Eigenflow

GUIDE · EIGENFLOW

Une nuit de notre fabrique de reels

Claude Code, heure par heure : la boucle, les modèles, les garde-fous

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté AUTOMATIQUE, voici le guide. Notre fabrique de reels tient en peu de pièces : une file de liens, une tâche du Mac qui passe toutes les dix minutes, Claude Code lancé sans interface avec une méthode écrite, et un bouton Valider sur Telegram. Ce guide raconte une vraie nuit, celle du 20 au 21 novembre 2026, avec les heures relevées dans le journal de la file et dans les fichiers. Puis la boucle complète d'une vidéo, et comment monter la tienne avec des modèles écrits pour ce guide.

Les heures sont celles de Paris. Tout a été relu le 21 novembre 2026 : les pages d'Anthropic, de Google, de Meta, d'Instagram, de Telegram et d'Apple citées en fin de guide, et nos propres fichiers.

À savoir avant de lire

- La nuit, la fabrique produit sans t'attendre. Elle ne publie qu'après ton bouton Valider : rien de ce qu'elle a fait la nuit du 20 au 21 novembre n'est sorti avant la validation d'Abyane, le lendemain à 12 h 27.
- Il te faut un Mac allumé et réveillé, un abonnement Claude qui inclut Claude Code (Pro au minimum, 20 $ par mois hors taxes), une clé Gemini pour la voix, une appli Meta pour publier par l'API d'Instagram et un bot Telegram pour valider.
- Nos chiffres : 60 reels publiés du 1er au 21 novembre 2026, au moins deux chaque jour, 887 133 vues cumulées (relevé du 21 novembre par l'API d'Instagram).
- Cette nuit-là, une production a pris entre 1 h 12 et 2 h 38. Tout passe par l'abonnement à Claude, et pour une nuit entière une formule Pro ne suffit pas.
- Anthropic autorise ton abonnement dans Claude Code non modifié, pour ton propre usage. Pas pour faire tourner Claude au service d'autres personnes.
- Ce guide donne la marche à suivre et des modèles écrits pour lui. Ce n'est pas le code de notre fabrique.

---

1. La nuit du 20 au 21 novembre, heure par heure

Les heures viennent du journal de la file (écrit en heure UTC, une heure de moins que Paris) et, pour le reel 155, de l'heure de création de chaque fichier. La file ne fait qu'une production à la fois.

Avant la nuit

Vendredi 20 novembre, entre 19 h 02 et 19 h 12, Abyane envoie neuf liens de reels à notre bot Telegram, dont huit sans un mot. C'est tout ce qu'il fait pour cette nuit. Chaque lien est un reel qui a marché chez un autre créateur ; la file en tire un reel à nous, avec nos preuves.

La nuit

- **22 h 01 à 23 h 46** : reel 151, un fichier de passation contre le contexte qui se dégrade. Le vrai test est fait sur une copie de l'une de nos propres sessions de travail.
- **23 h 51 à 1 h 03** : reel 152, Claude qui remplit des candidatures de stage. Un vrai formulaire est rempli dans un navigateur sans profil, avec tous les envois bloqués.
- **1 h 11 à 2 h 52** : reel 153, onze skills pour une boutique en ligne, écrites et testées par nous.
- **1 h 38** : Abyane, encore debout, regarde le 152 sur Telegram et touche « À revoir », avec une remarque sur l'accroche.
- **3 h 02 à 3 h 30** : reprise du 152, sans lui. La file relève l'abandon dans les 3 premières secondes de nos 74 reels à plus de 2 500 vues, voit ce qu'ont en commun les meilleures accroches, et refait la sienne sur ce schéma : le titre entier et six vrais formulaires dès la première image.
- **3 h 32 à 6 h 10** : reel 154, une vidéo entière à partir d'un seul mot. Trois vrais tests tournent en parallèle, de 33 à 48 minutes chacun.
- **6 h 13 à 8 h 12** : reel 155, des prospects locaux trouvés avec Google Maps. Le détail suit.
- **8 h 00** : le serveur publie sur Instagram le reel 116, validé le 13 novembre. Il sort à l'heure pile, sans le Mac : la publication est confiée au serveur dès la validation.
- **8 h 10** : le contrôle de sortie du 116. En ligne sur Instagram et TikTok, son intact, réponse automatique en place. YouTube a refusé le Short (limite du jour), noté pour Abyane.
- **8 h 13 et 8 h 33** : deux reels validés plus tôt (131 et 132) trouvent une place libre et sont programmés.
- **8 h 43 à 10 h 23** : reel 156, la place d'une entreprise dans les réponses IA de Google.
- **12 h 27** : Abyane valide d'un coup les reels 151 à 155 sur Telegram, puis le 156 à 12 h 29. Le 151 est programmé dans les deux minutes.

