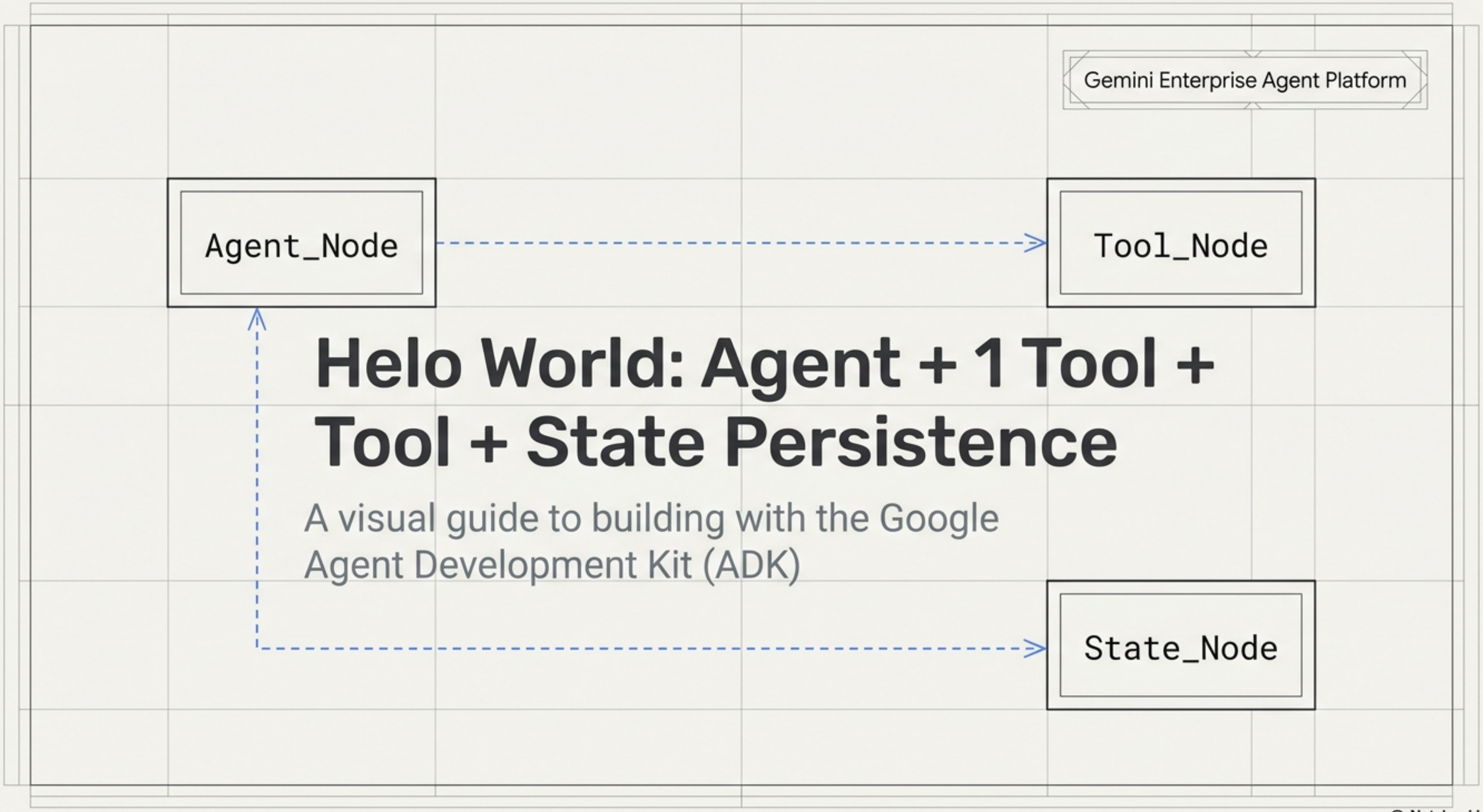

Gemini Enterprise Agent Platform
Agent_Node
Tool_Node
Helo World: Agent + 1 Tool + Tool + State Persistence
A visual guide to building with the Google Agent Development Kit (ADK)
State_Node

