Eigenflow

GUIDE GRATUIT · EIGENFLOW

Jarvis sur ton Mac

Deux claps, et ta journée se lance : le code de hex.gar adapté au Mac par Claude Code, testé le 29 octobre 2026

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

octobre 2026

Tu as commenté JARVIS. Voici ce qu'on a fait le 29 octobre 2026 au soir sur un MacBook Pro (puce Apple, macOS 26) : récupérer le Jarvis de hex.gar sur GitHub, demander à Claude Code de l'adapter au Mac, puis le lancer et taper deux fois dans les mains. Toutes les commandes et toutes les sorties de ce guide viennent de cette soirée. Versions : Python 3.12.7, Claude Code 2.1.280 avec Claude Opus 5.5, sur l'abonnement Claude (aucune clé d'API).

Ce que fait Jarvis

Un script Python, `jarvis.py`, écoute le micro en continu. Quand il entend deux bruits secs, plus de sept fois plus forts que le bruit de fond de la pièce et espacés de 0,05 à 0,35 seconde, il considère que tu as tapé deux fois dans tes mains.

Il lance alors, une seule fois par démarrage, ta routine : un morceau sur Spotify, Claude dans Chrome, une phrase d'accueil dite à voix haute, puis ton éditeur de code.

L'original est sur GitHub : github.com/hectorg2211/jarvis (400 étoiles et 150 forks le 29/10). Il a été écrit pour Windows. Il place les fenêtres avec les fonctions de Windows, fait parler Jarvis avec ElevenLabs (il faut une clé), ouvre aussi la page BTC de Binance, et finit par Cursor.

Avant tout : le code n'a pas de licence

Le dépôt est public, mais il n'a aucune licence : le champ est vide sur GitHub, on l'a vérifié le 29/10. Dans ce cas, l'auteur garde ses droits. GitHub te laisse lire le code et le copier dans ton compte. Fais-le tourner chez toi, mais ne le republie pas et ne le revends pas, même modifié.

C'est pour ça que ce guide ne contient pas le code. Il te donne le lien, la demande exacte à faire à Claude Code et ce qu'on a mesuré. Ce n'est pas un conseil juridique.

Ce qu'il te faut

- un Mac avec un micro (celui du MacBook suffit) ;

- Python 3 : tape `python3 --version` dans le Terminal pour vérifier ;

- Claude Code, avec un abonnement Claude ;

- Spotify et Google Chrome, si tu gardes les mêmes actions que nous (tout se change, voir plus bas).

Chez nous, du téléchargement au premier lancement : six minutes, dont trois pour Claude Code.

Étape 1 · Récupérer le code

Sur la page du dépôt, clique sur le bouton vert **Code**, puis **Download ZIP**, et double-clique sur le fichier pour l'extraire. Tu peux aussi le faire en trois lignes dans le Terminal, c'est ce qu'on a fait :

