GOOGLE ADS · CLAUDE · MCP

Brancher Claude sur ton Google Ads

Le setup complet, de zéro au premier appel réel. Les 5 pièces, les 3 pièges qui te coûtent une soirée, et les erreurs traduites en français.

Temps estimé : ~1 h | Niveau : débutant OK | @Lezardoloris

> *L'IA ne remplace pas la stratégie. Elle supprime le temps mort entre la question et la donnée. Ce n'est pas la qualité de tes décisions qui change, c'est le nombre que tu en prends dans une journée.*

Etienne Hurpin · Google Ads e-commerce · Vérifié sur les docs officielles Google et Anthropic, juillet 2026

Ce qu'il y a dedans

Six étapes, deux tableaux de secours. Suis-les dans l'ordre, tu es branché en une heure.

LE PRINCIPE

Pourquoi ça vaut le coup

> Aujourd'hui, entre « je me demande si mes PMax cannibalisent le Search sur les termes de marque » et la réponse, il y a 20 minutes d'UI. Quatre écrans, deux exports, un tableau croisé. Résultat : tu ne poses pas la question.

Tu ne l'évites pas parce qu'elle est mauvaise. Tu l'évites parce qu'elle coûte 20 minutes et que tu en as douze comme ça. Alors tu gardes les trois qui te semblent les plus rentables, et tu jettes les neuf autres. Ces neuf-là, c'est là que sont les trucs que tu ne sais pas encore.

Une fois branché, la question coûte 15 secondes. Tu écris ce que tu veux savoir, Claude écrit la requête, interroge le compte, te rend les lignes. Tu poses les douze. Tu en poses trente. C'est tout ce qui change, et c'est énorme.

Ce que tu auras à la fin de ce guide

Une conversation où tu écris « sors-moi le coût, les conversions et le ROAS de toutes les campagnes sur 30 jours, triées par dépense » et où tu reçois les vraies lignes de ton vrai compte, en 15 secondes, sans ouvrir l'interface Google Ads.

Ce que ça ne fait pas

Ça ne décide pas à ta place. La donnée arrive plus vite, la lecture reste la tienne. Un ROAS de 2,4 n'est bon ou mauvais qu'au regard de ta marge, et ça, aucun serveur ne le sait. Si ta stratégie est mauvaise, tu la valideras juste plus vite.

LA CARTE

Les 5 pièces du puzzle

Tout le monde bloque au même endroit : on croit qu'il y a une seule clé, il y en a cinq, et elles viennent de trois endroits différents. Voilà la carte avant de commencer.

La pièce	Où tu la prends	À quoi elle sert
Developer token	Google Ads API Center	Le droit d'utiliser l'API. 22 caractères. Lié à ton MCC.
Client ID + secret	Google Cloud Console	L'identité de ton application OAuth.
Refresh token	Généré en local, une fois	La preuve que TU autorises cette app à lire TON compte.

La pièce	Où tu la prends	À quoi elle sert
Login customer ID	Ton MCC	Dit à Google par quel manager tu passes.
Le serveur MCP	Ta machine	Traduit « pose-moi cette question » en appel API.

La règle qui évite 90 % des allers-retours

Le developer token vient de **Google Ads**. Le client ID vient de **Google Cloud**. Ce sont deux consoles différentes, deux comptes potentiellement différents. La quasi-totalité des blocages vient de mélanger les deux.

Combien ça coûte à faire tourner

Rien de plus que ce que tu paies déjà. Côté Anthropic, un abonnement **Claude Pro ou Max** suffit : pas besoin de crédits API. Côté Google, l'**API Google Ads est gratuite** en lecture (tu paies tes campagnes, pas les requêtes). Le seul « coût », c'est l'heure de setup ci-dessous.

1 Le developer token

Direction ads.google.com/aw/apicenter. Attention, il faut être connecté avec un compte **manager (MCC)**, pas un compte annonceur simple. Si tu n'as pas de MCC, crée-le d'abord, c'est gratuit et ça prend deux minutes.

Tu remplis le formulaire d'accès API. Le site de la société doit être en ligne, et l'email de contact réellement surveillé (Google écrit dessus). Tu ressors avec une chaîne de 22 caractères.

Le niveau d'accès, c'est TOUT. Et l'info a changé.

Ton token arrive avec un niveau. Ce niveau décide si tu peux lire un vrai compte ou seulement un bac à sable. La plupart des guides encore en ligne racontent la version d'avant.