The ADK Portability Promise

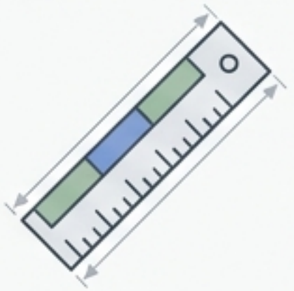
Simplicity

Pure Python

No system dependencies.
Installed via a single command:

```
pip install google-adk
```

Speed



< 10 Lines

The configuration code required to run a fully stateful, tool-using agent.

Testing

Built-in Local UI

Test agents interactively via `adk web` without writing a custom test harness.

Performance

Sub-second

Cold starts for pre-warmed instances on the Agent Runtime.

Anatomy of an ADK Tool

```
def log_expense(amount: float, category: str):
    """Use this tool when a user mentions spending money."""
    return f"Logged {amount} in {category}"
```

Trigger Logic

The model reads this—not the function name—to decide when to invoke the tool.

Parameter Schema

ADK infers the JSON schema automatically at runtime.

Model Input

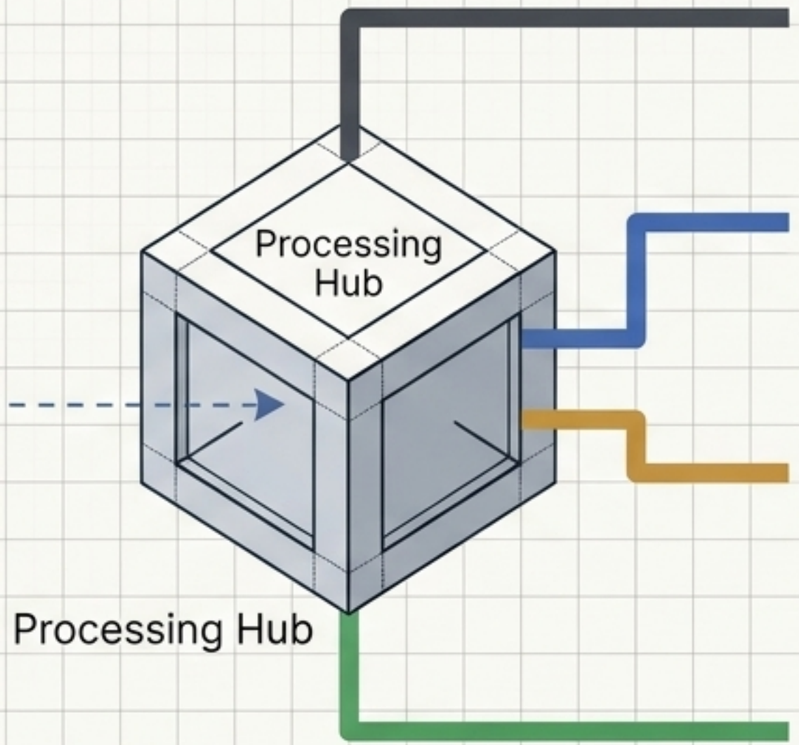
Returns must be strings or JSON-serializable values for the model to read.

Zero Decorator Magic. No `@tool` required.

Wiring the Agent's Instruction

agent.py

```
agent = Agent(
    instruction=system_prompt,
    tools=[log_expense]
)
```



Persona:
You are a helpful budget tracker.

