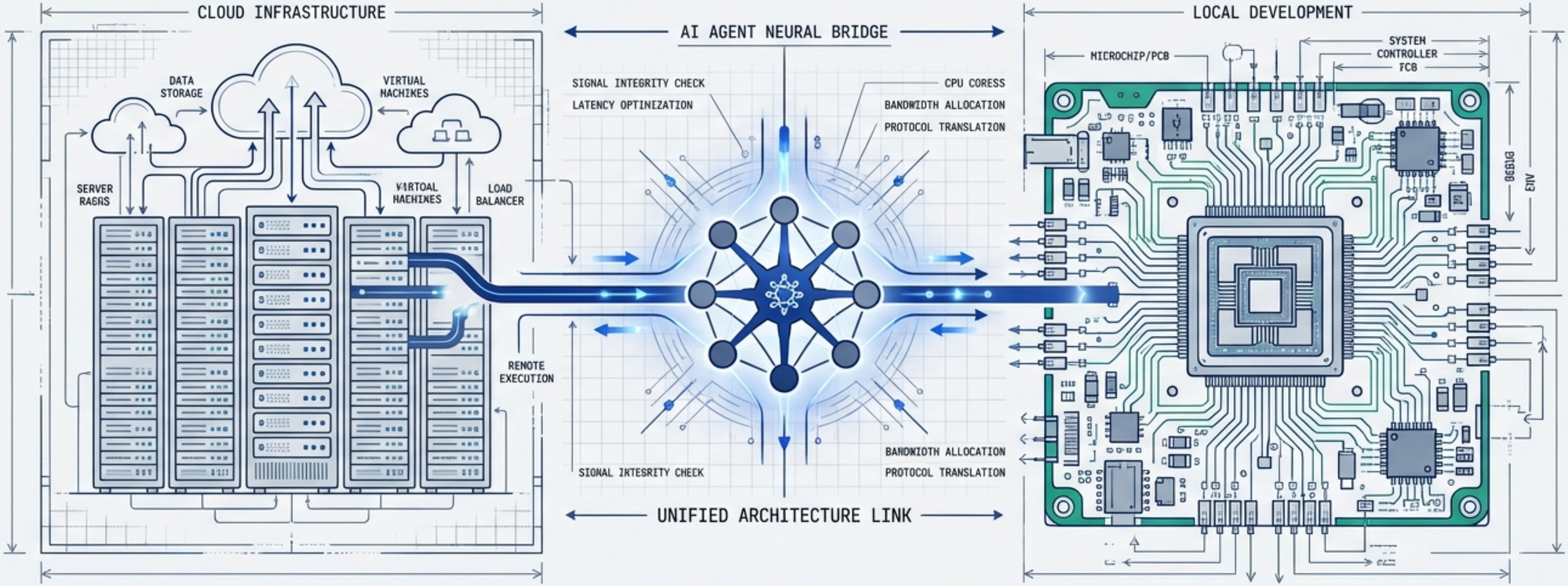
Desarrollo de Agentes de IA con Google ADK: Guía de Implementación Unificada

