



Guide gratuit • second volet

# Comprendre l'IA qui travaille pour vous

Agents, skills, MCP, hooks, fichiers d'instructions.  
Ce que c'est, comment ça marche, et les pièges.

Version 1, octobre 2026

**AuraStack AI** • Christophe Cazenave • Drôme

## À qui sert ce guide

Un vendredi soir, dans un restaurant d'une quarantaine de couverts, les commandes tombent toutes en même temps et déjà deux plats partent à la mauvaise table, un troisième ressort froid parce qu'il a trop attendu, et un client, qui patiente depuis quarante minutes, voit la table d'à côté servie avant lui. Personne n'a mal fait son travail, il est juste arrivé trop de commandes d'un coup et personne en cuisine ne pouvait suivre, même en se démenant.

Ce guide parle des machines à qui on voudrait confier une partie de ce désordre. De quoi elles sont faites, comment on en parle, et comment reconnaître celles qui tiennent la charge.



*Le premier guide expliquait ce qu'est une IA. Celui-ci explique comment on la met au travail.*

Le guide précédent répondait à une question simple : qu'est-ce que c'est, et qu'est-ce que ça vaut. Il s'arrêtait volontairement au seuil, en disant qu'au niveau quatre l'assistant comprend la demande, prépare le travail et le fait valider. Il ne disait pas comment.

Ce guide-ci ouvre la boîte. Vous allez y trouver les mots qu'on emploie quand on configure réellement un système : outils, MCP, fichier d'instructions, skills, hooks, agents, sous-agents. Chacun a son chapitre, et chaque chapitre suit la même logique : ce que c'est, comment ça marche, à quoi ça sert, et les pièges. Les quatre derniers chapitres s'écartent un peu de ce plan, parce qu'ils comparent et assemblent au lieu de définir.

Le restaurant du début revient à chaque chapitre. Ce n'est pas un décor choisi au hasard : une cuisine contient déjà, en vrai et depuis longtemps, à peu près tous les mécanismes dont il est question ici.

Et d'abord le mot qui donne son titre à ce guide. Une IA devient un agent quand elle décide elle-même de la suite : quel outil appeler, dans quel ordre, quand recommencer, quand s'arrêter. Les skills, les serveurs et les barrières sont là pour encadrer cette liberté de décider. Le chapitre 7 y revient en détail.

## DEUX PRÉCAUTIONS D'USAGE

**Ce guide est fait pour comprendre, pas pour construire.** Suivre une conversation où l'on parle de skills et de hooks, savoir ce qu'on vous propose réellement quand on vous vend un agent, et juger ce qu'on vous livre. Il ne suppose à aucun moment que vous mettiez les mains dedans, mais il vous arme si vous décidez de le faire. Vous n'avez pas besoin de savoir programmer, et si vous voyez passer des noms de fichiers c'est volontaire : ce sont ces mots-là qu'on emploiera devant vous.

**Je sépare partout le principe du nom de marque.** Beaucoup de ces mots viennent d'un outil précis et sont présentés ailleurs comme le vocabulaire universel de l'IA. C'est faux et ça vieillit mal. Dans chaque chapitre, vous aurez d'abord le principe, qui existe partout, puis les noms que chaque éditeur lui donne.

## Le sommaire

### 1 Le contexte, la ressource qui manque toujours

Pourquoi la machine ne retient rien, et pourquoi tout le reste découle de là.

### 2 Les outils, ce qui sépare un assistant d'un agent

Le moment où elle cesse de parler et se met à faire.

### 3 MCP, la prise standard

Brancher un logiciel une fois et le garder le jour où l'on change d'assistant.

### 4 Le fichier d'instructions

Les consignes de la maison, lues avant de commencer. Et leur limite.

### 5 Les skills, le savoir-faire à la demande

Vos procédures rangées dans des tiroirs qui ne s'ouvrent qu'au besoin.

### 6 Les hooks, la règle qui ne se discute pas

Le seul mécanisme du guide qui empêche au lieu de persuader.

### 7 Les agents et les sous-agents

Ce qui fait vraiment un agent, et pourquoi il en délègue une part.

### 8 Les workflows, et quand il ne faut pas d'agent

Une question suffit à trancher, et elle vous évitera des factures.

### 9 L'assistant d'entreprise, ce qui tient l'ensemble

Le schéma complet, et les deux critères qui séparent un jouet d'un système.

### 10 Ce que ça coûte, et comment le savoir

Où part l'argent, et comment le voir avant la facture.

### 11 Dans quel ordre tout ça se met en place

L'ordre des étapes, et ce qu'il vous dit sur ce qu'on vous propose.

## Le contexte, la ressource qui manque toujours

L'extra du samedi arrive à onze heures et il ne connaît pas l'entreprise, ni la cuisine, peut-être un peu le métier. Il faut lui redire où sont les torchons, qui annonce les plats, qu'on ne touche pas au frigo du bas, et la semaine d'après un autre extra se présentera et il faudra recommencer les explications. Une IA travaille exactement comme ça, à chaque échange.



*Une machine non plus ne peut pas tout porter. Tout ce qui suit revient à choisir ce qu'on lui met dans les bras.*

### Ce que c'est

Un modèle de langage ne se souvient de rien. Si votre outil a l'air de se souvenir de vous d'une semaine sur l'autre, ce n'est pas le modèle : le logiciel autour de lui a pris des notes et il les lui remet sous les yeux à chaque fois. C'est tout le sujet de ce chapitre. À chaque fois qu'il répond, on lui redonne l'intégralité de ce qu'il doit savoir : votre question, l'historique de la conversation, les documents utiles, les consignes permanentes, la liste des outils à sa disposition. Ce paquet s'appelle le contexte et sa taille maximale s'appelle la fenêtre de contexte.

Autrement dit, un commis qui arrive chaque matin sans rien savoir de la veille. Vous pouvez lui redire l'essentiel avant de commencer, il le prendra, mais il faudra recommencer demain et le temps passé à redire est du temps de service en moins.

### Comment ça marche

Cette fenêtre est large mais elle a une limite. Trois conséquences très concrètes.

Reprenez la cuisine du vendredi soir. Le nombre de bras est fixe, les commandes ne le sont pas, et quand ça déborde, ce n'est pas la dernière commande qui saute. C'est celle du milieu, celle que plus personne ne regarde.

- **Elle se remplit.** Une longue session finit par saturer. Les outils gèrent ça par une compression, un résumé automatique du début de la conversation. Vous y gagnez de la place et vous y perdez du détail, sans être prévenu de ce qui est parti.
- **Elle se paie.** Tout ce qui entre dans le contexte est renvoyé à chaque échange. Les outils réduisent la facture : le début de la conversation est gardé en cache et refacturé moins cher. Moins cher n'est pas gratuit, et le cache tombe dès que vous modifiez ce début.
- **Elle se dilue.** Plus le contexte est chargé, moins chaque élément pèse. Une instruction importante noyée dans quarante pages de documentation est une instruction que la machine va traiter comme le reste.

## À quoi ça sert de le savoir

Parce que c'est la clé de lecture de tout le reste. Le fichier d'instructions donne le strict nécessaire en permanence. Un skill ne charge un savoir-faire qu'au moment où il sert. Les sous-agents existent pour que le travail d'exploration ne pollue pas le fil principal. Le RAG existe pour aller chercher le bon document au moment où l'on en a besoin, au lieu de tous les garder ouverts. C'est le sigle anglais de « génération augmentée par la recherche » : la machine cherche d'abord, puis répond à partir de ce qu'elle a trouvé.

Ces quatre mécanismes répondent tous à la même question : qu'est-ce qu'on met dans les bras de la machine, et quand.

### LES PIÈGES

**Tout mettre dans les consignes permanentes.** C'est le réflexe du débutant, et le plus coûteux. Une consigne permanente est rechargée et payée à chaque tour, en plus de diluer les autres.

**Croire que la machine a lu ce que vous avez fourni.** Fournir n'est pas lire. Sur un contexte très chargé, un élément placé au milieu est nettement moins bien pris en compte qu'un élément placé au début ou à la fin.

**Ne pas voir la compression passer.** La plupart des outils l'annoncent à l'écran, prenez l'habitude de regarder. Ce qu'ils ne diront pas, c'est ce qui est parti au résumé. Si vos échanges longs dérivent soudainement, si l'assistant oublie une consigne posée une heure plus tôt, c'est souvent qu'il y a eu compression. La parade est de redonner l'essentiel, ou de couper la session en plusieurs.

## Les outils, ce qui sépare un assistant d'un agent

Quelqu'un passe en cuisine et lance qu'il n'y a plus de citrons. L'information est entendue et comprise, mais à la fin du service la commande de citrons ne sera pas passée, parce que chacun se sera dit que quelqu'un d'autre le ferait. Entre comprendre et faire, il y a les outils.



*Sans outils, il ne peut que commenter le travail. Avec, il le fait.*

### Ce que c'est

Un outil, c'est une fonction que le modèle a le droit d'appeler : lire un fichier, envoyer une requête à un site, interroger une base, écrire sur le disque, envoyer un mail. Le modèle ne l'exécute pas lui-même, il demande et le programme qui l'héberge exécute, puis lui rend le résultat.

### Comment ça marche

Chaque outil est décrit au modèle par un nom, une phrase qui dit à quoi il sert, et la liste de ce qu'il attend. Le modèle choisit lequel appeler d'après cette description, exactement comme vous choisiriez un tournevis d'après son étiquette.

Beaucoup le découvrent trop tard : **un outil mal décrit est un outil mal utilisé**. Un outil mal décrit sera mal appelé, ou pas appelé du tout et le modèle inventera une réponse à la place. Ce n'est pas un problème d'intelligence du modèle, c'est un problème d'étiquette sur le tournevis.

### À quoi ça sert

Les outils sont la frontière exacte entre les niveaux du premier guide. Un chatbot n'a pas d'outils, il produit du texte. Un assistant connecté en a quelques-uns, en lecture. Un assistant d'entreprise en a beaucoup, en écriture, avec des droits précis sur chacun.

Ils sont aussi votre point de contrôle. Vous ne pouvez pas empêcher un modèle de penser une bêtise, mais vous pouvez parfaitement l'empêcher de la faire, en ne lui donnant pas l'outil, ou en exigeant une validation avant l'appel.

#### LES PIÈGES

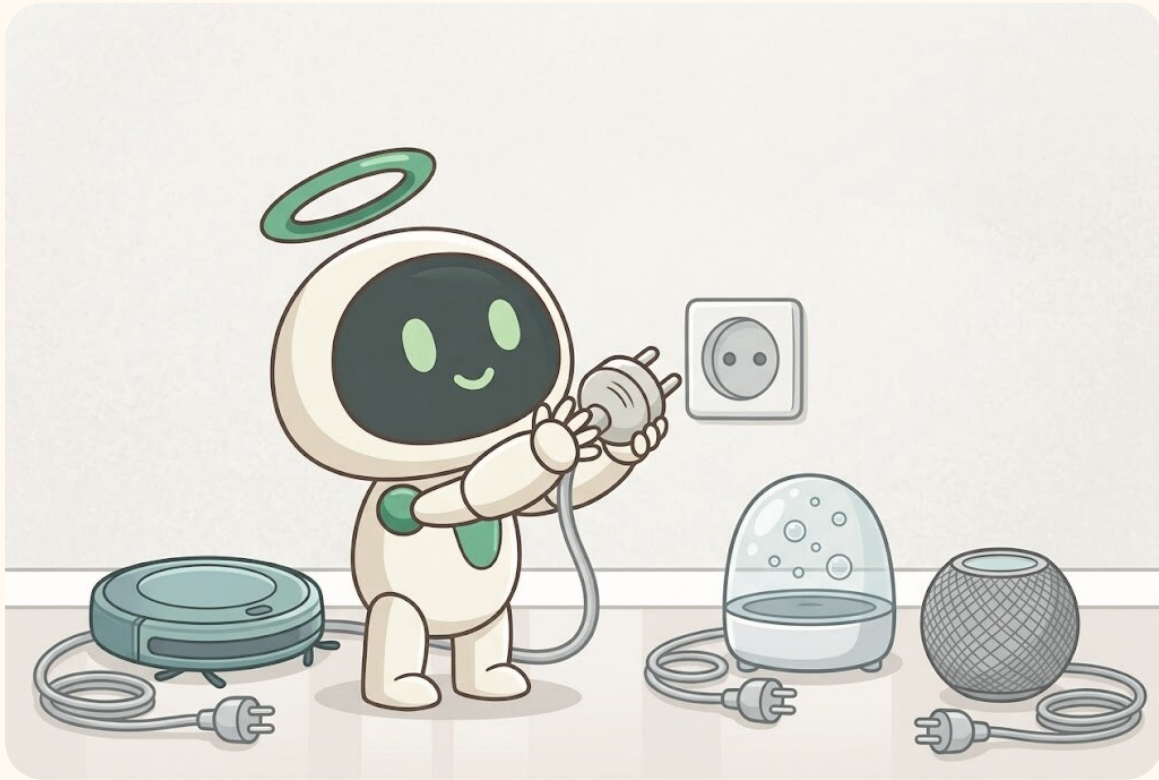
**Donner trop d'outils.** Chaque outil disponible occupe du contexte et augmente les chances d'un mauvais choix. Un assistant avec cent outils ne fait pas cent fois plus de choses. Il se trompe plus souvent de page.

**Confondre disponible et autorisé.** Un outil présent est un outil qui sera appelé un jour. Si vous ne voulez pas qu'il soit appelé, retirez-le, ne comptez pas sur une consigne écrite pour l'en dissuader.

**Croire un compte rendu d'exécution.** Le modèle rédige le récit de ce qu'il a fait et ce récit n'est pas une preuve. Il peut annoncer une vérification qu'il n'a pas faite. La seule preuve, c'est la trace de l'appel d'outil, et le résultat sur le disque.

## MCP, la prise standard

Le jour où on a changé de caisse il a fallu ressaisir la carte entière, retaper tous les prix, refaire les tickets, et pendant trois services on a pris les commandes sur un carnet en attendant que ça tourne. Le logiciel avait changé et tout le travail derrière était à refaire. MCP existe pour que ça n'arrive pas.



*Une seule prise, et tout ce qui a le bon embout se branche dessus.*

### Ce que c'est

MCP veut dire Model Context Protocol. C'est un protocole ouvert qui normalise la façon dont on branche des outils et des données sur un assistant. L'image que ses auteurs emploient eux-mêmes est celle d'une prise USB-C : un seul format de prise et tout ce qui le respecte se branche partout.

### Comment ça marche

D'un côté un serveur MCP, qui expose des outils : votre messagerie, vos fichiers, une base de données, un logiciel métier. De l'autre un client MCP, l'assistant. Le client demande au serveur la liste de ce qu'il sait faire et peut ensuite l'appeler. Écrit une fois, un serveur fonctionne avec tous les assistants compatibles.

Un mot sur qui tient ce format, parce que ça vous concerne. MCP a été créé par Anthropic, puis **donné à l'Agentic AI Foundation, sous la Linux Foundation, le 9 décembre 2025**. Le même jour, OpenAI y donnait AGENTS.md et Block y donnait goose. Parmi les membres qui paient la cotisation la plus élevée : AWS, Anthropic, Block, Bloomberg, Cloudflare, Google, Microsoft et OpenAI. Ce n'est donc plus le format propriétaire d'un éditeur et c'est un argument solide quand on vous reproche de dépendre d'un fournisseur.

## À quoi ça sert

À ne pas refaire le travail. Si vous faites développer une connexion à votre logiciel de devis sous forme de serveur MCP, elle vous suivra le jour où vous changerez d'assistant. Si vous la faites développer en direct dans un outil, elle partira à la poubelle avec lui.

### LES PIÈGES

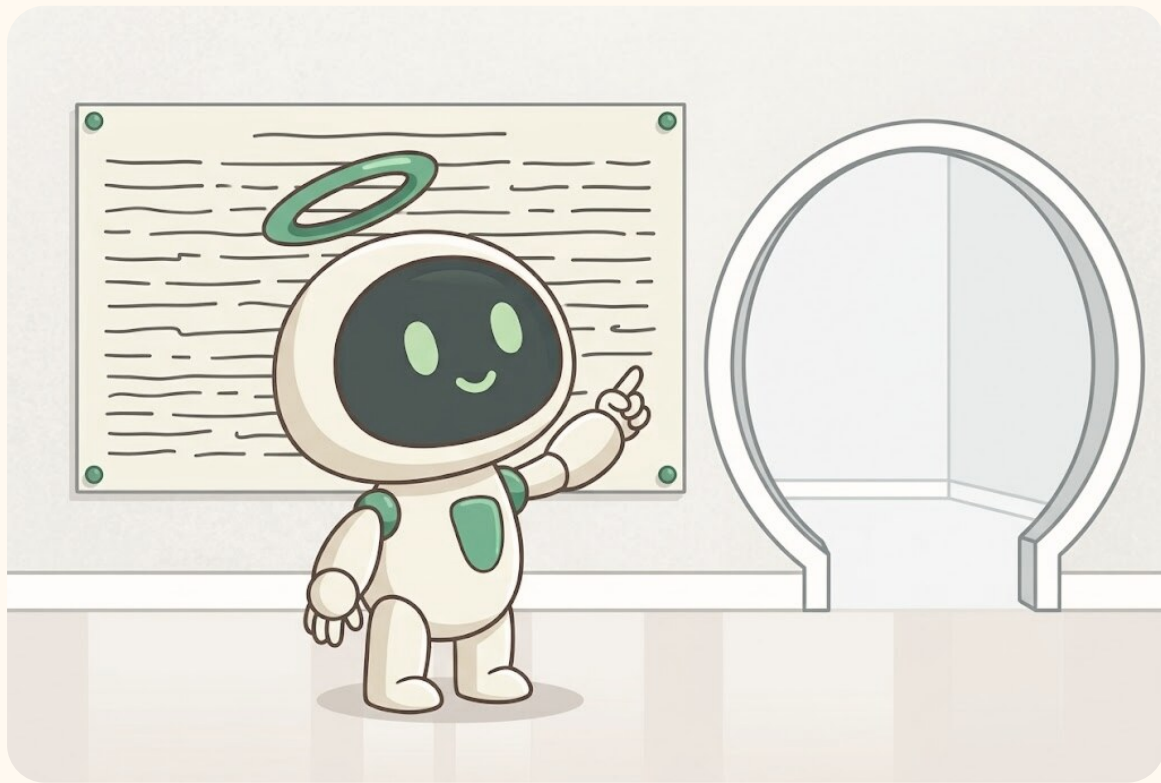
**Installer des serveurs trouvés au hasard.** Il en existe des milliers, publiés par n'importe qui. Et il y en a deux sortes, qui ne présentent pas le même risque. Celui qu'on installe tourne sur votre machine, avec vos droits : c'est un logiciel, regardez qui l'écrit avant. Celui auquel on se connecte tourne chez quelqu'un d'autre. Vous ne lui ouvrez pas votre poste, vous lui donnez une autorisation sur l'un de vos comptes, et elle reste valable tant que vous ne la retirez pas. Dans ce cas, lisez ce que l'écran de consentement réclame, et jusqu'où.

**Empiler les serveurs.** Chaque serveur ajoute ses outils à la liste. Avec les outils récents, la liste complète n'est plus chargée au démarrage, donc ça sature rarement. Le vrai problème est ailleurs : plus vous avez d'outils qui se ressemblent, plus l'assistant finira par choisir le mauvais.

**Confondre MCP et sécurité.** MCP normalise le branchement, il ne décide pas de ce qui est permis. Les droits, c'est à vous de les fixer. Vous le faites deux fois : au chapitre 2 quand vous donnez les outils, et au chapitre 6 quand vous posez les barrières.

## Le fichier d'instructions

Ce qu'on dit au nouveau le premier jour tient en quatre phrases : on ne sert pas une assiette ébréchée, on annonce avant de poser, le frigo du bas est pour le lendemain, et quand on ne sait pas on demande. Le fichier d'instructions est cette liste-là, avec une différence : il est relu en entier avant chaque échange.



*Les consignes de la maison, lues avant de toucher à quoi que ce soit.*

### Ce que c'est

Un fichier texte, posé à la racine du dossier. Vous y écrivez ce que l'assistant doit savoir en permanence : à quoi sert ce projet, quelles règles respecter, quelles commandes lancer, ce qu'il ne faut jamais faire. Il est lu en entier, à chaque session. On l'écrit en Markdown, c'est-à-dire du texte ordinaire avec quelques signes pour marquer les titres et les listes.

### Le principe, puis les noms

Le principe est le même partout. Les noms, non.

- **AGENTS.md** est la convention commune, née d'une collaboration entre plusieurs éditeurs et donnée à l'Agentic AI Foundation. Attention au vocabulaire : c'est une *convention*, pas une spécification. Il n'existe aucun document normatif, aucun champ obligatoire. C'est du texte libre.
- **CLAUDE.md** chez Anthropic, avec une hiérarchie : politique de l'organisation, puis utilisateur, puis projet, puis fichier local non versionné.
- **GEMINI.md** chez Google.
- **.github/copilot-instructions.md** chez GitHub.
- **.cursor/rules/** chez Cursor, avec une extension particulière.

Et un piège que vous rencontrerez pour de bon : **par défaut, Claude Code ne lit AGENTS.md que s'il n'y a pas de CLAUDE.md**. Si les deux existent, il ne lit que le sien. Il suffit qu'un collègue pose son propre fichier sur son poste : chez lui, le fichier commun n'est plus lu, et personne ne s'en aperçoit. Un réglage permet de lui faire lire les deux : si votre équipe tient à un fichier commun, vérifiez que ce réglage est bien activé sur tous les postes.

## À quoi ça sert

À ne pas répéter la même chose à chaque conversation. Tout ce que vous vous surprenez à réexpliquer trois fois par semaine a sa place dans ce fichier.

### LES PIÈGES

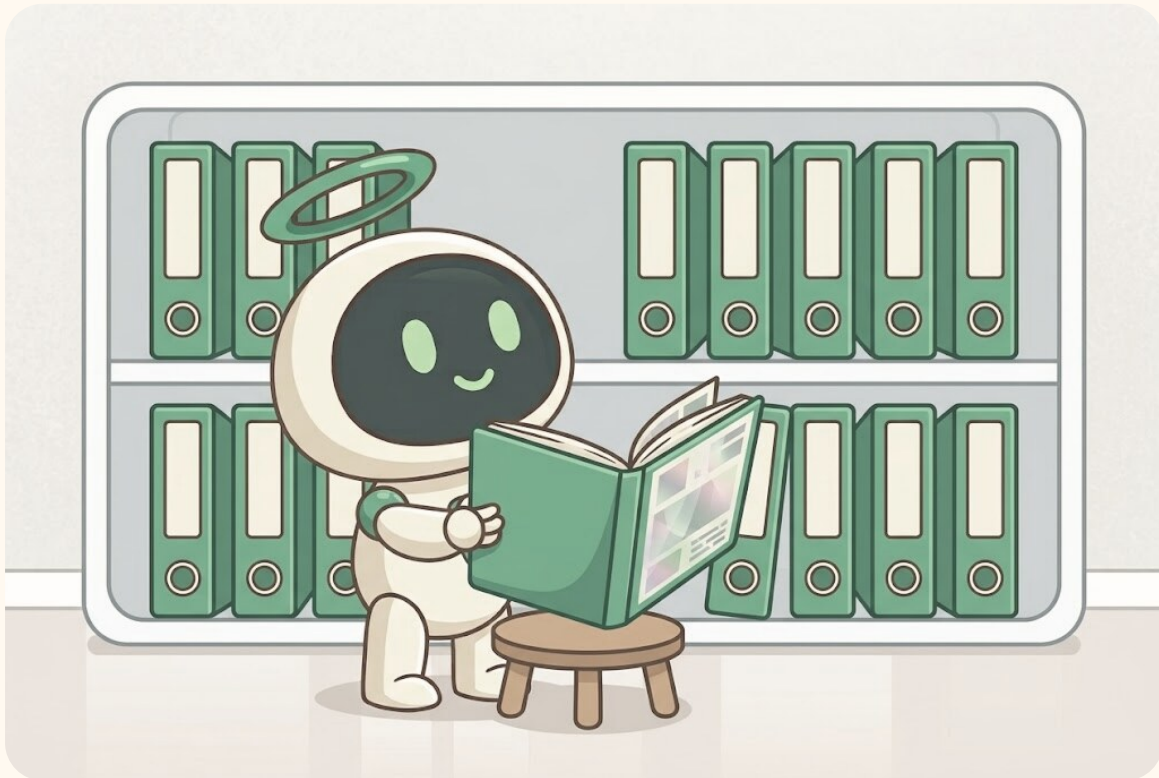
**En faire une encyclopédie.** C'est l'erreur la plus fréquente. Ce fichier est relu à chaque échange et vous le payez chaque fois. Plus il est long, moins l'assistant fait attention au reste. Vingt lignes utiles valent mieux que six pages exhaustives. Ce qui ne sert que parfois relève du chapitre suivant.

**Y écrire des interdictions et croire le travail fait.** Un fichier d'instructions persuade, il n'empêche pas. Écrire « ne jamais supprimer de fichier » réduit le risque, ça ne le supprime pas. Pour empêcher réellement, il faut un hook ou un retrait d'outil.

**Le laisser vieillir.** Un fichier qui décrit le projet d'il y a six mois est pire que pas de fichier : il oriente activement dans la mauvaise direction.

## Les skills, le savoir-faire à la demande

Le cahier de recettes reste fermé la plupart du temps. On l'ouvre le jour où on refait un plat qu'on n'a pas fait depuis six mois, ou quand un apprenti doit apprendre à le faire, et bien sûr on ne lit pas tout le cahier, uniquement la recette qui nous intéresse. Même si un apprenti en profite toujours pour en lire une ou deux de plus.



*Tout est rangé, et il n'ouvre que celui dont il a besoin aujourd'hui.*

### Ce que c'est

Un dossier qui contient un fichier `SKILL.md` et, si besoin, des gabarits, des documents de référence et des scripts. Le `SKILL.md` commence par deux informations attendues : un nom et une description qui dit ce que le skill fait et dans quel cas l'employer.

### Comment ça marche

C'est le mécanisme le plus élégant du lot et il tient en trois temps.

1. Au démarrage, l'assistant ne charge que le nom et la description de chaque skill. Quelques lignes, presque rien.
2. Quand la tâche correspond à une description, il charge le corps du `SKILL.md`, la procédure complète.
3. S'il a besoin d'un document annexe cité dans la procédure, il le charge seulement à ce moment-là.

Conséquence pratique : **vous pouvez avoir cinquante skills sans saturer quoi que ce soit**, là où cinquante procédures collées dans le fichier d'instructions seraient ingérables.

Bonne nouvelle sur le vocabulaire, c'est le cas le plus simple du guide. Le format vient d'Anthropic, a été publié comme standard ouvert, et a été repris tel quel par les autres. Même nom, même fichier `SKILL.md`, chez Claude, ChatGPT et Codex, GitHub Copilot, Cursor, Gemini CLI, le français Mistral Vibe et une quarantaine d'autres. Deux

choses changent d'un outil à l'autre : l'endroit où se range le dossier, et les réglages précis, qui ne suivent pas. Un skill qui a besoin d'un logiciel particulier ne fonctionnera pas ailleurs tel quel. La liste à jour des outils compatibles est publiée sur [agentskills.io](https://agentskills.io).

## À quoi ça sert

À sortir une procédure de la tête de celui qui la connaît. Tant que la façon de faire reste dans la tête du chef, elle part avec lui le jour où il s'en va. Écrite au gramme près, elle se transmet, elle se corrige et une machine peut la suivre. Votre procédure de relance client ou votre façon de rédiger un devis tiennent dans le même cahier.

### LES PIÈGES

**Une description vague.** C'est la seule chose que l'assistant lit pour décider d'ouvrir cette fiche. « Gestion des documents » ne déclenchera jamais rien. Écrivez dans la description les mots que vous emploierez vous-même en formulant la demande.

**Des skills qui se marchent dessus.** Deux descriptions proches, et le choix devient un tirage au sort. Un skill, un périmètre net.

## Les hooks, la règle qui ne se discute pas

Au pass, le chef prend l'assiette, l'examine, essuie une tache ou une coulure avec un papier, et elle part. Si elle ne va pas elle repart en cuisine et personne ne discute l'ordre du chef. Tout ce qui précède dans ce guide relevait de la persuasion, ici on passe à la contrainte.



*Interdit d'accès. Une porte fermée, pas une consigne qu'on suit quand on y pense.*

### Ce que c'est

Un hook est un petit programme qui se déclenche tout seul à un moment précis : juste avant que l'assistant utilise un outil, juste après, à l'ouverture d'une session, ou quand il a fini de répondre. Il peut observer, enrichir, modifier, et surtout bloquer.

### Comment ça marche

Vous déclarez dans un fichier de configuration qu'à tel événement, tel script doit s'exécuter. Le script reçoit les informations de l'événement et répond. S'il répond un refus, l'action n'a pas lieu. Point final, quelle que soit la conviction du modèle.

La distinction avec le chapitre 4 est la chose la plus importante de ce guide. **Un fichier d'instructions est un conseil que le modèle suit en général. Un hook est une règle qu'il ne peut pas enfreindre.** La documentation d'Anthropic le dit d'ailleurs sans détour : si vous voulez qu'une action soit bloquée même quand le modèle est persuadé du contraire, il faut un hook, pas une consigne écrite.

Le hook est la forme la plus générale de cette contrainte, ce n'est pas la seule. Vous pouvez aussi lui retirer l'outil, comme au chapitre 2, ou cocher un refus dans les réglages. L'action est bloquée aussi sûrement, et sans écrire une ligne de code. Le hook devient nécessaire quand la règle dépend du cas : bloquer selon le destinataire, selon le montant, selon l'heure.

Côté noms, le mot est le même chez les quatre grands outils de développement assisté, ceux d'Anthropic, d'OpenAI, de Google et de Cursor. En revanche, les détails ne se transportent pas : un hook écrit pour l'un est à réécrire pour l'autre. Vous n'avez pas à connaître ces détails pour comprendre le principe, mais il faut savoir qu'ils existent avant de choisir un outil.

#### POUR CEUX QUI IRONT VOIR DE PRÈS

Le moment où l'on intercepte un appel d'outil, juste avant qu'il parte, s'appelle `PreToolUse` chez Anthropic et chez OpenAI, `preToolUse` ou `beforeShellExecution` chez Cursor, et `BeforeTool` chez Google. Même idée, quatre écritures.

## À quoi ça sert

À rendre une garantie vérifiable. « L'assistant n'envoie jamais un mail seul » n'est une promesse sérieuse que si un hook bloque l'envoi. Autrement, c'est une intention. C'est aussi par là que passent les journaux d'audit, les formatages automatiques, et le rappel de contexte au démarrage.

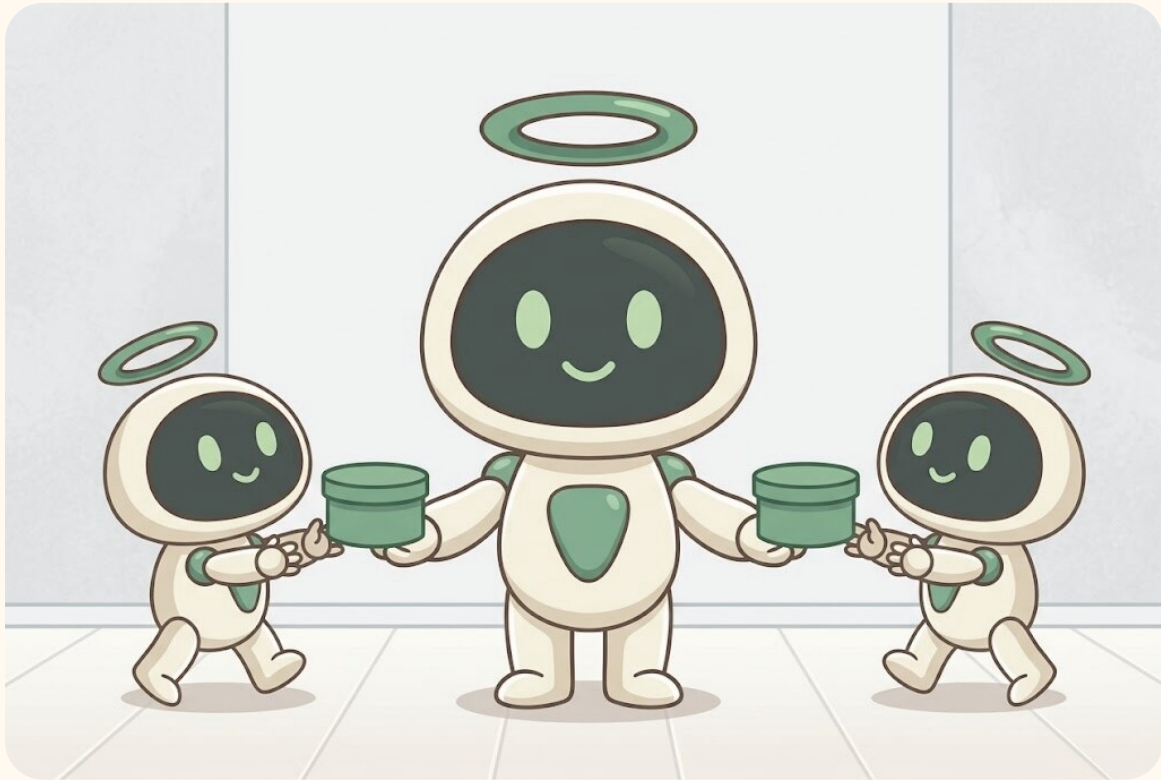
#### LES PIÈGES

**Croire qu'une consigne écrite suffit.** Le seul piège qui compte vraiment. Tant que votre garde-fou est une phrase dans un fichier, il tiendra la plupart du temps, et c'est là le comportement le plus dangereux : assez fiable pour qu'on baisse la garde, pas assez pour qu'on puisse le garantir à un client.

**Un hook lent, ou muet.** Il s'exécute à chaque événement concerné, donc un script qui prend deux secondes rend l'assistant pénible en une demi-heure. Et s'il bloque sans rien dire, vous passerez une heure à chercher pourquoi l'assistant « n'arrive pas ». Faites-lui toujours écrire pourquoi il a refusé.

## Les agents et les sous-agents

Le commis dit au chef de partie que la douze est sans gluten. Le chef de partie annonce la douze au pass, et le mot gluten est resté en route. Le chef voit une assiette qui a l'air conforme à ce qu'il connaît, sauf qu'il y a du gluten dedans.



*L'agent principal confie une part du travail à un autre, et n'en récupère que le résultat.*

### Ce qu'est un agent

Une IA devient un agent quand elle décide elle-même de la suite : quel outil appeler, dans quel ordre, quand recommencer, quand s'arrêter. Ce n'est pas sa spécialisation qui en fait un agent, c'est son autonomie. Une étape IA dans un scénario fixé d'avance n'est pas un agent, même si on vous la vend sous ce nom.

### Ce qu'est un sous-agent

Un sous-agent est un agent secondaire, lancé par le principal pour une tâche délimitée, **avec sa propre fenêtre de contexte**. Il travaille de son côté et ne renvoie qu'un résumé.

Les raisons sont des reprises directes des chapitres précédents. D'abord le contexte : la recherche qui remplit quarante pages remplit la mémoire du sous-agent, pas la vôtre. Vous ne recevez que sa conclusion. Ensuite les droits : on donne au sous-agent moins d'outils qu'au principal. Un agent chargé de lire et de résumer n'a aucune raison d'avoir le droit d'écrire.

Le terme est commun aux quatre mêmes outils. Les formats de définition, non, chacun écrit ses sous-agents à sa façon. À noter, parce que c'est pour l'instant le seul pont entre écosystèmes sur ce point précis : Cursor sait lire directement les dossiers d'agents d'Anthropic et d'OpenAI.

#### POUR CEUX QUI IRONT VOIR DE PRÈS

Un sous-agent se décrit dans un petit fichier : du Markdown avec un en-tête chez Anthropic, Cursor et Google, un format appelé TOML chez OpenAI. Même idée, deux écritures. Dans tous les cas on y trouve un nom et une description de ce à quoi il sert. La restriction des droits, en revanche, ne s'écrit pas partout de la même façon : une liste explicite des outils autorisés chez les uns, un simple interrupteur « lecture seule » chez les autres.

## À quoi ça sert

À deux choses. Traiter un gros dossier sans saturer la session, et limiter les droits de celui qui n'a qu'à lire. Dans une petite structure, trois suffisent : un qui cherche, un qui vérifie, un qui rédige.

#### LES PIÈGES

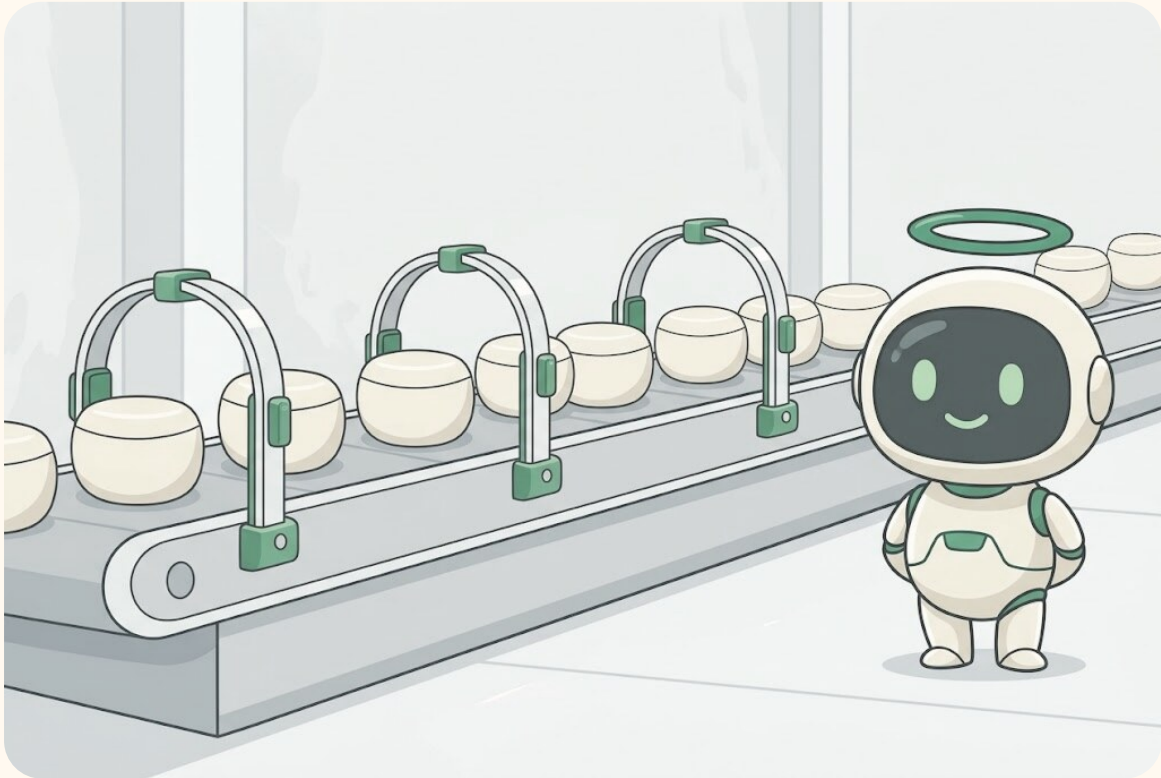
**Déléguer pour déléguer.** Un sous-agent démarre à vide, il ne sait rien de votre conversation. Pour une tâche courte, le temps passé à lui expliquer le contexte coûte plus cher que de la faire soi-même.

**La copie de la copie.** Le principal ne reçoit qu'un résumé écrit par le délégué. Si le délégué s'est trompé, le principal héritera de l'erreur en la croyant vérifiée et vous n'aurez aucun moyen de le voir dans la réponse finale.

**Plusieurs agents en parallèle sur le même sujet.** Ça a l'air impressionnant en démonstration. En pratique, deux agents qui travaillent sans se voir produisent deux résultats qu'il faut ensuite réconcilier à la main et personne ne parle de cette partie-là dans les vidéos.

## Les workflows, et quand il ne faut pas d'agent

L'inventaire, on compte le sec, puis le frais, puis le surgelé, on reporte et on commande ce qui manque, et c'est le même geste à chaque fois depuis des années. Le planning, on le commence le dimanche soir, et le lundi quelqu'un pose un congé, le mardi quelqu'un est malade, et on reprend tout parce qu'un samedi soir sans équipe n'est pas envisageable, sauf à fermer le restaurant ce soir-là.



*Toujours la même entrée, le même parcours, la même sortie. Personne ne décide en route.*

### Ce que c'est

Un workflow est un enchaînement d'étapes écrit à l'avance. On connaît le déclencheur, l'ordre, et le résultat attendu. Rien n'est décidé en cours de route. Certaines étapes peuvent faire appel à une IA, cela ne change rien à la nature de l'ensemble.

L'inventaire en est un. On compte le sec, puis le frais, puis le surgelé, on reporte et on commande ce qui manque. L'ordre ne change jamais et personne n'a rien à trancher en cours de route.

### Comment choisir entre les deux

Une seule question suffit et elle ne parle pas de technologie : **est-ce que les étapes sont connues d'avance ?**

Si oui, faites un workflow. Il sera moins cher, plus rapide, reproductible à l'identique, et vous saurez toujours ce qu'il a fait. Si non, si l'enchaînement dépend de ce qu'on trouve en route, alors un agent se justifie, avec ses garde-fous.

Dans une petite entreprise, la majorité des besoins réels tombe du premier côté. Relancer les impayés à trente jours, ranger les pièces jointes, produire un récapitulatif hebdomadaire : rien de tout cela ne demande qu'une machine improvise.

#### LES PIÈGES

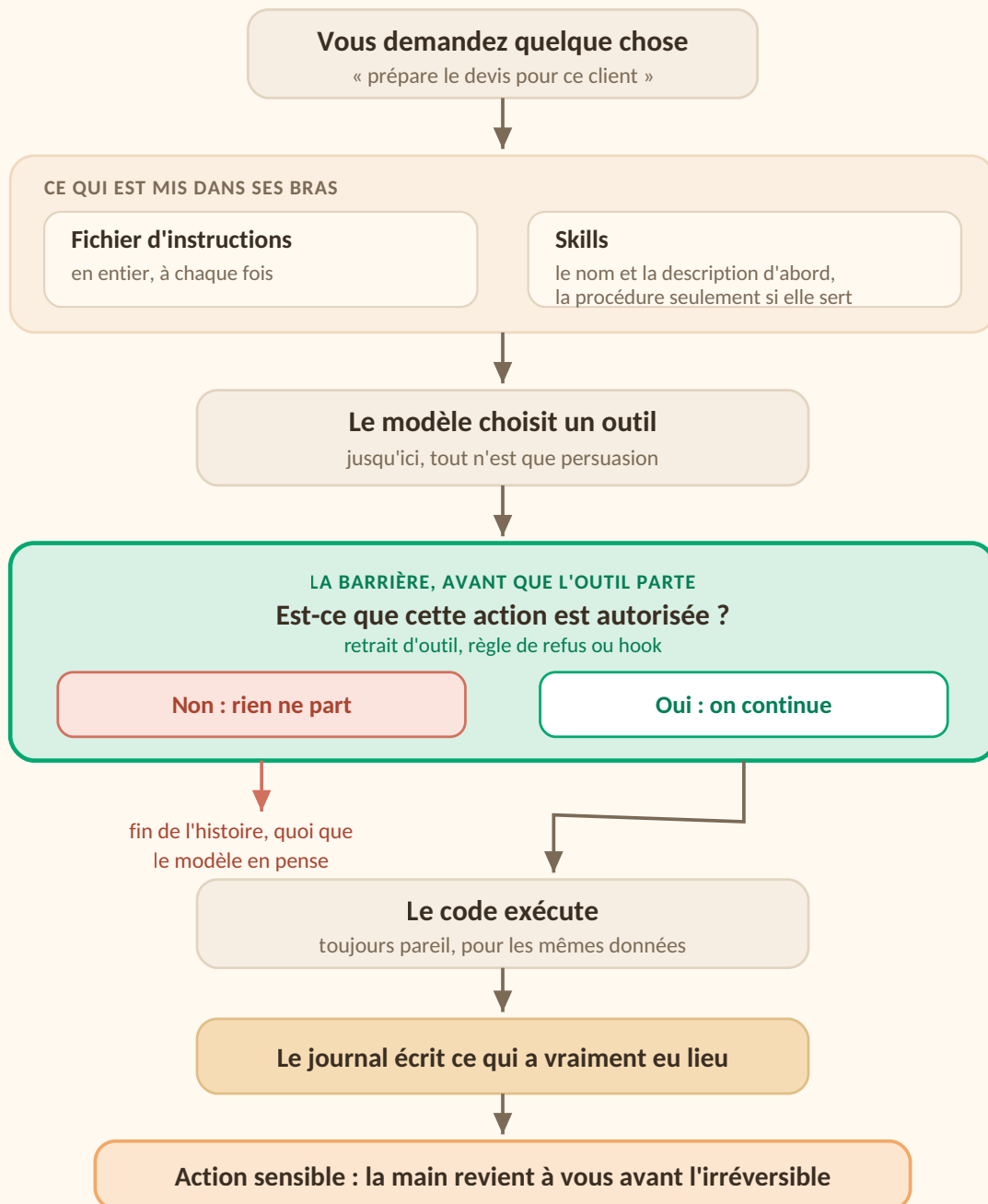
**Prendre un agent pour un workflow trop simple.** Vous payez un modèle pour réfléchir à une question dont vous connaissez déjà la réponse et vous introduisez une part de hasard là où il n'y en avait pas besoin.

**Prendre un workflow pour un besoin qui ne l'est pas.** L'erreur symétrique, et elle se voit bien : le workflow enfle, on rajoute une branche pour chaque exception, et au bout de quelques mois personne n'ose plus y toucher. Le jour où vous comptez plus de branches que d'étapes, c'était un agent qu'il fallait.

**Automatiser un processus bancal.** Une automatisation ne répare rien, elle répète. Si le processus est mauvais à la main, il sera mauvais plus vite et à plus grande échelle.

## L'assistant d'entreprise, ce qui tient l'ensemble

Le relevé des températures se note matin et soir, sur une feuille, même les jours où tout va bien. Personne ne le fait par plaisir, on le fait parce que le jour où un produit pose problème, c'est la feuille qui dit ce qu'il en était et pas le souvenir de celui qui était là. Avec le cahier de recettes et le chef au pass, ça fait trois petites choses qui, ensemble, font qu'un service tourne sans que le patron soit partout à la fois. Les mécanismes de ce guide s'assemblent de la même façon.



Le trajet d'une seule demande, du début à la fin. Tout ce qui est au-dessus de la barrière se discute, tout ce qui est en dessous se vérifie.

## Ce que c'est

C'est ce qu'on appelle parfois un OS agentique, un terme commercial pour une idée simple : au lieu d'un assistant qu'on relance à la main, un socle qui fait tourner les agents. Il distribue les droits, note ce qui a été fait et s'arrête quand c'est à vous de décider.

## Les six pièces, côte à côte

Chacune a eu son chapitre. Mises l'une à côté de l'autre, on voit surtout ce qui les distingue : le moment où elles entrent en jeu.

| La pièce                      | Quand elle entre en jeu                        | Ce qu'elle fait                                     | En cuisine                              |
|-------------------------------|------------------------------------------------|-----------------------------------------------------|-----------------------------------------|
| Fichier d'instructions        | À chaque échange, lu en entier                 | Donne les consignes de base. Il persuade.           | Ce qu'on dit au nouveau le premier jour |
| Skill                         | Seulement quand la tâche correspond            | Apporte une procédure précise                       | La page du cahier de recettes           |
| Outil, souvent branché en MCP | Quand le modèle décide de s'en servir          | Lui permet d'agir, pas seulement de parler          | Passer la commande de citrons           |
| Hook                          | À un moment précis, quoi qu'en pense le modèle | Bloque ou laisse passer. Il empêche.                | Le chef au pass                         |
| Sous-agent                    | Quand l'agent principal délègue                | Fait une partie du travail à part et rend un résumé | Le chef de partie                       |
| Journal                       | Tout le temps                                  | Garde la trace de ce qui a vraiment été fait        | Le relevé des températures              |

Et par-dessus le tout, l'**arrêt de validation**, qui rend la main à l'humain avant l'irréversible.

## Les deux critères qui font la différence

**Est-ce que je peux savoir ce qui a été fait ?** Pas ce que l'assistant raconte avoir fait, mais ce qui s'est réellement passé : quels outils appelés, avec quels arguments, quels fichiers écrits. Sans journal, vous n'avez aucun moyen de vérifier et vous vous retrouvez à croire un récit.

**Est-ce que ça s'arrête tout seul au bon endroit ?** Un système sérieux refuse par défaut et demande. Il ne demande pas la permission une fois sur deux, il la demande systématiquement pour la catégorie d'actions que vous avez définie comme sensible : l'envoi à un client, l'engagement de prix, l'écriture comptable, la suppression.

## LES PIÈGES

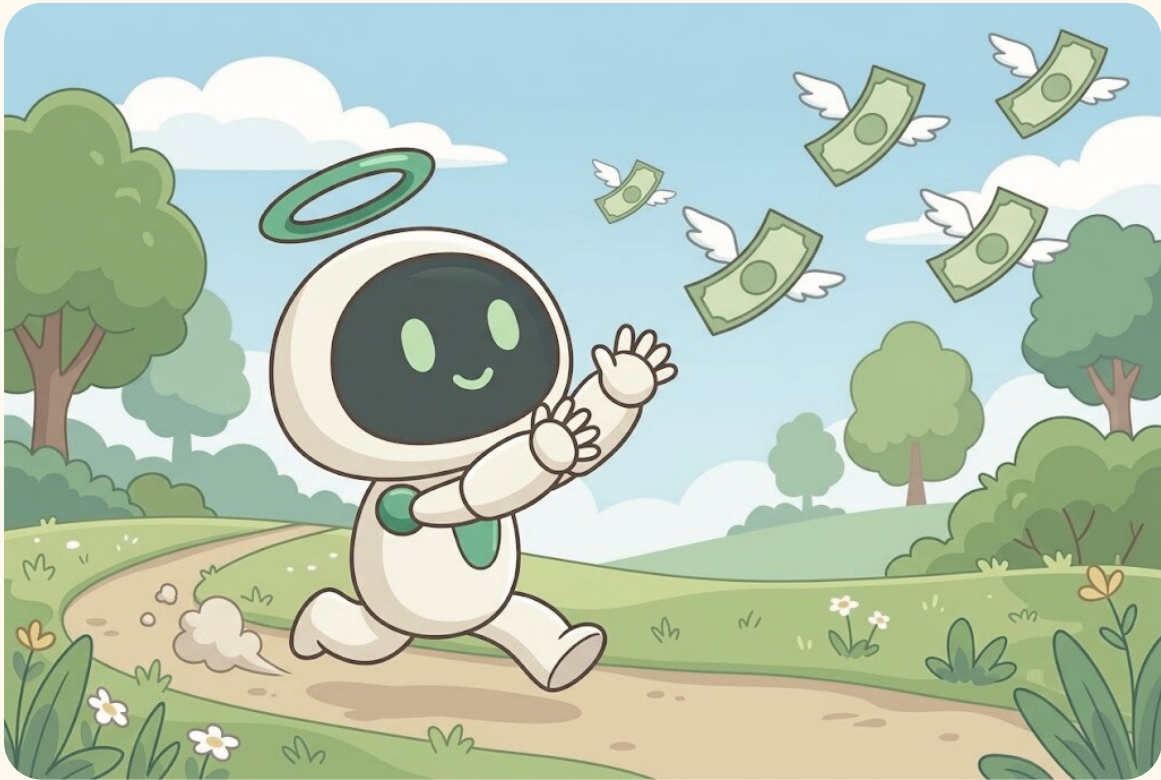
**Le compte rendu qui tient lieu de preuve.** C'est le piège numéro un de tout ce guide et il se voit rarement. Un modèle rédige volontiers « j'ai vérifié dans la base, le client existe » sans avoir appelé le moindre outil. Le récit est plausible, cohérent, et faux. Exigez que ce genre de phrase soit produite par le programme, à partir de ce qu'il a vraiment trouvé, et jamais rédigée par le modèle.

**L'autonomie donnée d'un coup.** Un système qui gagne des droits doit les gagner progressivement, sur preuve. Au début, vous validez tout. Deux mois plus tard, regardez ce que vous n'avez jamais eu à corriger : c'est là, et seulement là, que vous pouvez lui lâcher la main.

**Le travail déjà fait, resservi comme neuf.** Un cas vécu qui vaut avertissement : un système relancé sur une tâche déjà terminée peut ressortir l'ancien compte rendu rédigé au présent, avec toutes ses coches vertes. Rien n'a été exécuté, et pourtant l'écran dit que tout va bien. Si votre système sait rejouer, vérifiez qu'il le dit en toutes lettres et en premier.

## Ce que ça coûte, et comment le savoir

Un extra appelé le samedi matin est payé à l'heure. Un salarié, lui, a un salaire fixe et, heures supplémentaires mises à part, il coûtera la même chose toute l'année. Les deux façons de payer une IA sont exactement celles-là, et aucune démonstration n'en parle.



*Sans plafond ni compteur, ça s'envole, et on s'en aperçoit à la facture.*

### Les deux façons de payer

Avant de parler de montants, il faut savoir lequel des deux systèmes vous concerne, parce qu'ils ne se surveillent pas du tout pareil.

**L'abonnement forfaitaire.** Un prix fixe par mois et par personne. C'est ce que vous payez quand vous utilisez un assistant grand public dans votre navigateur. Vous connaissez la dépense d'avance, c'est confortable. En échange il y a une limite d'usage : au-delà d'un certain volume l'outil ralentit ou vous fait attendre, et cette limite n'est pas toujours annoncée.

**La facturation à l'usage.** Vous payez ce que vous consommez, sans plafond par défaut. C'est le mode d'un système qui tourne tout seul, qui se déclenche à l'arrivée d'un mail ou la nuit. Il suit exactement votre besoin, et c'est aussi lui qui peut déraper pendant que vous dormez.

La règle pratique est simple. Une personne qui travaille à la main avec un assistant prend un abonnement. Un système qui agit tout seul passe à l'usage. Là, le plafond et le compteur ne sont plus facultatifs, et vous verrez plus loin comment les poser.

Le piège classique est de construire sa maquette sur un abonnement, où l'on ne voit jamais passer un centime, puis de découvrir le coût réel le jour de la mise en production. Mesurez sur une tâche réelle avant de vous engager.

## Comment on paie à l'usage

On paie au token, c'est-à-dire au morceau de mot, en entrée comme en sortie. Reprenez le chapitre 1 : tout ce qui est dans le contexte repart à chaque question, donc est payé à chaque question. La mise en cache donne une remise sur ce qui n'a pas changé, c'est tout. Un fichier d'instructions de six pages n'est pas payé une fois, il revient dans toutes vos questions de la journée, et la remise n'efface pas la facture.

## Où part réellement l'argent

- **Les consignes permanentes trop longues.** Le coût le plus bête, et le plus facile à corriger. C'est exactement pour ça que les skills existent.
- **Les agents qui tournent en rond.** Un agent qui n'arrive pas à finir recommence. Chaque tentative coûte et une boucle non plafonnée peut coûter très cher en une nuit.
- **Les serveurs MCP bavards.** Chaque outil disponible occupe de la place, tout le temps, même inutilisé.
- **Les longues sessions.** Plus la conversation avance, plus le contexte est lourd, plus chaque réponse coûte.

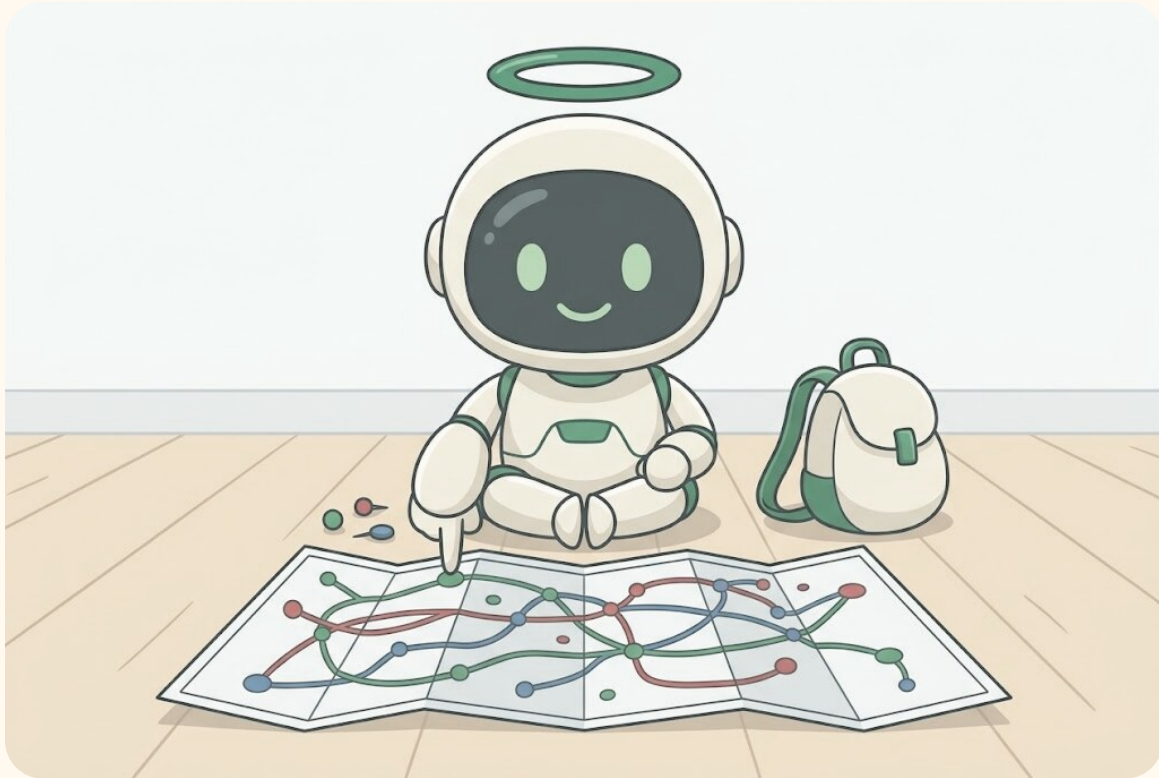
### LES PIÈGES

**Confondre les deux plafonds.** Celui qui vous protège la nuit se règle sur chaque lancement : tant d'essais au maximum, tant d'euros au maximum. Il coupe une boucle folle tout de suite. Le plafond mensuel chez votre fournisseur, lui, ne fait que borner les dégâts du mois. Vérifiez lequel vous avez réellement activé, tous les outils ne proposent pas les deux.

**Découvrir la facture à la fin du mois.** Regardez la consommation dès la première semaine. Les ordres de grandeur se stabilisent vite.

## Dans quel ordre tout ça se met en place

On ne change pas toute la carte d'un coup. On met le plat en test, en plat du jour ou en plat en plus, marqué sur un tableau et pas sur la carte, et on reste attentif aux retours des clients. Il ne passe à la carte qu'ensuite, et c'est seulement là qu'on en teste un second. La suite est rangée dans le même ordre, et elle vous servira aussi à juger le prestataire qui vient vous vendre quelque chose.



*Un premier plat à la carte, et on avance.*

### On commence toujours par écrire

La première chose qui se met en place est le fichier d'instructions, et il tient en vingt lignes : ce que fait la maison, comment on nomme les choses, ce qu'on ne fait jamais. On travaille une semaine avec, on note tout ce qu'il a fallu réexpliquer, et ces notes font la deuxième version. Si l'on vous propose autre chose avant ça, demandez pourquoi.

### Puis un savoir-faire, puis un branchement

La procédure qu'on réexplique le plus souvent devient un skill. Un seul au départ, avec une description soignée, puisque c'est elle qui décide de tout. Le test qui compte : il doit se déclencher quand vous demandez la chose avec vos mots de tous les jours, pas seulement avec ceux de la description.

Vient ensuite un seul serveur MCP, celui qui fait gagner le plus de temps, et l'on regarde ce qu'il coûte avant d'en brancher un deuxième. Un prestataire qui vous en branche six le premier jour vous vend du volume, pas du résultat.

### Ensuite seulement, la porte qui se ferme

Une porte se ferme, une seule, sur l'action que vous ne voulez jamais voir partir sans vous. Dans la plupart des cas c'est une ligne de refus dans les réglages de l'outil, et personne n'a besoin de programmer pour l'écrire. Le hook sert quand la règle dépend du cas, bloquer selon le destinataire ou selon le montant. À partir de là, vous ne demandez

plus un coup de main à l'assistant, vous vous appuyez sur lui. C'est la seule étape qui fait qu'une règle est vraiment tenue. C'est aussi la question à poser à qui vous installe quelque chose : montrez-moi ce qui bloque.

Les sous-agents viennent après, le jour où l'on bute sur le contexte, pas avant. Tant que le problème n'est pas apparu, la solution ne sert à rien, et quelqu'un qui en parle dès la première réunion n'a pas regardé votre besoin.

## Un mail, de bout en bout

Les onze chapitres montrent chacun une pièce. Celui-ci les fait tourner ensemble, sur une seule demande, du moment où elle arrive jusqu'au moment où la réponse part.

Mardi matin. Un mail arrive dans la boîte du restaurant : une table pour vingt personnes un vendredi de décembre, dont deux convives sans gluten. Suivons-le.

### Ce qu'on lui met dans les bras

L'assistant reçoit ce mail et les consignes de la maison. Rien d'autre. Il ne sait pas ce qui s'est dit hier au téléphone, il ne sait pas que le service de samedi dernier a été difficile, il ne sait rien de vous. **C'est le chapitre 1** : tout ce qu'il doit savoir doit lui être redonné, à chaque fois.

Les consignes, elles, sont relues en entier avant chaque échange, et l'une d'elles dit qu'au-delà de douze couverts la demande passe par le patron. C'est le fichier d'instructions du chapitre 4.

### Il ouvre la bonne page

Il reconnaît une demande de groupe et ouvre la procédure groupe, restée fermée jusque-là. Elle dit quoi vérifier et dans quel ordre : le plan de salle, ce qui est déjà pris ce soir-là, les plats compatibles. **C'est un skill**, la page du cahier qu'on n'ouvre que pour ce plat-là, et c'est tout le chapitre 5.

### Il va voir

Il lit le logiciel de réservation. En lecture seule : personne ne lui a donné le droit d'écrire dedans, et ce n'est pas un oubli. Le branchement qui lui permet de le lire a été fait une fois et survivra au prochain changement d'assistant. Un outil, et un branchement qui tient : les chapitres 2 et 3.

Le vendredi en question paraît libre.

Il envoie ensuite un second assistant, plus petit, fouiller le cahier de recettes pour la liste des plats qui tiennent sans gluten. Celui-là démarre à vide, fait son travail et rend sa liste. Un sous-agent, exactement comme au chapitre 7.

### Ce qu'aucune machine ne pouvait savoir

Une table de huit a été promise de vive voix quinze jours plus tôt, un soir de coup de feu, et n'a jamais été saisie dans le logiciel.

L'assistant ne s'est pas trompé. Il a lu le logiciel, le logiciel dit que la salle est libre, et le logiciel a raison au sens où l'information n'existe nulle part. Le journal ne la retrouvera pas, rien ne la retrouvera, parce qu'elle n'a jamais été écrite. **Voilà ce que ce guide n'avait pas encore dit : on ne garde la trace que de ce qui est écrit quelque part.**

Avant de confier quoi que ce soit à une machine, oubliez la question de son intelligence. Demandez-vous plutôt ce qui, chez vous, n'est écrit nulle part et vit seulement dans la tête de quelqu'un.

### Ça bloque

Il prépare la réponse au client. Au moment d'envoyer, rien ne part. Pas parce qu'il hésite, pas parce qu'il a changé d'avis : parce qu'une barrière est posée sur l'action d'envoyer un mail à un client, et qu'elle ne se discute pas. **C'est le chapitre 6**, et c'est le chef qui regarde l'assiette avant qu'elle sorte.

## La main revient à vous

Vous voyez la proposition. À côté, le journal vous montre ce qui a vraiment été consulté, et quand : le logiciel de réservation à 9h12, trois pages du cahier de recettes, rien d'autre. Vous savez donc ce sur quoi la réponse est bâtie, et ce qui n'a pas été regardé. C'est à ça que sert le journal du chapitre 9.

Ce que le journal ne peut pas vous montrer, c'est la table de huit. Ça, c'est vous qui vous en souvenez. Vous corrigez, vous validez, le mail part.

## Et le lundi suivant

L'inventaire se fait seul, toujours dans le même ordre, sans que rien n'ait été décidé. Le planning du mois, lui, ne se fera pas seul, parce qu'un congé posé la veille n'est pas une étape prévisible. **C'est le chapitre 8.** Demandez-vous toujours si les étapes sont prévisibles : c'est ce qui vous évitera les factures les plus inutiles.

### CE QUE CETTE ANNEXE NE FAIT PAS

Elle ne vous dit pas comment installer tout ça. Les étapes d'installation d'un outil précis seraient fausses dans six mois, sur un sujet où ce guide vous répète lui-même que tout bouge vite. Ce qui ne bouge pas, ce sont les pièces et la façon dont elles s'emboîtent, et c'est ce que vous venez de voir tourner une fois en entier.

### POUR VÉRIFIER PAR VOUS-MÊME

Ce guide vous demande de ne rien croire sur parole. Il serait mal venu de faire exception pour lui. Les sources de référence, celles des auteurs des formats et non des commentateurs :

**modelcontextprotocol.io** pour MCP, sa documentation et ses recommandations de sécurité.

**agents.md** pour la convention AGENTS.md.

**agentskills.io** pour le format des skills et la liste à jour des outils compatibles.

**aaif.io** pour l'Agent AI Foundation, qui héberge MCP et AGENTS.md. Le format des skills, lui, est géré en dehors de la fondation.

Pour les hooks, les sous-agents et les fichiers d'instructions, il n'existe pas de source commune : c'est la documentation de l'éditeur que vous aurez choisi qui fait foi, et elle seule.

## Pour finir

Deux choses à retenir de ce guide. La première : un fichier d'instructions persuade, un hook empêche. Le reste en découle, et c'est ce qui sépare un assistant impressionnant d'un système à qui on peut confier du travail.

La seconde : ne croyez jamais un compte rendu sur parole, même bien écrit, même avec des coches vertes. Demandez la trace.

Ce guide est gratuit et le restera. Rien à remplir, aucune adresse à laisser, et vous pouvez le transmettre à qui vous voulez.

Si vous voulez en parler, écrivez-moi.

**Christophe Cazenave** · AuraStack AI · Drôme

[ccazenave@aurastackai.fr](mailto:ccazenave@aurastackai.fr)

Guide gratuit, version 1, octobre 2026. Second volet, après « Comprendre l'IA sans jargon ». Rédigé avec l'aide d'une intelligence artificielle, relu et corrigé par Christophe Cazenave. Illustrations créées avec une intelligence artificielle. Les informations sur les formats et leur gouvernance ont été vérifiées en octobre 2026 ; ce domaine évolue vite, vérifiez l'état courant avant toute décision. Les marques citées appartiennent à leurs propriétaires et sont mentionnées à titre d'exemple, sans lien commercial avec AuraStack AI.