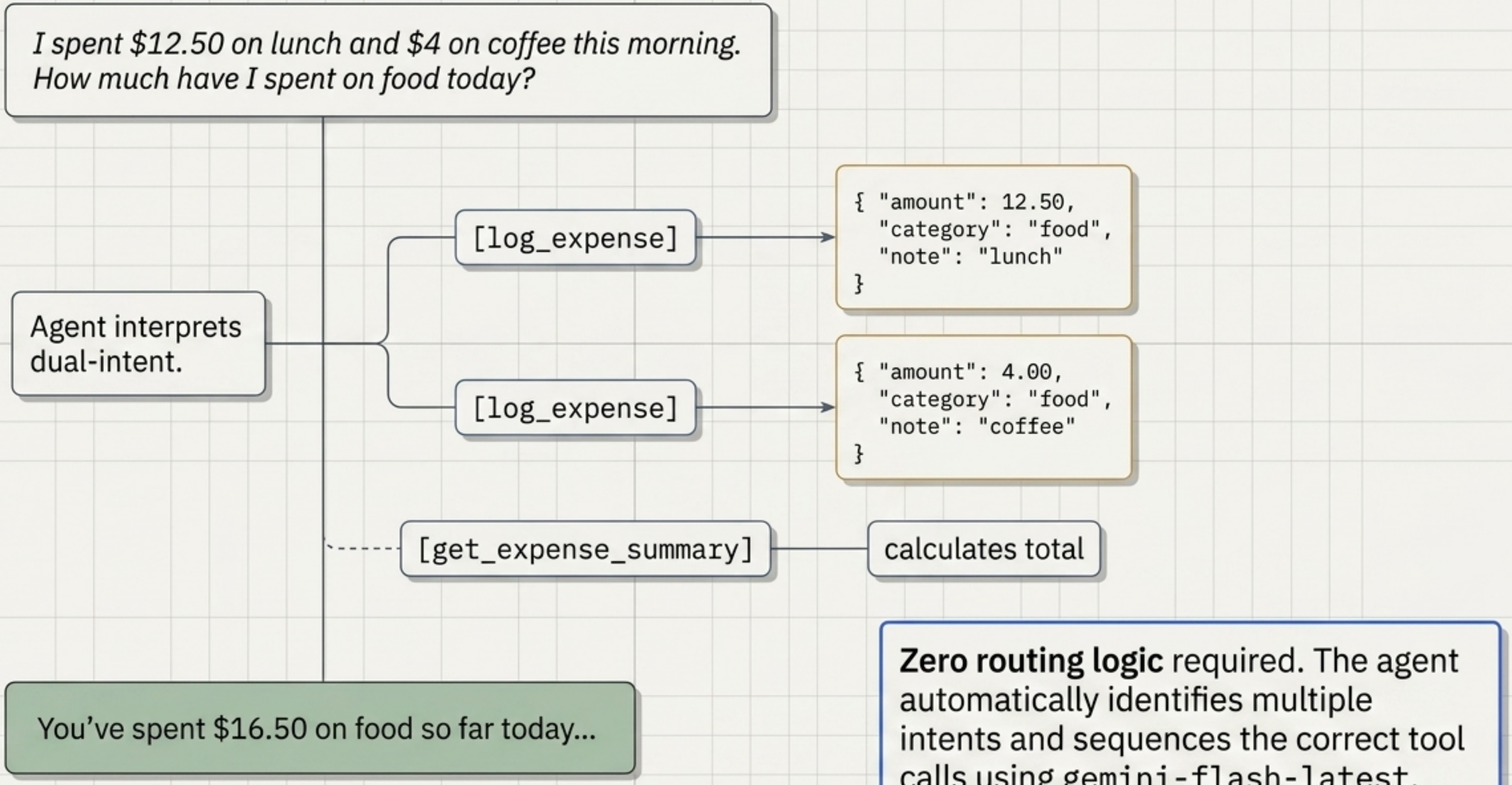
Rules:
Explicit conditions for when to call specific tools.

Guardrails:
Do not hallucinate financial data.