Niveau	Comptes lisibles	Opérations / jour	Validation
Test Account	Comptes de test seulement	0 en production	Automatique
Explorer	Test + production	2 880	Souvent automatique
Basic	Test + production	15 000	~5 jours ouvrés
Standard	Test + production	Illimité	~10 jours ouvrés

La bonne nouvelle que personne n'a mise à jour

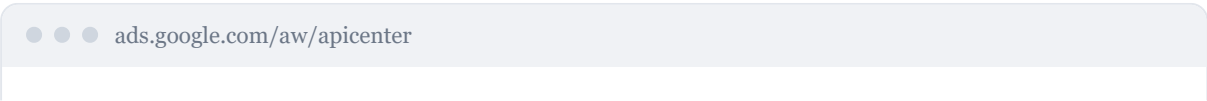
Le niveau **Explorer** lit les comptes de production, jusqu'à 2 880 opérations par jour. Autrement dit : tu n'as plus besoin d'attendre l'accès Basic pour brancher ton vrai compte. Les guides qui disent « demande Basic et attends une semaine » datent d'avant Explorer. Pour un usage perso, 2 880 opérations par jour, tu ne les atteins jamais.

Le piège en miroir

Un token **Test Account** ne lira JAMAIS un vrai compte. Les comptes de production diffusent de vraies annonces, les comptes de test n'en diffusent aucune, et les deux ne communiquent pas. Si tu es en Test, ça renverra `DEVELOPER_TOKEN_NOT_APPROVED` à l'infini.

Vérifie ton niveau, ne le devine pas

Deux pages officielles de Google se contredisent sur le niveau attribué par défaut. L'une dit Explorer, l'autre dit Test. Les deux cas existent réellement. Donc : va lire le niveau réel dans l'API Center, ne suppose pas.



Jeton de développeur		`ABcdeFGH93KL-NOPQ_STUv`
Niveau d'accès	`Explorer`	`← ce que tu veux voir`
Email de contact API		`toi@tondomaine.com`

Si « Niveau d'accès » affiche Test Account Access, arrête-toi ici et demande la montée de niveau via le menu déroulant : rien ne marchera avant.

Ce qu'Explorer ne fait pas

Explorer bloque le Keyword Planner (`KeywordPlanService` , `KeywordPlanIdeaService`), la facturation et la création de comptes. Pour lire et analyser tes campagnes, aucune importance. Si ton but c'est de sortir des volumes de recherche, il te faudra Basic.

2 Google Cloud : l'app OAuth

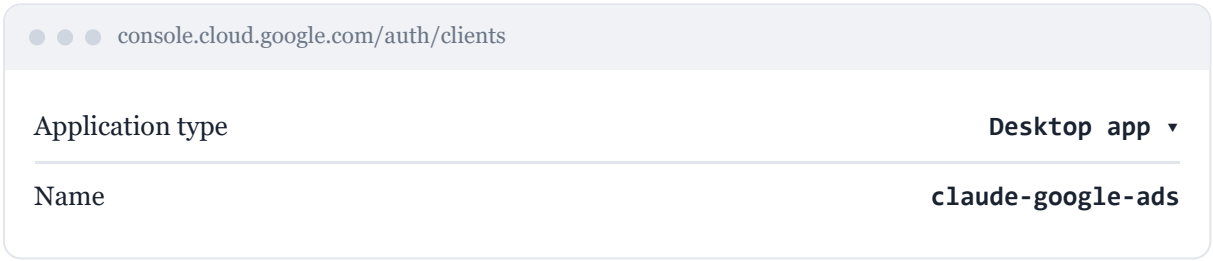
Deuxième console, deuxième logique. Ici tu crées l'*application* qui va parler à l'API.

Activer l'API

Un lien, un clic : console.cloud.google.com/flows/enableapi?apiid=googleads.googleapis.com. Tu choisis (ou crées) un projet, tu valides. Pas besoin d'activer la facturation, l'API Google Ads n'en demande pas.

Créer le client OAuth

Attention, l'interface a été renommée. On ne cherche plus « APIs & Services », mais **Google Auth platform → Clients** → console.cloud.google.com/auth/clients. Create Client → Application type : **Desktop app** → un nom (visible seulement par toi) → Create.



Desktop app : l'accès hors-ligne est activé par défaut, et aucune URL de redirection à configurer. Avec « Web application » tu devrais gérer les redirect URIs à la main. Aucun intérêt ici.

Tu ressors avec un **Client ID** et un **Client secret**. Garde-les de côté.

Le piège des 7 jours, celui qui te fera revenir dans une semaine

Ton projet Cloud a un « publishing status ». S'il reste sur **Testing**, avec un type d'utilisateur **External**, Google émet un refresh token qui **expire au bout de 7 jours**. Le scope Google Ads (`adwords`) n'est pas dans la liste exemptée. Concrètement : ça marche, et sept jours plus tard tout casse avec `invalid_grant` .

Le fix, maintenant : Google Auth platform → **Audience** → bouton **Publish app**. Le statut passe « In production ». C'est le correctif que la doc Google elle-même donne pour `invalid_grant` .

Ne régénère pas en boucle

Maximum **100 refresh tokens** par compte Google et par client ID. Au-delà, Google invalide le plus ancien **sans prévenir**. Pendant un setup où on relance le script dix fois, c'est exactement comme ça qu'on casse le token qui marchait.

3 Le refresh token

Le client ID dit « qui est l'application ». Le refresh token dit « et cet humain-là l'autorise à lire ce compte-là ». C'est la pièce qu'on génère une fois, en local. Le scope, exactement celui-là, rien d'autre :

```
https://www.googleapis.com/auth/adwords
```

Le plus simple, en Python, avec le client secret téléchargé depuis Cloud :

```python
# pip install google-auth-oauthlib
from google_auth_oauthlib.flow import InstalledAppFlow

flow = InstalledAppFlow.from_client_secrets_file(
    "client_secret.json",
    scopes=["https://www.googleapis.com/auth/adwords"],
)
creds = flow.run_local_server(port=0)
print("REFRESH TOKEN:", creds.refresh_token)
```

Ton navigateur s'ouvre, tu choisis le compte Google qui a accès au MCC, tu acceptes. Le refresh token s'affiche dans le terminal.

> **L'écran « Google n'a pas validé cette application »**
> C'est normal, l'app est la tienne et elle n'est pas vérifiée. **Advanced → Go to {nom} (unsafe)**. Tu autorises ta propre application à lire ton propre compte, il n'y a personne d'autre dans la boucle.

> **Le compte Google compte**
> Choisis, sur l'écran de consentement, le compte qui a réellement accès à ton MCC. Erreur classique : générer le token avec le compte perso alors que le MCC appartient au compte pro. Ça donne un token valide qui ne voit aucun compte, et une heure perdue. *(Note : la page OAuth Playground que citent encore beaucoup de tutos renvoie désormais une 404. Ne la cherche pas.)*

4 Le fichier .env

Cinq lignes. Les identifiants vivent ici, en local, et **jamais dans la config MCP** : c'est le fichier que tu ne commites pas.

```
# google-ads-mcp/.env
GOOGLE_ADS_DEVELOPER_TOKEN=ABcdeFGH93KL-NOPQ_STUv
GOOGLE_ADS_CLIENT_ID=xxxx.apps.googleusercontent.com
```

```
GOOGLE_ADS_CLIENT_SECRET=GOCSPX-xxxxxxxxxxx
GOOGLE_ADS_REFRESH_TOKEN=1//0xxxxxxxxxxxxxxx
GOOGLE_ADS_LOGIN_CUSTOMER_ID=9832314241   # SANS tirets
GOOGLE_ADS_ALLOW_WRITE=false              # commence en lecture seule
```

Les tirets. Vraiment.

L'interface Google Ads affiche ton MCC sous la forme `983-231-4241`. L'API le refuse sous cette forme. Le header `login-customer-id` se donne **sans tirets** : `9832314241`. Un .env recopié depuis l'UI échoue, avec un message qui ne parle jamais de tirets.

Commence en lecture seule

`GOOGLE_ADS_ALLOW_WRITE=false`. Tant que tu n'as pas confiance, Claude lit et propose, tu exécutes. Tu ouvriras l'écriture quand tu auras vu 50 requêtes passer proprement. Un serveur qui peut modifier des budgets sur un compte à 2 000 € / jour, ça se mérite.

5 Brancher le serveur

Une seule commande. Deux pièges dedans, tous les deux documentés, tous les deux vicieux.

```
claude mcp add --scope user google-ads \
  -- C:\...\google-ads-mcp\.venv\Scripts\python.exe C:\...\google-ads-mcp\server.py
```

Piège 1 : le double tiret

Le `--` sépare les options de Claude de la commande du serveur. Tout ce qui suit est passé tel quel. Sans lui, Claude Code essaie d'interpréter les arguments de ton serveur comme les siens, et se plante.

Piège 2 : --env juste avant le nom

Si tu utilises `--env`, ne colle jamais le nom du serveur juste après : le CLI lit le nom comme une paire KEY=value et rejette la commande. Ici on n'a pas le problème : les identifiants sont dans le .env, pas dans la config.