Estrategias de arquitectura para entornos Google Colab y Desarrollo Local

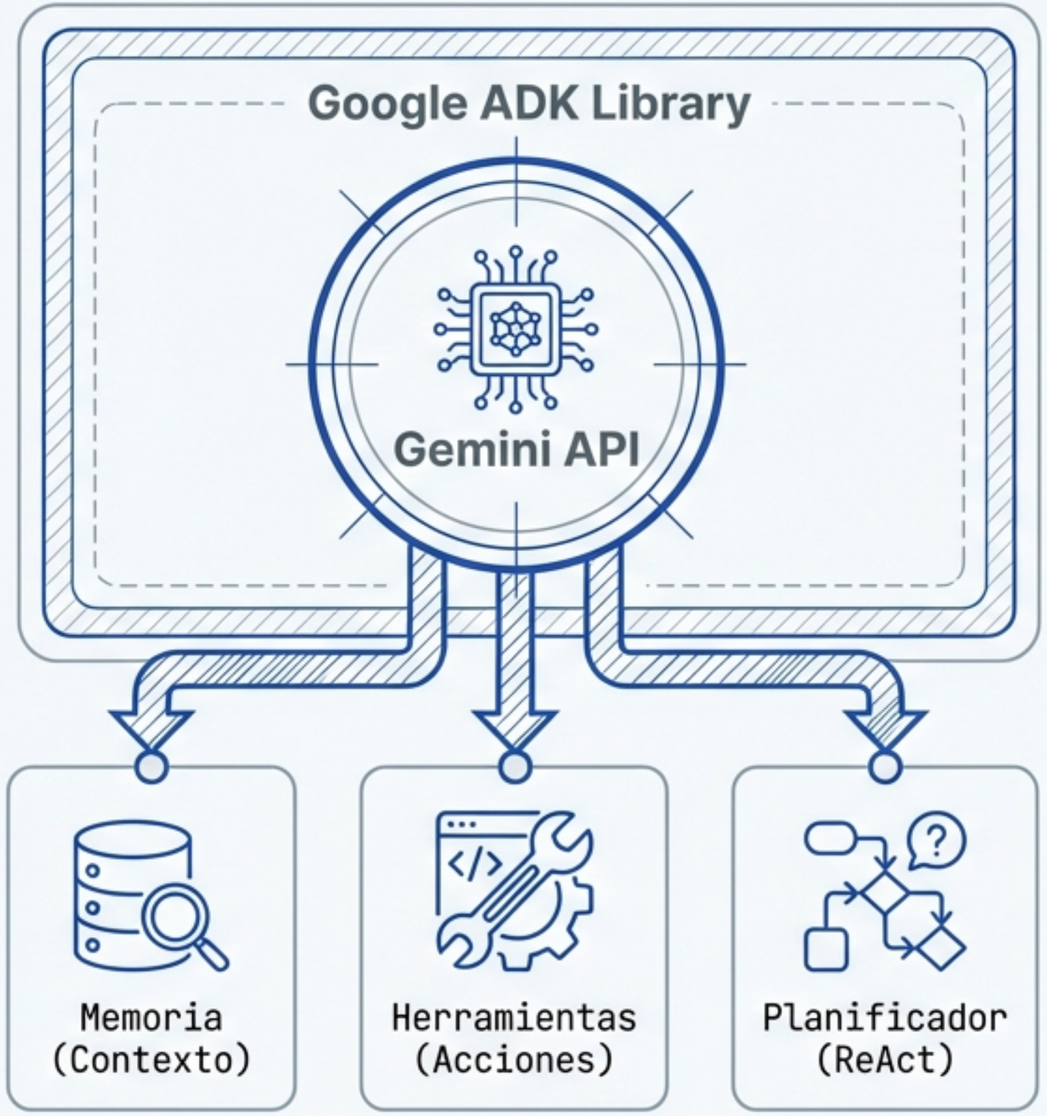


Basado en las guías técnicas oficiales 2026 para google-cloud-aiagent

NotebookLM

El Ecosistema ADK: Orquestando Inteligencia con Vertex AI y Gemini

El Google Agent Development Kit (ADK) permite construir agentes autónomos capaces de razonar, utilzar, utilizar herramientas y mantener memoria.



Modelos SOTA: Integración nativa con gemini-1.5-pro y gemini-1.5-flash.

Razonamiento: Planificadores ReAct incorporados para la toma de decisiones compleja.

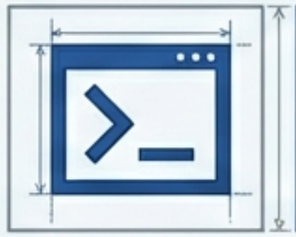
Flexibilidad: Código base unificado que escala desde notebooks experimentales hasta contenedores de producción.

NotebookLM

Selección del Entorno: Cloud Prototyping vs. Ingeniería Local

Google Colab (Ruta Rápida)

- **Hardware:** GPU NVIDIA T4/V100 recomendada.
- **Memoria:** RAM mínima 12GB (25GB+ para agentes complejos).
- **Almacenamiento:** ~80GB temporal (requiere montar Google Drive).
- **Ideal para:** Experimentación rápida, sin configuración de hardware.