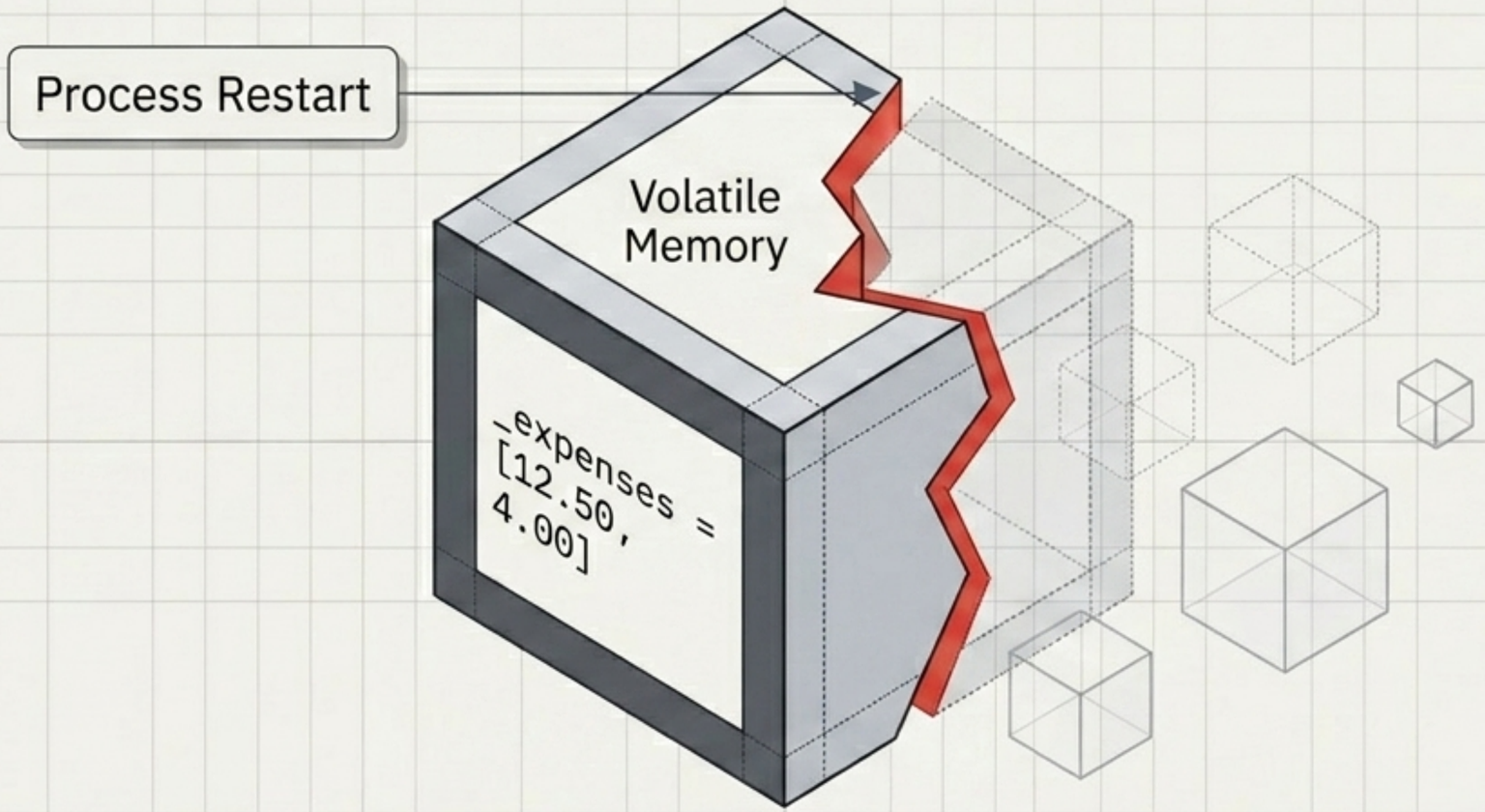
Constraints:
Keep outputs concise.

The **instruction** field acts as the **system prompt**. It sets the persona, rules, and guardrails, shaping the model's behavior around the tools you provided.

