



De prompts sueltos a agentes

- Hasta ahora vimos:
 - Transformers, atención, fine-tuning
 - Prompting basico y RA
- Limitación: todo eso suele ser **una llamada aislada** al modelo
- En producción necesitamos:
 - Múltiples pasos
 - Estado / memoria
 - Llamar APIs, BDs, RAG, etc.
 - Controlar errores, reintentos, límites

¿Qué es un agente?

Un agente es un sistema que usa un LLM para **tomar decisiones paso a paso**, llamar herramientas y **mantener estado** para cumplir un objetivo del usuario.

Componentes típicos:

- Objetivo o tarea (goal)
- Contexto / memoria
- Modelo de lenguaje (LLM)
- Tools (acciones externas)
- Ciclo de razonamiento:
 - Pensar → decidir → actuar → observar → repetir

Comparación:

- Prompt “clásico”: función pura `input → output`
- Agente: **bucle** `estado + input → nuevo estado + acciones + output`

¿Por qué agentes?

Casos donde un sólo prompt no alcanza:

- Flujos multi-paso:
 - “Reservá un vuelo, verificá clima, proponé 2 opciones y fijate mi presupuesto”
- Integraciones complejas:
 - CRM + base de tickets + documentación interna
- Ajuste incremental:
 - El usuario corrige, agrega contexto, cambia de idea
- Tareas largas:
 - Investigar, resumir, proponer plan, iterar sobre feedback

Beneficios:

- Automatizan workflows, no sólo respuestas.
- Hacen explícito el flujo de decisiones.

Anatomía de un agente LLM

Usuario → (Mensaje) → **Agente** → (Respuestas + acciones)

Dentro del agente:

- **State:**

- Historial de mensajes
- Memoria (resumen, entidades, preferencias)
- Variables de trabajo (ej. “ticket_id”, “carrito_compra”)

- **Reasoner (LLM):**

- Interpreta el estado actual
- Decide próxima acción (tool a llamar / respuesta / pregunta)

- **Tools:**

- Funciones externas (APIs, RAG, bases de datos, etc.)

- **Policies:**

- Reglas de seguridad, límites, fallback

¿Qué es una “tool”?

Tool = acción externa que el LLM puede pedir ejecutar.

Ejemplos:

- buscar_productos(query)
- consultar_stock(product_id)
- run_sql(query_sql)
- rag_search(question)
- enviar_correo(destinatario, asunto, cuerpo)
- programar_reunion(fecha, asistentes)

Características:

- Tienen **input bien tipado**
- Tienen **output estructurado**
- Hacen algo **fuera del LLM** (datos frescos, efectos en el mundo real)

Diseño de tools

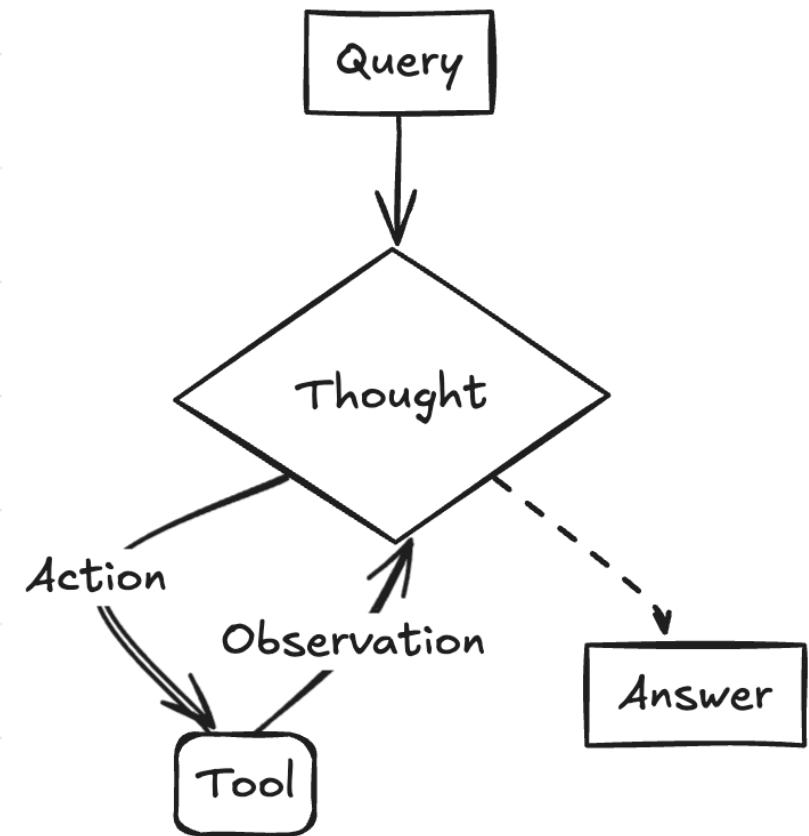
Buenas prácticas:

- Inputs y outputs **explícitos** (tipos, campos, unidades)
- Hacer tools **pequeñas y composable**s:
 - Mejor 3 tools simples que 1 mega-tool confusa
- Pensar en **idempotencia**:
 - ¿Qué pasa si el LLM la llama dos veces?
- Manejo de errores:
 - Mensajes claros de “no encontré X”, “timeout”, etc.
- Limitar poder:
 - Tools peligrosas (borrar datos, enviar dinero) con más checks / sandbox

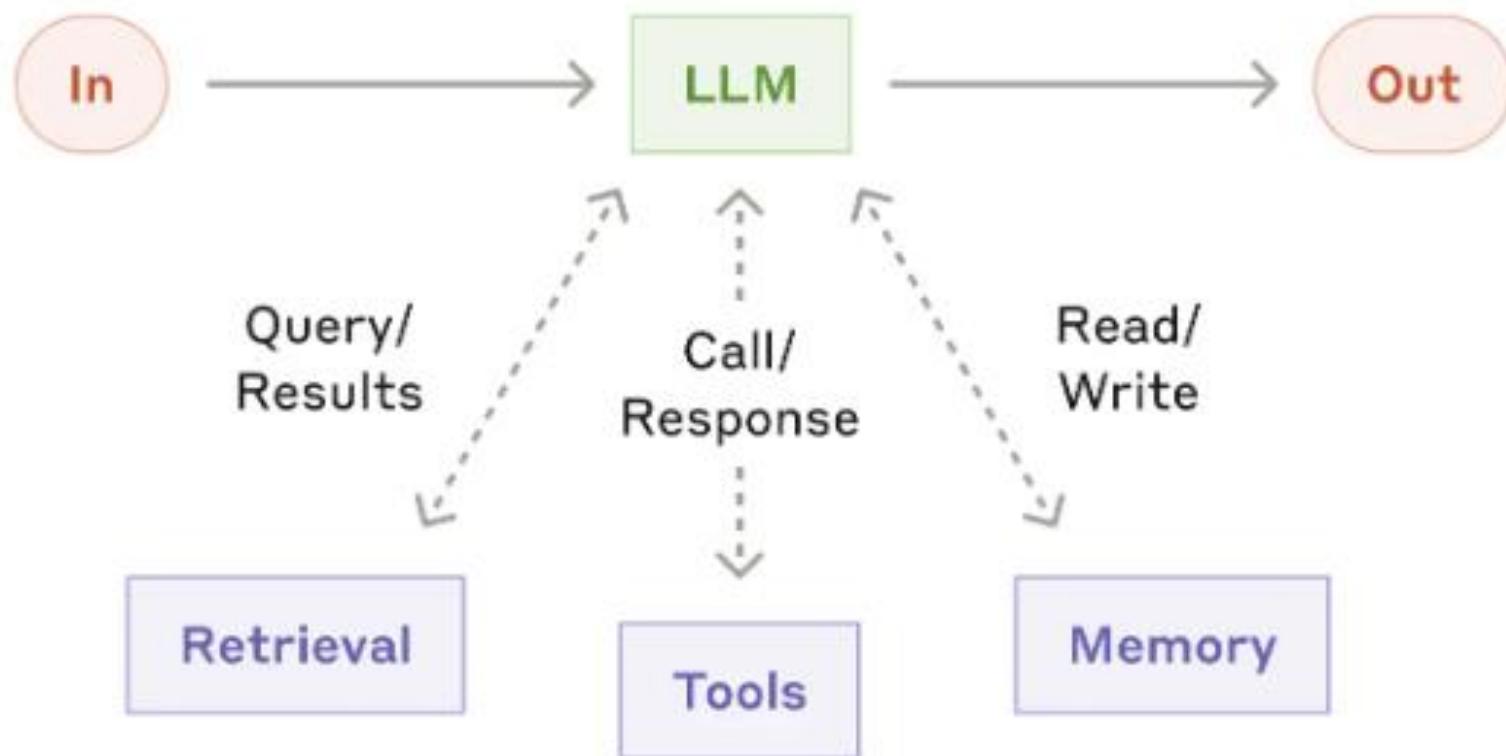
LLM + tools: ciclo de interacción