Le reel 155, minute par minute

- **6 h 13** : la tâche démarre.
- **6 h 17** : l'original est dans son dossier, avec sa transcription et ses planches.
- **6 h 18** : la skill du dépôt dont parle la vidéo est copiée pour le test. Claude la lira ; personne ne la lance.
- **6 h 22** : les huit répliques sont dites par la voix de la marque, une requête par réplique, en moins d'une demi-minute.
- **6 h 23** : la voix est assemblée et relue mot à mot par Whisper.
- **6 h 29 à 6 h 31** : les cartes, faites des vraies positions d'OpenStreetMap, et les données d'animation.
- **6 h 35 à 6 h 52** : le vrai test, trois villes en parallèle. Premier essai sans mode plan : Claude répond comme s'il avait lancé l'outil et invente des chiffres. Deuxième essai en mode plan : il écrit son plan et la commande exacte, sans rien inventer. C'est le second qu'on montre ; le premier est raconté dans le guide du reel, comme mise en garde.
- **7 h 01** : la fiche d'étude, ce qu'on vise image par image. À 7 h 05 et à 7 h 19, deux rendus d'essai comparés à cette fiche.
- **7 h 22** : la musique et les bruitages.
- **7 h 24** : script, master, version sans musique et version de publication. Earshot écoute le mix, Whisper relit la voix dans le mix final.
- **7 h 25** : planches de contrôle, légende, fiche de réponse automatique.

- **7 h 29** : la vérification du guide trouve une erreur à l'écran. La vidéo soulignait une clause des conditions de Google Maps qui ne visait pas ce cas. L'image est refaite avec la bonne clause.
- **7 h 33 à 7 h 44** : le guide (images, texte, PDF, vérification réplique par réplique).
- **8 h 09** : dernier encodage. À **8 h 12**, la vidéo part sur Telegram avec trois boutons : Valider, À revoir, Rejeter.

Une heure cinquante-neuf du lancement au message.

Ce que la nuit a fait, et ce qu'elle n'a pas fait

- Cinq reels prêts à valider (151 à 155) et une reprise, de 22 h 01 à 8 h 12 : 9 h 44 de production, une tâche après l'autre.
- Rien de facturé en plus de l'abonnement.
- Elle n'a choisi aucun sujet : les liens venaient d'Abyane, la veille au soir.
- Elle n'a rien publié de ce qu'elle a fait cette nuit-là. La sortie de 8 h était une vidéo validée une semaine plus tôt.

La vidéo qui t'a amené ici sort de la même file : lien envoyé le 20 novembre à 19 h 08, production lancée le 21 à 17 h 20, de jour cette fois. Comme les autres, elle n'est sortie qu'après le bouton Valider.

---

2. La boucle complète d'une vidéo

La veille, et la règle des 3 fois la médiane

Dans la nuit du 31 octobre au 1er novembre, de 22 h 50 à 1 h 30, Claude Code a relevé les 40 derniers posts de 91 comptes Instagram de notre niche, par un service de collecte payant (Apify, 7 $ pour cette partie) : 3 044 posts, dont 2 614 reels.

La règle : un post est un gagnant s'il fait au moins 3 fois la médiane de son propre compte. On compare un créateur à lui-même, pas aux autres : 50 000 vues, c'est énorme pour un petit compte et faible pour un gros. Sur les 2 227 posts publiés depuis le 1er septembre, 352 gagnants, dont 125 au-dessus de 10 fois.

Cette collecte n'est pas refaite chaque nuit. Elle a dressé la carte des formats et des sujets qui marchent. Au jour le jour, c'est Abyane qui repère les reels et envoie les liens ; la file relève ensuite les vues de chaque original avant de le refaire.

