Build Your First AI Agent

Agent Frameworks, RAG, Cloud Deployment

Olga Mirensky

Senior Staff SRE (Site Reliability Engineer)

Our (Ambitious) Agenda

A little bit of theory: Agents, Frameworks, RAG

A little bit about tooling: LiteLLM, uv, models

Explore Agent code

Clone repo → Adapt → Run Agent locally → Explore → [RAG]

Cloud (Vertex AI) Demo

Questions

AI Agents and Frameworks

What is an AI agent?

Special **type of application** leveraging LLM

We need frameworks! (maybe)

ADK - Agent Development Kit

LangGraph

Microsoft Agent Framework

OpenAI SDK

CrewAI and 100s more

ADK Framework

Python | TypeScript | Go | Java

```python
from google.adk import Agent
from google.adk.tools import google_search

agent = Agent(
    name="researcher",
    model="gemini-flash-latest",
    instruction="You help users research topics tho
    tools=[google_search],
)
```

https://adk.dev

RAG - Retrieval Augmented Generation

RAG combines information retrieval with text generation to enhance LLM output by incorporating additional sources.

- internal knowledge bases
- data (e.g. customer reports, invoices)

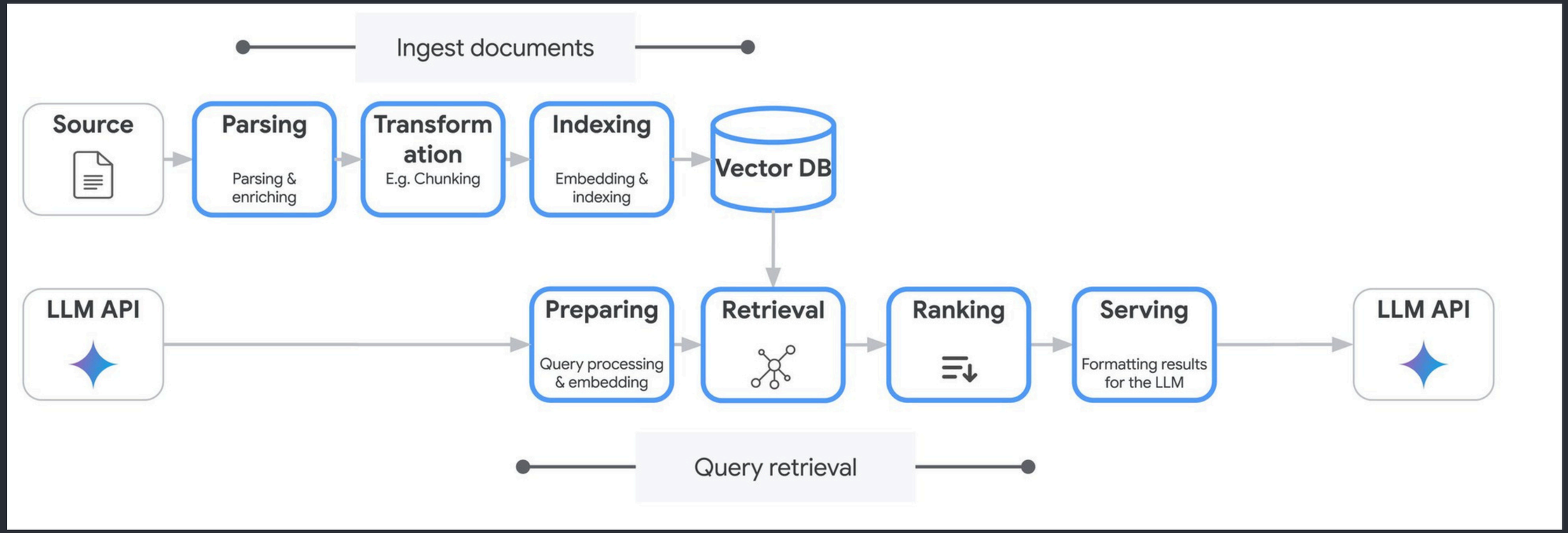
But... we have 1M contex window, why still RAG?

Utilize projects effectively: Projects use retrieval-augmented generation (RAG), which allows Claude to work with larger amounts of information by only loading relevant content into the context window.

https://support.claude.com/en/articles/8606394-how-large-is-the-context-window-on-paid-claude-plans

1M context window is prone to problems with recall accuracy, often suffering from the "lost in the middle" phenomenon where the model overlooks critical information buried in the middle of the prompt.