Patrón típico:

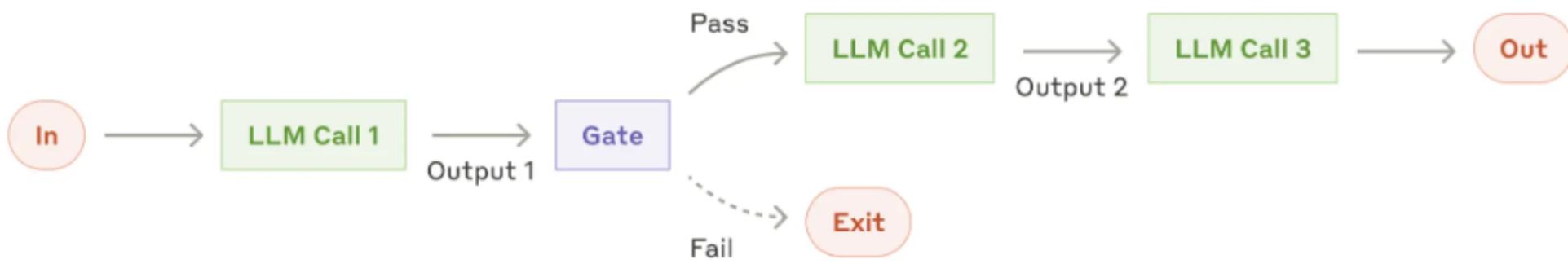
- Usuario manda mensaje
- Agente arma un “prompt de pensamiento”:
 - Historial relevante
 - Estado actual
 - Descripción de las tools disponibles
- El LLM decide:
 - Responder directamente
 - O pedir usar una tool con ciertos argumentos
- El runtime ejecuta la tool
- El resultado vuelve al LLM
- El LLM genera la respuesta final (o pide otra tool)
- Se actualiza el estado / memoria