Le choix du sujet

Un lien envoyé au bot Telegram entre dans la file. Première vérification, avant toute dépense : ce sujet a-t-il déjà été fait ? La file cherche le lien et le sujet dans nos dossiers et dans notre banque de sujets. Depuis le 7 novembre, six liens sur 87 se sont arrêtés là.

Le relevé de l'original

La vidéo, sa transcription par Whisper (modèle large-v3-turbo, sur le Mac), une planche à une image par seconde et les trois premières secondes à dix images par seconde. Puis, réplique par

réplique, ce que dit la voix et ce que montre l'image : l'accroche, la structure, chaque plan. Le démontage que montre la vidéo est fait sur notre propre reel 29 (540 456 vues), pas sur celui d'un autre créateur.

La vérification des faits

Chaque fait, chiffre et prix se vérifie sur la page officielle du jour, en euros pour la France. Quand la vidéo parle d'un outil, on le teste vraiment, sans lancer son code : Claude Code dans un dossier vide, sans notre configuration (option `--safe-mode`), avec la consigne de l'outil. Ce qui est faux se corrige dans le script, ce qui est invérifiable en sort.

Le script

On garde la mécanique d'une vidéo qui a marché : l'ordre des idées, la longueur des phrases, le type d'accroche. La file transpose les répliques en français une par une, remplace les exemples, les outils et les chiffres par les nôtres, vérifiés, et corrige ce qui est faux. Elle ne cite jamais l'original, ni son nom ni ses vues.

L'accroche suit nos règles : 56 règles et modèles dans nos fichiers de méthode. Contexte, raison de rester et retournement avant 5 secondes ; le titre complet et plusieurs vrais résultats dès la première image ; 10 gabarits ; 19 accroches chiffrées relevées dans la veille. Nos accroches de ce type perdent 27 à 35 % des gens dans les trois premières secondes, nos meilleurs chiffres.

La voix

La voix de la marque est une voix conçue chez Gemini à partir d'une description, pas un clone de la voix d'Abyane. Modèle Gemini 3.8 Flash TTS, une requête par réplique, 25 requêtes au plus par reel. Whisper relit chaque réplique seule. Un mot avalé, on réécrit la phrase plutôt que de la régénérer telle quelle : « ce reel », lu « Sril », est devenu « cette vidéo ». Les mots sont ensuite horodatés pour les sous-titres mot à mot.

L'image en code

Les reels de la file sont animés en code : chaque scène est une page web animée, ou une composition Remotion, rendue image par image dans un Chrome sans fenêtre, en 1080 × 1920 à 30 images par seconde. Chaque affirmation a sa preuve à l'écran : une vraie capture, un vrai chiffre avec sa source en petit. Un style neuf à chaque reel ; avant l'envoi, la planche du nouveau reel se compare à celle de notre reel de référence.

Le son

Une instrumentale énergique de notre banque, différente de celles des six derniers reels, calée sur le moment fort. Des bruitages écrits en code sur chaque mot qui révèle et chaque coupe. La voix à -13,5 LUFS, la musique dessous. On garde toujours un master sans musique.

Les contrôles

Whisper relit le mix final : un mot masqué par la musique se voit là. Earshot, un outil libre (licence MIT), écoute le mix et le compare à des films de lancement professionnels : niveau, coupes sans son, largeur stéréo. Une planche à deux images par seconde se relit en entier, et un script mesure le rythme : un nouvel élément à l'écran toutes les 0,5 seconde environ.

Le guide

Pendant le rendu, un autre agent écrit le guide envoyé en DM et vérifie chaque réplique sur les pages officielles du jour : juste, à nuancer ou faux, avec la source. S'il trouve une erreur, la vidéo est retouchée avant l'envoi. Au 155, c'était la clause mal soulignée.

La validation

La vidéo arrive sur Telegram avec trois boutons : Valider, À revoir, Rejeter. « À revoir » avec un mot lance une reprise ; « Rejeter » l'abandonne. Sans « Valider », rien ne sort. Après trois versions refusées, la vidéo revient dans une session de travail avec Abyane.

