

Déplacer Claude vers un autre PC

Comment emporter la mémoire, les règles et les skills — pour n'importe quel projet

Version 2.4 · 27 octobre 2026

Hans Tandt

Table des matières

1. À quoi sert ce guide	3
2. Où Claude range son cerveau	3
2.1 Les cinq choses les plus importantes	4
2.2 Les parties que l'on saute, et à quoi sert chacune	5
2.3 Trois choses différentes, faciles à confondre	6
2.4 Tenez une note de session, et mettez-la à jour à chaque fois	6
3. Avant de copier quoi que ce soit	7
4. Tout récolter avec deux consignes	7
4.1 Les sept questions de l'export	8
4.2 Ce que l'export laisse sur le disque	8
4.3 Vérifier avant de débrancher	9
4.4 Elle écrit ce qui manque	9
5. Ce qu'il ne faut pas emporter	10
6. Tout remettre en place sur le nouveau PC	11
6.1 Les quatre questions de l'import	11
6.2 Elle fusionne, elle n'écrase pas	12
6.3 Le nom du dossier mémoire sera différent	12
6.4 Programmes et variables d'environnement	13
7. Vérifier que cela a marché	13
7.1 S'il manque encore quelque chose	14
8. Liste de contrôle	14
9. Glossaire	14

1. À quoi sert ce guide

Vous travaillez avec Claude sur un PC depuis des mois. Claude a appris votre façon de travailler, ce qu'est ce projet, et quelles erreurs éviter. Vous avez maintenant besoin du même Claude sur un autre PC.

Copier le dossier du projet ne suffit pas. L'essentiel de ce que Claude a appris est rangé ailleurs, et une copie du dossier le laisse sur place. Ce guide montre quoi emporter et comment le remettre en place.

REMARQUE — Valable pour tout projet

Rien ici n'est propre à un projet ou à une entreprise. Suivez les mêmes étapes pour un manuel, un programme ou un dossier de documents.

REMARQUE — Qui l'a créé

Ces prompts et ce manuel ont été créés par Hans Tandt (hanstandt.be). Ils sont gratuits sous licence MIT : vous pouvez les utiliser, les copier et les adapter, à condition de garder cette ligne.

2. Où Claude range son cerveau

Claude range ce qu'il sait à deux endroits. Un seul des deux suit quand vous copiez le dossier du projet.

DANS LE DOSSIER DU PROJET	DEHORS, DANS VOTRE PROFIL UTILISATEUR
copiez le dossier et ceci suit	reste sur place si vous ne le copiez pas
CLAUDE.md	~/ .claude/CLAUDE.md
.claude/rules/	~/ .claude/rules/
.claude/skills/	~/ .claude/skills/
.claude/settings.json	~/ .claude/settings.json
.claude/agents/	~/ .claude/agents/ commands/
.mcp.json	~/ .claude/plugins/
vos propres fichiers de notes	projects/<name>/memory/

L'essentiel du cerveau est à droite.

Copier le dossier du projet n'emporte que la colonne de gauche. La droite est la partie que l'on perd.

La colonne de droite se trouve dans votre profil utilisateur, dans un dossier caché appelé `.claude`. Sous Windows c'est `C:\Users\<votre nom>\.claude`. Il est partagé par tous les projets du PC, et c'est précisément pour cela qu'une copie de projet ne le contient pas.

2.1 Les cinq choses les plus importantes

Quoi	Ce que cela contient	Où cela se trouve
Mémoire automatique	Ce que Claude a compris de vous et du projet, écrit par Claude lui-même	<code>~/.claude/projects/<projet>/memory/</code>
Skills	Vos méthodes : comment produire un manuel, comment vérifier avant de dire que c'est fini	<code>~/.claude/skills/</code>
CLAUDE.md personnel	Vos règles pour tous les projets de ce PC	<code>~/.claude/CLAUDE.md</code>
CLAUDE.md du projet	Les règles de ce projet-ci	Dans le dossier du projet
Votre note de session	Les décisions et leurs raisons, écrites par vous	Dans le dossier du projet

PIÈGE — Les skills sont ce que l'on perd le plus facilement

Les skills vivent hors du projet. Si vous ne copiez que le dossier du projet, tous les skills que vous avez écrits restent sur l'ancien PC. Ce sont de petits fichiers qui portent le plus de travail : emportez-les en premier.