Les scopes, en une ligne

Scope	Chargé où	Stocké dans
local (défaut)	Ce projet, toi seul	`~/.claude.json`
project	Ce projet, partagé (git)	`.mcp.json`
user	Tous tes projets	`~/.claude.json`

Pour un serveur Google Ads que tu veux partout, c'est `--scope user`. Et surtout pas `project` : ce fichier part dans git.

Sur Claude Desktop plutôt que Claude Code, c'est un fichier JSON : `%APPDATA%\Claude\claude_desktop_config.json`.

```
{
  "mcpServers": {
    "google-ads": {
      "command": "C:\\...\\google-ads-mcp\\.venv\\Scripts\\python.exe",
      "args": ["C:\\...\\google-ads-mcp\\server.py"],
      "env": {}
    }
  }
}
```

Chemins absolus, toujours. Et le python du .venv, pas le python système : c'est lui qui a les dépendances. Si Claude Desktop ne voit rien, les logs sont dans %APPDATA%\Claude\logs (mcp-server-google-ads.log).

Vérifier : `claude mcp list`, puis dans une conversation `/mcp`. Le serveur doit apparaître connecté. S'il est en échec, lance la commande du serveur à la main dans un terminal : l'erreur Python s'affiche en clair, alors qu'elle est avalée par MCP.

Tu n'as pas encore de server.py ? Pars de l'officiel Google

Le plus rapide pour débuter, c'est le serveur maintenu par Google (lecture seule, pas de refresh token, il suit les versions d'API tout seul) :

```
# cloner + installer
git clone https://github.com/googleads/google-ads-mcp
cd google-ads-mcp
pip install -e .
```

Tu l'authentifies en ADC (`gcloud auth application-default login`) au lieu du refresh token, et l'étape 3 + le piège des 7 jours disparaissent. On compare les deux serveurs en fin de guide.

6 Le premier appel réel

Le moment de vérité. Tu ne codes rien, tu écris une phrase.

```
> liste mes comptes Google Ads
● google-ads · list_accounts
{"ok": true, "login_customer_id": "98XXXXXXXX",
 "accounts": [
   {"id": "98XXXXXXXX", "name": "COMPTE ADMIN", "currency": "EUR", "is_manager": true},
   {"id": "54XXXXXXXX", "name": "[marque 1]", "currency": "EUR", "is_manager": false}
 ]}
```

Sortie réelle de mon serveur, identifiants et marques masqués. Si tu vois ça, les 5 pièces sont bonnes : token, OAuth, refresh, MCC, serveur.

Ensuite, la vraie chose. Tu n'écris pas de GAQL, tu poses la question :

```
> sur le compte X, sors le coût, les conversions et la valeur de conv
  de chaque campagne sur 30 jours, triées par dépense
● google-ads · gaql_query
SELECT campaign.name, campaign.status,
       metrics.cost_micros, metrics.conversions,
       metrics.conversions_value
FROM campaign
WHERE segments.date DURING LAST_30_DAYS
ORDER BY metrics.cost_micros DESC

→ 10 lignes. 15 secondes. Zéro écran Google Ads ouvert.
```

C'est lui qui écrit le GAQL. Toi tu poses la question en français. C'est exactement là que le temps mort disparaît.

Le détail qui trompe : cost_micros

Google renvoie les coûts en **micros**. `cost_micros: 45230000` = **45,23 €**. Divise par 1 000 000. Un serveur correct le fait pour toi, mais si tu lis le JSON brut et que tu paniques sur un budget à 45 millions, c'est ça.

QUAND ÇA CASSE

Les erreurs, traduites

Les messages de l'API Google Ads décrivent le symptôme, jamais la cause. Voilà la traduction, celle qui te fait gagner la soirée.

Le message	Ce que ça veut VRAIMENT dire	Le fix
`DEVELOPER_TOKEN_NOT_APPROVED`	Token en niveau Test, tu tapes sur un vrai compte.	Monte en Explorer ou Basic dans l'API Center.
`USER_PERMISSION_DENIED`	Tu passes par un MCC mais tu n'as pas dit lequel. Erreur n°1.	Renseigne `login-customer-id`, sans tirets.
`invalid_grant`	Refresh token mort. 9 fois sur 10 : projet resté en « Testing », 7 jours écoulés.	Publish app → In production. Puis régénère le token.
`CUSTOMER_NOT_FOUND`	Aucun compte pour cet ID. Souvent des tirets, ou un compte trop récent.	Enlève les tirets. Compte neuf : attends 5 min.
`ACCESS_TOKEN_SCOPE_INSUFFICIENT`	Token généré sans le scope adwords.	Régénère avec `.../auth/adwords`.
`RESOURCE_EXHAUSTED`	Quota du jour atteint (2 880 en Explorer).	Attends, ou monte en Basic.

