Eigenflow

GUIDE TESTÉ · NOVEMBRE 2026

MCP ou API : la différence en une page

Ce que cache une intégration, une vraie requête tools/list, et comment brancher un serveur MCP à Claude, avec Stripe en exemple.

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté MCP, voici le guide. Le pense-bête de la vidéo d'abord : les API sont faites pour les développeurs, MCP pour les agents IA, et un serveur MCP appelle en général une API en dessous.

Ensuite, la preuve. La commande ci-dessous pose à un vrai serveur MCP public, celui de DeepWiki (une documentation des dépôts GitHub écrite par IA), la question qu'une appli d'IA pose avant de travailler : « quels outils as-tu ? ». Colle-la telle quelle dans le Terminal d'un Mac ou d'un Linux.

À COLLER

```
curl -s https://mcp.deepwiki.com/mcp \
  -H "Content-Type: application/json" \
  -H "Accept: application/json, text/event-stream" \
  -H "MCP-Protocol-Version: 2026-07-28" \
  -H "Mcp-Method: tools/list" \
  -d '{"jsonrpc":"2.0","id":1,"method":"tools/list","params":{"_meta":
{"io.modelcontextprotocol/protocolVersion":"2026-07-
28","io.modelcontextprotocol/clientInfo":
{"name":"essai","version":"1.0"},"io.modelcontextprotocol/clientCapabilities":
{}}}}'
```

Le 22 novembre 2026 à 4 h 47 (heure de Paris), le serveur nous a répondu en 0,53 s : trois outils, chacun avec son nom, sa description et la liste des paramètres qu'il attend. Sans compte ni clé : ce serveur public répond à tout le monde. La partie 5 lit cette réponse champ par champ.

Ce guide reprend ce que la vidéo résume en une minute : la différence entre une API et MCP, le calcul 24 contre 11, ce qu'il y a vraiment dans une intégration (avec les pages de Stripe), comment un client et un serveur se parlent, comment brancher un serveur MCP dans Claude, ce que MCP coûte en contexte et en sécurité, et quand choisir l'un ou l'autre.

Pages officielles relues le 22 novembre 2026. On n'a aucun lien avec Anthropic.

À savoir avant de lire

- MCP se pose par-dessus les API. Un serveur MCP est un programme qui, le plus souvent, appelle l'API d'un service et présente ses fonctions à l'IA comme des outils décrits en toutes lettres.
- Le calcul de la vidéo tient : 3 applis d'IA et 8 outils, c'est 24 connecteurs à écrire sans standard, 11 morceaux de code avec MCP. Les 24 branchements restent, chacun avec sa connexion à autoriser.
- La version du protocole en vigueur est celle du 28 juillet 2026 (2026-07-28). Elle a supprimé la poignée de main du début : chaque requête porte sa version. Le serveur de DeepWiki

répond déjà dans cette version.

- Dans l'appli Claude, un serveur MCP distant se branche par Personnaliser > Connecteurs ; la formule gratuite n'en accepte qu'un. Dans Claude Code : `claude mcp add --transport http <nom> <adresse>`.
- Les outils pèsent dans le contexte. Chez nous, le 17 novembre 2026, 308 outils MCP chargés d'avance faisaient 260 274 tokens avant le premier mot. Claude Code les charge à la demande par défaut : 12 131.
- Le serveur MCP officiel de Stripe est en bêta publique. Il passe par quatre outils génériques pour « une grande partie de l'API », pas pour toute.
- Claude Pro coûte 18 € HT par mois en France (21,60 € TTC), ou 15 € HT par mois payés à l'année (180 € HT).
- Si un agent lit les données de tes clients, la formule compte : en Team, Enterprise ou par l'API, Anthropic s'engage comme sous-traitant (partie 10).

---

1. API ou MCP, la différence en une page

L'API : une porte pour un développeur

Une API (interface de programmation) est la porte qu'un service ouvre aux programmes. Stripe en a une pour les paiements, Google pour Gmail et l'Agenda, Slack pour les messages. Pour s'en servir, un développeur lit la documentation, choisit les adresses à appeler (les endpoints), règle la connexion avec une clé ou un jeton, et écrit le code qui fait exactement l'appel voulu.

Exemple chez Stripe : pour créer un paiement de 10 €, le code envoie une requête POST à `https://api.stripe.com/v1/payment_intents` avec `amount=1000` et `currency=eur`. Le développeur sait d'avance quel appel faire, et il l'écrit en dur dans son programme.

MCP : une prise commune pour les applis d'IA

MCP (Model Context Protocol) est un protocole : une façon commune de parler, que respectent d'un côté des applis d'IA (Claude, ChatGPT, Cursor, VS Code…) et de l'autre des programmes appelés serveurs MCP. Le site officiel le compare à une prise USB-C : « Think of MCP like a USB-C port for AI applications ».

La différence qui compte : avec MCP, l'appli d'IA demande au serveur, pendant qu'elle tourne, la liste de ce qu'il sait faire (la requête `tools/list`). Chaque outil arrive avec un nom, une description en langage courant et la liste de ses paramètres. Le modèle lit ces descriptions, choisit l'outil, remplit les paramètres et demande l'appel. Personne n'a écrit cet appel d'avance.

Ce qui ne bouge pas

L'API reste en dessous. Le serveur MCP de Stripe a par exemple un outil `stripe_api_read`, qui sert à « lire des données avec n'importe quelle méthode GET de l'API Stripe » : il appelle l'API de Stripe à ta place. MCP ajoute une couche qui décrit les outils à l'IA. La doc, les clés, les limites et les centimes de l'API sont toujours là, simplement gérés par celui qui a écrit le serveur.

Le pense-bête, en quatre lignes

- Une API : faite pour un développeur, qui lit la doc au moment de coder et écrit les appels.
- MCP : fait pour un agent IA, qui découvre les outils au moment de travailler et choisit lui-même l'appel.
- Un serveur MCP appelle en général une API en dessous.
- Une tâche fixe avec un seul service : l'API suffit. Une IA qui doit trouver et combiner plusieurs outils : MCP (partie 9).

---

2. 3 × 8 = 24 contre 3 + 8 = 11

Le calcul

Reprends l'exemple de la vidéo : trois applis d'IA (disons Claude, ChatGPT et l'assistant de ton logiciel métier) et huit outils (messagerie, agenda, appli de finance, CRM, Slack, GitHub, base de données, fichiers).

Sans standard, chaque appli parle à chaque outil à sa façon. Il faut un connecteur par paire : 3 × 8 = 24 connecteurs, chacun écrit pour une appli et un outil précis. Un neuvième outil, et ce sont trois connecteurs de plus. Une quatrième appli : huit de plus.

Avec un standard, chaque appli apprend une seule fois à parler MCP (c'est le client), et chaque outil le parle une seule fois (c'est le serveur) : 3 + 8 = 11 morceaux de code. Un neuvième outil ne demande qu'un serveur de plus, que les trois applis savent déjà utiliser.

C'est l'argument d'Anthropic depuis le premier jour. Son annonce du 25 novembre 2024 dit que chaque nouvelle source de données « requires its own custom implementation », et que MCP remplace ces intégrations éparpillées « with a single protocol ». La spécification cite son modèle : le Language Server Protocol, qui a fait la même chose entre les éditeurs de code et les langages de programmation.

Ce que le calcul ne dit pas

- Les 24 branchements existent toujours. Chaque appli doit être reliée à chaque serveur, et chaque liaison a sa connexion. Chez Stripe, une session OAuth ouverte depuis un client MCP dure 14 jours au plus, puis il faut l'autoriser de nouveau. Ce qui change : un branchement se fait en collant une adresse et en se connectant, sans écrire de code.
- Le travail est fait une fois, et par ceux qui connaissent l'outil, quand l'éditeur publie son propre serveur. Stripe, Notion et Sentry ont le leur. Quand l'éditeur n'en publie pas, le serveur vient d'un tiers : à toi de juger s'il est sérieux (partie 8).
- MCP ne fait pas la logique métier. Faire du « montant » de Stripe le « prix » de ton logiciel reste à faire, par le modèle ou par toi.

Pour une PME

Toi, tu branches ; les serveurs, d'autres les écrivent. La question à te poser pour chaque outil que tu utilises : a-t-il un serveur MCP officiel ? Si oui, le brancher se fait sans code (partie 7).

3. Ce qu'il y a dans une intégration : l'exemple de Stripe

La vidéo parle de « pages de code ». Voici ce que chaque étape veut dire, avec les pages de Stripe relues le 22 novembre 2026. C'est ce qu'un développeur doit écrire et tenir à jour, pour chacun des 24 connecteurs.

- Lire la doc. La référence de l'API Stripe couvre les paiements, les clients, les factures, les abonnements, les remboursements, les litiges, la TVA. Chaque objet a ses champs, ses méthodes, ses exemples.
- Régler la connexion, et l'empêcher d'expirer. Stripe identifie chaque requête par une clé secrète : `sk_test_…` pour les essais et, en production, une clé restreinte `rk_live_…` limitée aux droits utiles. Chez Google, la connexion passe par OAuth et le jeton d'accès expire : l'exemple de la doc de Google renvoie `"expires_in": 3920`, soit 3 920 secondes, un peu plus d'une heure. Le code doit le renouveler avec un jeton de rafraîchissement, sinon tout s'arrête au bout d'une heure.
- Faire correspondre les champs. Stripe compte en centimes : « Toutes les requêtes API attendent des montants exprimés dans l'unité mineure de la devise », par exemple 1000 pour facturer 10 USD (ou 10 € : l'euro a aussi deux décimales). Pour le yen, qui n'a pas de centimes, 10 veut dire 10 JPY. Si ton logiciel stocke 10,00, il faut convertir dans les deux sens.
- Parcourir les pages. Une liste Stripe renvoie 10 objets par défaut, 100 au plus (paramètre `limit`). Pour la suite, le code rappelle avec `starting_after` et l'identifiant du dernier objet reçu, tant que `has_more` vaut `true`.
- Rester sous les limites. En production, Stripe accepte 100 requêtes par seconde au total, 25 dans un environnement de test, et 25 par seconde sur chaque endpoint sauf mention contraire. Au-delà : une erreur `429 Too Many Requests`.
- Réessayer sans payer deux fois. Si la connexion coupe pendant un paiement, tu ne sais pas s'il est passé. Stripe prévoit une clé d'idempotence (`Idempotency-Key`) : on réessaie avec la même clé et les mêmes paramètres pendant 24 heures, sans risque de créer un second paiement. Stripe conseille d'espacer les essais de plus en plus (« exponential backoff »).
- Gérer chaque type d'erreur. Le tableau des codes de Stripe compte 400, 401, 402, 403, 404, 409, 424, 429 et 500 à 504, plus des types comme `card_error` (carte refusée) ou `idempotency_error`. Une carte refusée ne se traite pas comme une panne chez Stripe.
- Suivre les versions. La version en cours de l'API Stripe est 2026-11-18.endive. Stripe publie une version par mois sans changement cassant, et deux fois par an une version majeure qui peut demander de modifier le code.

Spécifier les montants dans les requêtes API

Sauf indication contraire, les devises sont exprimées en unités décimales. Toutes les requêtes API attendent des `montants` exprimés dans l'unité mineure de la devise. Par exemple, définissez le `montant` comme suit :

- `1000` pour facturer 10 USD (ou toute autre devise à deux décimales).
- `10` pour facturer 10 JPY (ou toute autre devise à zéro décimale).

La page « Devises » de Stripe, relue le 22 novembre 2026 : 1000 pour facturer 10 USD.

Avec le serveur MCP de Stripe, la connexion, la recherche de la bonne méthode et l'appel lui-même sont faits par le serveur. Les conventions de l'API restent : le modèle doit toujours envoyer 1000 pour 10 €. Quand Claude te demande d'approuver un appel qui écrit, lis le montant.

---

4. Les trois rôles, et comment ils se parlent

Hôte, client, serveur

La spécification en définit trois :

- L'hôte : l'appli d'IA, celle que tu ouvres (Claude Desktop, Claude Code, VS Code). Il gère les autorisations, te demande ton accord et transmet les outils au modèle.
- Le client : la connexion à l'intérieur de l'hôte. La spécification écrit « Connectors within the host application ». Un client parle à un seul serveur : trois serveurs branchés, trois clients.
- Le serveur : le programme qui propose les outils (et, on le verra, des ressources et des prompts). Il tourne sur ton ordinateur ou chez l'éditeur.

L'exemple de la documentation officielle : VS Code ouvre une connexion vers le serveur MCP de Sentry, hébergé chez Sentry, et une autre vers un serveur de fichiers qui tourne sur l'ordinateur. Deux serveurs, deux clients.

Une règle de la spécification qui rassure : un serveur ne voit ni toute la conversation ni les autres serveurs. « Full conversation history stays with the host. »

JSON-RPC 2.0 : la forme des messages

Tous les messages sont du JSON au format JSON-RPC 2.0. Il y en a trois sortes :

- la requête : un numéro (`id`), une méthode (`tools/list` , `tools/call` …) et des paramètres ;
- la réponse : le même numéro, avec un résultat (`result`) ou une erreur (`error`) ;
- la notification : sans numéro, sans réponse attendue, par exemple pour dire que la liste des outils a changé.

stdio et HTTP streamable : le chemin des messages

La spécification définit deux transports :

- stdio, pour un serveur local. L'appli d'IA lance le serveur comme un petit programme sur ton ordinateur et lui écrit sur son entrée standard, une ligne de JSON par message. Pas de réseau. Les identifiants passent par l'environnement du programme, pas par OAuth.
- HTTP streamable, pour un serveur distant. Chaque message part en requête POST vers une seule adresse (pour Stripe, https://mcp.stripe.com) ; le serveur répond par un objet JSON ou par un flux d'événements (SSE). La connexion passe en général par OAuth.

Claude Code accepte aussi l'ancien transport SSE seul, que sa documentation dit abandonné (« deprecated »).

Ce qui a changé le 28 juillet 2026

Jusqu'à la version 2025-11-25, une session commençait par une poignée de main (`initialize`). La version 2026-07-28 l'a supprimée : « MCP is a stateless protocol ». Chaque requête porte sa version et les capacités du client ; en HTTP, la version passe aussi dans l'en-tête `MCP-Protocol-Version` , et la méthode dans `Mcp-Method` . C'est pour ça que la commande du début tient en une requête. Un client qui sait parler les deux versions se rabat sur `initialize` quand le serveur est ancien.

Le serveur de DeepWiki nous l'a montré : sans l'en-tête `Mcp-Method` , il a répondu par l'erreur -32020, que la spécification appelle HeaderMismatch.

---

5. La découverte, en vrai

La question : tools/list

Pour savoir ce qu'un serveur sait faire, le client lui envoie `tools/list` . On l'a fait le 22 novembre 2026 avec le serveur public de DeepWiki, https://mcp.deepwiki.com/mcp. Voici le premier des trois outils de sa réponse, tel quel, mis en forme sur plusieurs lignes par nous :

À COLLER

```
{
  "name": "ask_wiki_question",
  "title": "Ask Wiki Question",
  "description": "Ask any question about a GitHub repository's codebase and get
an AI-powered answer\ngrounded in its DeepWiki.",
  "inputSchema": {
    "properties": {
      "repoName": {
        "anyOf": [
          { "type": "string" },
          { "items": { "type": "string" }, "type": "array" }
        ],
        "description": "GitHub repository or list of repositories (max 10) in
owner/repo format."
      },
      "question": {
        "description": "The question to ask about the repository.",
        "type": "string"
      }
    },
    "required": ["repoName", "question"],
    "type": "object",
    "additionalProperties": false
  }
}
```

Les deux autres :

- `read_wiki_contents` : « View documentation about a GitHub repository. » Un paramètre, `repoName`.
- `read_wiki_structure` : « Get a list of documentation topics for a GitHub repository. » Même paramètre.

La réponse complète fait environ 1,8 ko pour trois outils (on a aussi retiré ici un champ `outputSchema` et un champ `_meta` propre au serveur). Notre première requête, vers 4 h 20, a reçu la réponse au format des versions précédentes du protocole (un flux SSE, sans champ `resultType`). La même question posée à 4 h 43 dans la version 2026-07-28 a renvoyé les mêmes trois outils, avec `"resultType": "complete"`.

Ce que chaque champ veut dire

- `name` : l'identifiant de l'outil, unique sur ce serveur. C'est lui que le client renvoie pour l'appeler. Obligatoire.
- `title` : un nom lisible, pour l'affichage. Facultatif.
- `description` : ce que fait l'outil, en langage courant. C'est ce que le modèle lit pour choisir. Facultative d'après le schéma officiel, présente sur les trois outils de DeepWiki.
- `inputSchema` : les paramètres, au format JSON Schema. Ici, `repoName` (un dépôt au format propriétaire/nom, ou une liste de 10 au plus) et `question`, tous deux obligatoires (`required`), et rien d'autre accepté (`additionalProperties: false`). Obligatoire.

Le modèle remplit lui-même les arguments à partir de ce schéma. Pour la question « comment marche useEffect dans React ? », le client enverrait une requête `tools/call` de cette forme

(notre exemple, jamais envoyé, sans les métadonnées `_meta` que la spécification omet aussi dans ses exemples) :

```
À COLLER

{"jsonrpc":"2.0","id":2,"method":"tools/call","params":
{"name":"ask_wiki_question","arguments":
{"repoName":"facebook/react","question":"How does useEffect work?"}}}
```

Avec une API classique, ce travail (trouver le bon appel, remplir les bons champs) est fait par un développeur, au moment où il écrit le code.

Quand la liste change

Un serveur peut ajouter ou retirer des outils. La spécification prévoit un message pour prévenir le client : `notifications/tools/list_changed`. À deux conditions : le serveur l'a annoncé dans ses capacités (`"listChanged": true`), et le client a ouvert un flux d'écoute (`subscriptions/listen`). Le verbe de la spécification est SHOULD, une recommandation.

Interrogé par `server/discover` (la requête qui donne les capacités d'un serveur), DeepWiki annonce `"tools": {"listChanged": false}` : il ne préviendra pas. Claude Code, lui, recharge la liste quand la notification arrive ; sa doc conseille de reconnecter le serveur depuis `/mcp` pour voir un changement plus vite.

Autre détail de la même réponse : ses instructions citent 22 autres outils marqués « private mode only ». La spécification permet qu'une liste dépende des droits de celui qui la demande : sans compte, tu en vois trois.

---

6. Outils, ressources, prompts

Un serveur peut offrir trois sortes de choses. La spécification les range selon qui décide de s'en servir.

- Les outils (tools) : des actions que le modèle peut lancer. Chercher, créer une facture, écrire un fichier. « Model-controlled » : le modèle choisit, avec ton accord. Méthodes `tools/list` et `tools/call`.
- Les ressources (resources) : des données à lire, chacune repérée par une adresse (URI), comme le contenu d'un fichier ou un historique git. « Application-controlled » : c'est l'appli, ou toi, qui décide de les joindre. Méthodes `resources/list` et `resources/read`. Dans Claude Code, tu les cites avec @, comme un fichier.
- Les prompts : des modèles de messages préparés par le serveur, avec des paramètres. « User-controlled » : c'est toi qui les lances. Dans Claude Code, ils deviennent des commandes de la forme `/serveur:prompt`.

Une nuance par rapport à la vidéo : les ressources sont « des données qu'il peut lire », mais le modèle ne les lit que si l'appli les lui donne. La spécification laisse chaque appli libre de sa façon de faire.

DeepWiki annonce les trois capacités. On n'a demandé que la liste de ses outils : ressources et prompts à tester.

---

7. Brancher un serveur MCP dans Claude, pas à pas

L'exemple : le serveur MCP officiel de Stripe, https://mcp.stripe.com, en bêta publique. On n'a pas ouvert de compte Stripe pour ce guide : les étapes viennent des pages de Stripe et d'Anthropic du 22 novembre 2026, on ne les a pas faites (pas testé).

Dans l'appli Claude (web et bureau)

Le plus court : le connecteur officiel Stripe du répertoire de Claude. D'après Stripe :

1. Sélectionne « Connect to Claude » et ajoute le connecteur à ton compte.

2. Connecte-toi à Stripe et accorde les autorisations dans la fenêtre OAuth.

3. Dans une conversation, clique sur l'icône « + », puis « Connecteurs », et active Stripe.

Pour un serveur absent du répertoire, passe par un connecteur personnalisé. Sur Pro ou Max, d'après l'aide de Claude en français :

1. Va dans Personnaliser > Connecteurs.

2. Clique sur « + Ajouter », puis « Ajouter un connecteur personnalisé ».

3. Donne un nom, colle l'adresse du serveur exactement comme le fournisseur l'écrit, chemin compris, puis « Continuer ».

4. Vérifie les réglages d'authentification que Claude a détectés, puis « Continuer ».

5. Sous Authentification, choisis « Se connecter maintenant », « Se connecter si nécessaire » ou « Pas de connexion ». Sous Client OAuth, garde « Utiliser l'identité publiée de Claude (Recommandé) », sauf si la doc du serveur demande autre chose.

6. Clique sur « Ajouter ». Ensuite, active le connecteur conversation par conversation, avec « + » puis « Connecteurs ».

À savoir :

- La formule gratuite accepte un seul connecteur personnalisé. Sur Team et Enterprise, seuls les propriétaires (et, sur Enterprise, un rôle dédié) peuvent en ajouter ; chaque membre s'y connecte ensuite avec son propre compte.
- La connexion part des serveurs d'Anthropic, pas de ton ordinateur, même avec l'appli de bureau. Le serveur MCP doit donc être joignable depuis Internet, sur le port 443. Un serveur installé sur le réseau interne de ta boîte ne répondra pas tant que le pare-feu n'accepte pas les adresses d'Anthropic.
- Pour modifier un connecteur, il faut le supprimer et le rajouter.

Dans Claude Code

La commande que donne Stripe :

```
À COLLER

claude mcp add --transport http stripe https://mcp.stripe.com/
```

Lance ensuite `claude`, tape `/mcp` et connecte-toi à Stripe (OAuth). Pour vérifier, `claude mcp list` affiche l'état de chaque serveur (« Connected », « Needs authentication »…), `claude mcp get stripe` donne le détail, `claude mcp remove stripe` le retire.

Trois réglages à connaître :

- La portée. Par défaut (portée `local`), le serveur ne sert que dans le dossier où tu l'as ajouté, et reste privé. `--scope project` l'écrit dans un fichier `.mcp.json` à partager avec ton équipe ; `--scope user` le rend disponible dans tous tes projets.
- Une clé au lieu d'OAuth. Pour un agent qui tourne seul, Stripe propose une « clé API d'agent », à garder dans une variable d'environnement et à citer dans `.mcp.json`, jamais en clair dans la commande. À partir du 11 janvier 2027, son serveur MCP refusera les clés secrètes à accès complet.
- La durée. Une session OAuth Stripe dure 14 jours au plus. Ensuite, `/mcp` et reconnexion.

Commence par l'environnement de test

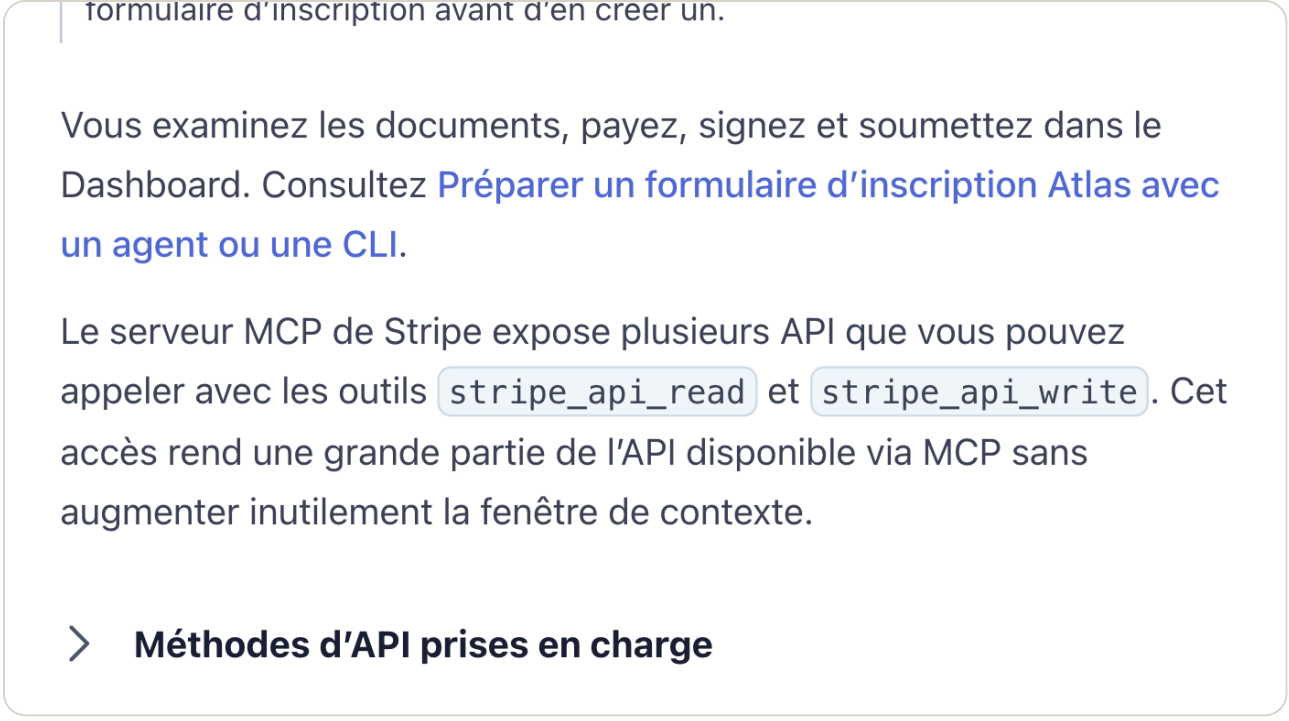
Stripe permet d'autoriser un client MCP sur un environnement de test (sandbox), avec d'autres droits qu'en production. Branche d'abord celui-là. Dans tes paramètres utilisateur Stripe, la rubrique Sessions OAuth liste chaque client autorisé, et tu peux en révoquer un à tout moment.

---

8. La partie honnête

Un serveur n'expose pas toujours toute l'API

Le serveur de Stripe liste 16 outils. Pour l'API elle-même, il en a quatre, génériques : `stripe_api_search` (chercher une méthode par mot-clé), `stripe_api_details` (le détail de ses paramètres), `stripe_api_read` (toute méthode GET) et `stripe_api_write` (POST, PATCH, PUT et DELETE). Stripe écrit : « Cet accès rend une grande partie de l'API disponible via MCP sans augmenter inutilement la fenêtre de contexte. » Une grande partie : la page liste 166 méthodes prises en charge (notre compte). Avant de compter sur un usage, vérifie que ta méthode y est.



La page MCP de Stripe, relue le 22 novembre 2026 : « une grande partie de l'API ».

Autre leçon de cette page : un seul nom d'outil peut couvrir beaucoup d'actions. `stripe_api_write` sert à créer un client comme à rembourser. Stripe prévient que « la confirmation dépend de l'action, et non seulement du nom de l'outil » : un remboursement demande ta confirmation dans le Dashboard (24 heures pour l'approuver), la création d'un client non.

Chez DeepWiki, on l'a vu : trois outils publics, et 22 autres réservés au mode privé, avec un compte.

Le poids des outils dans le contexte

Avant ta première question, l'appli donne au modèle la définition de chaque outil branché : nom, description, paramètres. Ça se compte en tokens, l'unité de mesure du modèle, et ça entame ta limite d'usage.

Notre mesure du 17 novembre 2026, sur notre Claude Code (version 2.1.325, modèle Opus 5.5), dans un dossier vide, avec la phrase « Réponds seulement : ok » :

- tous nos serveurs MCP branchés, recherche d'outils coupée (`ENABLE_TOOL_SEARCH=false`), 308 outils chargés d'avance : 297 334 tokens d'entrée ;
- aucun serveur MCP : 37 060 tokens ;
- le réglage par défaut, outils chargés à la demande : 49 191 tokens.

Les définitions des 308 outils pesaient donc 260 274 tokens, environ 845 par outil, avant le premier mot. Avec le chargement à la demande, 12 131. Un seul de nos connecteurs, de génération d'images et de vidéos, en apportait 175.

Claude Code charge à la demande par défaut : « Only tool names and server instructions load at session start ». Le guide officiel des clients MCP dit pourquoi : tout charger d'avance « wastes tokens, increases latency, and degrades model performance ». Il conseille de passer au chargement à la demande dès que les définitions dépassent 1 à 5 % du contexte.

Dans l'appli Claude, on n'a pas mesuré (à tester). L'aide de Claude conseille de couper, dans le menu « Recherche et outils », les outils qui ne servent pas à la conversation en cours. Branche les

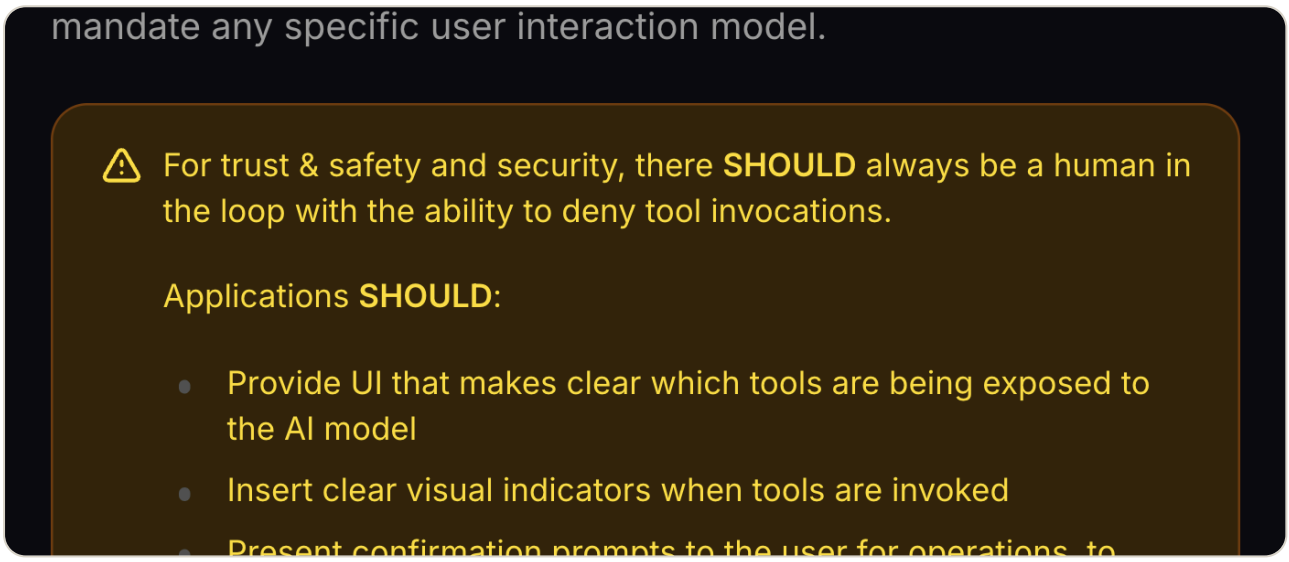
serveurs dont tu te sers, et coupe les autres.

La sécurité

La spécification MCP pose le principe :

« For trust & safety and security, there SHOULD always be a human in the loop with the ability to deny tool invocations. »

Dans une spécification, SHOULD veut dire recommandé : on peut s'en écarter pour une raison valable, le mot de l'obligation est MUST. La vue d'ensemble de la même spécification va plus loin dans ses principes, « Hosts must obtain explicit user consent before invoking any tool », tout en reconnaissant que le protocole ne peut pas les imposer lui-même (« MCP itself cannot enforce these security principles at the protocol level »).



La page Tools de la spécification MCP 2026-07-28 : un humain doit pouvoir refuser chaque appel.

Trois risques concrets :

- L'injection de prompt. Un serveur malveillant, ou un contenu lu par un serveur (un mail, une page web), peut cacher des instructions qui font agir Claude. L'aide de Claude : « Les serveurs MCP malveillants peuvent inclure des instructions cachées qui tentent de faire en sorte que Claude effectue des actions involontaires. » La doc de Claude Code : « Servers that fetch external content can expose you to prompt injection risk. »
- Le mélange de serveurs. Stripe demande d'activer l'approbation des outils dans ton client et de « faire preuve de prudence lorsque vous utilisez le MCP Stripe avec d'autres serveurs afin d'éviter les attaques par injection de prompts ». Un mail piégé lu par ton connecteur de messagerie, et un serveur qui peut rembourser branché dans la même conversation : c'est le scénario à éviter.
- L'approbation en un clic. L'aide de Claude dit de ne cliquer sur « Autoriser toujours » que pour un serveur et un outil de confiance. Elle prévient aussi qu'en mode Recherche, Claude « peut invoquer les outils de vos connecteurs automatiquement sans approbation supplémentaire ».

Ce qu'on en retient :

- Branche des serveurs publiés par l'éditeur de l'outil, ou par quelqu'un que tu connais.
- Donne-leur le moins de droits possible : environnement de test, clé restreinte, autorisations OAuth limitées.

- Garde l'approbation manuelle pour tout ce qui écrit, paie ou envoie. Dans Claude Code, n'ajoute pas de règle qui autorise `mcp__stripe__stripe_api_write` sans te demander.
- Ne mets pas dans la même conversation un serveur qui lit du contenu extérieur et un serveur qui peut payer.

---

9. API ou MCP : comment choisir

La règle de la vidéo : un workflow fixe avec un seul service, appelle l'API ; une IA qui doit trouver et combiner des outils toute seule, MCP. Elle tient, avec une précision pour ceux qui ne codent pas (plus bas).

L'API directe, quand :

- la tâche est toujours la même, au même moment ;
- elle tourne sans personne devant l'écran ;
- le volume compte : avec MCP, chaque appel passe par le modèle, avec ses tokens et son temps ;
- tu veux le même résultat à chaque fois.

MCP, quand :

- une personne travaille dans Claude et pose des questions qui changent ;
- la réponse demande de croiser plusieurs outils ;
- personne n'a le temps d'écrire et d'entretenir un connecteur pour chaque question.

Les deux exemples qui suivent sont les nôtres, pas des cas testés.

Exemple 1 : la boutique en ligne et sa compta (API)

Une boutique qui encaisse avec Stripe veut, chaque nuit, recopier les paiements de la veille dans son logiciel de compta. C'est le même appel tous les jours (la liste des paiements de la journée, page par page), sans décision à prendre, sur des dizaines ou des centaines de lignes. Un script ou un outil d'automatisation qui appelle l'API de Stripe le fait sans modèle et sans tokens, avec la gestion des erreurs et des limites de la partie 3. MCP n'ajouterait que du coût et de l'incertitude.

Exemple 2 : le gérant qui relance ses impayés (MCP)

Le gérant d'une agence de huit personnes facture ses abonnements avec Stripe et écrit à ses clients depuis sa messagerie. Lundi, il demande à Claude : « Quels clients ont une facture impayée depuis plus de 15 jours ? Prépare un brouillon de relance pour chacun. » Mardi, ce sera « Combien d'abonnements ont été annulés ce mois-ci ? ». Les questions changent à chaque fois. Avec le connecteur Stripe et un connecteur de messagerie, Claude cherche la bonne méthode (`stripe_api_search`), lit les factures (`stripe_api_read`) et prépare les brouillons, sans une ligne de code. Le gérant approuve chaque appel, et il relit les brouillons avant tout envoi.

Le cas du milieu

Une tâche fixe, et personne pour coder ? Tu peux la confier à Claude avec un connecteur, à la main, chaque semaine. Ça marche à trois conditions : relire chaque appel, accepter que chaque

passage entame ta limite d'usage, et accepter que deux passages ne rendent pas forcément le même résultat (la CNIL rappelle la « nature probabiliste » de ces systèmes). Dès que la tâche devient quotidienne et sans surprise, elle mérite un vrai script.

---

10. Le cadre

On n'est pas avocats. Ce qui suit se lit au conditionnel, et ne remplace pas l'avis de ton délégué à la protection des données (DPO) si tu en as un.