2.2 Les parties que l'on saute, et à quoi sert chacune

Ces cinq-là sont les plus importantes. Le profil en contient généralement davantage, et voici la partie que l'on saute, parce que son nom ne dit rien. Chacune est un petit dossier. Chacune est du travail déjà fait.

Quoi	Ce que cela contient	Ce que vous perdez sans cela
<code>rules/</code>	Des règles personnelles classées par sujet, sorties de votre <code>CLAUDE.md</code> personnel pour le garder court	Elles ont la même force que le <code>CLAUDE.md</code> lui-même. Rien ne signale leur absence.
<code>agents/</code>	Les sous-agents que vous avez écrits — un assistant avec ses propres instructions pour un type de tâche	Ils cessent d'exister. Aucune erreur : le travail se fait de façon ordinaire, en plus lent.
<code>commands/</code>	Vos propres commandes, chacune une instruction enregistrée que vous utilisez souvent	Une habitude cesse de marcher. Vous la tapez et rien ne se passe.
<code>output-styles/</code>	La forme des réponses — la longueur, le ton, le niveau d'explication	Claude ne sonne plus pareil sur le nouveau PC et vous ne savez pas dire pourquoi.
<code>workflows/</code>	Des tâches en plusieurs étapes, enregistrées et lancées d'un coup	Généralement le plus long à reconstruire de mémoire dans ce tableau.
<code>plugins/</code>	Des ensembles de skills et de commandes venus d'une place de marché	Ceux-là s'installent, ils ne se copient pas. Emportez la liste, pas le dossier.
<code>.mcp.json</code>	Les serveurs MCP qu'un projet peut atteindre — les outils extérieurs dont il peut se servir. Celui-ci est dans le projet, pas dans le profil	Le projet perd ses outils et paraît cassé sans raison visible.

REMARQUE — Les mêmes noms existent des deux côtés

Un projet peut avoir son propre dossier `.claude\` contenant `rules\`, `skills\`, `agents\`, `commands\` et `workflows\`, ainsi que `.mcp.json` et `CLAUDE.local.md`. Ceux-là suivent une copie du dossier. Ceux du profil utilisateur, non. Quand l'export liste les deux, vérifiez de quel côté de la ligne chacun se trouve.

2.3 Trois choses différentes, faciles à confondre

	Qui l'écrit	Combien de temps cela dure
Transcription	Enregistrée automatiquement	Supprimée après un certain temps. Ne comptez pas dessus.
Mémoire automatique	Claude, pour lui-même	Reste, mais vit hors du projet
Note de session	Vous, aidé de Claude	Reste, et suit le projet

La note de session est la seule des trois que vous maîtrisez, qui survit, et qu'un nouveau Claude peut lire dès le premier jour. C'est le fichier le plus précieux de la reprise, et c'est celui que personne ne pense à écrire.

2.4 Tenez une note de session, et mettez-la à jour à chaque fois

Une note de session est un fichier unique dans le projet. Elle consigne ce qui a été fait, ce qui a été décidé et pourquoi, et ce qui reste ouvert. Écrivez-la en phrases simples, pour un lecteur qui n'était pas là.

Faites-en une habitude : à la fin de chaque séance de travail, demandez à Claude de mettre la note à jour. Il a toute la séance devant lui et rédigera en une minute ce qui vous prendrait vingt. Relisez ensuite ce qu'il a écrit : c'est vous qui savez si c'est vrai.

À mettre dedans	À laisser dehors
Ce qui a changé, et à quelle date	Tout ce que Claude peut lire dans les fichiers eux-mêmes
Ce qui a été décidé, et la raison	Le récit détaillé de la conversation
Ce qui a été essayé puis écarté, et pourquoi	Les compliments, le remplissage, ce que vous savez déjà
Ce qui reste ouvert, et l'étape suivante	Tout ce dont vous n'êtes pas sûr

ASTUCE — La raison compte plus que la décision

« Nous utilisons LibreOffice pour le PDF » ne vaut pas grand-chose. « Nous utilisons LibreOffice parce que Word se bloque en automatique » empêche le suivant de défaire le choix.