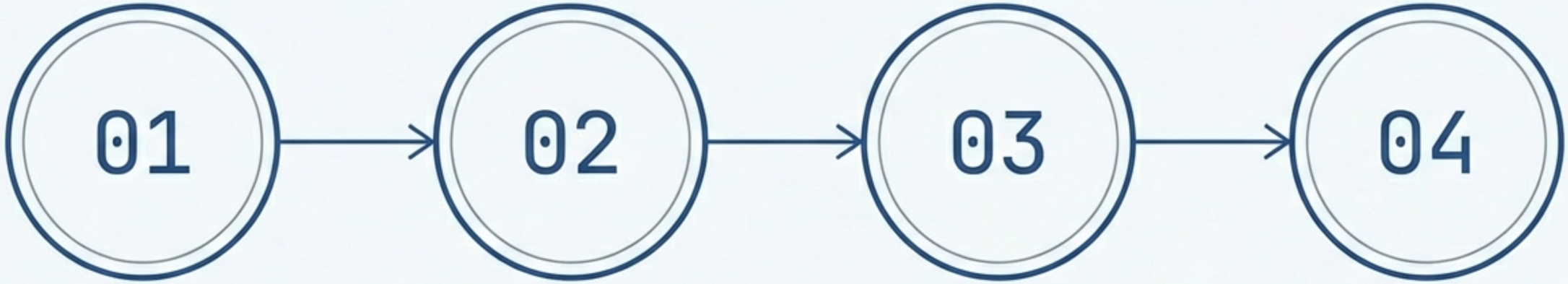


Desarrollo Local (Control Total)

- **Hardware:** CPU Intel i7/i9 11ª Gen o Ryzen 7/9. RAM 32GB recomendada.
- **Software:** Windows 11 (WSL2), Ubuntu 22.04 LTS o macOS Monterey+.
- **Almacenamiento:** SSD NVMe requerido.
- **Ideal para:** Privacidad, producción, control de latencia, integración Docker.

NotebookLM

Prerrequisito Universal: Configuración de Google Cloud Platform (GCP)



Proyecto GCP

Crear proyecto y habilitar facturación.

Habilitar APIs

Activar aiplatform.googleapis.com (Vertex AI).

Service Account

Crear cuenta con rol roles/aiplatform.user.

Llave Maestra

Descargar credenciales JSON.

Seguridad: Nunca suba el archivo JSON a repositorios públicos.

NotebookLM

Configuración del Entorno A: Google Colab

```
# 1. Validación de GPU
!nvidia-smi -L

# 2. Instalación del ADK y dependencias
!pip install --upgrade pip
!pip install google-cloud-aiagent google-cloud-aiplatform
!pip install langchain google-generativeai

# 3. Autenticación y Montaje
from google.colab import auth, drive
auth.authenticate_user()
drive.mount('/content/drive')
```

Las sesiones gratuitas expiran a las 12 horas. Es vital montar Drive para persistencia.