À COLLER

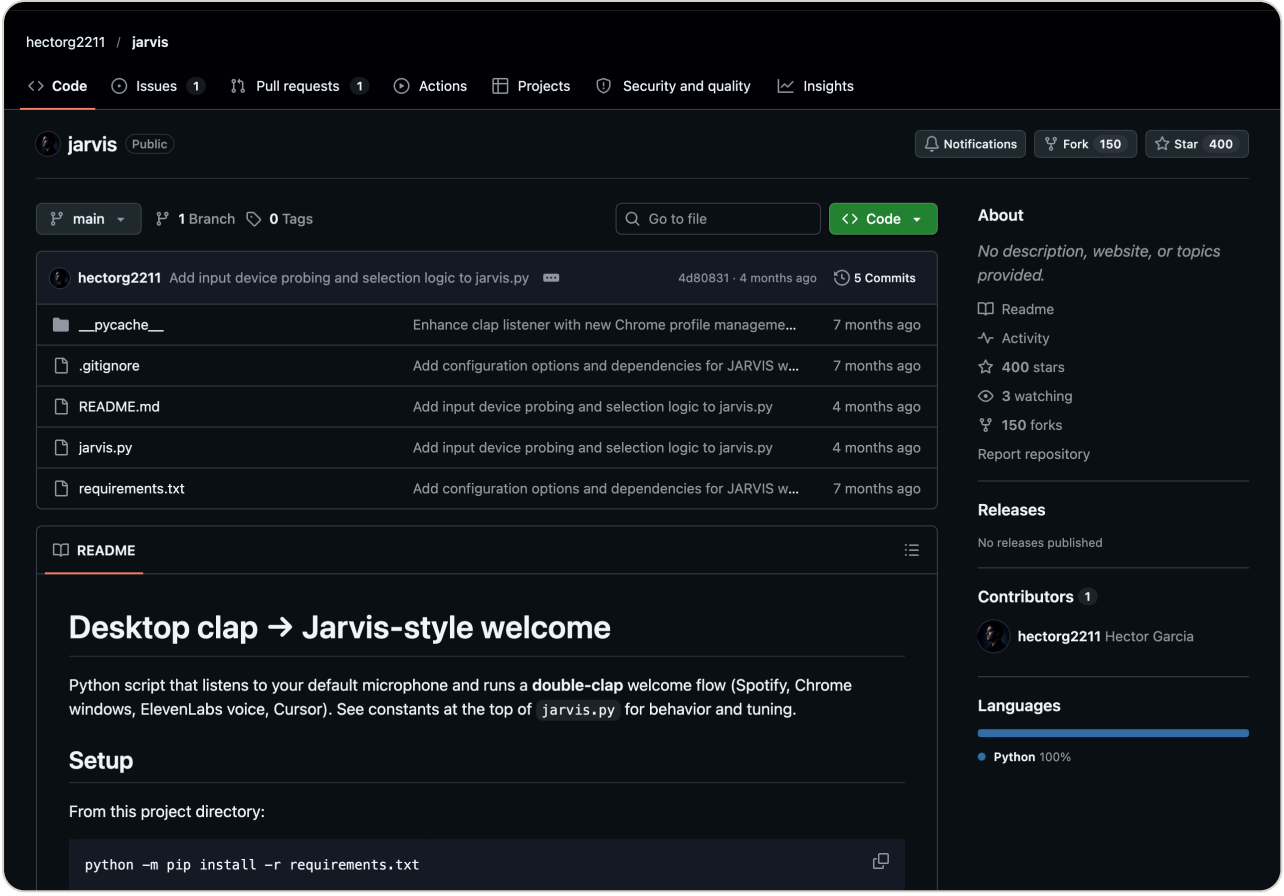
```
cd ~/Desktop
curl -L https://codeload.github.com/hectorg2211/jarvis/zip/refs/heads/main -o
jarvis-main.zip
unzip jarvis-main.zip && cd jarvis-main
```

Le ZIP contient un dossier `__pycache__` avec un fichier Python déjà compilé. Il ne sert à rien et on ne peut pas le relire, alors on l'a supprimé avant de lancer quoi que ce soit :

À COLLER

```
rm -r __pycache__
```

On a aussi lu les 1 051 lignes de `jarvis.py` avant de le lancer : il écoute le micro, ouvre des pages et des applis, et appelle ElevenLabs pour la voix. Rien d'autre.



La page du dépôt le 29/10, vue sans compte : cinq fichiers, 400 étoiles, 150 forks, et aucune licence dans la colonne de droite.