Autonomous Multi-Step Tool Execution



The Volatility Problem



By default, variables live in memory. When the process restarts, the expenses vanish. Real agents require state persistence that survives restarts without forcing the developer to manage complex database integrations.

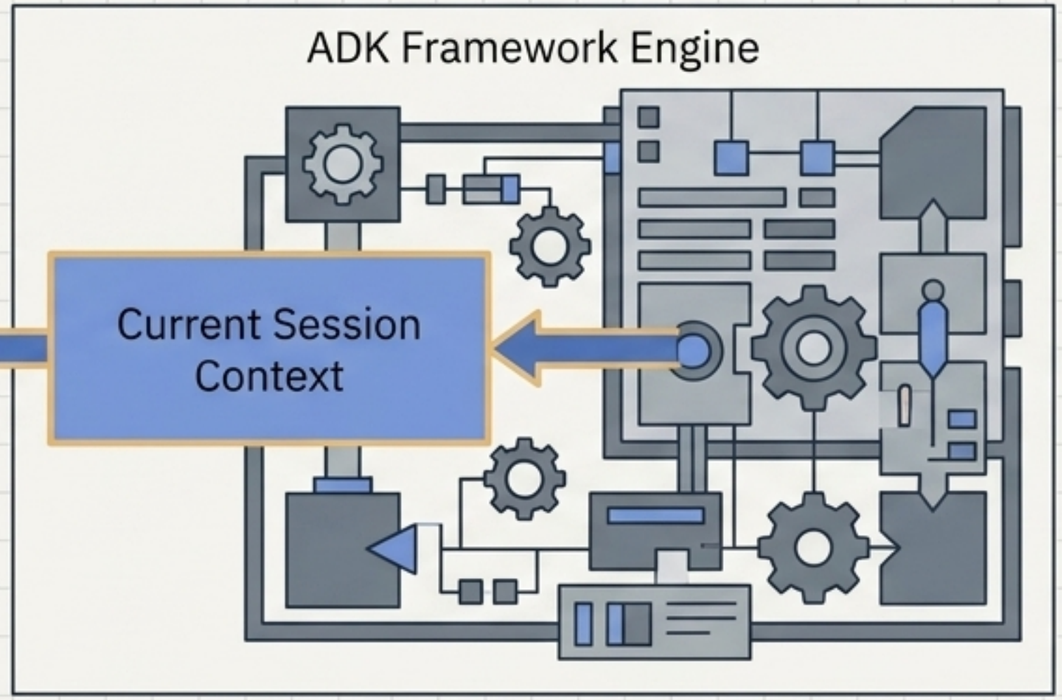
The Grand Split: Session State vs. Memory Bank

	Session State	Memory Bank
Scope	One conversation	All conversations for a user
Duration	Until session expires	Long-term (days to indefinite)
Content	Raw context & structured `state dict`	Distilled **Memory Profiles**
Latency	Sub-millisecond (local `dict`)	Low-latency retrieval (`indexed`)
Creator	You (explicit writes)	The Platform (model `distillation`)
Reader	Your tools (explicit read)	Agent's instruction context (automatic)

Use Session State for today's shopping cart. Use Memory Bank for the user's lifelong dietary preferences.

Magic Routing via Dependency Injection

```
def log_expense(
    session: Session,
    amount: float):
```



You do not pass the session manually. ADK reads the Session type annotation in your function signature and automatically injects the active session state when invoking the tool.