PIÈGE — Ne laissez pas la note devenir un journal

Une note que personne ne lit équivaut à pas de note. Tenez-vous aux décisions, aux raisons et aux points ouverts. Le détail terminé va dans les documents du projet.

3. Avant de copier quoi que ce soit

Faites d'abord ces trois choses sur le PC source. Elles prennent dix minutes et déterminent ce que vous récupérerez vraiment.

1. Vérifiez que le projet a un `CLAUDE.md`. S'il n'y en a pas, demandez à Claude de lancer `/init`, puis relisez et corrigez. C'est le fichier qui explique le projet à un Claude tout neuf.
2. Mettez la note de session à jour. Demandez à Claude de l'actualiser, puis relisez-la. Sur le nouveau PC, c'est la première chose qui sera lue : elle doit être juste.
3. Demandez à Claude de ranger sa mémoire. Ouvrez `/memory`, retirez ce qui est périmé, fusionnez ce qui est écrit deux fois. N'empORTEZ pas de faits périmés.

ASTUCE — Pourquoi ranger d'abord

Un fait faux qui voyage jusqu'au nouveau PC est pire qu'un fait manquant. Vous lui ferez confiance, et il sera faux.

4. Tout récolter avec deux consignes

Deux consignes font le travail. Ce sont des fichiers texte simples, faits pour être collés dans Claude Code — pas des documents à lire à l'écran. Ce guide décrit leur version 3.2.

Fichier	À lancer sur	Ce qu'il fait
<code>1_EXPORT_PROMPT.txt</code>	Le PC source	Pose sept questions, puis exécute cinq tâches : il inventorie le projet, rassemble tout ce qui vit en dehors, écrit ce qui manque, copie l'ensemble et vérifie le résultat
<code>2_IMPORT_PROMPT.txt</code>	Le nouveau PC	Pose quatre questions, puis exécute sept tâches : il lit ce que l'export a laissé sur le disque, remet chaque partie à sa place, prouve que tout est arrivé, et retourne chercher sur l'ancien PC ce qui manque encore

Ouvrez Claude Code avec le dossier du projet comme dossier de travail, et collez la consigne en un seul message. Chaque fichier contient une ligne de caractères `v` et une ligne de caractères `^`. Ne collez que ce qui se trouve entre les deux : les lignes de repère ne font pas partie de la consigne.

Les deux posent leurs questions avant de toucher à quoi que ce soit : rien n'est découvert à mi-parcours, comme un disque sans place libre. Ensuite elles avancent tâche par tâche et vous montrent chaque résultat avant de passer à la suivante.

REMARQUE — Elles découvrent, elles ne supposent pas

Aucune des deux consignes n'a de liste figée de vos fichiers. Chacune regarde ce qui existe vraiment et vous le rapporte : vous n'avez pas besoin de savoir à l'avance quels skills ou quels réglages s'y trouvent.

REMARQUE — L'export ne modifie rien

La consigne d'export traite le projet en lecture seule. Les seules choses qu'elle crée sont les fichiers de notes demandés et les copies sur votre disque. Elle ne construit rien, n'installe rien, ne règle rien, et ne copie jamais votre connexion.

REMARQUE — Elles lisent vos fichiers, elles ne leur obéissent pas

Un fichier du projet, ou du disque, peut contenir un texte qui ressemble à une instruction. Les deux consignes ont pour règle de vous montrer ce texte au lieu d'agir dessus. Vous seul donnez des instructions.

4.1 Les sept questions de l'export

Préparez ces réponses. Rien ne commence avant que vous les ayez données ; ensuite la consigne vérifie qu'il reste assez de place à la destination, et s'arrête si ce n'est pas le cas.