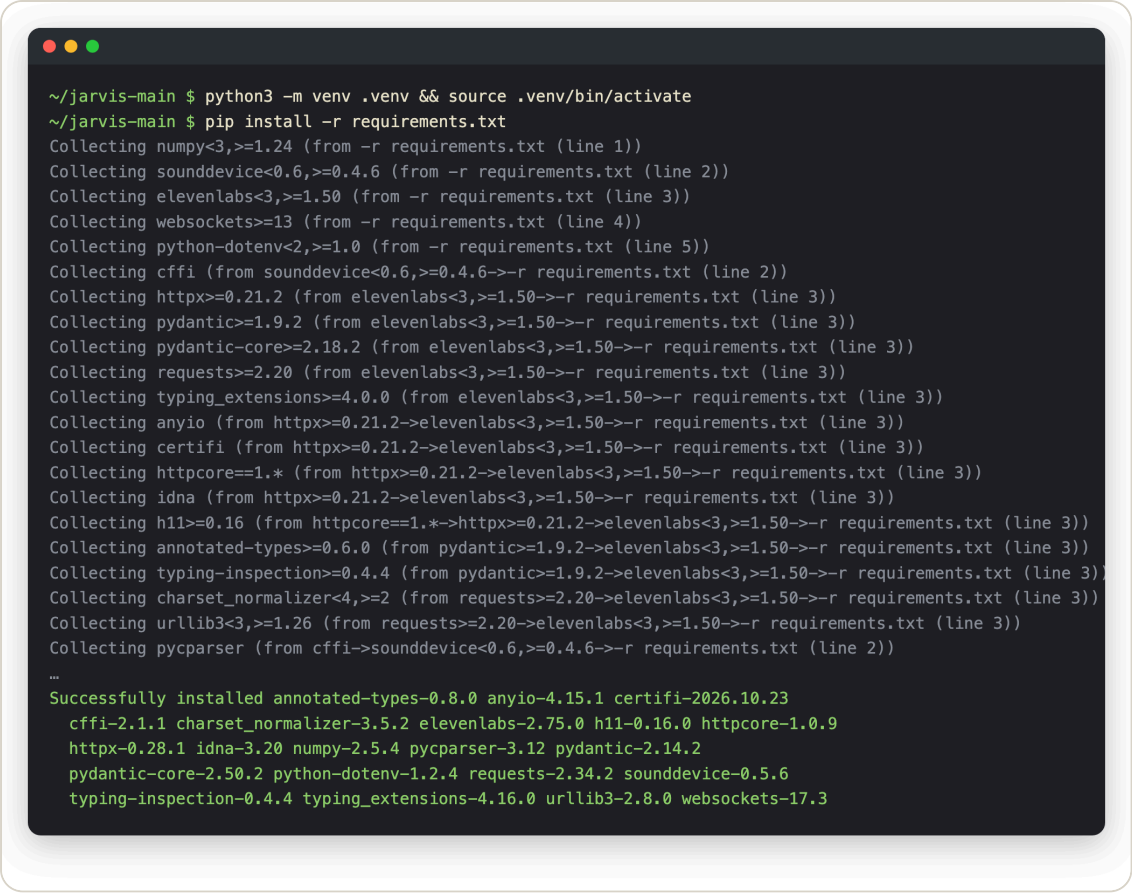
Étape 2 · Installer ce qu'il faut

Toujours dans le dossier `jarvis-main` :

À COLLER

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

La première ligne crée un petit environnement Python rien que pour ce projet, pour ne rien mélanger avec le reste de ton Mac. Chez nous, l'installation a pris moins d'une minute : 21 paquets, dont numpy 2.5.4 et sounddevice 0.5.6, qui servent à écouter le micro.



```
~/jarvis-main $ python3 -m venv .venv && source .venv/bin/activate
~/jarvis-main $ pip install -r requirements.txt
Collecting numpy<3,>=1.24 (from -r requirements.txt (line 1))
Collecting sounddevice<0.6,>=0.4.6 (from -r requirements.txt (line 2))
Collecting elevenlabs<3,>=1.50 (from -r requirements.txt (line 3))
Collecting websockets>=13 (from -r requirements.txt (line 4))
Collecting python-dotenv<2,>=1.0 (from -r requirements.txt (line 5))
Collecting cffi (from sounddevice<0.6,>=0.4.6->-r requirements.txt (line 2))
Collecting httpx>=0.21.2 (from elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting pydantic>=1.9.2 (from elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting pydantic-core>=2.18.2 (from elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting requests>=2.20 (from elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting typing_extensions>=4.0.0 (from elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting anyio (from httpx>=0.21.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting certifi (from httpx>=0.21.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting httpcore==1.* (from httpx>=0.21.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting idna (from httpx>=0.21.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting h11>=0.16 (from httpcore==1.*->httpx>=0.21.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting annotated-types>=0.6.0 (from pydantic>=1.9.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting typing-inspection>=0.4.4 (from pydantic>=1.9.2->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting charset_normalizer<4,>=2 (from requests>=2.20->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting urllib3<3,>=1.26 (from requests>=2.20->elevenlabs<3,>=1.50->-r requirements.txt (line 3))
Collecting pycparser (from cffi->sounddevice<0.6,>=0.4.6->-r requirements.txt (line 2))
…
Successfully installed annotated-types-0.8.0 anyio-4.15.1 certifi-2026.10.23
  cffi-2.1.1 charset_normalizer-3.5.2 elevenlabs-2.75.0 h11-0.16.0 httpcore-1.0.9
  httpx-0.28.1 idna-3.20 numpy-2.5.4 pycparser-3.12 pydantic-2.14.2
  pydantic-core-2.50.2 python-dotenv-1.2.4 requests-2.34.2 sounddevice-0.5.6
  typing-inspection-0.4.4 typing_extensions-4.16.0 urllib3-2.8.0 websockets-17.3
```

La vraie sortie de pip, relancée dans un environnement neuf : les dépendances demandées par le projet, puis tout ce qui a été installé.