Le piège que personne ne raconte : ton antivirus

Celui-là m'a coûté une soirée entière. La bibliothèque Google Ads parle en **gRPC**, qui utilise sa **propre pile SSL en C** et ignore le magasin de certificats Windows. Si tu as un antivirus qui fait de l'interception TLS (Avast, Kaspersky, ESET…), il présente son propre certificat, gRPC ne le reconnaît pas, et **tous** les appels meurent en `CERTIFICATE_VERIFY_FAILED`. Pendant ce temps ton navigateur marche parfaitement, donc tu cherches partout sauf au bon endroit.

```
# AVANT d'importer google.ads
import os
from pathlib import Path
os.environ.setdefault(
    "GRPC_DEFAULT_SSL_ROOTS_FILE_PATH",
    str(Path(__file__).resolve().parent / "win_ca_bundle.pem"),
)
```

Ajoute `truststore.inject_into_ssl()` pour le reste des appels HTTP. Les deux sont nécessaires : ils ne couvrent pas la même pile.

CHOISIR

Quel serveur prendre

Critère	Officiel Google	Serveur perso / communautaire
Auth	ADC, pas de refresh token	Refresh token (piège 7 jours)
Écriture	Non	Oui, si tu la codes
Version d'API	Suivie par Google	À ta charge
Qui répare	Google	Toi

Si tu débutes, prends l'officiel

Repo github.com/googleads/google-ads-mcp. Lecture seule, trois outils, pas de refresh token. Tu supprimes d'un coup l'étape 3, le piège des 7 jours et la limite des 100 tokens. Tu perds l'écriture, ce qui au début est plutôt une bonne nouvelle.

Le truc à vérifier avant de copier un serveur communautaire

Beaucoup de serveurs sur GitHub tapent l'API REST avec une version **écrite en dur** (`API_VERSION = "v19"`). La v19 est morte le **11 février 2026** : ces serveurs échouent à 100 % aujourd'hui, et l'erreur ne dit pas « version morte ». La parade : prends un serveur qui passe par la **bibliothèque officielle** `google-ads` (Python) plutôt que du REST à la main. La bibliothèque porte la version, tu la mets à jour avec `pip install -U` .

Deux limites à connaître

Sortie tronquée : Claude Code coupe la sortie d'un outil MCP à 25 000 tokens par défaut (`MAX_MCP_OUTPUT_TOKENS`). Filtre et agrège dans le GAQL, pas après. **Ton compte est exposé :** le serveur expose tes données à l'agent. C'est le but, mais c'est à décider consciemment, surtout sur des comptes clients.

LE VRAI PROPOS

Ce que ça change au quotidien

Avant, ma boucle : intuition → 20 minutes d'UI → réponse → décision. Trois ou quatre fois par jour, les gros sujets seulement. Les petites questions « tiens, je me demande si… », je ne les posais jamais. Maintenant : intuition → 15 secondes → réponse → question suivante. La vraie différence n'est pas la vitesse d'une question, c'est que **j'en pose trente**. Et sur trente, il y en a deux que je n'aurais jamais posées, qui trouvent un truc.

La bascule

Quand une question coûte 20 minutes, tu ne poses que celles dont tu crois déjà connaître la réponse. Quand elle coûte 15 secondes, tu poses celles dont tu n'as aucune idée. Ce sont les seules qui t'apprennent quelque chose.

Trois questions pour commencer

- « Quelles campagnes ont dépensé plus de 100 € sur 30 jours sans une seule conversion ? »
- « Compare le CPA des 7 derniers jours à celui des 30 derniers, campagne par campagne, et montre-moi ce qui dérive. »
- « Sur quels search terms je paie le plus cher sans jamais convertir ? »

Chacune : une phrase, 15 secondes. Chacune : 20 minutes d'UI avant. C'est tout le sujet.

Tu veux que je regarde ton compte ?

Google Ads e-commerce. Je regarde, je te dis ce qui saigne, et si je ne vois rien d'exploitable je te le dis aussi.

Me parler sur WhatsApp

Tu connais quelqu'un que ça peut aider ? Si tu me présentes un client qui signe, je te reverse 10 % du premier mois (setup + mois 1), avec un plancher de 100 €. Versé une fois, sans paperasse.

Vérifié sur les sources primaires (developers.google.com/google-ads/api, code.claude.com/docs, modelcontextprotocol.io) en juillet 2026. Les niveaux d'accès, quotas et versions d'API bougent : revérifie l'API Center avant de conclure que ton setup est cassé.