La publication

Après « Valider », Claude Code prend le premier créneau libre (8 h, 13 h ou 18 h), dépose la vidéo sur notre serveur et lui confie la publication. Une tâche y tourne toutes les 5 minutes et publie à l'heure pile par l'API officielle d'Instagram, avec la mention IA, même si le Mac dort. TikTok et YouTube Shorts partent par Buffer au même créneau quand il a de la place ; sinon la file les rattrape plus tard, sans retenir Instagram. La réponse automatique au mot à commenter se crée sur le serveur à la sortie. Neuf minutes après l'heure, un contrôle vérifie que tout est en ligne et que le son n'a pas été coupé.

La mesure

Toutes les 3 heures, le serveur relève par l'API les chiffres de chaque reel sorti depuis moins de 8 jours : vues, portée, enregistrements, partages, temps moyen, abandon dans les 3 premières secondes.

Les reels d'essai

Un reel d'essai est montré d'abord à des gens qui ne te suivent pas. Deux usages chez nous :

- le jumeau : 48 heures après la sortie d'un reel au numéro impair, sa version d'essai, avec une autre accroche, une autre musique, l'image recadrée et une vitesse de 1,05. Les numéros pairs servent de témoins, jusqu'au bilan du 28 novembre ;
- le top 30 : les reels qui ont le plus de vues, classés de nouveau avant chaque lot, repartent en essai, faits revérifiés.

Cinq essais par jour au plus. Du 10 au 21 novembre : 37 essais, 248 722 vues chez des non-abonnés. Notre reel 29 (540 456 vues) a eu sa version d'essai avec une autre accroche : 54 602 vues de plus.

Apprendre

Chaque retour d'Abyane devient une ligne de la méthode ou une fiche de mémoire, avec le pourquoi : ce qu'il aime comme ce qu'il refuse. Les chiffres aussi. La nuit du 20 au 21 novembre, à 3 h, pour refaire une accroche, la file a classé nos reels par abandon à 3 secondes : nos meilleures accroches (27 à 35 %) posent toutes le titre complet et une grande image de la promesse dès la première image. C'est devenu une règle de la méthode le jour même, et la production suivante la lit avant d'écrire.

3. Monter la tienne

Six pièces, dans cet ordre. Les modèles ci-dessous ont été écrits pour ce guide. Le lanceur a été testé le 21 novembre avec un faux Claude (production prête, production coupée au délai, enfants arrêtés avec elle), le fichier launchd validé par `plutil -lint`, les boutons testés contre un faux serveur Telegram, et les options de permission par un vrai `claude -p`. Rien n'a été envoyé ni publié. Pour un script de départ plus complet (relais Telegram, état, routeur), notre guide « Monter ton système de vidéos avec Claude Code » le donne en entier.

Étape 1 : le dossier et ta méthode

Un dossier `fabrique` avec :

- `CLAUDE.md` : tes règles, lues à chaque lancement (ton, cible, ce qui est interdit, ce que tu veux voir à l'écran) ;
- `METHODE.md` : la marche d'une vidéo, étape par étape. Chaque erreur corrigée y ajoute une ligne, avec le pourquoi ;
- `consignes/production.md` : le prompt de l'étape 3 ;
- `productions/` : un dossier par vidéo ;
- `musique/` : tes morceaux, avec leur licence.

Commence par faire deux ou trois vidéos à la main avec Claude Code, en session. Ce que tu corriges pendant ces sessions devient `METHODE.md`. Une file sans méthode écrite produit vite, et mal.

Étape 2 : la file

Un fichier `file.json`. Tu y ajoutes un élément par lien, à la main ou par ton bot.

À COLLER

```
{
  "elements": [
    {
      "id": "e001",
      "lien": "https://www.instagram.com/reel/XXXXXXXXXX/",
      "note": "garde l'accroche, prends mes exemples à moi",
      "statut": "a_faire",
      "essais": 0
    }
  ]
}
```

Les statuts : `a_faire`, `en_cours`, `pret` (vidéo faite), `envoye` (sur Telegram), puis `valide`, `revoir` ou `rejete`, et `erreur` après trois essais ratés. Seul ton bouton met `valide`.

Étape 3 : le prompt de production

Il va dans `consignes/production.md`. Le lanceur remplace les quatre mots entre accolades.