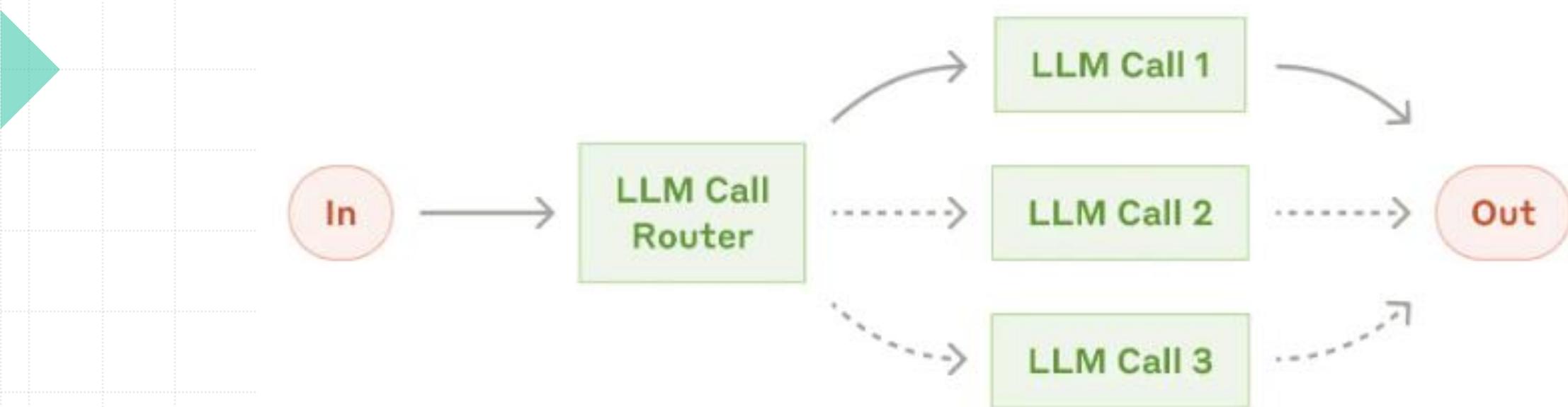
Patrones comunes de agentes – Basic LLM



Patrones comunes de agentes – Prompt chaining



Patrones comunes de agentes – Routing



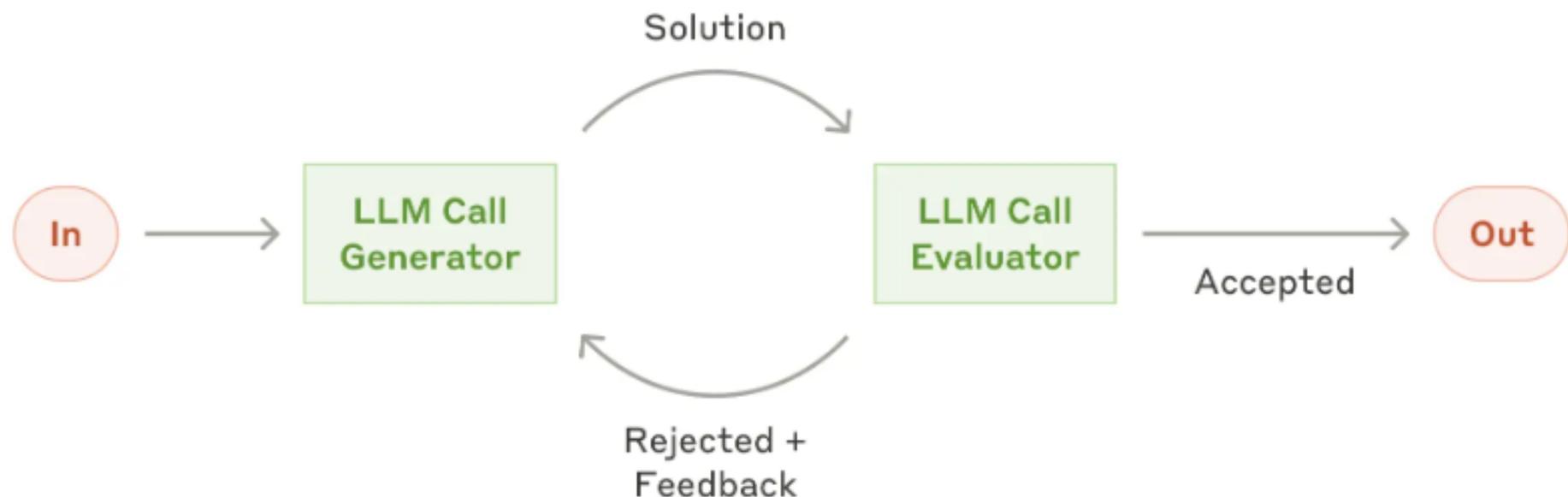
Patrones comunes de agentes – Parallelization



Patrones comunes de agentes – Orchestrator



Patrones comunes de agentes – Evaluator



La pila de protocolos

Contexto actual:

- Grandes players (Google, Anthropic, OpenAI, Microsoft...) están empujando **estándares** para:
 - Conectar agentes a datos y herramientas
 - Hacer que agentes **hablen entre sí**
 - Conectar agentes con **interfaces de usuario** ricas y en tiempo real

Tres capas clave que vamos a mirar:

- **MCP** – Model Context Protocol → agentes ↔ datos / herramientas
- **A2A** – Agent2Agent Protocol → agentes ↔ agentes
- **AG-UI** – Agent–User Interaction Protocol → agentes ↔ frontends

MCP – Model Context Protocol

¿Qué es MCP?