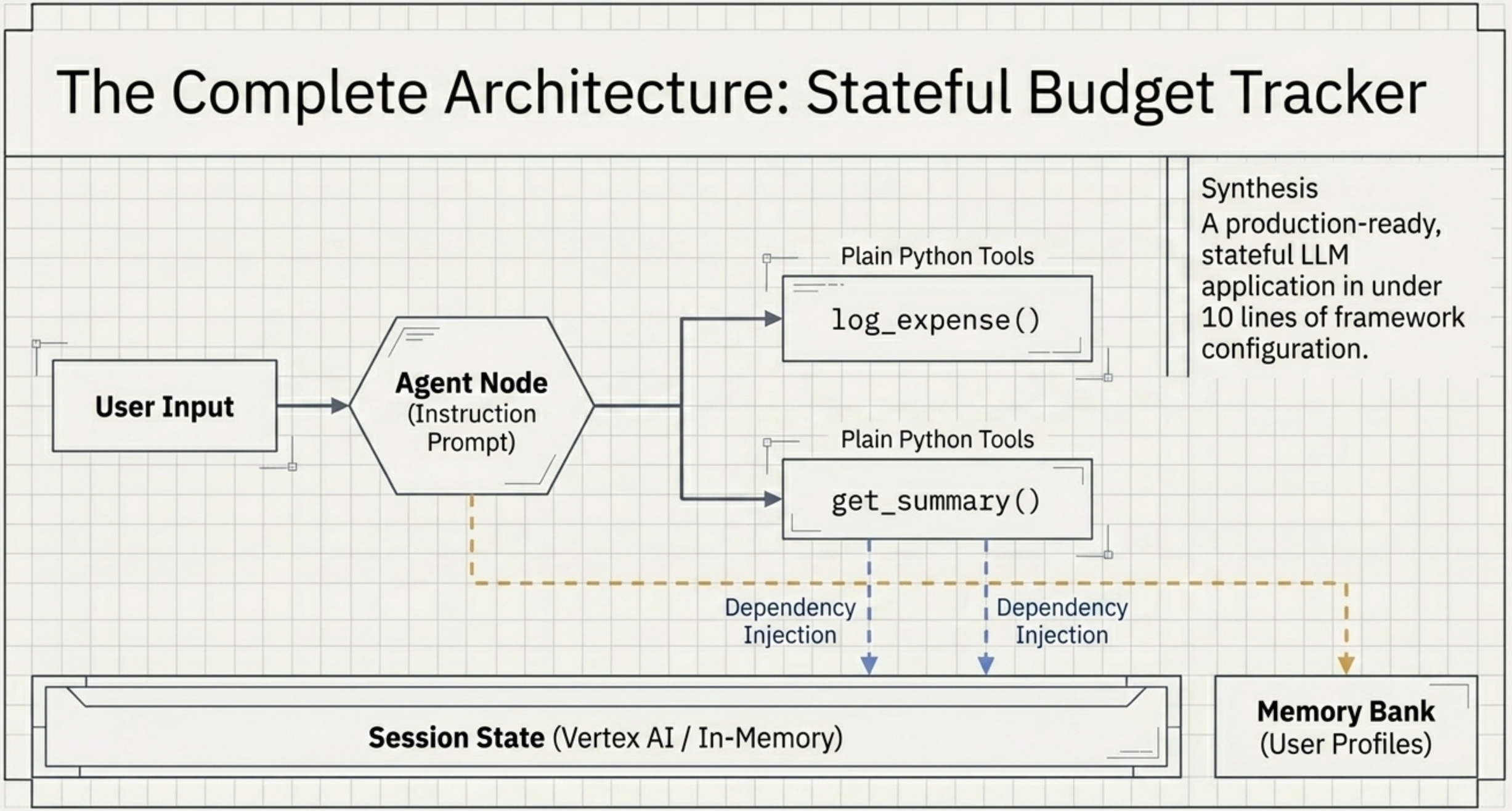
The Portability Toggle

Local Development -> InMemorySessionService()

Production -> VertexAiSessionService()

Zero code changes to your agent or tools. Develop locally with in-memory services. Deploy to Vertex AI with a single configuration swap.

The Complete Architecture: Stateful Budget Tracker



What’s Next: Scaling to Multi-Agent Orchestration

You now have a single-agent system with state. In Chapter 3, discover how to use the Agent Registry to make specialist sub-agents discoverable by a master Supervisor agent.