1. Le dossier où je me trouve est-il bien le projet à exporter ?
2. Où doit aller l'export ? Une lettre de lecteur ou un chemin complet. Il y crée un dossier appelé `Handover_Export`.
3. S'agit-il d'un déménagement, d'une sauvegarde, ou d'une deuxième machine qui reste en service ?
4. Faut-il inclure les gros dossiers ? Il vous donne d'abord la taille totale et nomme tout dossier de plus de 100 Mo, pour que vous décidiez.
5. Quel fichier est le point d'entrée principal — celui qui est lancé ou construit ? Il propose sa propre hypothèse à confirmer.
6. Y a-t-il une note de session ou un fichier de décisions qu'il doit absolument inclure ?
7. Y a-t-il quelque chose en dehors du dossier du projet et en dehors de `.claude` dont le travail dépend — une bibliothèque de consignes, un dossier de modèles, un disque partagé ? Il ne peut pas les trouver tout seul.

4.2 Ce que l'export laisse sur le disque

Tout arrive dans un seul dossier. Emportez ce dossier entier vers le nouveau PC, avec son contenu et ses noms tels quels : la seconde consigne vous demande où il se trouve, puis lit dedans.

Handover_Export\	
project\	le projet, sans ce qui ne doit pas voyager
brain\	
skills\	tous les skills de l'ancien PC
memory\	les fichiers .md de la mémoire
rules\ agents\	commands\ output-styles\ workflows\
CLAUDE.md	vos règles personnelles, tous projets
settings.json	réglages et hooks
HANDOVER_MANIFEST.md	chaque fichier, avec une empreinte
HANDOVER_BRAIN.md	ce qui n'a jamais été écrit nulle part
SOURCE_FACTS.md	anciens chemins, programmes, variables
robocopy_log.txt	la preuve de ce qui a été copié

Ce que produit l'export. Ce dossier est la reprise.

Les trois fichiers de notes	À quoi cela sert
HANDOVER_MANIFEST.md	Chaque fichier du projet avec sa taille et une empreinte, pour prouver sur le nouveau PC que vous avez la bonne version et que rien n'a été perdu en route
HANDOVER_BRAIN.md	Les décisions et leurs raisons, ce qui a été convenu de vive voix sans jamais être écrit, les questions restées sans réponse. Celui-là, lisez-le vous-même : c'est la part qu'aucune copie de dossier ne contient
SOURCE_FACTS.md	L'ancien chemin du projet, l'ancien nom du dossier mémoire, les programmes, les variables d'environnement, et ce qui doit être fusionné plutôt que remplacé. C'est le fichier que lit la consigne d'import

REMARQUE — C'est un dossier, pas un fichier zip

Rien n'est compressé ici. La consigne d'import lit `SOURCE_FACTS.md` directement sur le disque, et la vérification avant de débrancher ouvre des fichiers copiés pour prouver qu'ils sont arrivés entiers : les deux sont plus simples sur un dossier ordinaire, et un zip abîmé ne vous laisse rien du tout. Si vous envoyez l'export par le réseau au lieu de le transporter, compressez tout le dossier `Handover_Export` pour le trajet et décompressez-le avant de lancer l'import.

4.3 Vérifier avant de débrancher

L'export se termine par une vérification. Lisez-la, ne la parcourez pas. Tant que le disque est encore dans le PC source, vous pouvez corriger ce qu'elle trouve ; une fois parti, ce disque est peut-être votre seule copie.

PIÈGE — Une copie qui annonce un succès ne prouve rien

Un outil de copie peut annoncer un succès et avoir tout de même écrit un fichier tronqué. L'export ouvre deux ou trois des fichiers copiés — un `SKILL.md` et un `.md` de mémoire — et confirme qu'ils sont complets, ni vides ni coupés.

Elle confirme aussi ceci avant votre départ — et s'il manque quelque chose, c'est le moment de l'apprendre :

- Chacun des trois fichiers de notes est là, avec sa taille
- `project\`, `brain\skills\` et `brain\memory\` sont bien arrivés
- Rien de ce qui ne doit pas voyager n'a atteint le disque — `.credentials.json` avant tout
- Le nombre de fichiers et la taille totale correspondent à ce que la copie a annoncé
- Une liste simple de tout ce qui est sur le disque, pour voir la récolte entière d'un coup

4.4 Elle écrit ce qui manque

Le chapitre 3 vous demande de mettre `CLAUDE.md` et la note de session en ordre avant de commencer. On ne le fait pas toujours, et une note de session maigre voyage aussi mal qu'une note absente.

L'export ne se contente donc plus de le signaler. Sa troisième tâche propose d'écrire ce qui manque, à partir de la séance en cours : un `CLAUDE.md` s'il n'y en a pas, et une nouvelle entrée datée pour la note de session. Elle vous montre chaque ligne avant, et n'écrit rien sans votre accord.

REMARQUE — Elle ne fait qu'ajouter

Elle peut créer un fichier qui n'existe pas, ou ajouter à la fin d'un fichier existant. Elle ne réécrit ni n'effacera jamais une ligne que vous avez écrite. Si vous dites non, elle laisse le fichier tranquille et continue : un manque que vous connaissez n'est pas un problème.

ASTUCE — À faire même si vous ne déménagez pas

Lancez la troisième tâche de la consigne d'export sur un projet qui ne bouge pas, et vous obtenez un `CLAUDE.md` et une note de session écrits pendant que le travail était encore frais. Cela vaut la peine, déménagement ou pas.

5. Ce qu'il ne faut pas emporter

Certaines choses ne doivent jamais être copiées. Elles sont soit privées, soit reconstruites toutes seules sur le nouveau PC, et les copier provoque des pannes.

Ne pas copier	Pourquoi
<code>.credentials.json</code>	Votre connexion. Reconnectez-vous sur le nouveau PC. Jamais sur une clé USB.
Clés API, jetons, mots de passe	Jamais copiés en clair. L'export les nomme et écrit SECRET - set by hand on the new PC. Vous les configurez vous-même là-bas.
<code>.venv</code> , <code>node_modules</code>	Construits pour un seul PC. Refaites-les avec <code>py -m venv</code> et <code>npm install</code> .
<code>__pycache__</code> , <code>build</code> , <code>dist</code> , <code>_internal</code>	Fichiers générés. Ils reviennent tout seuls.
<code>cache</code> , <code>shell-snapshots</code> , <code>telemetry</code> , <code>debug</code> , <code>sessions</code> , <code>file-history</code> , <code>history.jsonl</code>	Fichiers de travail de l'outil lui-même. Sans intérêt pour vous.
<code>_archive</code> , les très gros fichiers <code>.zip</code>	Le plus souvent d'anciens instantanés de ce que vous copiez déjà.
<code>.git</code>	Volumineux, et mieux rétabli par un clone depuis votre dépôt distant. Lisez d'abord l'avertissement ci-dessous.

PIÈGE — Si le projet est un dépôt git

L'export laisse `.git` de côté : l'historique ne voyage donc pas avec lui. Clonez plutôt le projet depuis son dépôt distant sur le nouveau PC.

Si le projet n'a pas de dépôt distant, son historique n'existe que sur l'ancien PC. Copiez `.git` à la main avant d'effacer quoi que ce soit.

6. Tout remettre en place sur le nouveau PC

C'est la seconde consigne qui fait ce travail. Vous préparez le PC, vous collez la consigne, et vous répondez à ses questions. Elle exécute ensuite sept tâches : ce qui manque sur ce PC, les fichiers du projet (en vérifiant le point d'entrée principal contre l'empreinte du manifeste, pour que vous sachiez que vous avez la bonne version), les skills et les réglages, la mémoire, les variables d'environnement, une liste de contrôle, et enfin ce qui manque encore — en vous montrant chaque résultat avant de passer à la suite.

1. Branchez le disque qui contient `Handover_Export`.
2. Créez le dossier où le projet va vivre. Vide, c'est parfait.
3. Ouvrez Claude Code avec ce dossier comme dossier de travail. C'est important : cela fait créer à Claude le dossier mémoire sous le bon nouveau nom.
4. Collez `2_IMPORT_PROMPT.txt` en un seul message, sans les lignes de repère.

ASTUCE — Laissez la consigne faire les étapes minutieuses

Remettre la mémoire en place demande de retrouver un dossier dont le nom a changé avec le chemin. Rétablir les réglages demande de fusionner deux fichiers au lieu d'en remplacer un. Ce sont exactement les étapes à confier à une machine plutôt qu'à faire à la main en fin de journée.

6.1 Les quatre questions de l'import

Elle pose ces quatre questions, puis lit `SOURCE_FACTS.md` et vous montre l'ancien chemin du projet, l'ancien nom du dossier mémoire, les programmes et les variables dont vous aurez besoin, et tout ce qui doit être fusionné plutôt que remplacé. Si le dossier de destination n'est pas vide, elle en liste le contenu et s'arrête pour vous demander.

1. Où se trouve `Handover_Export` ? Une lettre de lecteur ou un chemin complet.
2. Le dossier où je me trouve est-il bien là où le projet doit vivre ?
3. Ce PC est-il neuf, ou a-t-il déjà des skills et des réglages Claude à conserver ?
4. Faut-il installer les programmes manquants, ou seulement les lister ?

6.2 Elle fusionne, elle n'écrase pas

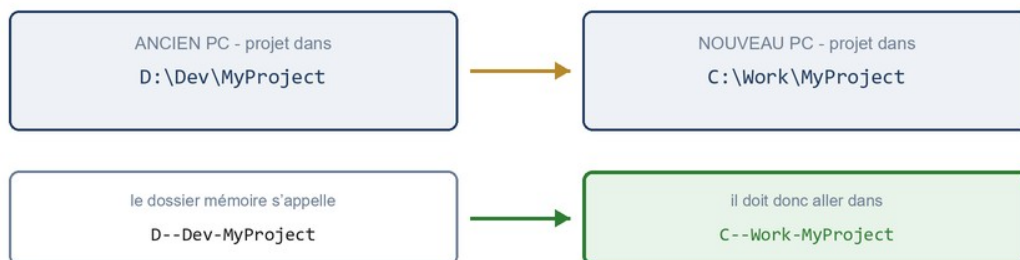
Le nouveau PC a peut-être déjà sa propre installation de Claude. Remplacer les fichiers en bloc jetterait les permissions et les hooks construits là-bas. La consigne est écrite pour ne pas le faire.

Quoi	Comment c'est traité
Skills	Des dossiers : ils se posent côte à côte. Un skill du même nom n'est jamais écrasé : les deux vous sont montrés — dates, tailles, journal des modifications — et vous choisissez.
CLAUDE.md personnel	Un fichier unique. Si ce PC n'en a pas, il est copié. S'il en a un, les deux vous sont montrés en entier et une version fusionnée vous est proposée. Rien n'est écrit avant votre accord.
settings.json	Un fichier unique, traité de la même façon : les deux versions, les clés qui diffèrent, et une fusion proposée à votre accord.
rules\, agents\, commands\, output-styles\, workflows\	Copiés fichier par fichier. Tout ce qui existe déjà ici est laissé de côté, et on vous dit lesquels.
Les fichiers propres à une machine dans un skill	node_modules, __pycache__ et profile.json sont laissés de côté exprès. Ils appartiennent à l'ancien PC — profile.json est le fichier qui nomme qui signe les documents là-bas.
Plugins	Pas copiés. La consigne liste les plugins et les places de marché de l'ancien PC et vous donne la commande pour installer chacun ici. Un plugin s'installe, il ne se copie pas.

6.3 Le nom du dossier mémoire sera différent

C'est l'étape qui échoue le plus souvent. Le dossier mémoire porte le nom du chemin complet du projet. Placez le projet ailleurs et le nom change.

LE DOSSIER MÉMOIRE PORTE LE NOM DU CHEMIN



Copiez les fichiers dans le NOUVEAU nom. L'ancien n'est plus jamais lu.

Copiez les fichiers de mémoire dans le dossier portant le NOUVEAU chemin, pas l'ancien.