Étape 3 · Faire adapter le code au Mac par Claude Code

D'après son code, lancé tel quel sur un Mac, le script ne trouve pas Chrome (les pages partent dans ton navigateur par défaut), ne lance Cursor que si sa commande `cursor` est installée, et reste muet sans clé ElevenLabs. Plutôt que de corriger à la main, on a ouvert Claude Code dans le dossier :

À COLLER

```
claude
```

Puis on lui a collé cette demande, mot pour mot :

À COLLER

```
Ce projet Jarvis a été écrit pour Windows. Adapte jarvis.py pour qu'il
marche sur mon Mac, sans toucher à la détection du double clap.

Au double clap, je veux :
1. le morceau ouvert dans l'appli Spotify ;
2. Claude ouvert dans Google Chrome (https://claude.ai/new) ;
3. la phrase d'accueil dite en français par une voix intégrée du Mac,
   sans ElevenLabs et sans clé ;
4. un nouveau Terminal ouvert dans ce dossier, avec Claude Code lancé
   dedans (la commande `claude`), à la place de Cursor.

Retire Binance. Pour ouvrir les applis, utilise la commande `open` de
macOS. Ne lance pas jarvis.py : je le teste moi-même.
```

Pour le test, notre session était bridée : elle pouvait lire et modifier les fichiers, vérifier la syntaxe et lister les voix du Mac, mais pas lancer le script. Chez toi, une session normale fait l'affaire.

Résultat : **3 minutes 11 secondes**, 23 échanges, l'équivalent de 0,98 $ au tarif de l'API, payé par l'abonnement. Claude Code a :

- traduit la phrase d'accueil et l'a confiée à la voix Thomas du Mac (la commande `say`) ;

- remplacé toutes les ouvertures de Windows par la commande `open` : Spotify dans l'appli, Claude dans Chrome ;

- remplacé Cursor par un nouveau Terminal qui lance `claude` dans le dossier du projet ;

- retiré Binance, ElevenLabs et tout le code propre à Windows ;

- retiré `elevenlabs` et `websockets` de `requirements.txt`, et réécrit le README pour macOS.

Le fichier passe de 1 051 lignes à 447. On a vérifié avec `diff` que la partie qui écoute le micro et reconnaît le double clap est restée identique, au caractère près. Seul un message d'erreur a changé : il parle de macOS au lieu de Windows.

```
# test/log-claude.jsonl · session neuve, mode restreint, Opus 5.5, abonnement

21:07:19  Bash   ls -la && cat jarvis.py && cat *.txt *.md 2>/dev/null | h…
21:07:23  Read   jarvis.py
21:07:23  Bash   cat README.md requirements.txt .gitignore; git log --onel…
21:07:39  Bash   say -v '?' | grep -i fr_; which claude python3; ls -d /Ap…
21:07:43  Bash   say -v '?'
21:07:47  Bash   ls -d /Applications/Spotify.app "/Applications/Google Chr…
21:07:48  Bash   command -v claude
21:08:02  Edit   jarvis.py
21:08:11  Edit   jarvis.py
21:08:12  Edit   jarvis.py
21:08:34  Bash   grep -n "^def _elevenlabs_pcm_sample_rate\|^def main() ->…
21:08:38  Grep   ^def _elevenlabs_pcm_sample_rate|^def main\(\) -> int:|if…
21:08:53  Write  (fichier de travail) actions.py
21:08:57  Write  (fichier de travail) logs.py
21:09:01  Write  (fichier de travail) splice.py
21:09:05  Bash   (fichier de travail) splice.py
21:09:06  Read   jarvis.py
21:09:12  Read   jarvis.py
21:09:56  Write  jarvis.py
21:10:01  Write  requirements.txt
21:10:09  Write  README.md
21:10:15  Bash   python3 -m py_compile jarvis.py && echo OK

terminé en 191 s · 23 tours · équivalent API 0.98 $ (payé par l'abonnement)
```