À COLLER

```
Tu travailles seul, en mode autonome, pour la file de production de mes reels.
Personne ne te répond pendant la session. Écris en français.

Élément : {{ID}}
Lien du reel de référence : {{LIEN}}
Ma note : {{NOTE}}

AVANT TOUT
- Lis CLAUDE.md et METHODE.md de ce dossier. Mes règles passent avant tes
habitudes.
- Si ce sujet a déjà été produit (cherche le lien et le sujet dans
productions/), arrête-toi avec le statut "deja_fait".

LES ÉTAPES
1. Relevé : la vidéo de référence est dans productions/{{ID}}/original/. Si elle
n'y est pas, arrête-toi avec le statut "question". Transcris-la avec Whisper,
fais une planche à 1 image par seconde (10 images par seconde pour les 3
premières), et note, réplique par réplique, ce que dit la voix et ce que montre
l'image.
2. Faits : vérifie chaque fait, chiffre et prix sur la page officielle du jour.
Note la source et la date dans productions/{{ID}}/sources.md. Un fait faux se
corrige, un fait invérifiable sort du script.
3. Script : garde l'ordre des idées et le rythme de la référence, mais écris
avec mes exemples, mes outils et mes chiffres vérifiés. Vérifie l'accroche
contre les règles de METHODE.md. Écris productions/{{ID}}/script.md.
4. Voix : génère chaque réplique avec la voix de ma marque (25 requêtes au plus
pour ce reel). Relis chaque réplique seule avec Whisper. Un mot avalé : réécris
la phrase plutôt que de la régénérer telle quelle.
5. Image : une scène par réplique, en code, rendue image par image en 1080 x
1920 à 30 images par seconde. Chaque affirmation a sa preuve à l'écran : capture
réelle, vrai chiffre avec sa source. Aucun texte demandé à un modèle d'image ou
de vidéo.
6. Son : une musique de musique/ dont la licence permet la publication, la voix
à -13,5 LUFS, la musique sous la voix. Garde aussi une version sans musique.
7. Contrôles : Whisper sur le mix final, planche à 2 images par seconde relue en
entier. Corrige avant de finir.
8. Légende de 5 lignes au plus dans productions/{{ID}}/legende.txt.

CE QUE TU NE FAIS JAMAIS
- Publier, programmer, envoyer un message, créer un compte, faire un commit.
- Lancer le code d'un outil ou d'un dépôt dont parle la vidéo : tu le lis, tu
montres sa page.
- Arrêter un processus par son nom (pkill, killall). Seulement ce que tu as
lancé, par son numéro.
- Citer ou montrer le reel de référence, son auteur ou ses vues.
- Terminer ton tour en attendant une tâche de fond : attends son retour avant
d'écrire ton résultat.

TON RÉSULTAT (obligatoire, même en cas d'échec)
Écris {{RESULTAT}} en JSON :
{"statut": "pret" | "question" | "deja_fait" | "erreur", "video": "chemin absolu
du MP4 de publication", "message": "une phrase de 15 mots au plus pour moi",
"question": "seulement si statut = question"}
```

Étape 4 : le lanceur, avec sa garde de temps

`lanceur.py` fait un passage : il prend le premier élément à faire, lance Claude Code et lit le résultat. Trois garde-fous dedans :

- `-p` (ou `--print`) lance Claude Code sans interface : il fait la tâche, puis s'arrête. Pas de session qui reste ouverte.
- Un délai maximal : passé 4 heures, Claude est coupé, avec tout ce qu'il a lancé, par numéro de groupe de processus. Jamais par son nom : d'autres tâches tournent sur ton Mac.
- `--permission-mode dontAsk` avec une liste d'outils autorisés : tout ce qui n'est pas dans la liste est refusé sans question. Ajoute à la liste les commandes dont ta méthode a besoin.

