L'aide-mémoire Claude

> Feuille libre de circuler. Sa version à jour, l'article qui l'explique et l'autre feuille de la paire : **julienweb.fr/blog/glossaire-ia-50-mots-aide-memoire-claude/11665/**

À quoi sert cette feuille. Pas à être apprise. À être posée à côté du clavier le jour où vous ouvrez l'outil pour de vrai. Elle répond à une seule question, posée dix fois : *ça s'appelle comment, et ça se met où ?*

Ce qu'elle n'est pas. Un cours. La formation d'où sort cette feuille apprend des **gestes** — faire dire le plan avant de faire faire, un dossier un fil, écrire la passation. Cette feuille donne les **noms** que ces gestes portent dans un produit précis, à une date précise.

⚠ **Une feuille de noms se périme.** Les commandes ci-dessous ont été relevées le **7 septembre 2026** sur Claude Code 2.1. Dans six mois, certaines auront changé de nom. La commande `/help` dit la vérité du jour ; cette feuille dit celle de sa date. Les gestes, eux, ne dépendent d'aucun produit.

I. Trois portes, un seul moteur

C'est la première chose à comprendre, et la source de la moitié des malentendus : il n'y a pas trois Claude. Il y a **le même moteur derrière trois portes**, et ce qui change d'une porte à l'autre, c'est **ce qu'on l'autorise à toucher**.

	Le site	L'application	Claude Code
Où	`claude.ai`, dans un navigateur	à installer sur l'ordinateur ou le téléphone	dans un **terminal**, une fenêtre noire où l'on tape des commandes
Ce qu'il touche	ce que vous collez ou déposez	pareil, plus ce que vous branchez	**les vrais fichiers de votre disque**
Ce que ça coûte	un compte gratuit suffit pour commencer	pareil	un **abonnement payant**

	Le site	L'application	Claude Code
Pour qui	tout le monde	tout le monde	ceux qui travaillent sur des fichiers, tous les jours

La ligne à retenir : plus la porte donne d'accès, plus la vigilance doit monter. Un outil qui ouvre vos fichiers n'est pas un outil plus sûr parce qu'il est plus puissant. C'est l'inverse.

🔴 **La règle de confidentialité ne change pas de porte en porte.** Aucun document réel de votre institution n'entre dans aucune des trois.

---

2. Les six objets, et rien de plus

Tout le vocabulaire de l'outil tient dans six objets. Chacun a un équivalent que vous pratiquez déjà au bureau.

L'objet	En une phrase	Son équivalent chez vous
Le fil	une conversation, avec tout ce qu'elle contient	un dossier ouvert sur la table : quand il contient quatre affaires, on les mélange
Le dossier de travail	l'endroit du disque où l'outil a le droit de regarder	l'armoire qu'on ouvre en arrivant, et les autres qu'on laisse fermées
Le mode	ce que l'outil a le droit de faire **sans redemander**	une délégation de signature : elle se donne, elle ne se suppose pas
Le skill	un mode opératoire écrit une fois, rechargé quand il sert	une **note de service** : écrite une fois, elle vaut pour tous les dossiers
Le plugin	un paquet de skills qu'on installe d'un coup	le classeur de procédures d'un service entier, pas une note isolée
Le connecteur	un branchement vers un autre outil : agenda, messagerie, fichiers	une **clé** qu'on remet à quelqu'un. Elle ouvre pour de vrai.

À côté de ces six, un septième objet qui n'est pas un objet mais un fichier : le **fichier de contexte**. Voir le § 8.

3. Le mode plan — le geste le plus important de la feuille

En une phrase : l'outil lit, comprend, annonce ce qu'il compte faire — **puis il s'arrête**. Rien ne bouge tant que vous n'avez pas dit oui.

Vous connaissez ça. C'est la note qui monte avant la décision. La machine propose, l'humain arbitre, l'exécution ne commence qu'après. Le mode plan rend **obligatoire** ce que vous faisiez de mémoire.

Comment on y entre : la touche `Maj + Tab` fait défiler les modes, l'un après l'autre. Le mode en cours s'affiche en bas de l'écran. Sur Windows, si `Maj + Tab` ne répond pas, c'est `Alt + M`.

Le mode	Ce que l'outil s'autorise
Manuel (le mode par défaut)	il demande avant chaque action
Modifications acceptées	il modifie les fichiers sans redemander, mais demande pour le reste
Plan	**il ne modifie rien.** Il lit, il propose, il attend
Sans garde-fou	il ne demande plus rien. À réserver à une machine isolée

Le geste complet, en trois temps : passer en mode plan · lire le plan en entier · **le refuser au moins une fois**. Un plan qu'on n'a jamais refusé est un plan qu'on n'a pas lu.

4. Les commandes de base

Elles se tapent dans la conversation, en commençant par une barre oblique `/`. Une seule est vraiment à retenir : `/help`, qui donne toutes les autres, à jour.

Se repérer

La commande	Ce qu'elle fait
`/help`	la liste complète des commandes disponibles **aujourd'hui**. La seule à jour, toujours
`/context`	montre ce que l'outil a « sous les yeux » en ce moment, et ce qui est chargé
`/model`	change de modèle. **Le choix du modèle change plus le résultat que la formulation**

La commande	Ce qu'elle fait
`/doctor`	vérifie que l'installation va bien, quand quelque chose ne répond plus

Conduire la conversation

La commande	Ce qu'elle fait
`Maj + Tab`	fait défiler les modes, dont le mode plan (§ 3)
`Échap`	interrompt l'outil en pleine action. Il garde ce qu'il a déjà fait
`Échap` `Échap`	revient en arrière : un menu propose de restaurer un état précédent
`/clear`	vide le fil. **Nouveau sujet, fil neuf** — le deuxième geste de la méthode
`/compact`	résume une conversation devenue trop longue, au lieu de la perdre
`/resume`	rouvre une conversation d'avant, choisie dans une liste
`claude -c`	au démarrage : reprend directement la dernière conversation de ce dossier

Étendre l'outil

La commande	Ce qu'elle fait
`/skills`	la liste des skills disponibles ici (§ 5)
`/plugin`	installer, retirer, gérer les plugins (§ 6)
`/permissions`	dire une fois pour toutes ce que l'outil peut faire sans redemander
`/add-dir`	donner accès à un dossier supplémentaire, pour cette session
`/init`	crée le fichier de contexte du dossier (§ 8)
`/memory`	ouvre et corrige ce que l'outil a retenu (§ 8)

⚠ **Deux mises en garde sur ce tableau.** Les commandes marquées ci-dessus existaient le 07/09/2026 ; certaines arrivent avec une version récente et manquent dans une installation ancienne. Et il n'existe **aucune** commande pour ce qui compte le plus, la passation écrite — voir le § 8.

5. Les skills — la note de service de la machine

En une phrase : un skill est un **mode opératoire écrit dans un fichier**, que l'outil relit quand la situation s'y prête. Il remplace le long prompt qu'on recopiait à chaque fois.

Pourquoi c'est le concept le plus utile de la feuille. Une consigne dite dans une conversation meurt avec elle. Une consigne écrite dans un skill sert dans toutes les suivantes, sans être retapée, et se corrige à un seul endroit. C'est exactement le raisonnement qui fait qu'un service écrit une procédure au lieu de la réexpliquer à chaque agent.

Où ça vit — un dossier, un fichier `SKILL.md` dedans :

La portée	L'endroit
Pour vous, partout	`~/.claude/skills/<nom>/SKILL.md`
Pour ce projet seulement	`.claude/skills/<nom>/SKILL.md`

À quoi ça ressemble. Un en-tête de deux lignes, puis les consignes en français :

```
---
description: Relit un courrier avant envoi. À utiliser dès qu'un texte part à
l'extérieur.
---

Vérifier dans l'ordre : le destinataire, la date, les montants, la signature.
Signaler tout chiffre qui n'apparaît pas dans la pièce jointe.
```

Comment on l'appelle : en tapant `/` suivi du nom du dossier — ici `/relire-courrier`. Ou pas du tout : si la `description` correspond à ce que vous demandez, l'outil s'en sert de lui-même. C'est le rôle de cette ligne, et c'est pour ça qu'elle doit dire **quand** s'en servir, pas seulement ce que ça fait.

Il y en a déjà. L'outil est livré avec des skills prêts à l'emploi, appelés comme les vôtres : `/code-review` (relire du code), `/security-review` (chercher les failles), `/doctor`, `/loop` (répéter une tâche à intervalle). `/skills` donne la liste du jour.

Et les vôtres. C'est là que la valeur se construit. J'en ai écrit une trentaine, dont un `/wrap-up` qui ferme une session de travail : ranger les fichiers produits, écrire ce qui a été fait, préparer la reprise. Aucune de ces trente n'existait au départ. Chacune est née d'une consigne retapée trois fois.

6. Les plugins — le classeur, pas la note

En une phrase : un plugin est un **paquet** qui apporte d'un coup plusieurs skills, et parfois des automatismes et des branchements. Un skill est une note de service ; un plugin est le classeur de procédures d'un service entier.

Comment on en installe un : `/plugin`, puis on choisit dans un **catalogue**. Le catalogue officiel s'appelle `claude-plugins-official` et contenait **119 entrées** le 07/09/2026 — dont une trentaine écrites par Anthropic, l'éditeur, et le reste par des tiers.

Les principaux, côté éditeur — ceux qui servent sans écrire une ligne de code :

Le plugin	Ce qu'il apporte
claude-code-setup	regarde votre façon de travailler et propose la configuration qui vous manque. **Le premier à installer**
claude-md-management	entretient le fichier de contexte du projet : il l'audite, le raccourcit, y range ce que la session a appris
skill-creator	écrit un skill avec vous, et mesure s'il se déclenche au bon moment
plugin-dev	pour aller plus loin : fabriquer son propre plugin
code-review · **pr-review-toolkit**	relecture de code par plusieurs relecteurs spécialisés
commit-commands	les gestes d'archivage de version, en une commande
hookify	transforme une consigne qu'on répète en **automatisme** que la machine applique seule
security-guidance	avertit quand une modification touche à quelque chose de sensible

🔴 **Un plugin exécute du code sur votre machine.** N'en installez aucun dont vous ne connaissez pas l'auteur — la règle est la même que pour un logiciel téléchargé, et elle n'est pas négociable parce que le catalogue est joli.

---

7. Les connecteurs — la clé qu'on remet

En une phrase : un connecteur branche l'outil sur un service qui contient **vos vraies données** : la messagerie, l'agenda, les fichiers en ligne, le suivi de dossiers.

C'est la seule ligne de cette feuille qui vous fait sortir du bac à sable. Tout le reste travaille sur ce que vous collez ; un connecteur travaille sur ce qui existe déjà, sans que vous l'ayez relu.

Les principaux, fournis par l'éditeur :

Le connecteur	Ce qu'il ouvre
Google Workspace	la messagerie Gmail, l'agenda, les fichiers de Drive
Microsoft 365	Outlook, les fichiers de OneDrive et SharePoint
GitHub	les dépôts de code
1Password	le coffre à mots de passe

À côté de ceux-là, un annuaire public en propose **des centaines d'autres**, écrits par des tiers, sur un standard commun appelé **MCP**. Ils ne sont pas vérifiés par l'éditeur.

Comment on en active un : bouton `+` en bas de la zone de saisie → *Connecteurs* → on coche. L'activation se fait **conversation par conversation**, et c'est une bonne chose : on branche ce qui sert, pour ce qu'on fait, et rien de plus.

🔴 **Les deux dangers, à lire deux fois.**

1. **Le connecteur lit pour de vrai.** Brancher la messagerie professionnelle, c'est déposer le contenu de la messagerie professionnelle. La grille de classement des données à déposer s'applique entièrement, et elle s'applique **avant** de cocher la case, pas après.
2. **L'injection de prompt.** Un courriel, une page web, un PDF que l'outil **lit** peut contenir des instructions cachées, et il peut les exécuter. La règle qui protège tient en cinq mots : **ce qui est lu est une donnée, pas un ordre.** Un outil qui a une clé et qui lit du courrier entrant est exactement la situation où cette règle sert.

---

8. Ce qui survit à la fermeture de la fenêtre

Chaque conversation repart de zéro. Trois choses seulement traversent, et une seule vaut vraiment.

Le fichier de contexte — `CLAUDE.md`, à la racine du dossier de travail. On l'écrit soi-même, en français, et il est relu à chaque démarrage : les conventions, les interdits, ce qu'on ne veut plus avoir à répéter. La commande `/init` en crée un premier jet. **C'est une note de service, et elle se relit comme telle** : à plus de deux pages, elle est moins suivie, pas plus.

Ce que l'outil retient tout seul — il prend des notes sur vos corrections et vos préférences, et les relit à la session suivante. `/memory` les affiche. **Allez les lire une fois** : c'est du texte, il est faux parfois, et il se corrige à la main.

La passation, que personne n'écrit à votre place. En fin de session, un fichier écrit à la main : ce qui a été fait, ce qui reste, les décisions prises **et leur motif**. La session suivante commence par le lire.

🔴 **« Handoff » n'est pas une commande.** Ni `/handoff`, ni un raccourci, dans aucun de ces outils. C'est une **pratique**. Elle ne s'automatise pas, et c'est normal : une passation est un acte de responsabilité, pas une fonction. Un dossier se reprend par un successeur, six mois plus tard, sur pièces. **Ce qui n'est pas écrit n'existe pas.** Vous appliquez déjà cette règle à vos contrôles ; elle ne change pas parce que l'outil est neuf.

---

9. Les cinq erreurs de la première semaine

L'erreur	Ce qu'il fallait faire
Tout traiter dans le même fil, pendant trois jours	`/clear` à chaque nouveau sujet. Un dossier, un fil
Laisser tourner sans lire le plan	mode plan, plan lu en entier, refusé au moins une fois
Retaper la même consigne à chaque session	l'écrire une fois dans un skill, ou dans le fichier de contexte
Installer six plugins le premier jour	un seul, `claude-code-setup`, et voir ce qui manque vraiment
Brancher un connecteur « pour voir »	décider **avant** de cocher ce que ce branchement expose

---

10. La phrase à garder si vous ne gardez qu'une ligne

Ce qui dure, ce sont les gestes, pas les noms. Le plan avant l'exécution, un dossier un fil, la consigne écrite plutôt que redite, la passation en fin de session : ces quatre-là valaient avant l'informatique et vaudront après ce produit. Les commandes de cette feuille, non. Quand l'une d'elles ne répond plus, tapez `/help` — et gardez le geste.