Chaque action de Claude Code pendant la session, à la seconde : il lit le projet, cherche les voix françaises du Mac, modifie jarvis.py, puis vérifie qu'il compile.

```
# run/jarvis.py, écrit par Claude Code (extraits)

def play_song(uri: str) -> None:
    u = uri.strip()
    if not u:
        return
    _open(["-a", "Spotify", _spotify_app_uri(u)], "Spotify")


def open_claude_in_chrome() -> None:
    if not OPEN_CLAUDE_CODE_IN_CHROME:
        return
    url = (os.environ.get("CLAUDE_CODE_URL") or "https://claude.ai/new").strip()
    _open(["-a", "Google Chrome", url], "Claude in Chrome")

def run_double_clap_actions() -> None:
    """Run outside the mic loop so sleeps do not stall capture."""
    play_song(SONG_URI)
    open_claude_in_chrome()
    if JARVIS_WELCOME_ENABLED and JARVIS_WELCOME_PHRASE.strip():
        delay = max(0.0, JARVIS_AFTER_SONG_DELAY_S)
        if delay:
            time.sleep(delay)
        threading.Thread(target=say_jarvis_welcome, daemon=True).start()
    open_claude_code_terminal()


$ wc -l jarvis-main/jarvis.py run/jarvis.py
  1051 jarvis-main/jarvis.py   (original, Windows)
   447 run/jarvis.py           (adapté au Mac)

$ diff # la boucle qui écoute et détecte le double clap, avant / après
identique
```

Ce qu'il a écrit pour le Mac (extraits), le nombre de lignes avant et après, et la preuve que la boucle d'écoute n'a pas bougé.

Étape 4 · Le lancer, puis taper deux fois dans tes mains

```
À COLLER

python3 jarvis.py
```

Au premier lancement, macOS te demandera sans doute l'accès au micro pour l'appli Terminal : accepte. Le script affiche ensuite ses réglages, la liste de tes micros, puis celui qu'il a choisi. Chez nous :

```
À COLLER

INFO Listening (double clap: 0.05–0.35s apart, rate=44100, block=40 ms,
spike_ratio=7.0, cooldown=0.45s). Ctrl+C to stop.
INFO Using default microphone [2]: Micro MacBook Pro (probe rms=0.00118)
```

Tape deux fois dans tes mains, vite. Si Spotify, Chrome, la voix et Claude Code se lancent, c'est bon. Pour arrêter : `Ctrl+C`.

Ce qu'on a mesuré

On a fait deux essais, et on te les raconte tels quels.

Premier essai, 21 h 10. On voulait jouer un double clap enregistré par les haut-parleurs du Mac, mais le Mac était en sourdine : le clap n'est jamais sorti. À 21 h 11 min 12, ce sont deux bruits secs de la pièce, à 0,33 seconde d'écart, que le micro a captés et pris pour un double clap. Quatre secondes après, un Terminal s'ouvrait, et Claude Code y tournait une seconde plus tard, dans le dossier du projet. Le micro marche, et tu sais maintenant que deux coups secs suffisent à le déclencher.

Essai contrôlé, 21 h 12. Le même double clap (deux vrais claps enregistrés, de la bibliothèque libre VCSL, à 0,24 seconde d'écart), envoyé directement dans une entrée audio virtuelle, BlackHole, que le script écoute grâce au réglage `JARVIS_INPUT_DEVICE=BlackHole`. Ce qui s'est passé :

- 21:12:44 : `Double clap detected (gap=0.255s)` ;

