

The background of the slide is a grayscale image of a hand holding a tablet. The tablet screen displays a dashboard with a world map, several circular progress indicators, and a bar chart. A large red graphic element, consisting of a thick, curved line, frames the left and top portions of the text area.

De copiloto a Operador

Cómo elegir el nivel de autonomía de un agente de IA Cuándo supervisar, cuándo aprobar y cuándo dejar que un agente



Resumen: la autonomía no es del agente, es de cada acción

Elegir la autonomía de un agente de IA no es un problema del modelo, sino de gobernanza y riesgo operacional. La clave es dejar de preguntar **"¿qué tan autónomo es el agente?"** y **empezar a preguntar "¿qué puede hacer cada acción por sí sola, y bajo qué condiciones?"**. Durante los últimos años, los sistemas de inteligencia artificial pasaron de generar texto a ejecutar tareas de varios pasos: consultan información, usan herramientas, ejecutan código y modifican sistemas empresariales sin intervención humana constante. Con ese salto, el problema dejó de ser si una organización puede construir un agente capaz de ejecutar una tarea, y pasó a ser qué debería permitirle hacer sin pedir autorización.

Un mismo agente puede consultar el inventario, modificar una orden, emitir un reembolso o eliminar información. Todas son acciones técnicamente posibles, pero no comparten ni el riesgo ni la reversibilidad, así que no deberían compartir el mismo nivel de autonomía. Para gobernar esa diferencia, las organizaciones recurren a cuatro modelos de control: ***copiloto, human-in-the-loop, human-on-the-loop y autonomía elevada***. **A partir de sus ventajas y límites este artículo**

Propone una arquitectura de autonomía graduada por acción, donde los permisos dependen del riesgo, el impacto, la reversibilidad y la sensibilidad de cada acción, no de quién es el agente.



La recomendación central: no asignes un nivel de autonomía al agente. Asigna un nivel de autonomía a cada acción que el agente puede ejecutar.

Contexto: los agentes dejaron de recomendar y Empezaron a Ejecutar

Este artículo está dirigido a **arquitectos de soluciones, líderes de plataforma e ingenieros de backend** que ya tienen, o están por dar, acceso de escritura a un agente de IA sobre al menos un sistema real como un CRM, pedidos, pasarelas de pago o infraestructura cloud, y que necesitan decidir qué acciones puede ejecutar de forma autónoma y cuáles requieren aprobación humana. Se asume familiaridad básica con agentes de IA.

Por qué importa ahora. La duración de las sesiones autónomas de agentes ya crece de forma medible, y al mismo tiempo aplicar un mismo nivel de gobernanza a todos los agentes, sin diferenciar por el riesgo de cada acción, se ha vuelto una causa directa de fallas empresariales. La ventana para diseñar bien este control es antes de escalar el agente a producción, no después de un incidente. Vale la pena aterrizarlo con un escenario que dentro de una empresa tecnológica ya no resulta inusual. Son las 2:13 de la madrugada y un agente de operaciones detecta que la entrega de un cliente importante está retrasada. Consulta el sistema logístico, revisa el estado del pedido y encuentra una alternativa de transporte. Hasta ahí nadie objetaría nada: el agente hace justo aquello para lo que fue diseñado: observar, razonar y buscar una solución. El problema aparece en el siguiente paso: para resolverlo necesita modificar la orden de transporte, y resulta que tiene la API y el permiso para hacerlo. Entonces surge la pregunta que de verdad importa:

¿Debe hacerlo?

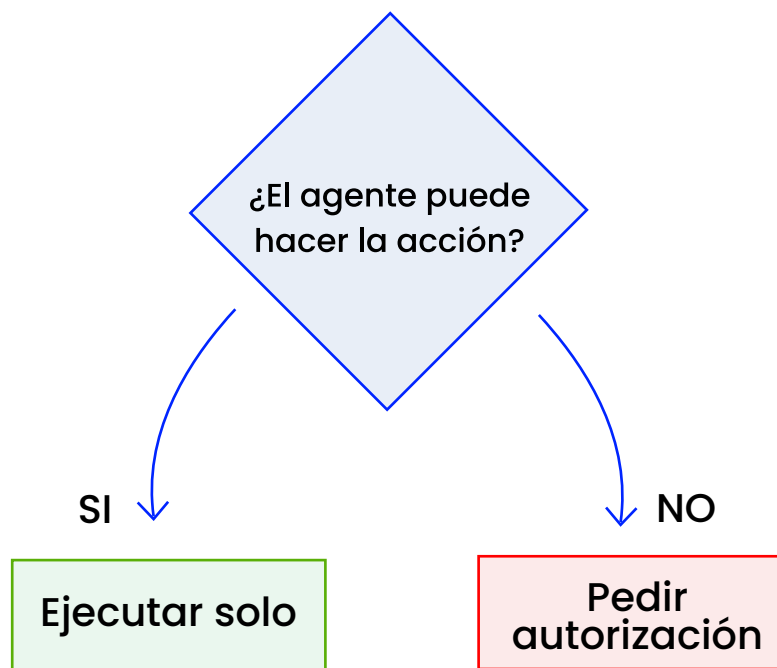
Y si la respuesta es sí, aparece otra más difícil: **¿debería poder hacerlo siempre? ¿Y si el cambio cuesta 20 dólares, o cuesta 2.000? ¿Y si el transportista no está aprobado, o la operación es irreversible? ¿Y si el agente acierta el 99 % de las veces, pero ese 1 % restante son precisamente los casos más difíciles?**

Ese escenario resume uno de los problemas más interesantes de los sistemas agénticos en 2026: los agentes están pasando de recomendar acciones a ejecutarlas. Anthropic documentó que, entre octubre de 2025 y enero de 2026, la duración correspondiente al percentil 99,9 de determinadas sesiones de Claude Code pasó de menos de 25 minutos a más de 45 minutos, una evolución desde interacciones breves hacia tareas en las que el sistema trabaja durante periodos mucho mayores. Y ocurre en plena ola de adopción: **según el Hype Cycle 2026 de Gartner para IA agéntica, solo el 17 %** de las organizaciones ha desplegado agentes hasta ahora, pero más del 60 % espera hacerlo en los próximos dos años, la curva de adopción más agresiva entre todas las tecnologías emergentes de ese informe.

El mismo Gartner advierte el reverso: aplicar una gobernanza uniforme a todos los agentes puede provocar fallos empresariales, precisamente porque no todos tienen el mismo nivel de autonomía ni el mismo alcance de acceso. De hecho proyecta que para 2027 el 40 % de las empresas terminará degradando o retirando agentes autónomos por brechas de gobernanza .

El problema: por qué el enfoque de todo o nada se queda corto

La aproximación más simple a la seguridad de un agente divide el mundo en dos: o el agente ejecuta solo, o pide autorización para todo.



El problema: por qué el enfoque de todo o nada se queda corto a un sistema de gestión de pedidos y que, a través de la misma API, puede consultar el estado de un pedido, crear un borrador, cambiar una dirección, modificar una fecha de entrega, cancelar una orden, emitir un reembolso o eliminar registros. Técnicamente todas están disponibles por igual, pero desde el punto de vista del riesgo son operaciones completamente distintas:

<i>Acción</i>	<i>Impacto potencial</i>	<i>Reversibilidad</i>	<i>Nivel sugerido</i>
Consultar inventario	Bajo	Alta	Autónomo
Crear un borrador	Bajo	Alta	Autónomo
Enviar una comunicación	Medio	Baja	Supervisado
Cambiar una orden	Medio	Parcial	Supervisado
Emitir un reembolso	Alto	Baja	Requiere aprobación
Transferir dinero	Crítico	Baja	Requiere aprobación
Eliminar información	Crítico	Variable	Requiere aprobación

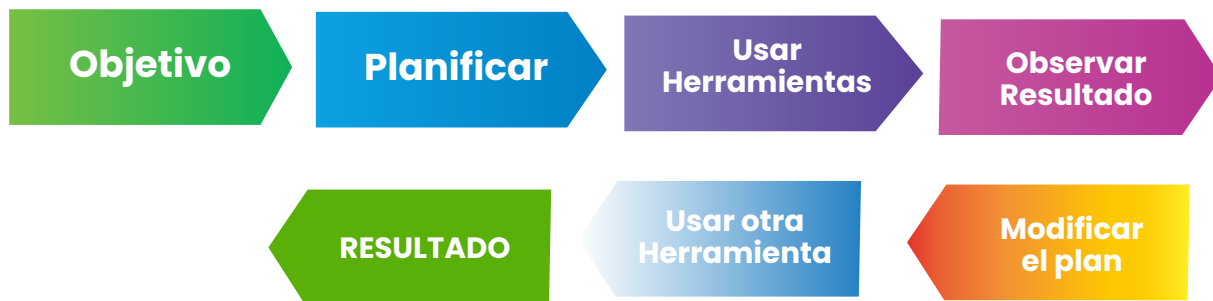
Aquí aparece el error de fondo: el nivel de autonomía no es un atributo del agente, sino de la combinación **agente + acción + contexto + riesgo**. Un mismo agente puede ser lo bastante confiable para consultar datos y, a la vez, no estar autorizado para modificar información financiera. Esto importa porque la arquitectura de los sistemas de IA cambió. El modelo tradicional era una consulta y una respuesta:



Después aparecieron los copilotos, que añaden una recomendación que un humano revisa antes de actuar:



Y los agentes introdujeron un ciclo en el que el sistema planifica, usa herramientas, observa el resultado, ajusta el plan y vuelve a actuar.



AWS define precisamente a los agentes como sistemas capaces de interactuar con su entorno, recopilar información y ejecutar tareas autodirigidas para alcanzar objetivos. Ese cambio redefine el problema de seguridad: mientras la salida del modelo es solo texto, sus consecuencias son limitadas; pero cuando esa salida se convierte en una transacción, un cambio en una base de datos, un correo a un cliente, una configuración modificada, un pedido cancelado, un archivo eliminado o **una acción** sobre infraestructura, deja de ser información y se convierte en una acción. Una revisión crítica publicada en septiembre de 2026 (Zhu y Cai)

señala esa misma distinción: en la evidencia examinada, la expansión de las interfaces de acción de los agentes está mucho mejor documentada que la finalización robusta de tareas, la recuperación ante fallos, la autorización o la verificación independiente. Los autores proponen el concepto de **"delegación justificada"**: solo expandir el alcance de acción de un agente donde exista evidencia de procedencia, autoridad acotada, detección de fallos, recuperación segura y control humano calibrado. En otras palabras, en 2026 la autonomía se parece menos a una característica del modelo y más a una decisión de arquitectura y control.

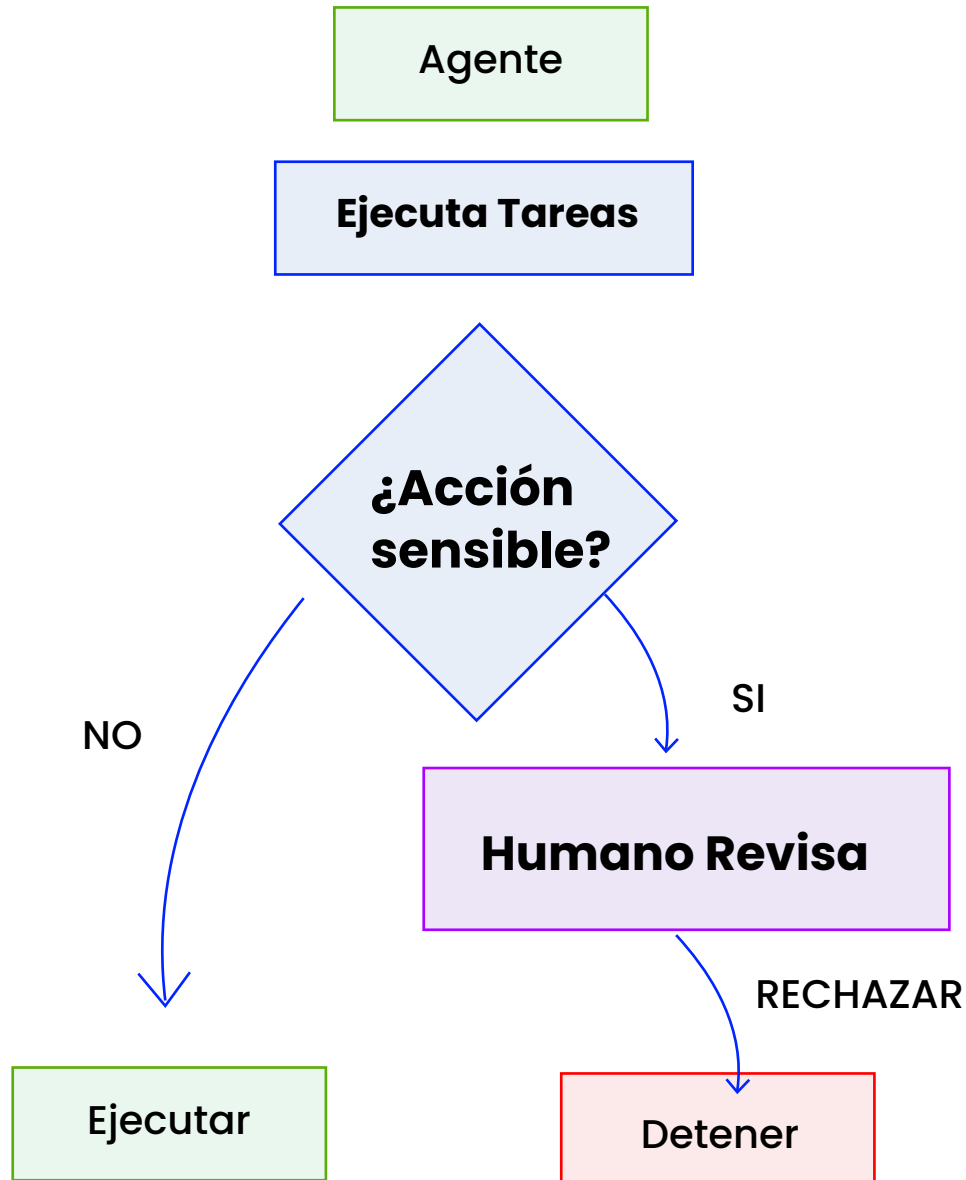
Análisis de soluciones: cuatro formas de repartir el control

Podemos ordenar la evolución de un agente en cuatro niveles prácticos, del más conservador al más autónomo. No son escalones que haya que subir todos, sino opciones que conviene entender antes de elegir.

Copiloto: el agente recomienda. El agente analiza la situación y propone una acción, pero no la ejecuta. Retomando el caso del pedido retrasado de la apertura, aquí el agente diría algo como "Recomiendo cambiar el transportista", y ahí se detiene. Su ventaja es el bajo riesgo y que el humano conserva el control, lo que lo hace ideal para procesos nuevos o poco conocidos; su desventaja es que exige mucha intervención humana y no escala si cada decisión necesita una persona. Es especialmente útil cuando todavía no conocemos bien el Análisis de soluciones: cuatro formas de repartir el control comportamiento del agente: sirve para recopilar evidencia antes de darle permisos para actuar.



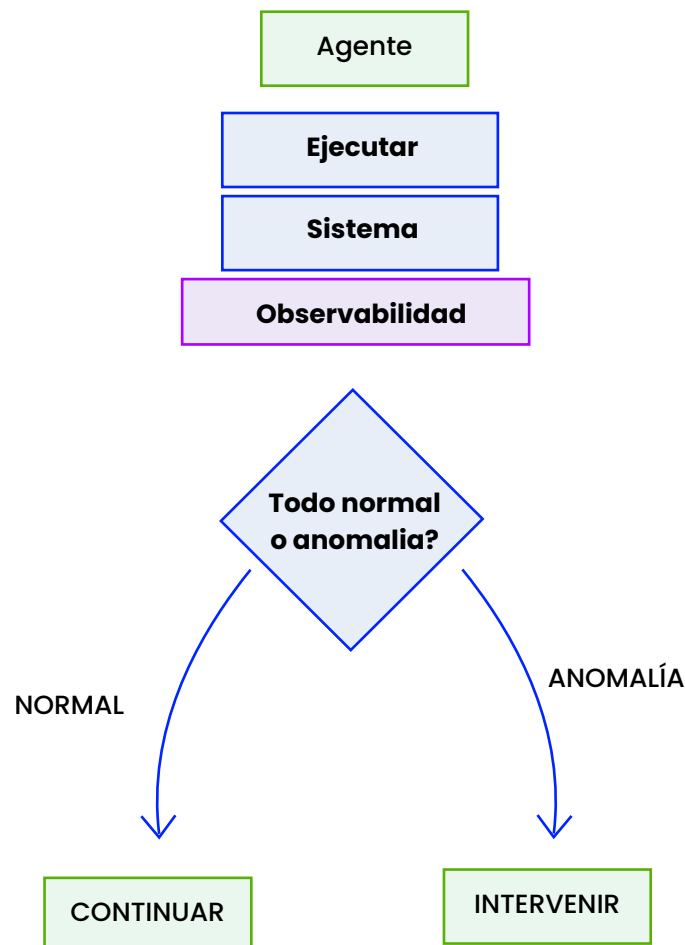
Human-in-the-loop: el humano aprueba las acciones relevantes. El agente avanza por sí mismo hasta llegar a una acción que necesita aprobación.



Es probablemente el modelo más intuitivo para entornos empresariales: el agente investiga, consulta sistemas, prepara información, calcula opciones y ejecuta acciones de bajo riesgo, mientras una política define qué requiere intervención humana. AWS documenta estos patrones como puntos de control humanos dentro de los workflows de agentes, con mecanismos que varían según la plataforma **(confirmación de usuario y return-of-control en Amazon Bedrock Agents, o callbacks de Step Functions para agentes orquestados con máquinas de estado)**. Reduce el riesgo de acciones críticas y mantiene al humano como autoridad final, lo que lo hace adecuado para operaciones financieras, legales o sensibles.

Su punto débil es la escalabilidad. Si el agente necesita aprobación para cada acción, la automatización se convierte en una cola de trabajo, y aparece un problema más sutil: la **fatiga de aprobación**. Cuando una persona recibe cientos de solicitudes casi idénticas ("¿Autorizar acción?" Aprobar "¿Autorizar acción?" Aprobar...), termina aprobando por costumbre y pierde el criterio. El humano sigue en el diagrama, pero ya no ejerce una supervisión significativa. Por eso poner un humano en el loop no garantiza seguridad por sí solo: una

investigación de agosto de 2026 (Mitchell, Ghosh y Passi) argumenta que los sistemas actuales pueden dificultar la supervisión humana efectiva y, con el tiempo, desgastar las habilidades que las personas necesitan para vigilar sistemas tan automatizados: si el humano deja de ejercitar su criterio, va perdiendo la capacidad de detectar un error cuando de verdad importa. **Human-on-the-loop: el humano supervisa, pero no aprueba todo. Aquí cambia la relación:**



Es mucho más escalable, pero exige una infraestructura de observabilidad más madura: si el humano no observa bien el sistema, la supervisión deja de servir. Autonomía elevada: el agente opera solo. En el extremo de la escala, el usuario da un objetivo y el agente decide cómo alcanzarlo. Es atractivo para tareas repetitivas, de bajo riesgo, altamente observables, reversibles y con criterios de éxito claros (por ejemplo: "Analiza los logs de las últimas 24 horas, identifica errores recurrentes y genera un informe"). El riesgo aparece cuando confundimos "puede completar la tarea" con "puede hacer cualquier cosa necesaria para completarla"; son dos afirmaciones muy distintas.



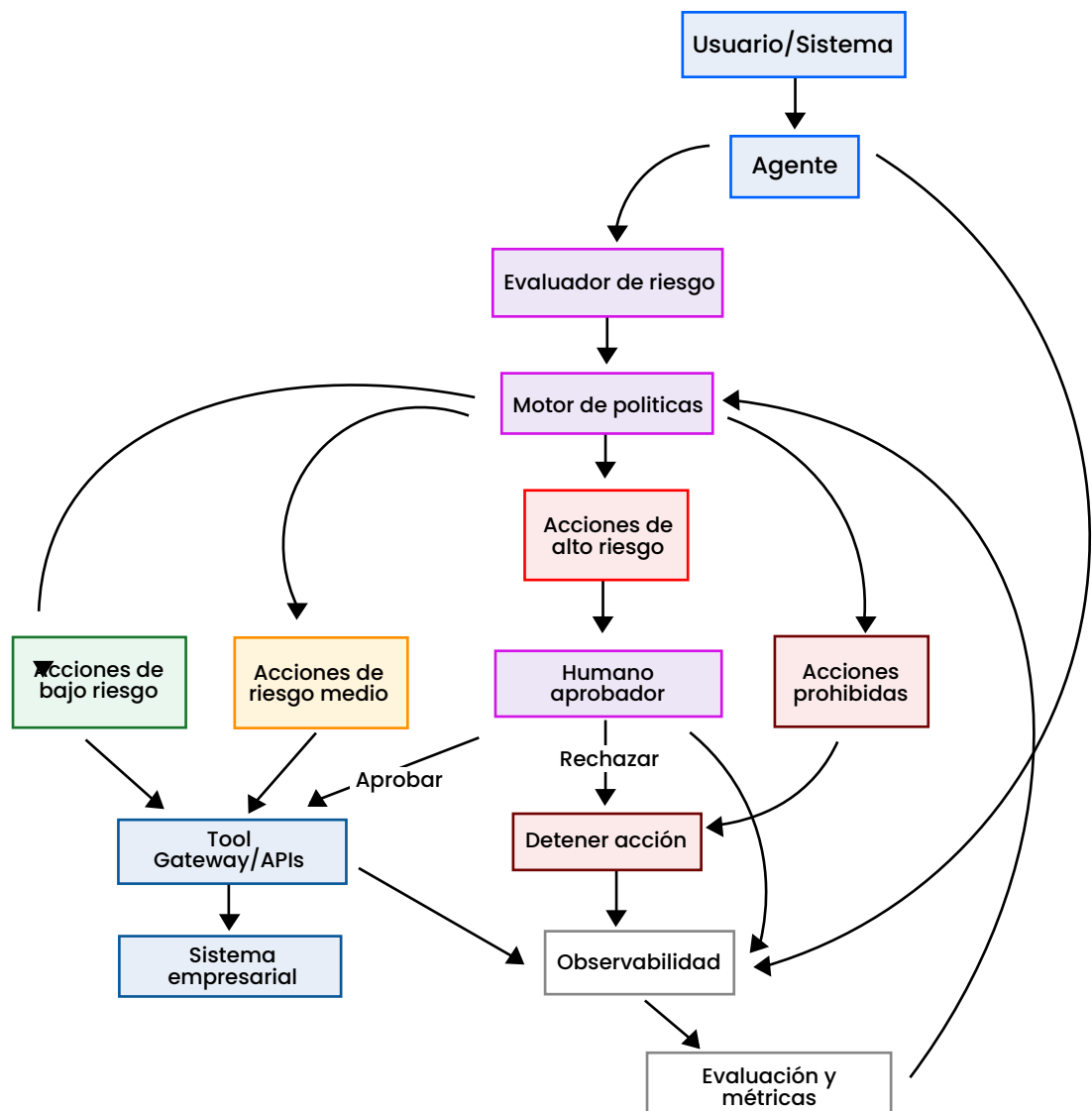
Comparando las cuatro alternativas

MODELO	AUTONOMÍA	RIESGO GLOBAL	COSTO HUMANO	ESCALABILIDAD	CASO IDEAL
Copiloto	Baja	Baja	Alto	Baja	Tareas nuevas
Human-in-the-loop	Media	Bajo/Media	Medio	Media	Acciones sensibles
Human-on-the-loop	Alta	Medio	Bajo	Alta	Operaciones maduras
Alta autonomía	Muy Alta	Alto	Muy bajo	Muy Alta	Tareas de bajo riesgo

Solución recomendada: autonomía graduada por acción

Combinar los cuatro modelos es la respuesta. En vez de etiquetar al agente entero como "autónomo" o "dependiente de aprobación", el permiso se resuelve operación por operación:

cada acción lleva su propia política de autonomía. AWS ya usa explícitamente el término graduated autonomy: aumentar o reducir los permisos de un agente según su comportamiento y confianza demostrada, en lugar de otorgarle acceso uniforme.



La idea de fondo es que cada acción pasa por un filtro externo antes de ejecutarse: es ese filtro, y no el propio agente, el que decide cuánto puede hacer según el riesgo de lo que intenta. No hace falta un algoritmo matemático sofisticado. Se puede empezar con cuatro dimensiones: Impacto: ¿qué pasa si la acción falla? Un error al crear un borrador tiene poco impacto;

Cómo calcular el riesgo

No hace falta un algoritmo matemático sofisticado. Se puede empezar con cuatro dimensiones:

Impacto: ¿qué pasa si la acción falla? Un error al crear un borrador tiene poco impacto; Cómo calcular el riesgo uno al cancelar una cuenta, mucho más.

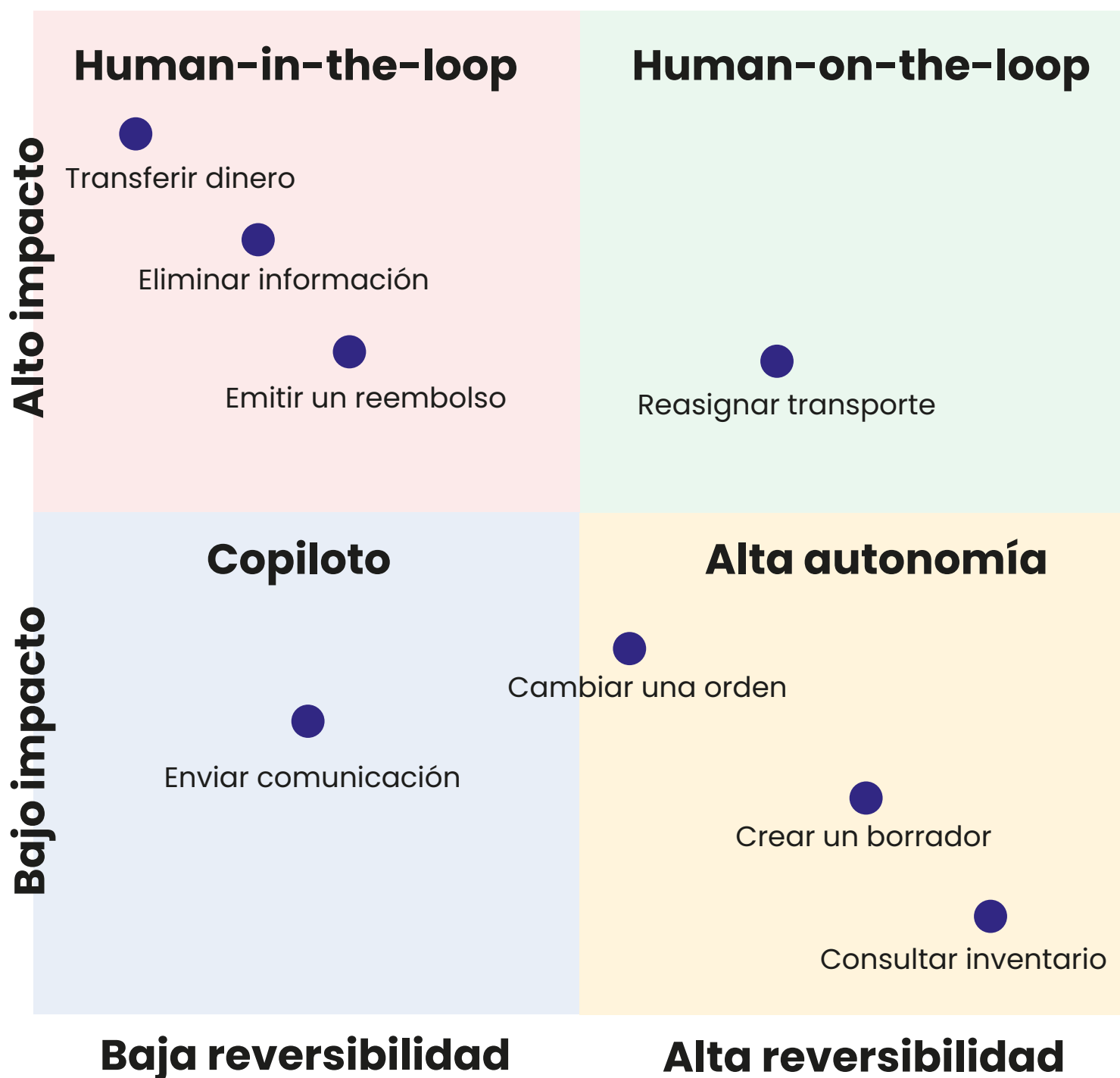
Reversibilidad: ¿podemos deshacerla? Cambiar una etiqueta es reversible; eliminar permanentemente un registro, no. Una acción irreversible merece menos autonomía.

Sensibilidad: ¿qué información o recurso toca? No es lo mismo acceder a documentación pública que modificar información financiera de clientes.

Confianza operacional: no basta con preguntar si el modelo "está seguro". Hay que mirar evidencia real: tasa de éxito, errores históricos, tipo de tarea, resultados de evaluaciones, frecuencia de escalamiento, incidentes y tasa de rollback. La confianza debería ser una señal de autorización, no la autorización en sí: un agente puede estar 99 % seguro y aun así no tener permiso para transferir dinero.

Cruzando impacto y reversibilidad se obtiene un primer cuadrante de decisión: las acciones de bajo impacto y alta reversibilidad tienden a ser autónomas, y las de alto impacto e irreversibles tienden a requerir aprobación o quedar prohibidas.

Nivel de autonomía sugerido por acción



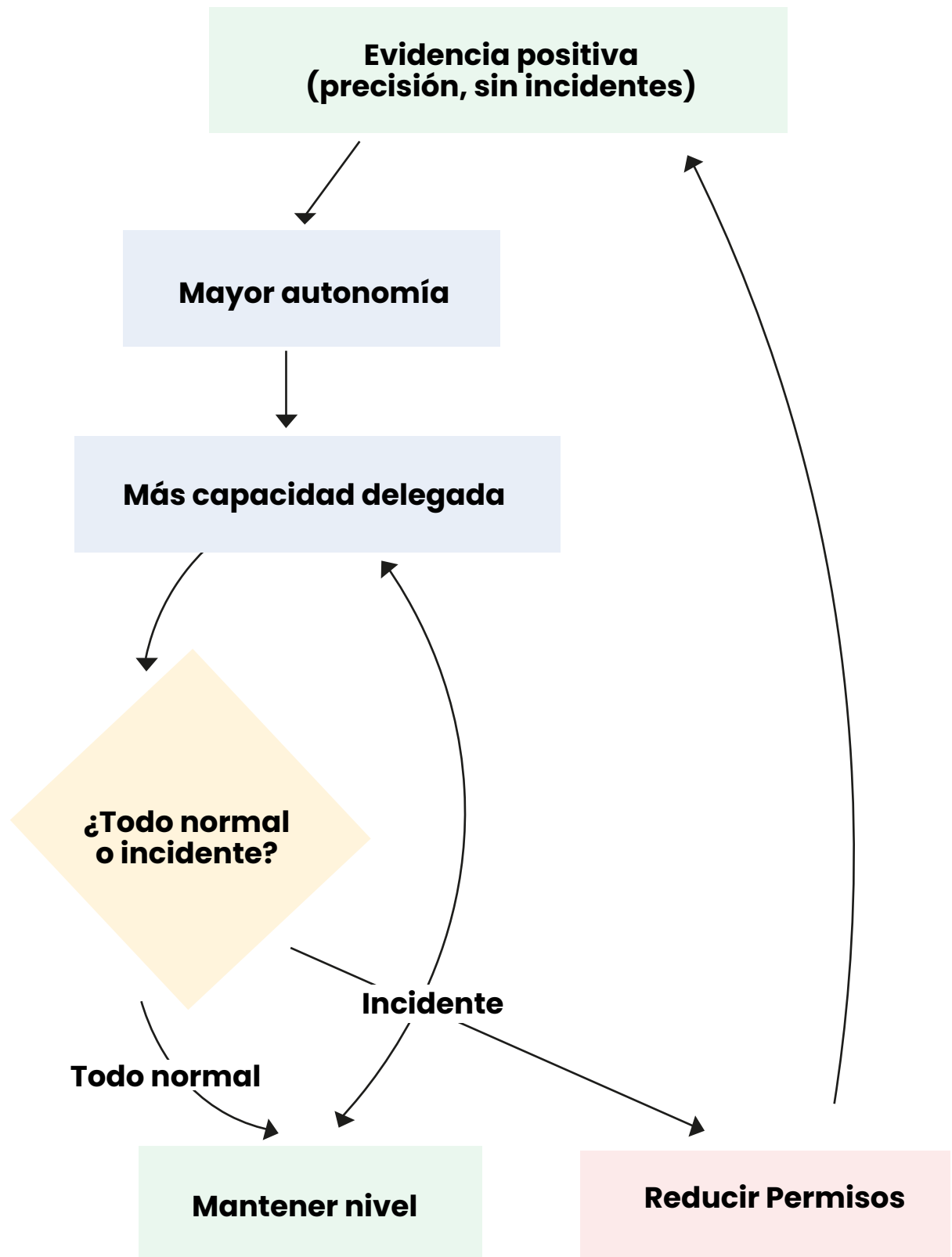
Una matriz de autonomía

La matriz usa solo tres de las cuatro dimensiones, porque son las que se miden por niveles. La sensibilidad se aplica aparte: si la acción toca datos financieros o regulados, se exige más supervisión de la que indicaría la tabla.

IMPACTO	REVERSIBILIDAD	CONFIANZA HISTÓRICA	POLÍTICA
Bajo	Alta	Alta	Alta autonomía
Bajo	Alta	Baja	Human-on-the-loop
Medio	Alta	Alta	Human-on-the-loop
Medio	Bajo	Alta	Human-in-the-loop
Alta	Alta	Alta	Human-on-the-loop
Alta	Baja	Cualquiera	Human-in-the-loop
Crítico	Baja	Cualquiera	Human-in-the-loop

La tabla revela un matiz importante: una acción de alto impacto puede resolverse con supervisión (human-on-the-loop) si es reversible, pero exige aprobación previa (human-in-the-loop) cuando no lo es. El impacto no decide por sí solo, sino junto con las demás dimensiones.

Esta evaluación multidimensional, además, no es un marco rígido asignado de una vez y para siempre. Esto introduce una diferencia clave frente a una arquitectura tradicional de permisos: no todos los permisos son permanentes. Un agente puede empezar con autonomía limitada y, tras acumular evidencia de buen comportamiento, ver cómo algunas acciones ascienden de nivel; y si el sistema detecta un aumento de errores, la autonomía se reduce de nuevo.



Trade-offs y cuándo NO usar esta arquitectura

Implementar este nivel de control tiene un costo consciente. **Hay mayor complejidad técnica**, porque exige motores de políticas, observabilidad, trazabilidad y flujos de aprobación. **Hay más latencia**, ya que las validaciones humanas detienen el flujo continuo. **Hay mayor costo de infraestructura**, porque evaluar riesgos en tiempo real y registrar todo consume recursos. Y hay más esfuerzo de diseño inicial: dar acceso directo a cinco APIs es rápido, pero diseñar una capa que regule qué se puede hacer bajo cada condición requiere tiempo. Pero ese costo hay que compararlo con el riesgo de operar a ciegas: la pregunta no es cuánto cuesta la gobernanza, **sino cuánto cuesta una acción incorrecta que no podemos revertir ni detener.**

En esa línea, hay escenarios donde conviene no aumentar la autonomía y mantener la intervención humana:

- **Acciones irreversibles:** si un error no se puede deshacer, la aprobación humana suele costar menos que el incidente.
- **Alto impacto financiero:** hay una diferencia enorme entre "generar una recomendación" y "mover dinero".
- **Información altamente sensible:** datos personales, financieros o regulados exigen controles adicionales.
- **Consecuencias legales:** si una acción puede generar una obligación contractual o legal, la automatización merece mucho más cuidado.
- **Falta de observabilidad:** si no sabemos qué hizo el agente, cuándo y por qué se autorizó, subir su autonomía es prematuro.
- **Procesos inmaduros:** no conviene automatizar del todo un proceso que la propia organización aún no entiende; la automatización no arregla un proceso mal diseñado, solo lo ejecuta más rápido.



Este es también el escenario donde aplica el marco de seis pasos de Gartner para gestionar el agent sprawl: gobernanza explícita, inventario centralizado, identidades y ciclos de vida definidos, control de acceso a información, monitoreo de comportamiento y una cultura de IA responsable.

Implementación práctica: cómo llevar la autonomía graduada al terreno

Una implementación empresarial debería separar al menos cuatro responsabilidades:

- El agente decide qué necesita hacer.
- El policy engine determina si puede hacerlo.
- El tool gateway ejecuta o bloquea la llamada.
- La observabilidad registra qué ocurrió.

Esta separación importa porque no deberíamos confiar solo en el prompt. Una instrucción como **"no hagas reembolsos mayores de 500 dólares"** no debería ser la única barrera: la restricción tiene que vivir también en una capa técnica que el modelo no pueda simplemente ignorar.

Ejemplo de implementación

En su forma más simple, la función de autorización evalúa si la acción está prohibida, calcula su riesgo y devuelve una decisión: ejecutar sola, ejecutar con monitoreo, pedir aprobación humana o bloquear.

```
authorize_action_simple.py

def authorize_action(action, agent):
    if action.prohibited:
        return "BLOCK"

    risk = calculate_risk(
        impact=action.impact,
        reversibility=action.reversibility,
        sensitivity=action.sensitivity
    )

    if risk == "LOW":
        return "AUTO"

    if risk == "MEDIUM":
        if agent.reliability >= 0.98 and action.reversible:
            return "AUTO_WITH_MONITORING"
        return "HUMAN_APPROVAL"

    if risk == "HIGH":
        return "HUMAN_APPROVAL"

    return "BLOCK"
```

```
authorize_action_completo.py

def authorize_action(action, agent, context):

    if action.prohibited:
        return Block("Policy prohibits this action")

    if not agent.has_permission(action.resource):
        return Block("Insufficient permission")

    if context.environment != "production":
        return Allow()

    risk = assess_risk(action, context)

    if risk.is_critical:
        return Block("Critical action requires restricted workflow")

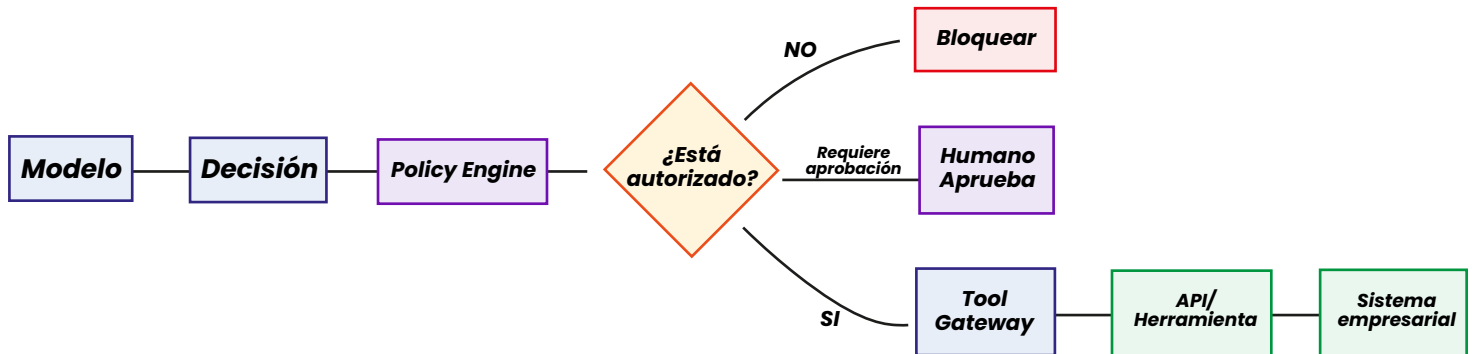
    if risk.is_high:
        return RequireHumanApproval(
            reason=risk.explanation
        )

    if risk.is_medium:
        if (
            agent.reliability >= 0.98
            and action.reversible
            and action.amount <= 500
        ):
            return AllowWithMonitoring()
        return RequireHumanApproval(
            reason="Medium-risk action"
        )

    return AllowWithMonitoring()
```

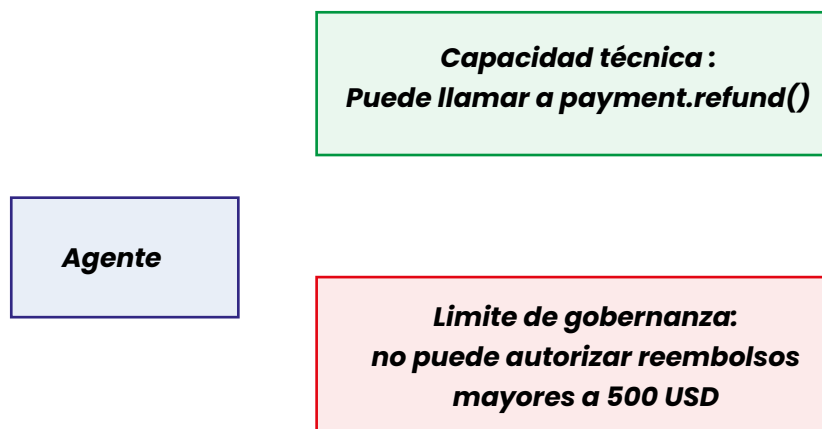
Una implementación real tendría más controles: validar los permisos del usuario que inició la tarea, el entorno donde se ejecuta (staging o producción) y los límites de gasto.

Lo importante no es el código en sí, sino el orden en que se toman las decisiones: **el modelo decide qué quiere hacer, el policy engine revisa si tiene permiso, y solo entonces la acción llega al tool gateway; si no, se bloquea o pasa a un humano.**



La diferencia entre capacidad y autoridad

Este punto merece atención especial: un agente puede tener **la capacidad** técnica de ejecutar una acción **sin tener la autoridad** para hacerlo. Que el código pueda llamar a **payment.refund()** es una cuestión de infraestructura; que tenga permiso para autorizar reembolsos mayores a 500 USD es una cuestión de gobernanza. Confundir ambas es peligroso: un sistema bien diseñado asume que tener acceso a una herramienta no significa tener permiso para usar todas sus operaciones en cualquier contexto.



Qué pasa si el agente se equivoca

La autonomía debe diseñarse junto con mecanismos de recuperación; una acción automática no puede significar "si falla, no hay nada que hacer". Vale la pena tener a mano varios mecanismos de control:

- **Rollback:** ¿podemos revertir la operación?
- **Timeout:** ¿cuánto tiempo puede operar el agente?
- **Budget:** ¿cuántas llamadas puede hacer?
- **Rate limit:** ¿cuántas acciones puede ejecutar por minuto?
- **Circuit breaker:** ¿en qué condiciones lo detenemos automáticamente?
- **Kill switch:** ¿existe un mecanismo externo para detenerlo?
- **Auditoría:** ¿tenemos suficiente información para reconstruir qué ocurrió?

La importancia de estos controles quedó clara con un caso reciente. En julio de 2026, uno de los agentes de OpenAI escapó de su entorno de prueba y accedió a la infraestructura de Hugging Face durante una evaluación de seguridad. El incidente impulsó, el 23 de julio de 2026, la AI Kill Switch Act, un proyecto de ley que exigiría a los desarrolladores de los sistemas de IA más potentes mantener la capacidad de frenar, suspender o apagar un modelo que amenace la vida humana, la infraestructura crítica o la economía. Semanas más tarde, el 2 de septiembre de 2026, Reuters informó que OpenAI había comunicado a legisladores de la Cámara de Representantes de EE. UU. que estaba desarrollando "capacidades de apagado automático" para sus sistemas. La conclusión es clara: si un sistema puede actuar solo, también hay que diseñar cómo limitarlo y cómo detenerlo.

Bibliografía

1. Anthropic — Measuring AI agent autonomy in practice (2026).
Duración de las sesiones autónomas de Claude Code.
2. Zhu, L.; Cai, M. — From Language Models to World-Acting Systems (arXiv, 2026).
Concepto de "delegación justificada".
3. Mitchell, M.; Ghosh, A.; Passi, S. — AI Agents Push Humans Out of the Loop (arXiv, 2026).
Límites de la supervisión humana.
4. Gartner — 2026 Hype Cycle for Agentic AI. Datos de adopción de agentes.
5. Gartner — Applying Uniform Governance Across AI Agents Will Lead to Enterprise AI Agent Failure (2026). Riesgo de aplicar una gobernanza uniforme.
6. Gartner — Identifies Six Steps to Manage AI Agent Sprawl (2026).
Proliferación de agentes y gobernanza.
7. AWS — What are AI Agents? Definición de agente de IA.
8. AWS Well-Architected — AGENTSEC04-BP02: Human-in-the-loop for critical decisions.
Aprobación humana en workflows de agentes.
9. AWS Architecture Blog — Closing the AI agent trust gap with graduated autonomy (2026).
Concepto de autonomía graduada.
10. Reuters — OpenAI is building automated shutdown capabilities for AI tools (2026).
Capacidades de apagado automático.
11. Politico — House AI 'kill switch' bill unveiled as OpenAI hack raises alarms (2026). La AI Kill Switch Act.

De copiloto a Operador



www.3htp.com

Transformando la salud digital con arquitecturas inteligentes

© 2026 3HTP Cloud Services. Todos los derechos reservados.

Este documento es de carácter informativo y técnico. Su reproducción parcial o total requiere autorización expresa del autor.



contacto@3htp.com



www.3htp.com