NotebookLM

Configuración del Entorno B: Estación de Trabajo Local

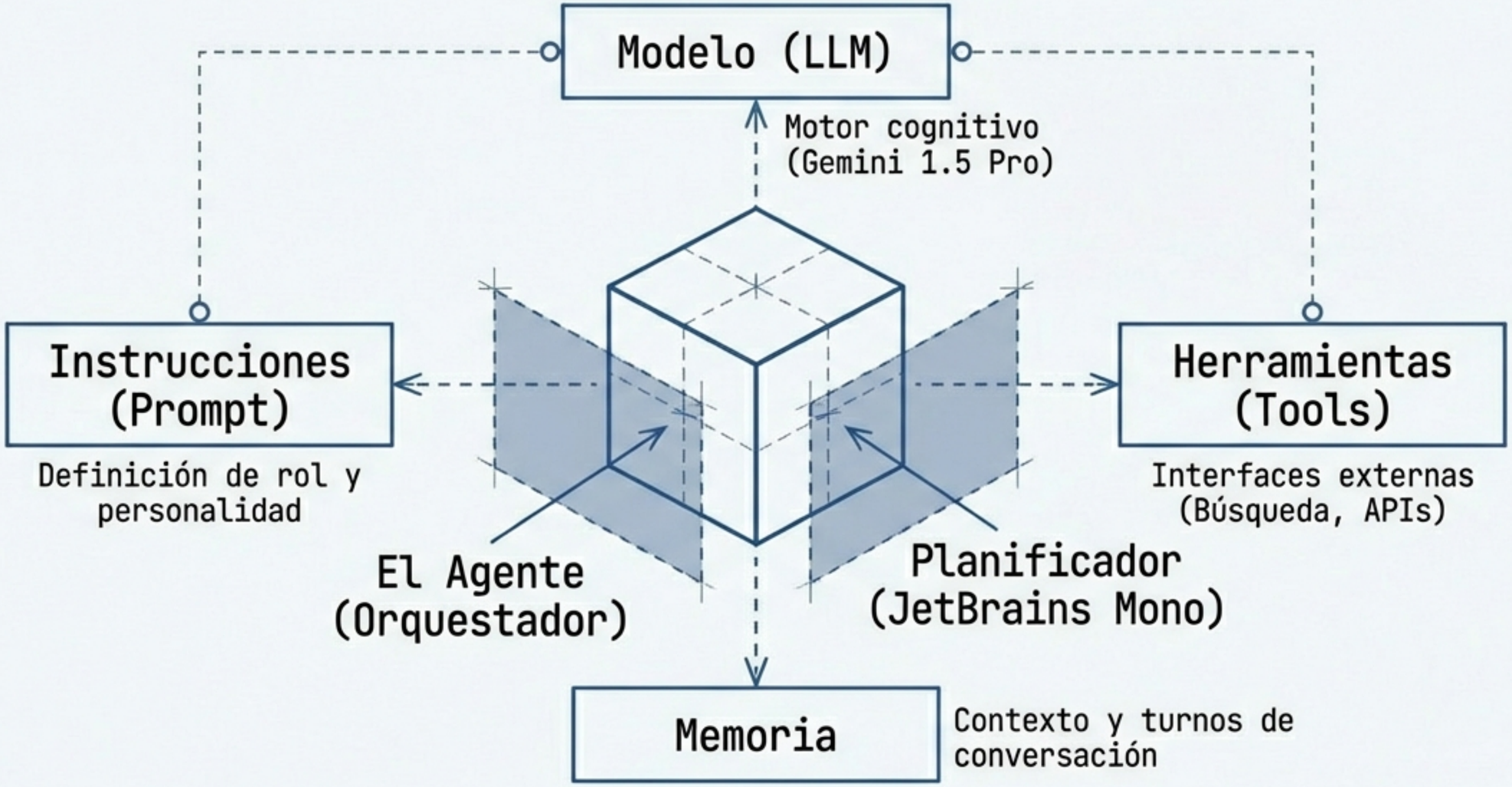
- ✓ Sistema Base: WSL2 instalado (wsl --install) si usa Windows. Ubuntu/Debian actualizado.
- ✓ Python: Versión 3.9 a 3.11 (3.12 puede tener incompatibilidades). Uso de venv.
- ✓ SDK: Instalar Google Cloud CLI y ejecutar gcloud init.

```
# Crear entorno virtual y activar
python3.11 -m venv ~/adk-env
source ~/adk-env/bin/activate

# Configurar variable de entorno crítica
export GOOGLE_APPLICATION_CREDENTIALS="~
/adk-key.json"
```

NotebookLM

Anatomía del Agente: Arquitectura de Componentes



NotebookLM

Inicialización del Núcleo del Agente

```python
from google.cloud import aiagent
from google.cloud import aiplatform

# Inicialización (Universal)
aiplatform.init(project="mi-proyecto-adk", location="us-central1")

agent = aiagent.Agent(
    name="agente-demo",
    model="gemini-1.5-pro", # Opciones: gemini-1.5-flash
    instructions="Eres un asistente especializado que responde con precisión."
)
```

Selección de Modelo

- Use **gemini-1.5-flash** para baja latencia.
- Use **gemini-1.5-pro** para razonamiento complejo.

NotebookLM

Persistencia y Contexto: Implementando Memoria

La memoria permite al agente recordar turnos anteriores para mantener el hilo de la conversación.

```
from google.cloud.aiagent import components

memory = components.ConversationMemory(
    max_turns=20,  # Límite de historial
    # Nota: Ajustar ruta según entorno (Drive vs Local)
    storage_path="./conversation_memory.json"
)
agent.add_component(memory)
```

Optimización: El parámetro max_turns gestiona la ventana de contexto para controlar los costos de tokens.

NotebookLM

Ampliando Capacidades: Integración de Herramientas

```python
# 1. Herramienta de Búsqueda Web
search_tool = components.WebSearchTool(
    api_key="TU_API_KEY",
    custom_search_engine_id="TU_CSE_ID"
)
agent.add_tool(search_tool)
```

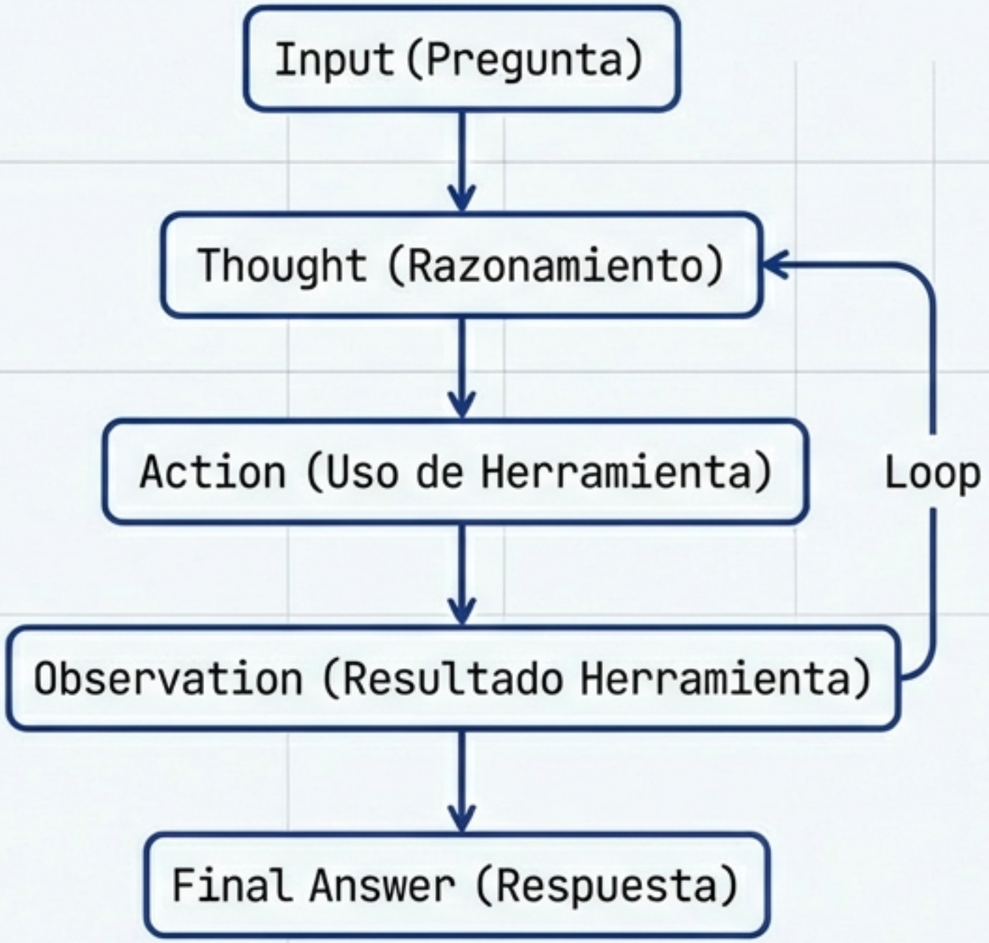
```python
# 2. Intérprete de Código (Sandbox)
code_tool = components.CodeInterpreterTool(
    safe_execution=True,
    timeout_seconds=30
)
agent.add_tool(code_tool)
```

NotebookLM

El Motor de Razonamiento: Planificador ReAct

El planificador permite al agente 'pensar' antes de actuar.

```
planner = components.ReActPlanner()
agent.add_component(planner)
```