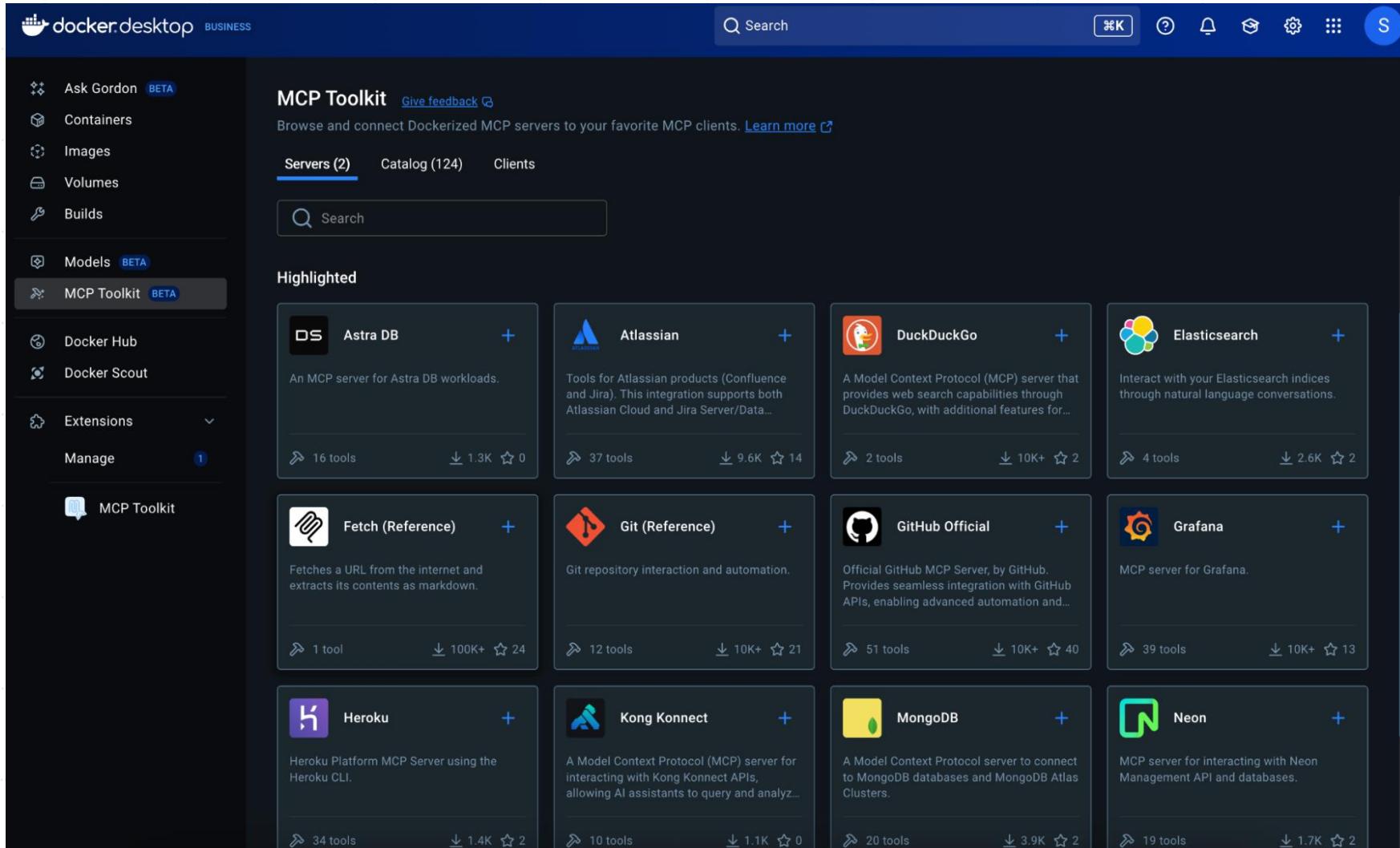
Estándar abierto para conectar aplicaciones de IA con fuentes de datos y herramientas externas.