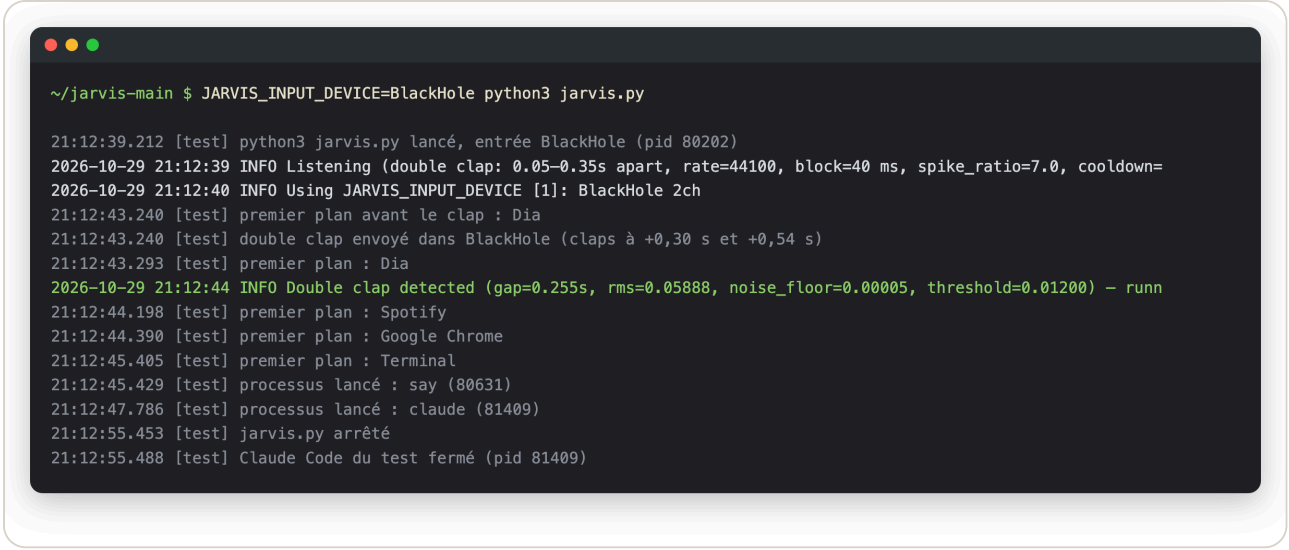
- + 0,4 s : Spotify passe au premier plan ;

- + 0,6 s : Chrome, sur claude.ai ;

- + 1,6 s : le Terminal, et la voix démarre ;

- + 4,0 s : Claude Code tourne dans le nouveau Terminal.

Le Mac était toujours en sourdine : on n'a donc pas entendu la voix, ni vérifié que Spotify lisait bien le morceau. La commande `open` affiche le morceau dans l'appli, et Claude Code a prévenu qu'elle ne garantit pas la lecture.



La sortie du script pendant l'essai contrôlé, avec nos relevés toutes les 0,1 seconde : l'appli au premier plan et les programmes qui démarrent.

Le régler à ta façon

Les réglages sont en haut de `jarvis.py`. Ceux qui comptent :

- **Il se déclenche tout seul** (porte qui claque, objet qui tombe) : monte `SPIKE_RATIO` de 7 à 9 ou 10.

- **Il rate tes claps** : descends `SPIKE_RATIO` vers 5, ou rapproche-toi du micro.

- **Tu claques lentement** : l'écart accepté va de `MIN_DOUBLE_GAP_S = 0.05` à `MAX_DOUBLE_GAP_S = 0.35` seconde. Passe le maximum à 0.5.

- **Un autre morceau** : remplace le lien dans `SONG_URI`.

- **Une autre page que Claude** : mets `CLAUDE_CODE_URL=https://…` dans un fichier `.env` à côté de `jarvis.py`.

- **Une autre voix** : `say -v '?' | grep fr_` liste les voix françaises de ton Mac (chez nous : Thomas, Amélie, Jacques, Eddy, Flo et d'autres). Mets `JARVIS_VOICE=Amélie` dans le `.env`.

- **Une autre phrase** : change `JARVIS_WELCOME_PHRASE`.

- **Un autre micro** : `python3 -c "import sounddevice as sd; print(sd.query_devices())"` donne la liste. Mets le numéro ou une partie du nom dans `JARVIS_INPUT_DEVICE`.

Tu peux aussi demander tout ça à Claude Code, en une phrase, comme à l'étape 3.

Ce qu'il faut savoir

- **Le micro reste ouvert tant que le script tourne.** Rien n'est enregistré : le script calcule un niveau sonore toutes les 40 millisecondes et l'oublie. Dans la version adaptée, rien ne sort du

Mac, à part l'ouverture des pages. L'original envoyait la phrase d'accueil à ElevenLabs pour la faire dire.

- **Jarvis reconnaît deux bruits secs, rien de plus.** Il ne comprend pas ce que tu dis. Le nom vient de l'assistant de Tony Stark dans Iron Man ; ici, tout tient dans la routine que tu lui donnes.

- **La routine ne se lance qu'une fois par démarrage.** Pour la rejouer, arrête le script et relance-le.

- **Mac en sourdine, Jarvis muet.** La voix passe par les haut-parleurs du Mac.

- **Le lancer à l'allumage du Mac**, avec un agent de démarrage (LaunchAgent) : c'est possible, mais on ne l'a pas testé. Si tu le fais, demande à Claude Code de l'écrire et de t'expliquer comment l'enlever.

Abyane · @abyane_nb · eigenflow.fr

Eigenflow