RAG In Action



https://docs.cloud.google.com/vertex-ai/generative-ai/docs/rag-engine/rag-overview

LiteLLM


AI Gateway to provide model access, fallbacks and spend tracking across 100+ LLMs. All in the OpenAI format.
https://litellm.ai
Request Pricing
Deploy LiteLLM On-Prem
User
LiteLLM
Cost Tracking
Batches API
Guardrails
Model Access
Budgets
LLM Observability
Rate Limiting
Prompt Management
s3 Logging
Pass-Through Endpoints

LiteLLM

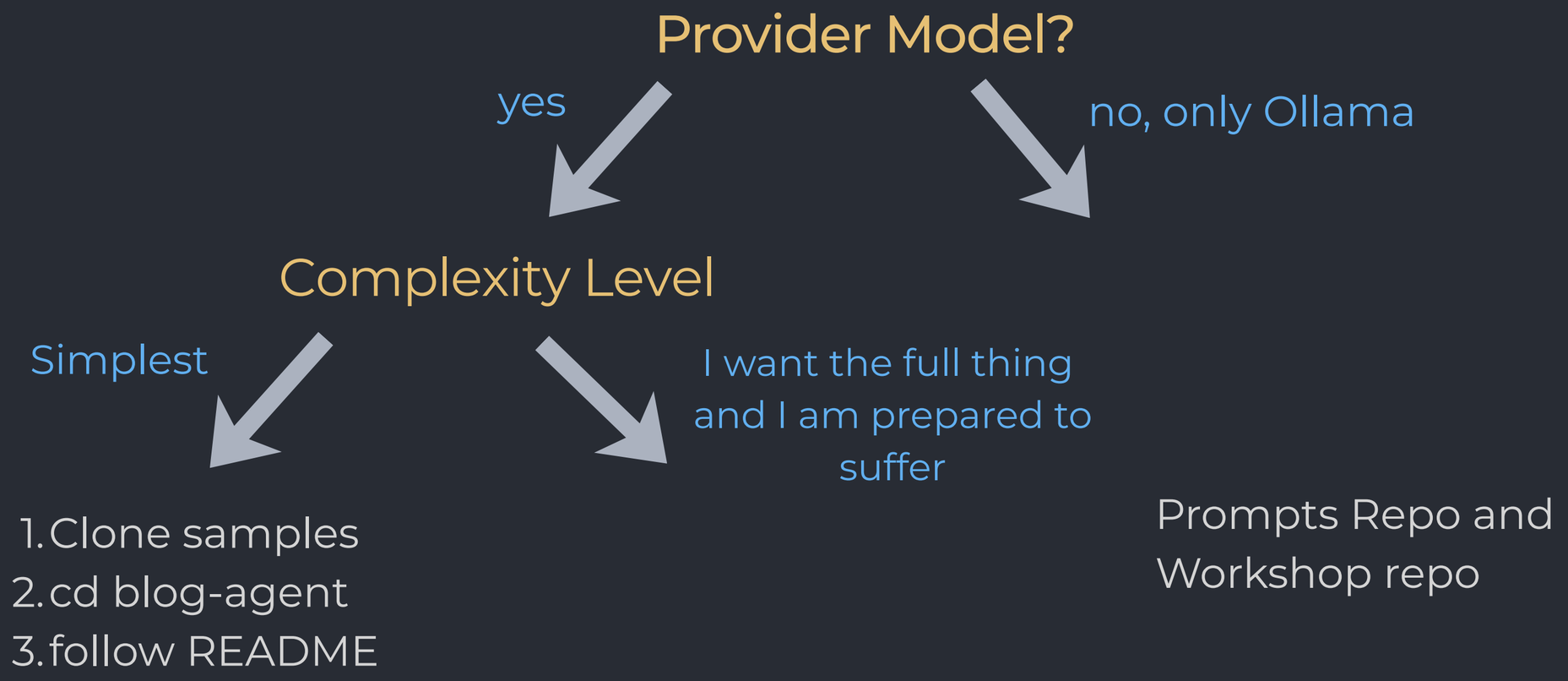
```python
from google.adk.agents import Agent


agent = Agent(
    name="agent-name",
    model=config.worker_model,
    ... rest of the config
)
```

```python
from google.adk.agents import Agent
from google.adk.models.lite_llm import LiteLlm

agent = Agent(
    name="agent-name",
    model=LiteLlm(config.worker_model),
    ... rest of the config
)
```

Decision Tree



Workshop Repositories

ADK Samples:

github.com/**google/adk-samples**/
path: python/agents/blog-writer

Prompts

https://github.com/olga-mir/workshop-prompts

Workshop-repo

https://github.com/olga-mir/workshop-prompts

Reference Links

Context Engineering, SwirlAI newsletter: https://www.newsletter.swirlai.com/p/state-of-context-engineering-in-2026

ADK Sample: https://github.com/google/adk-samples/tree/main/python/agents/blog-writer