NotebookLM

Nivel Avanzado: Creación de Herramientas Personalizadas

Transforme cualquier función Python en una habilidad del agente.

```python
def herramienta_calculadora(expresion: str) -> str:
    """Evalúa expresiones matemáticas básicas de forma segura."""
    # Lógica de cálculo...
    return f"Resultado: {result}"

# Wrapper para ADK
tool_wrapper = aiagent.Tool.from_function(
    func=herramienta_calculadora,
    name="calculadora",
    description="Realiza cálculos matemáticos"
)
agent.add_tool(tool_wrapper)
```

NotebookLM

Validación y Pruebas Unitarias

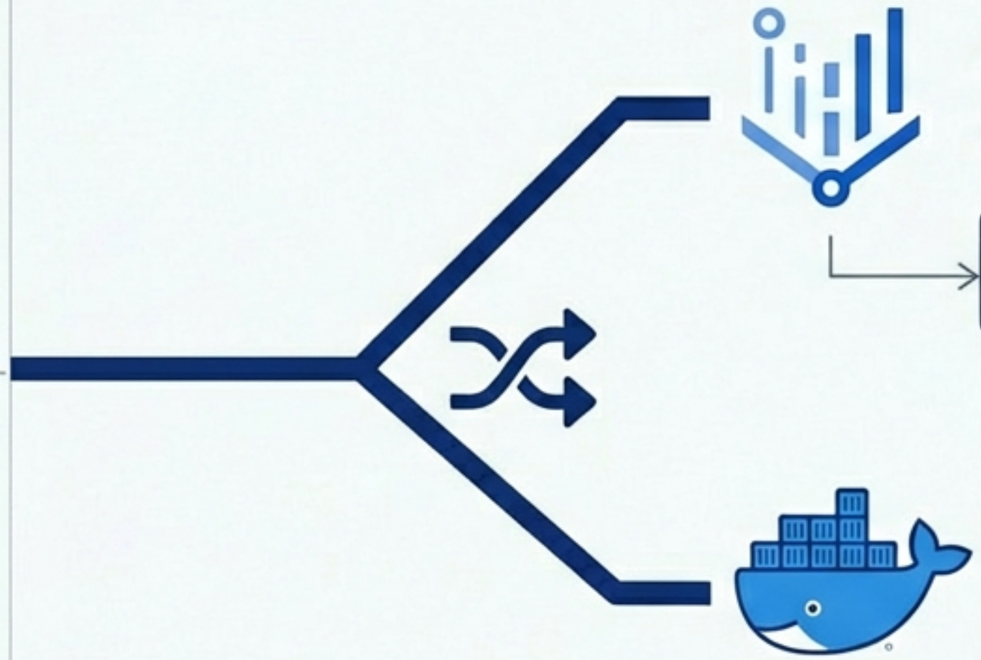
1. **Consulta Básica:** Verificar respuesta no nula (assertIsNotNone).
2. **Prueba de Memoria:** Inyectar dato y preguntar por él posteriormente.
3. **Persistencia:** Guardar estado (save), reiniciar y cargar (load).

```
response = agent.run("¿Cuál es el
precio de Bitcoin?")
print(f"Respuesta: {response.text}")

# Prueba de persistencia
agent.save("./estado_agente.json")
```

NotebookLM

Despliegue y Estrategias de Producción



Opción A: Vertex AI (Nube)

Endpoint gestionado, escalado automático.

```
agent.export_to_vertex(display_name="MiAgente")
```

Opción B: Contenedor Docker (Local)

Portabilidad total. Inyectar credenciales vía ENV.

```
FROM python:3.11-slim
```

Monitoreo: Configurar alertas de latencia (p99) y tasa de errores.

NotebookLM

Solución de Problemas y Mejores Prácticas

Problema	Causa Probable	Solución
Error Auth	API inhabilitada / JSON perdido	Verificar console.cloud.google.com y ruta.
Timeout	Respuesta lenta de herramienta	Reiniciar entorno (`!kill -9 -1`) o subir timeout.
Cuotas	Modelo Pro en capa gratuita	Solicitar aumento de cuota en GCP.

Mantenimiento: Revocar credenciales no usadas y actualizar librerías regularmente.

El desarrollo local ofrece control; la nube ofrece velocidad. El ADK le permite dominar ambos.

NotebookLM