La consigne s'en charge dans sa quatrième tâche. Elle trouve le dossier correspondant au chemin de ce PC, vérifie que ce n'est pas l'ancien nom inscrit dans `SOURCE_FACTS.md`, y copie les fichiers `.md`, puis relit `MEMORY.md` et vous dit combien d'entrées sont arrivées. Elle

signale aussi les souvenirs devenus faux ici — une ancienne lettre de lecteur, un ancien chemin, une note sur le PC source — et vous laisse décider.

Pour le faire à la main, ou pour contrôler son travail :

1. Lancez Claude Code une fois dans le dossier du projet. Cela crée le dossier au bon nom.
2. Regardez dans `C:\Users\<votre nom>\.claude\projects\` et repérez le nouveau nom.
3. Copiez les fichiers `.md` de l'ancien PC dans le dossier `memory\` qui s'y trouve.
4. Relancez Claude Code et demandez : « que sais-tu de ce projet ? »

PIÈGE — Copier dans l'ancien nom ne sert à rien

Un dossier portant l'ancien chemin n'est plus jamais lu. Les fichiers y resteront, auront l'air corrects, et seront invisibles pour Claude.

6.4 Programmes et variables d'environnement

Des fichiers arrivés ne font pas un projet qui tourne. Deux choses ne voyagent jamais : les programmes installés et les variables d'environnement. Les deux doivent être remises en place à la main.

La première tâche de la consigne d'import signale les programmes et les paquets manquants et imprime la commande exacte pour chacun. Sa cinquième tâche liste chaque variable d'environnement de l'ancien PC, vous dit si elle est déjà réglée ici et à quelle valeur, et vous donne la commande `setx` avec le chemin corrigé pour ce PC.

PIÈGE — Une nouvelle variable demande un nouveau terminal

`setx` écrit bien la variable, mais une fenêtre déjà ouverte ne la verra jamais. Fermez Claude Code et rouvrez-le ensuite.

7. Vérifier que cela a marché

Ne supposez rien. Demandez, et comparez avec ce que vous avez apporté.

PIÈGE — Relancez avant toute vérification

La mémoire n'est lue qu'au démarrage d'une session. Fermez Claude Code et rouvrez-le d'abord, sinon une installation parfaitement correcte répondra comme si elle ne se souvenait de rien.

Demandez à Claude	Vous devez obtenir
« Que sais-tu de ce projet ? »	Les faits de l'ancien PC, pas une réponse vide
« Quels skills as-tu ? »	La même liste que sur l'ancien PC
« Quels fichiers d'instructions as-tu chargés ? » (/context)	Vos <code>CLAUDE.md</code> du projet et personnel
Lancez la construction ou le test du projet	Le même résultat que sur l'ancien PC

La consigne d'import termine en imprimant quatre listes : ce qui a été copié, ce qui a été laissé de côté et pourquoi, ce qui attend votre décision, et ce qui n'a pas pu être fait du tout. Lisez-la avant de faire confiance au nouveau PC.

ASTUCE — Gardez la clé jusqu'à ce que tout passe

N'effacez ni le PC source ni la clé tant que chaque ligne ci-dessus n'a pas répondu correctement. D'ici là, c'est votre seule copie.

7.1 S'il manque encore quelque chose

Ne prenez pas la liste de contrôle pour la fin. Si une réponse est fausse, ou si le projet ne tourne pas, ce n'est pas fini — et l'ancien PC est toujours là, capable de répondre. La septième tâche de la consigne d'import s'en sert.

1. Elle dit exactement ce qui lui manque : un fichier nommé, un skill, un réglage, un fait.
2. Elle sépare cela en deux : ce que l'ancien PC peut encore donner, et ce que vous seul pouvez faire ici (installer un programme, ajouter un plugin, régler une variable).
3. Pour le premier groupe, elle vous écrit une consigne à coller dans Claude Code sur l'ANCIEN PC, en nommant le projet et ce qu'il lui faut.
4. Vous rapportez la réponse, et la liste de contrôle repasse. Refaites ce tour autant de fois qu'il le faut.

PIÈGE — Voilà pourquoi on n'efface rien encore

Toute la boucle dépend de l'ancien PC, encore là pour qu'on l'interroge. Gardez-le, et gardez le disque, jusqu'à ce que la liste passe. Les deux consignes le disent maintenant dans leur dernière ligne.

8. Liste de contrôle

- Le dossier du projet, sans `.venv`, `node_modules`, `build`, `dist`, `__pycache__`
- La note de session, mise à jour le jour même de la copie
- Le `CLAUDE.md` du projet et vos fichiers de notes (déjà dans le dossier)
- Tout le dossier `~/.claude/skills/`
- `~/.claude/CLAUDE.md` et `~/.claude/rules/`, s'ils existent
- `~/.claude/settings.json`
- `~/.claude/agents/`, `commands/`, `output-styles/`, `workflows/`, s'ils existent
- Les fichiers `.md` de `~/.claude/projects/<projet>/memory/`
- Une liste écrite des plugins installés — la liste, pas le dossier
- Une liste écrite des variables d'environnement et des programmes installés
- `HANDOVER_MANIFEST.md`, `HANDOVER_BRAIN.md` et `SOURCE_FACTS.md`
- `.git`, uniquement si le projet n'a pas de dépôt distant à cloner
- Jamais : `.credentials.json`

9. Glossaire

Terme	Signification
Mémoire automatique	Les notes que Claude écrit pour lui-même sur vous et sur le projet.
Note de session	Un fichier du projet où vous consignez ce qui a été décidé et pourquoi. C'est vous qui l'écrivez, et elle suit le dossier.

Terme	Signification
CLAUDE.md	Un fichier que vous écrivez pour dire à Claude comment travailler. Un par projet, plus un personnel.
Skill	Un dossier contenant une méthode écrite, avec un <code>SKILL.md</code> dedans.
Plugin	Un ensemble de skills et de commandes venu d'une place de marché. Il s'installe, il ne se copie pas.
Handover_Export	Le dossier que la consigne d'export crée sur votre disque. Il contient le projet, le cerveau et trois fichiers de notes.
SOURCE_FACTS.md	Le fichier que l'export écrit et que l'import lit : anciens chemins, programmes et variables.
Profil utilisateur	Votre dossier sur le PC : <code>C:\Users\<votre nom></code> . Il contient le dossier caché <code>.claude</code> .
Transcription	L'enregistrement d'une séance. Supprimé automatiquement après un certain temps.
Variable d'environnement	Un réglage rangé dans Windows, pas dans un fichier. Il ne suit pas une copie.
Hook	Une commande que Claude lance de lui-même à un moment donné, définie dans <code>settings.json</code> .
Serveur MCP	Un outil extérieur qu'un projet est autorisé à atteindre, nommé dans <code>.mcp.json</code> .

Historique des versions : Version 1.0 — 10 septembre 2026 — Première version, remplaçant une consigne propre à un seul projet par une méthode générale. Version 2.0 — 11 septembre 2026 — Mise en accord avec la version 3.0 des deux consignes : les questions posées, ce que l'export laisse sur le disque, la fusion au lieu de l'écrasement, l'historique git, les programmes et les variables. Version 2.1 — 11 septembre 2026 — Tout ce que traitent les deux consignes est désormais expliqué, avec la raison de l'emporter : les dossiers du profil qui n'étaient que listés (2.2), la vérification avant de débrancher (4.3), et la règle selon laquelle les consignes lisent vos fichiers sans leur obéir. Version 2.2 — 26 octobre 2026 — Mise en accord avec les consignes version 3.1 : l'export écrit désormais un `CLAUDE.md` ou une note de session manquante au lieu de seulement la signaler (4.4), il demande les dossiers qui vivent hors du projet (4.1), l'import peut retourner chercher sur l'ancien PC ce qui manque (7.1), et rien ne s'efface tant que cela n'est pas passé. Version 2.3 — 26 octobre 2026 — Dit clairement en 4.2 que l'export est un dossier et non un fichier zip, et quand il vaut tout de même mieux le compresser. Version 2.4 — 27 octobre 2026 — Mise en accord avec les consignes version 3.2 : les mots de passe, clés et jetons sont nommés mais jamais écrits sur le disque (chapitre 5). L'auteur et la licence MIT (chapitre 1). Versions néerlandaise et allemande.