ADK: https://adk.dev

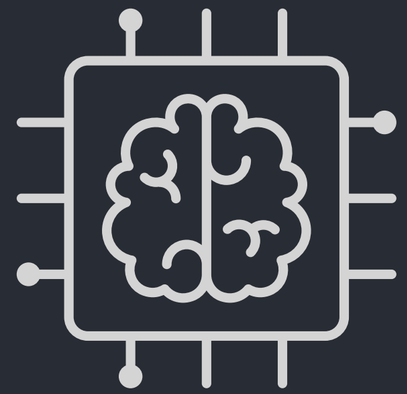
Appendix

Following slides are not part of the demo, but might be required to explain Agent behaviour with local models

Function Calling

`FunctionDeclaration`
(spec)

User free-text form

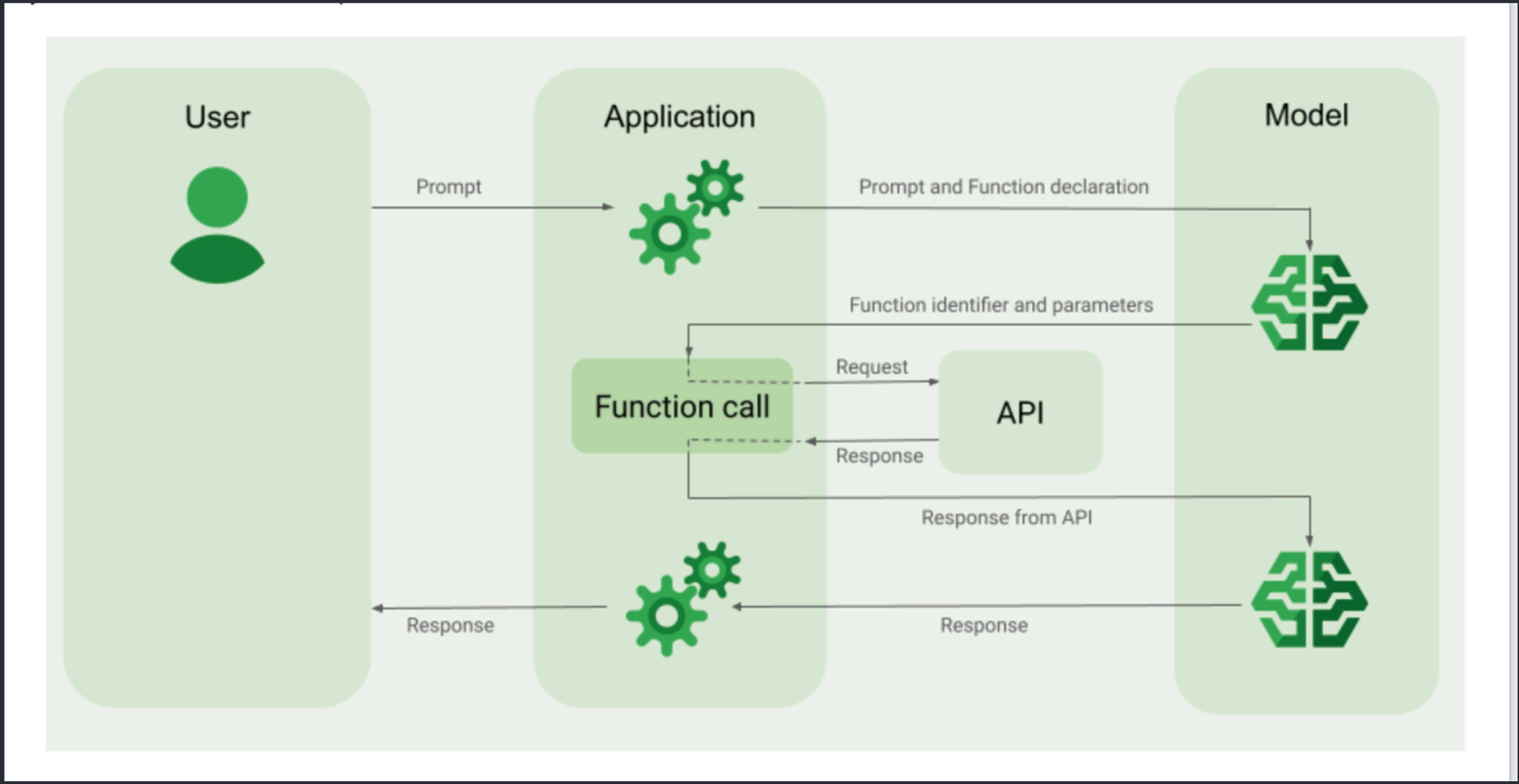


User: What Google Pixel phones available in stores near me?

LLM: here's what your user wants. In your code call this function <function_name> with these params <param1, param2>

Your program: Programatically calls the methods, returns response to user (or llm to generate user response)

Function Calling (cont.)



https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling