MANUAL BASE · IA AGÉNTICA

Un agente no es un chatbot con esteroides.

Es **un empleado nuevo**. Brillante, con memoria de pez, sin manos y sin llaves de tu oficina. Todo lo que la gente llama "conceptos técnicos" — contexto, memoria, tools, MCP, skills, loop — es en realidad **lo que le das a ese empleado para que sirva**. Este documento explica cada pieza: qué es, dónde vive, quién la escribe y cómo se dispara.

Sin código previo requerido Verificado con documentación oficial Agosto 2026

01 — EL PUNTO DE PARTIDA

La analogía que amarra todo lo demás

Si entiendes esta sola idea, el resto del documento es vocabulario. Si no la entiendes, ninguna definición te va a servir.

Imagina que contratas a alguien **brillante**. Leyó prácticamente todo lo que se ha escrito. Habla ocho idiomas, entiende contabilidad, sabe programar, redacta mejor que tu agencia.

Y llegó ayer.

No sabe cómo se llama tu empresa. No sabe qué vendes. No tiene llave de la oficina, ni acceso a tu correo, ni idea de cómo llenas una propuesta. Cada mañana llega con amnesia total: no se acuerda de lo que le corregiste ayer.

Ese es un modelo de lenguaje. Todo lo demás — *absolutamente todo lo demás* — es lo que haces para volverlo útil.

EN LA OFICINA	SE LLAMA	QUÉ RESUELVE
Lo que sabe del mundo porque ya lo estudió	**Modelo · LLM**	La materia prima. Es lo único que no construyes tú.
Lo que le pides hoy, de viva voz	**Prompt**	La instrucción puntual. Se muere cuando termina la conversación.
El manual de la empresa que lee cada mañana antes de empezar	**Contexto**	Deja de explicarle quién eres en cada mensaje.
La libreta donde apunta lo que le corregiste	**Memoria**	Que no repita el mismo error el lunes siguiente.
Sus manos: teclado, teléfono, coche, calculadora	**Tools**	Que pueda *hacer*, no solo opinar.
El llavero con las credenciales de tus sistemas	**MCP**	Que entre a tu correo, tu CRM, tu base de datos — sin que programes cada puerta.
Los instructivos de "así se hace aquí"	**Skills**	Que tu proceso salga igual de bien la vez 50 que la vez 1.
Su forma de trabajar: mira, decide, hace, revisa, repite	**Loop**	Que termine la tarea sola en vez de devolverte una pregunta.
El asistente al que le encarga una parte y le reporta	**Subagente**	Dividir trabajo grande sin saturar una sola cabeza.
Todo lo anterior, funcionando sin que lo supervises	**Agente**	Le das un objetivo, no una instrucción.

> Un chatbot es esa misma persona, brillante, pero amarrada a la silla: sin manos, sin llaves y con amnesia cada mañana.

> Por eso "la IA no me sirve para mi negocio" casi nunca es un problema de la IA. Es que la contrataste y nunca la diste de alta.

02 — ARQUITECTURA

Las seis capas de un agente

Un agente no es una cosa. Son seis capas apiladas, y cada una la escribe alguien distinto. El orden importa: cada capa se apoya en la de abajo.

01

El modelo

LO ELIGE TÚ · LO CONSTRUYE EL PROVEEDOR

El cerebro. Opus, Sonnet, GPT, Gemini. Es la única capa que no controlas, y es en la que todo el mundo pierde el tiempo comparando. Cambia el modelo y ganas quizá un 10% de calidad. Arregla las cinco capas de arriba y ganas 300%.

02

El loop

LO TRAE LA PLATAFORMA

El motor que convierte "responder" en "trabajar": observar, pensar, actuar, verificar, repetir hasta terminar. No lo escribes tú; viene con la herramienta que uses. Pero sí decides cuándo se detiene.

03

El contexto

LO ESCRIBES TÚ, UNA VEZ

Quién eres, qué vendes, cómo hablas, qué nunca debe hacer. Se carga solo al empezar cada sesión. Es la capa con mejor retorno por hora invertida y la que casi nadie escribe.

04

La memoria

LA ESCRIBE EL AGENTE · TÚ LA REVISAS

Lo que aprende de trabajar contigo: tus correcciones, tus preferencias, los tropiezos que ya resolvió. Se acumula sola. Aquí está el efecto compuesto: el agente de tu mes seis es notablemente mejor que el de tu semana uno, con el mismo modelo.

05

Las herramientas

LAS CONECTAS TÚ

Tools y MCP. Buscar en internet, leer un archivo, correr código, mandar un correo, consultar tu base de datos. Sin esta capa el agente solo puede hablar. Con ella, puede hacer.

06

Las skills

LAS ESCRIBES TÚ · UNA POR PROCESO

Tus procedimientos empaquetados: "así se arma una propuesta", "así se hace el reporte del viernes". Es donde tu conocimiento operativo deja de vivir en tu cabeza y empieza a ser un activo que se ejecuta.

LA REGLA DEL 90/10

El 90% de la gente pelea con la capa 01 — qué modelo es mejor, qué versión salió. El 10% que le saca resultados trabaja las capas 03 a 06. Son capas de **escribir documentos**, no de programar. Por eso esto no es una habilidad técnica: es una habilidad de operación.

03 — REFERENCIA

La matriz: los doce conceptos en una sola vista

Si solo te vas a llevar una cosa de este documento, llévate esta tabla. Está ordenada de lo más simple a lo más compuesto.

CONCEPTO	QUÉ ES, EN UNA FRASE	ANALOGÍA	DÓNDE VIVE	CÓMO SE ACTIVA
Modelo LLM	El motor que entiende y genera lenguaje.	El cerebro de la persona que contrataste.	En los servidores del proveedor. Tú lo consumes.	Siempre encendido. Lo eliges, no lo disparas.
Prompt	Lo que le pides en este momento.	La instrucción que le das de viva voz.	En el chat. En ningún archivo.	Lo escribes tú. Muere al cerrar la conversación.
Contexto	La información base que siempre debe tener presente.	El manual de la empresa, en su escritorio.	Un archivo de texto: `CLAUDE.md` o `AGENTS.md`.	Se carga solo al iniciar cada sesión. No lo pides.
Memoria	Lo que aprendió de trabajar contigo y conserva.	Su libreta de correcciones.	Archivos `MEMORY.md` que el propio agente mantiene.	El agente escribe cuando aprende; lee al empezar.
Tool herramienta	Una acción concreta que puede ejecutar.	Sus manos: el teclado, el teléfono, el coche.	Dentro de la plataforma o expuesta por un MCP.	El agente la llama solo, cuando la tarea la necesita.
MCP Model Context Protocol	El estándar que conecta al agente con sistemas externos.	El enchufe universal. USB-C para software.	`.mcp.json` en tu proyecto, o el menú de conectores.	Se conecta al abrir la sesión; sus tools quedan disponibles.
Skill habilidad	Un procedimiento tuyo, empaquetado y reutilizable.	El instructivo de "así se hace aquí".	Una carpeta con un `SKILL.md` adentro.	Por nombre (`/reporte`) o sola, si tu petición coincide.
Loop ciclo agéntico	El ciclo observar → pensar → actuar → verificar.	Su forma de trabajar hasta terminar.	Dentro del motor de la plataforma.	Arranca con tu objetivo; para cuando se cumple el criterio.
Subagente	Un agente especializado al que el principal le delega.	El asistente que hace una parte y reporta.	Un archivo `.md` por subagente, con sus propias reglas.	El agente principal lo invoca, o tú lo mencionas.

CONCEPTO	QUÉ ES, EN UNA FRASE	ANALOGÍA	DÓNDE VIVE	CÓMO SE ACTIVA
Hook gancho	Un script tuyo que corre siempre, decida lo que decida la IA.	El torniquete de la entrada: no negocia.	En la configuración: `settings.json`.	Automático, en momentos fijos (antes de X, después de Y).
Permisos	Lo que el agente tiene prohibido o permitido tocar.	El nivel de acceso de su gafete.	También en `settings.json`.	Se evalúan en cada acción, antes de ejecutarla.
Agente	Todo lo anterior operando junto, sin supervisión paso a paso.	El empleado ya dado de alta y trabajando.	No es un archivo: es la suma de las capas.	Tú das el objetivo. Él decide los pasos.

FÍJATE EN LA COLUMNA "DÓNDE VIVE"

Casi todo son **archivos de texto en carpetas**. Eso es la revelación que cambia el juego: construir un agente se parece más a escribir un manual de operación que a programar. Si sabes redactar un procedimiento claro para un empleado nuevo, ya sabes construir agentes. Lo único que falta es dónde se guarda cada cosa — la sección 06.

---

04 — DESHACER NUDOS

Las cuatro confusiones que traen todos

Estas cuatro parejas se confunden siempre. Resolverlas es la diferencia entre repetir palabras y entender el sistema.

A · Prompt, contexto, memoria y skill son todos texto. ¿Cuál es la diferencia?

Los cuatro terminan siendo texto que el modelo lee. La diferencia no está en *qué* son, sino en **cuándo entran** y **quién los escribe**. Esa sola distinción resuelve el 80% de la confusión.

	QUIÉN LO ESCRIBE	CUÁNDO ENTRA	CUÁNTO DURA	EJEMPLO
Prompt	Tú, en el momento	Cuando lo mandas	Esta conversación	"Hazme el reporte de agosto"
Contexto	Tú, una vez	Siempre, al arrancar	Hasta que lo edites	"Mi empresa fabrica aluminio arquitectónico; el cliente típico es un constructor"
Memoria	El agente	Al arrancar, si es relevante	Se acumula sola	"Jorge corrige siempre 'usuario' por 'cliente'"
Skill	Tú, una vez por proceso	Solo cuando hace falta	Permanente	"Para armar una propuesta: 1) revisa el catálogo, 2) calcula..."

> **Todo lo que le explicas más de dos veces al agente debería ser un archivo, no un prompt.**
>
> Si es sobre ti o tu negocio, va al contexto. Si es un procedimiento, va a una skill. Si es una corrección de estilo, la memoria la recoge sola.

B · Tool, MCP y API no son lo mismo

Esta la confunden hasta quienes ya programan, porque las tres palabras se usan como sinónimos en internet. No lo son.

API	**La puerta que tu sistema ya tenía.** Tu CRM, tu banco y tu correo ya tienen una desde antes de que existiera la IA. Es cómo un programa le habla a otro.
MCP	**El estándar de la chapa.** Antes, conectar una IA a un sistema requería programar esa conexión a la medida — una por cada sistema, por cada modelo. MCP define un formato único, así que quien publica el conector lo hace una vez y sirve para cualquier agente compatible. Es literalmente el argumento del USB-C.

Tool	**La acción concreta.** "Mandar correo", "buscar cliente", "crear evento". Una conexión MCP normalmente trae varias tools adentro. La tool es lo que el agente *llama*; el MCP es solo cómo llegó ahí.
En corto	La API es la puerta. El MCP es la cerradura estándar. La tool es abrir la puerta.

Dato de contexto: MCP nació en Anthropic, pero desde finales de 2025 lo administra la Agentic AI Foundation, bajo la Linux Foundation, junto con OpenAI, Google, Microsoft, Amazon y otros. Ya no es de una empresa. Eso importa cuando decides sobre qué construir: estás construyendo sobre un estándar de industria, no sobre el formato de un proveedor.

C · Skill, subagente y workflow

SKILL

Un procedimiento

Texto que le enseña *cómo* hacer algo. No tiene cabeza propia: se carga en la del agente. Es tu SOP, versionado.

Úsalo cuando: la tarea siempre se hace igual y quieres que salga igual.

SUBAGENTE

Un ayudante con su propia cabeza

Arranca con memoria limpia, hace lo suyo, regresa un resumen. Sirve para que la investigación pesada no contamine la conversación principal.

Úsalo cuando: la subtarea es grande, ruidosa o necesita otro criterio.

WORKFLOW

El guion que coordina

Define qué corre primero, qué corre en paralelo, qué pasa si algo falla. Es el jefe de proyecto, no el que ejecuta.

Úsalo cuando: ya tienes varias piezas y necesitas orden entre ellas.

D · Agente contra automatización

La confusión más cara, porque lleva a gastar en agentes donde una automatización de veinte pesos hacía el trabajo mejor.

Automatización	Las decisiones se toman **cuando la diseñas**. Si llega correo de X, mándalo a Y. Siempre igual. Es barata, instantánea y no se equivoca nunca.
Agente	Las decisiones se toman **cuando corre**. Lee el correo, juzga de qué se trata y decide qué hacer. Es más caro, más lento y a veces se equivoca.
Cómo elegir	¿Puedes escribir la regla completa en una hoja, sin "depende"? **Automatización.** ¿La regla depende de leer, entender o juzgar algo que cambia? **Agente.**
Lo normal	Los sistemas que funcionan en producción son mezclas: automatización para los pasos duros, agente solo en el paso donde de verdad hay criterio. Meter IA en pasos deterministas es la forma más común de tirar dinero.

05 — EL MOTOR

El loop, que es lo único que separa un agente de un chat

Un chat responde una vez y se calla. Un agente da vueltas hasta terminar. Ese ciclo es todo el asunto.

1. **Observa**

   Revisa qué tiene: tu petición, el contexto cargado, la memoria, qué herramientas están disponibles, qué ya hizo en pasos anteriores.

2. **Piensa**

   Decide cuál es el siguiente paso. Uno solo. No planea las quince acciones de golpe: elige la siguiente.

3. **Actúa**

   Ejecuta ese paso: llama una tool, lee un archivo, corre un cálculo, escribe algo.

4. **Verifica**

   Mira el resultado y se pregunta si ya terminó. Si no, vuelve al paso uno con información nueva. Si sí, entrega.

EL PASO QUE CASI TODOS SE SALTAN

La mayoría de las explicaciones describen el loop con tres pasos: observar, pensar, actuar. El cuarto — **verificar contra un criterio de terminado** — es el que separa un agente que sirve de uno que da vueltas para siempre o que se detiene a la mitad convencido de que acabó. Cuando escribas una skill, la línea más importante no es cómo hacer la tarea: es **cómo se ve la tarea bien hecha**.

Por qué se rompe un loop, en orden de frecuencia

- **No hay criterio de salida.** Nunca dijiste qué es "listo", así que entrega a la primera o no para nunca.
- **Le faltan manos.** Necesita leer un archivo y no tiene permiso de leer archivos. Te devuelve una disculpa en vez del trabajo.

- **El contexto está sucio.** Cargaste tanto texto de fondo que las instrucciones importantes se diluyeron.
- **Le pediste un objetivo que en realidad eran cinco.** "Arregla mi operación" no es un objetivo. "Revisa estas 40 facturas y márcame las que no cuadran con el contrato" sí.
- **Se topa con un permiso a cada paso.** Si todo requiere que apruebes, no es un agente: es un chat con pasos extra.

---

06 — LO FÍSICO

Las carpetas: dónde vive todo esto de verdad

Aquí es donde la teoría se vuelve archivos que puedes abrir con el Finder. Esta estructura es la de Claude Code, pero el patrón se repite en las demás herramientas con otros nombres.

TU PROYECTO — SE COMPARTE CON TU EQUIPO POR GIT

```
tu-proyecto/
├── CLAUDE.md              ← el contexto. Se carga en TODA sesión.
├── .mcp.json              ← qué sistemas externos puede tocar
└── .claude/
    ├── settings.json      ← permisos y hooks. Esto SÍ se obliga.
    ├── rules/             ← reglas por tema, se cargan según el caso
    │   └── propuestas.md
    ├── skills/            ← tus procedimientos, uno por carpeta
    │   └── reporte-semanal/
    │       ├── SKILL.md   ← obligatorio: qué hace y cuándo usarlo
    │       ├── plantilla.md
    │       └── scripts/
    │           └── armar.py
    └── agents/            ← subagentes especializados
        └── revisor.md
```

TU COMPUTADORA — PERSONAL, APLICA A TODOS TUS PROYECTOS

```
~/.claude/
├── CLAUDE.md              ← tus preferencias, en todos lados
├── settings.json          ← tus permisos por defecto
├── skills/                ← skills tuyas, disponibles siempre
└── projects/
    └── <proyecto>/memory/
        └── MEMORY.md      ← la memoria. La escribe el agente, no tú.
```

LA REGLA QUE ORDENA TODO

Dentro del proyecto va lo que es de ese negocio y lo comparte tu equipo. **En tu carpeta personal** (`~`) va lo que es tuyo y te sigue a todos lados. Si dudas: ¿le sirve a tu socio si abre este proyecto? Va en el proyecto. ¿Es tu manía personal de cómo quieres las respuestas? Va en tu carpeta.

Los tres archivos que importan, y qué NO poner en cada uno

CONTEXTO

CLAUDE.md

Sí: qué haces, cómo se llaman las cosas en tu negocio, tu tono, los comandos que corres siempre, lo que nunca debe tocar.

No: el estatus de esta semana. Se vuelve mentira en tres días y el agente la sigue creyendo.

Límite real: apunta a menos de 200 líneas. Pasado eso empieza a obedecerlo peor, no mejor. Lo que sobre, muévelo a reglas o skills.

MEMORIA

MEMORY.md

Sí: lo que el agente aprendió trabajando. Es él quien lo escribe; tú lo revisas y podas.

No: tu manual de operación. Eso va en contexto o en skills, porque quieres controlarlo tú, no que él lo reescriba.

Cómo se usa: le dices "recuerda esto" y lo guarda. Es la capa que hace que mejore solo con el tiempo.

PROCEDIMIENTO

SKILL.md

Sí: pasos, formato de salida, criterios de calidad, qué hacer si falta un dato.

No: teoría. El agente ya sabe qué es un estado de resultados. Lo que no sabe es cómo lo quieres *tú*.

Ventaja sobre un prompt guardado: puede traer plantillas, scripts y ejemplos en la misma carpeta.

CLAUDE.MD O AGENTS.MD — CUÁL USAR

Son el mismo tipo de archivo con distinto nombre. `AGENTS.md` se volvió el estándar abierto de la industria: hoy lo leen más de veinte herramientas y va por encima de 60,000 repositorios. `CLAUDE.md` es el nombre nativo de Claude Code.

Lo práctico: escribe tu contexto una sola vez en `AGENTS.md` y deja un `CLAUDE.md` de una línea que lo importe con `@AGENTS.md`. Así no mantienes dos copias que se van separando, y si mañana cambias de herramienta, tu contexto se va contigo.

---

07 — EL FORMATO

Anatomía de un SKILL.md, línea por línea

Una skill es una carpeta con un archivo de texto adentro. Eso es todo. El formato es tan simple que se explica completo en una pantalla.

SKILLS/REPORTE-SEMANAL/SKILL.MD

```
---                          ← el bloque de arriba es la ficha técnica
name: reporte-semanal        ← minúsculas y guiones. Igual que la carpeta.
description: Arma el reporte semanal de obra a partir de los
  correos de la semana. Usar cuando pida "el reporte", "cómo
  vamos esta semana", o invoque /reporte-semanal.
---                          ← de aquí para abajo, instrucciones normales

# Reporte semanal

Entregas un PDF de una página. No preguntas nada: si falta un
dato, decides hacia lo más simple y avisas al final qué asumiste.

## Pasos

1. Lee los correos de la semana con la etiqueta "obra".
2. Saca avance, bloqueos y lo que se decidió.
3. Llena `plantilla.md` de esta carpeta.
4. Conviértelo a PDF y preséntalo.

## Cómo se ve bien hecho

- Cabe en una página. Si no cabe, sobra texto, no falta espacio.
- Cada bloqueo tiene responsable y fecha. Sin dueño no es bloqueo.
- Cero adjetivos. Números o hechos.
```

> **Si tu skill no se dispara sola, el problema casi nunca son las instrucciones. Es la descripción.**
>
> El agente decide si usarla leyendo *solo* la línea `description` — el resto ni lo abre hasta que decide entrar. Escríbela con las palabras que tú realmente usas al pedir la tarea, no con las palabras "correctas".

Las dos partes, y por qué están separadas

- **La ficha (frontmatter).** Solo dos campos obligatorios: `name` y `description`. Esto es lo único que el agente tiene siempre a la vista, de todas tus skills a la vez. Por eso debe ser corto y debe decir *cuándo*, no solo *qué*.
- **El cuerpo.** Markdown normal. Se carga completo solo cuando la skill se activa. Aquí van los pasos, el formato, los criterios.

- **Los archivos vecinos.** Plantillas, ejemplos, scripts. Se leen solo si el cuerpo los menciona. Este mecanismo se llama *revelación progresiva* y es la razón de que puedas tener cincuenta skills sin saturar nada.

LA PRUEBA DE QUE FUNCIONA

Escribe la skill. Cierra todo. Al día siguiente pídele la tarea **con tus palabras normales**, sin nombrar la skill. Si se activa sola, quedó. Si no, arregla la `description` — casi siempre le faltan las palabras coloquiales con las que de verdad pides las cosas.

---

08 — ACTIVACIÓN

Las cinco formas de disparar un agente

"¿Cómo se triggerea?" tiene cinco respuestas, y elegir mal es lo que hace que un agente que funciona nunca se use.

FORMA	CÓMO SE VE	CUÁNDO CONVIENE	QUÉ NECESITAS
Implícita él decide	Pides algo normal y la skill se activa sola porque coincide con su descripción.	El 80% de los casos. Es la forma natural.	Nada extra. Solo una `description` bien escrita.
Explícita tú la llamas	Escribes `/reporte-semanal` y corre.	Cosas delicadas que no quieres que decida solo: mandar correos, publicar, cobrar.	Marcar la skill para que solo tú puedas invocarla.
Por horario cron	Todos los lunes 7:00 corre sin que nadie esté ahí.	Reportes, resúmenes, revisiones periódicas.	Un programador de tareas y una máquina prendida.
Por evento webhook	Llega un correo con cierta etiqueta y arranca.	Atención a clientes, entrada de pedidos, alertas.	Algo que escuche: un orquestador o un pequeño servidor.

FORMA	CÓMO SE VE	CUÁNDO CONVIENE	QUÉ NECESITAS
Por otro agente delegación	El agente principal le pasa una parte a un subagente.	Trabajo grande que conviene partir.	Subagentes definidos con su propio alcance.

LOS HOOKS NO SON UN TRIGGER, SON UN GUARDIA

Un hook corre **siempre**, en un momento fijo, decida lo que decida la IA. Antes de que toque un archivo, después de cada edición, al terminar la sesión. Esta distinción es de seguridad, no de vocabulario:

Lo que escribes en el contexto es **una sugerencia que el modelo lee y casi siempre sigue**. Lo que pones en hooks y permisos **se cumple aunque el modelo decida lo contrario**. Por eso "nunca borres la base de producción" no debe vivir en un archivo de instrucciones: debe vivir en los permisos.

09 — PLATAFORMAS

Dónde vive cada cosa, por nivel

La pregunta no es "cuál es mejor", es "en cuál estoy hoy y qué gano con subir uno". Estos niveles se apilan: casi nadie necesita el 5.

NIVEL	DÓNDE	QUÉ PUEDES HACER AHÍ	EL TECHO
0	Chat en navegador o celularclaude.ai y equivalentes	Contexto, memoria, skills, conectores MCP. Más de lo que la gente cree: aquí ya se pueden armar agentes útiles sin tocar una terminal.	No toca los archivos de tu computadora.
1	App de escritorioClaude Desktop, Cowork	Todo lo del nivel 0, más trabajar directo con tus carpetas y tus apps. Es el salto para quien no programa.	Corre mientras tu compu está prendida.

NIVEL	DÓNDE	QUÉ PUEDES HACER AHÍ	EL TECHO
2	Terminal o editorClaude Code, Codex, Cursor	La capa donde se construye. Aquí viven de verdad las carpetas de la sección 06: skills, subagentes, hooks, permisos, workflows.	Hay que perderle el miedo a una ventana negra.
3	API o SDK	Cuando el agente deja de ser tu asistente y se vuelve parte de tu producto: dentro de tu app, para tus clientes.	Ya es desarrollo de software. Necesitas quién.
4	Orquestadoresn8n, Make, Zapier	El pegamento: escuchan eventos, guardan colas, reintentan, avisan si algo truena. Llaman al agente solo en el paso que requiere criterio.	Se vuelve un dibujo de cajitas difícil de mantener si abusas.
5	Nube privada o empresarial	Cuando el dato legalmente no puede salir de tu perímetro. Mismos modelos, dentro de tu infraestructura.	Cuesta y requiere equipo. Casi nadie lo necesita.

> **Casi todo el valor de negocio está entre el nivel 1 y el 2.**
>
> Ahí es donde un dueño de empresa recupera horas reales de su semana. Los niveles 3 a 5 son para cuando ya no estás automatizando tu trabajo, sino construyendo un producto.

10 — PRODUCCIÓN

Cómo pasa de ”me funcionó” a ”corre solo”

"Subir a producción" suena a nube y servidores. En realidad son cuatro saltos, y los primeros dos no requieren infraestructura.

1

Corre cuando tú se lo pides

En tu compu, tú presente. Aquí vive el 100% de la gente el primer mes, y está perfecto: es donde descubres si la skill de verdad sirve. No automatices nada que no hayas corrido a mano diez veces.

2

Corre solo, en tu compu, por horario

El programador de tareas de tu sistema lo dispara los lunes a las 7. Cero infraestructura, cero costo extra. El único requisito es que tu máquina esté prendida. Aquí llega la mayoría de los negocios pequeños y les alcanza.

3

Corre en una máquina que no se apaga

Un servidor barato en la nube, un contenedor. Ahora sí necesitas: las llaves fuera del código, un registro de qué hizo, y una forma de apagarlo desde el teléfono.

4

Corre para otras personas

Cuando lo usan tus clientes o tu equipo. Aquí aparece lo aburrido y lo obligatorio: quién puede pedir qué, cuánto puede gastar, qué pasa si se cae, y un humano revisando los pasos irreversibles.

Lo que hay que tener antes de dejarlo solo

- **Llaves fuera del texto.** Nunca escribas una contraseña o un token dentro de un archivo de instrucciones. Van en variables de entorno, que es un archivo aparte que no se comparte.
- **Permisos explícitos.** Lista de lo que puede hacer sin preguntar y lista de lo que jamás. Se escribe una vez y se cumple siempre.
- **Un registro.** Qué hizo, cuándo, con qué resultado. Sin esto, cuando algo salga raro no vas a poder reconstruir qué pasó.
- **Un humano en el punto de no retorno.** Mandar el correo al cliente, hacer la transferencia, borrar registros. El agente prepara; una persona aprueba.
- **Un límite de gasto.** Un loop mal cerrado puede dar vueltas toda la noche. Ponle tope.

> **Nunca le des a un agente un permiso que no le darías a un becario de su primer día.**
>
> Es la regla más útil que existe para esto, y no requiere entender nada de seguridad informática. Al becario le das el correo para redactar borradores; no le das la firma del banco.

11 — DIAGNÓSTICO

Las ocho cosas que rompen un agente

Ordenadas por qué tan seguido pasan. Si el tuyo no jala, revísalas en este orden.

01

Contexto obeso

Metiste todo lo que sabes en el archivo de contexto. Pasadas unas 200 líneas, el modelo empieza a obedecer *peor*: las reglas importantes se pierden entre las accesorias. Menos texto, mejor seguido.

02

Descripción floja

La skill está perfecta y nunca se activa. Es la `description`: le faltan las palabras con las que de verdad pides la tarea.

03

Sin criterio de terminado

No escribiste cómo se ve bien hecho, así que entrega a medias o da vueltas. Toda skill debería cerrar con una lista de "esto es lo que tiene que cumplirse".

04

La memoria como basurero

Si dejas que se acumule sin podarla, se llena de cosas de un proyecto que ya cerró y arrastra decisiones viejas a trabajo nuevo. Ábrela una vez al mes y bórrale lo muerto.

05

IA donde iba un script

Le pides que sume una columna o que renombre archivos. Un script lo hace más rápido, más barato y sin equivocarse jamás. Reserva la IA para donde hay criterio.

06

Todo en el prompt

Cada vez escribes tres párrafos explicando quién eres. Eso es contexto que nunca guardaste. Es la fuga de tiempo más común y la más fácil de tapar.

07

Reglas duras escritas como sugerencia

"Nunca borres nada" en un archivo de instrucciones es una recomendación fuerte, no una barda. Si algo *no puede* pasar, va en permisos o en un hook.

08

Skills de origen desconocido

Una skill es texto que tu agente obedece, más scripts que puede correr. Ya se documentaron paquetes maliciosos en repositorios públicos que se veían normales y robaban credenciales. Instala solo de fuentes que puedas revisar, y ábrele los archivos antes.

12 — CONTROL DE CALIDAD

Fe de erratas de tu material de referencia

La tabla y el video que traías están bien orientados, pero tienen seis cosas que conviene corregir antes de enseñarlas — sobre todo si vas a pararte frente a gente.

LO QUE DICE	QUÉ PASA	CÓMO QUEDA
En la tabla, la limitación del asistente de código dice *"lleva la IA a producción con conocimiento del sistema"*.	Es un copiar-y-pegar: esa frase es la **ventaja de MCP**, dos columnas a la derecha. Una limitación no puede leerse como un beneficio.	La limitación real: solo ve el archivo abierto. No entiende tu sistema completo ni sabe por qué existe.
La tabla compara asistente de código, LLM, agente y MCP como cuatro cosas equivalentes.	Están en capas distintas. El LLM es el motor; el asistente y el agente son productos que lo usan; el MCP es un protocolo de conexión. Es como comparar motor, coche, camión y toma de corriente.	Ordénalas por capa (sección 02) y la confusión desaparece sola.

LO QUE DICE	QUÉ PASA	CÓMO QUEDA
La fila de "herramientas destacadas" lista Claude 4 y modelos de esa generación.	Es la fila que caduca más rápido. Esa generación ya quedó atrás; la línea de Anthropic hoy va por Opus 5, Sonnet 5 y Haiku 4.5.	No enseñes modelos, enseña categorías. Los nombres cambian cada trimestre; las capas no han cambiado.
El video describe el loop como observar, pensar, actuar.	Le falta el paso que hace que sirva: verificar contra un criterio de terminado.	Cuatro pasos, y el cuarto es el que separa un agente útil de uno que se cicla.
El video dice que crees un `memory.md` vacío y le pidas al agente que escriba ahí.	Era buen consejo cuando se grabó. Hoy las herramientas serias ya traen memoria automática: el agente crea y mantiene sus propios archivos sin que le montes nada.	Verifica si tu herramienta ya la trae encendida antes de inventar el mecanismo a mano.
El video presenta MCP como creación de Anthropic.	Cierto de origen, pero desde finales de 2025 lo administra la Agentic AI Foundation, bajo la Linux Foundation, con OpenAI, Google, Microsoft y Amazon adentro.	Importa para tu decisión: estás construyendo sobre un estándar de industria, no sobre el formato propietario de un proveedor.

UN SÉPTIMO, MÁS DE CRITERIO QUE DE DATO

El video clasifica dónde viven los agentes como "entornos de desarrollo, sistemas operativos y capas enterprise". Los ejemplos de la categoría intermedia no son sistemas operativos: son **aplicaciones que corren sobre el tuyo**. La distinción no es cosmética — cambia qué permisos les estás dando y qué puede pasar si algo sale mal.

13 — EJECUCIÓN

Siete días para tener uno funcionando

Una cosa al día. Ninguna toma más de una hora. Al séptimo día tienes un agente propio y sabes medir si sirve.

1

Lunes — escribe tu contexto

Pídele a la IA que *te entreviste* a ti sobre tu negocio: qué vendes, a quién, cómo hablas, qué odias, qué nunca debe hacer. Que la entrevista salga en un `AGENTS.md`. Media hora contestando preguntas vale más que dos horas escribiendo en blanco.

2

Martes — enciende la memoria

Revisa si tu herramienta ya trae memoria automática. Si sí, úsala tal cual y esa tarde corrígele tres cosas para que empiece a acumular. Si no, créale el archivo y pídele que apunte ahí cada corrección tuya.

3

Miércoles — conecta una sola herramienta

Una. La más aburrida que tengas: calendario o correo. Pídele algo mínimo — "resúmeme lo que llegó hoy". El objetivo del día no es el resultado, es ver el mecanismo funcionando.

4

Jueves — convierte tu tarea más repetida en skill

Elige lo que haces cada semana igual. Escríbelo como se lo explicarías a alguien que entra el lunes. Pasos, formato, y cómo se ve bien hecho. Guárdalo como `SKILL.md`.

5

Viernes — pruébala como desconocido

Sesión nueva. Pide la tarea con tus palabras normales, sin nombrar la skill. Si no se activa, arregla la descripción, no las instrucciones.

6

Sábado — ponle límites

Escribe qué puede hacer sin preguntarte y qué jamás. En permisos, no en instrucciones. Diez minutos que te ahorran un susto.

7

El séptimo — mídelo

Cronometra lo que esa tarea te tomaba y lo que te toma ahora. Multiplica por las veces al mes y por lo que vale tu hora. Ese número decide si vale la pena la skill número dos — y es el número con el que le vendes esto a alguien más.

LA TRAMPA DEL DÍA UNO

La tentación es empezar por lo espectacular: el agente que maneja toda la operación. Ese proyecto se abandona en la semana dos, siempre. Empieza por lo aburrido y repetido — es donde está el retorno medible y donde aprendes el mecanismo sin arriesgar nada.

14 — PARA IMPRIMIR

Glosario de una línea

La versión de bolsillo. Si alguien te suelta una de estas palabras en una junta, aquí está la traducción.

Agente
Le das un objetivo, no instrucciones. Decide los pasos solo.

Agent loop
Observar, pensar, actuar, verificar. Repetir hasta terminar.

AGENTS.md
El archivo estándar de contexto. Un README, pero para la IA.

Chatbot
Pregunta y respuesta. Sin manos, sin memoria, sin autonomía.

CLAUDE.md
El nombre del archivo de contexto en Claude Code.

Context engineering
Escribir bien lo que el agente siempre debe tener presente.

Contexto
El manual que lee antes de empezar cada sesión.

Hook
Un script que corre siempre, decida lo que decida la IA.

LLM
El modelo. El cerebro que entiende y genera lenguaje.

MCP
El enchufe estándar entre agentes y tus sistemas.

Memoria
Lo que aprendió de trabajar contigo y conserva entre sesiones.

Permisos
Qué puede tocar y qué no. Se cumple, no se sugiere.

Prompt

Revelación progresiva

Lo que le pides ahorita. Muere al cerrar la conversación.

Solo carga el detalle cuando lo necesita. Por eso caben muchas skills.

Skill
Un procedimiento tuyo, en una carpeta, reutilizable.

SKILL.md
El archivo que define una skill: ficha arriba, instrucciones abajo.

Subagente
Un ayudante con cabeza propia al que se le delega una parte.

Tool
Una acción concreta que el agente puede ejecutar.

Trigger
Lo que hace que arranque: tú, un horario, un evento u otro agente.

Workflow
El guion que coordina varias piezas y define el orden.

Nada de esto es programar. Es escribir un buen manual de operación.

Contexto, memoria y skills son documentos. Si sabes explicarle un proceso a alguien que entra el lunes, ya tienes la habilidad que importa. Lo demás es saber en qué carpeta va cada archivo — y eso está en la sección 06.