- Nace del ecosistema Anthropic / Claude, pero es abierto.
- Un puerto estándar para enchufar:
 - BDs, APIs, sistemas internos
 - Herramientas (search, cálculos, workflows)
 - Repositorios de prompts / workflows

Conceptos clave:

- **MCP server:** expone tools y recursos (archivos, queries, etc.)
- **MCP client:** tu agente / aplicación que consume esos tools
- Tools tienen:
 - Nombre único
 - Esquema de inputs/outputs tipado
 - Metadata de permisos y contexto

MCP – Model Context Protocol



The screenshot shows the Docker Desktop MCP Toolkit interface. The left sidebar includes options for Ask Gordon (BETA), Containers, Images, Volumes, Builds, Models (BETA), MCP Toolkit (BETA), Docker Hub, Docker Scout, Extensions, and Manage. The MCP Toolkit section is currently selected. The main area displays a grid of MCP servers and tools, each with a title, icon, description, and metrics (number of tools, star rating, and last update). The servers listed are Astra DB, Atlassian, DuckDuckGo, Elasticsearch, Fetch (Reference), Git (Reference), GitHub Official, Grafana, Heroku, Kong Konnect, MongoDB, and Neon.

Server/Tool	Description	Tools	Last Update	Rating
Astra DB	An MCP server for Astra DB workloads.	16	1.3K	0
Atlassian	Tools for Atlassian products (Confluence and Jira). This integration supports both Atlassian Cloud and Jira Server/Data...	37	9.6K	14
DuckDuckGo	A Model Context Protocol (MCP) server that provides web search capabilities through DuckDuckGo, with additional features for...	2	10K+	2
Elasticsearch	Interact with your Elasticsearch indices through natural language conversations.	4	2.6K	2
Fetch (Reference)	Fetches a URL from the internet and extracts its contents as markdown.	1	100+	24
Git (Reference)	Git repository interaction and automation.	12	10K+	21
GitHub Official	Official GitHub MCP Server, by GitHub. Provides seamless integration with GitHub APIs, enabling advanced automation and...	51	10K+	40
Grafana	MCP server for Grafana.	39	10K+	13
Heroku	Heroku Platform MCP Server using the Heroku CLI.	34	1.4K	2
Kong Konnect	A Model Context Protocol (MCP) server for interacting with Kong Konnect APIs, allowing AI assistants to query and analyze...	10	1.1K	0
MongoDB	A Model Context Protocol server to connect to MongoDB databases and MongoDB Atlas Clusters.	20	3.9K	2
Neon	MCP server for interacting with Neon Management API and databases.	19	1.7K	2

A2A – Agent2Agent Protocol

¿Qué es A2A?

Protocolo abierto para que agentes de distintos vendors y frameworks puedan comunicarse y coordinarse entre sí.

- Originalmente impulsado por Google y donado a la Linux Foundation.
- Objetivo: que un agente de:
 - Google / Vertex,
 - OpenAI / LangGraph,
 - Vendor X en otra nube
 - puedan hablar un mismo “idioma” estructurado.

¿Qué resuelve?

- Descubrimiento de agentes (“qué sabes hacer vos?”)
- Mensajes estructurados (intents, respuestas, errores)
- Seguridad / autenticación entre organizaciones
- Coordinación:
 - Agente “usuario” ↔ agente “comercio”
 - Agente “soporte” ↔ agente “facturación”

AG-UI – Agent–User Interaction Protocol

¿Qué es AG-UI?

Protocolo abierto y ligero para estandarizar cómo los agentes se conectan con aplicaciones de usuario (web, mobile, etc.).

- Se centra en el **canal tiempo real** entre:
 - Backend agéntico (LangGraph, Agent Framework, etc.)
 - Frontend (React, Next.js, móviles...)

Qué define AG-UI:

- Eventos estándar:
 - Mensajes de chat, streaming de tokens
 - Tool calls (y su render en UI)
 - Actualización de estado compartido (p. ej. un panel, formulario)
- Threads / sesiones desde el lado UI
- Patrones para:
 - Human-in-the-loop
 - Mostrar planes, pasos, errores.

Cómo encajan MCP, A2A y AG-UI

Capa UI (User ↔ Agente)

- AG-UI: eventos, estado compartido, streaming
- Frameworks: React + CopilotKit, Agent Framework, etc.

Capa Agente / Orquestación

- LangGraph, LangChain, Vertex ADK, OpenAI Agents SDK
- Definen:
 - Estado del agente
 - Tools internas (RAG, BDs, APIs)
 - Memoria, seguridad, routing

Capa Conectores / Datos / Otros agentes

- MCP: conectar con datos y tools externos estándar
- A2A: hablar con otros agentes, internos o de terceros
- Otros protocolos emergentes (pagos AP2, credenciales, etc.)