- Ce que lit l'agent part chez le fournisseur du modèle. Quand Claude lit une facture Stripe par un connecteur, son contenu entre dans la conversation, donc chez Anthropic. Avec un connecteur distant, la connexion part même des serveurs d'Anthropic. Et si le serveur MCP n'est pas celui de l'éditeur de l'outil, les données passent aussi par ce tiers.
- Tu restes responsable des données de tes clients. Pour la CNIL, le sous-traitant est « la personne physique ou morale (entreprise ou organisme public) qui traite des données pour le compte d'un autre organisme ». Un traitement confié à un sous-traitant doit être encadré par un contrat ; la CNIL renvoie aux articles 28.1, 28.3 et 28.10 du RGPD.
- La formule de Claude compte. Sur les formules commerciales (Team, Enterprise, API), Anthropic écrit que le client est le responsable du traitement et qu'Anthropic agit comme sous-traitant (« Processor »), avec un accord de traitement des données (DPA) intégré d'office à ses conditions commerciales, clauses contractuelles types comprises. Anthropic n'y entraîne pas ses modèles sur tes données, sauf si tu rejoins volontairement son programme partenaire. Sur Free, Pro et Max, c'est la politique de confidentialité grand public qui s'applique, et Anthropic Ireland y est responsable du traitement pour les utilisateurs européens. Team coûte 25 $ HT par siège Standard et par mois (20 $ payés à l'année), pour 2 à 150 personnes ; la page française l'affiche en dollars, le montant en euros n'y figure pas.
- Sur Pro, regarde le réglage d'entraînement. S'il est activé, tes conversations peuvent servir à améliorer les modèles. Anthropic précise que ça n'inclut pas « le contenu brut des connecteurs » (« raw content from connectors »), serveurs MCP compris, sauf ce qui est copié directement dans la conversation. L'aide le place dans les paramètres, rubrique Confidentialité.
- La CNIL, dans ses questions-réponses du 18 juillet 2024 sur l'IA générative, conseille d'éviter autant que possible de fournir des données personnelles à un système utilisé par API, dont « la maîtrise » est alors « quasi exclusivement dans les mains du fournisseur ». Elle demande aussi de vérifier si le prestataire peut réutiliser les données, d'associer le DPO et, selon les cas, de faire une analyse d'impact (AIPD).
- En pratique, pour une PME : un connecteur qui touche des données clients plutôt sur une formule commerciale, avec les droits au plus juste. Note ce traitement dans ton registre, et vérifie que ta politique de confidentialité dit à tes clients que leurs données peuvent passer par un outil d'IA.
- On n'a aucun lien avec Stripe ni avec DeepWiki.

---

11. La méthode

Guide écrit le 22 novembre 2026. Requêtes envoyées le même jour au serveur MCP public de DeepWiki (https://mcp.deepwiki.com/mcp) avec curl, sans compte ni clé : `tools/list` vers 4 h 20 (heure de Paris), au format des versions précédentes du protocole ; `tools/list` et `server/discover` à 4 h 43 dans la version 2026-07-28 ; la commande de ce guide à 4 h 47, réponse en 0,53 s. Aucun outil appelé : `tools/call` n'a jamais été envoyé. Une requête au serveur public de Context7 a reçu « Bad Request » et n'est pas exploitée ici.

Poids des outils : notre test du 17 novembre 2026, avec Claude Code 2.1.325 et Opus 5.5, trois passes de `claude -p` dans un dossier vide (tout chargé d'avance, chargement à la demande, aucun serveur), tokens d'entrée lus dans la réponse de Claude Code.

Pages relues le 22 novembre 2026 : la spécification MCP 2026-07-28 (vue d'ensemble, architecture, base, versions, transports, stdio, HTTP streamable, découverte, outils, ressources, prompts, schéma) et la documentation de modelcontextprotocol.io (présentation, architecture, bonnes pratiques des clients) ; les annonces d'Anthropic du 25 novembre 2024 et du 9 décembre 2025 ; la documentation de Stripe (MCP, devises, limites, pagination, idempotence, erreurs, versions, authentification, création d'un PaymentIntent) ; la documentation OAuth de Google ; la documentation MCP et permissions de Claude Code ; l'aide de Claude sur les connecteurs personnalisés, en français et en anglais ; la page des tarifs de Claude depuis la France ; le centre de confidentialité, la politique de confidentialité et le DPA d'Anthropic ; les pages de la CNIL sur le sous-traitant et l'IA générative. Prix en euros hors taxes, TTC calculé avec une TVA de 20 %.

Relu le 25 novembre 2026, avant la mise en ligne : la version en cours de la spécification est toujours 2026-07-28 ; la page MCP de Stripe dit toujours « bêta publique » et « une grande partie de l'API » ; la version en cours de l'API Stripe est toujours 2026-11-18.endive ; la formule Team coûte toujours 25 $ par siège Standard et par mois (20 $ à l'année) ; la commande de ce guide renvoie toujours les trois mêmes outils. La page des tarifs ne s'est affichée qu'en dollars ce soir-là : les prix en euros de Pro sont ceux du 22 novembre.

Pas testé : le connecteur Stripe, dans l'appli comme dans Claude Code ; un connecteur personnalisé dans l'appli ; les ressources et les prompts de DeepWiki ; un appel d'outil ; le poids des outils dans l'appli Claude ; la commande curl sous Windows.