À COLLER

```
#!/usr/bin/env python3
"""Un passage = une production au plus. Lancé toutes les 10 minutes par
launchd."""
import fcntl, json, os, signal, subprocess, sys
from contextlib import contextmanager
from datetime import datetime, timezone
from pathlib import Path

BASE = Path(__file__).resolve().parent
FILE = BASE / "file.json"
CONSIGNE = BASE / "consignes" / "production.md"
CLAUDE = str(Path.home() / ".local/bin/claude")  # chemin complet : launchd n'a
pas ton PATH
MODELE = "opus"                                  # toujours le donner
DELAI_MAX = 4 * 3600                             # au-delà, la production est
coupée
ESSAIS_MAX = 3
OUTILS = ("Read,Write,Edit,Glob,Grep,WebFetch,WebSearch,"
          "Bash(ffmpeg *),Bash(ffprobe *),Bash(python3 *),Bash(node
*),Bash(mkdir *),Bash(ls *)")


@contextmanager
def file_json():
    """Lit file.json sous verrou et le réécrit en entier à la sortie (jamais à
moitié)."""
    with open(BASE / "file.lock", "w") as verrou:
        fcntl.flock(verrou, fcntl.LOCK_EX)
        etat = json.loads(FILE.read_text())
        yield etat
        tmp = FILE.with_suffix(".tmp")
        tmp.write_text(json.dumps(etat, ensure_ascii=False, indent=1))
        os.replace(tmp, FILE)


def journal(msg):
    with open(BASE / "journal.log", "a") as f:
        f.write(f"{datetime.now(timezone.utc):%Y-%m-%dT%H:%M:%SZ} {msg}\n")
```

À COLLER

```
with file_json() as etat:
    # « en_cours » au début d'un passage = production coupée (Mac éteint, délai)
: elle repart
    for e in etat["elements"]:
        if e["statut"] == "en_cours":
            e["statut"] = "a_faire" if e["essais"] < ESSAIS_MAX else "erreur"
    el = next((e for e in etat["elements"] if e["statut"] == "a_faire"), None)
    if el:
        el["statut"], el["essais"] = "en_cours", el["essais"] + 1
if el is None:
    sys.exit(0)

dossier = BASE / "productions" / el["id"]
dossier.mkdir(parents=True, exist_ok=True)
resultat = dossier / "result.json"
resultat.unlink(missing_ok=True)
consigne = (CONSIGNE.read_text().replace("{{ID}}", el["id"]).replace("{{LIEN}}",
el["lien"])
            .replace("{{NOTE}}", el.get("note") or "aucune").replace("
{{RESULTAT}}", str(resultat)))

# le Mac ne se met pas en veille tant que ce passage tourne
subprocess.Popen(["caffeinate", "-i", "-w", str(os.getpid())])
cmd = [CLAUDE, "-p", consigne, "--model", MODELE, "--permission-mode",
"dontAsk",
       "--allowedTools", OUTILS, "--output-format", "json"]
journal(f"lance {el['id']} (essai {el['essais']})")
with open(dossier / "claude.log", "w") as log:
    p = subprocess.Popen(cmd, cwd=BASE, stdout=log, stderr=subprocess.STDOUT,
start_new_session=True)
    try:
        rc = p.wait(timeout=DELAI_MAX)
    except subprocess.TimeoutExpired:
        os.killpg(p.pid, signal.SIGTERM)  # Claude et ce qu'il a lancé, par
numéro, rien d'autre
        p.wait()
        rc = "délai dépassé"
journal(f"fin {el['id']} : {rc}")
```

À COLLER

```
with file_json() as etat:
    e = next(x for x in etat["elements"] if x["id"] == el["id"])
    if resultat.exists():
        r = json.loads(resultat.read_text())
        e.update(statut=r["statut"], video=r.get("video"),
message=r.get("message"))
    else:
        e["statut"] = "a_faire" if e["essais"] < ESSAIS_MAX else "erreur"
```

Deux pièges. Si la variable `ANTHROPIC_API_KEY` existe dans l'environnement, `claude -p` l'utilise toujours, d'après la doc d'Anthropic, et elle se paie à l'usage : vérifie qu'aucune clé ne traîne. Et une connexion Claude finit par expirer : `claude auth status --text` donne la date de fin (ligne « Sign-in ends »), à renouveler avant.

Étape 5 : la tâche launchd toutes les 10 minutes

Un fichier `~/Library/LaunchAgents/fr.toi.fabrique.plist`. Remplace TOI par ton nom d'utilisateur.

À COLLER

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>fr.toi.fabrique</string>
  <key>ProgramArguments</key>
  <array>
    <string>/usr/bin/python3</string>
    <string>/Users/TOI/fabrique/lanceur.py</string>
  </array>
  <key>WorkingDirectory</key>
  <string>/Users/TOI/fabrique</string>
  <key>StartInterval</key>
  <integer>600</integer>
  <key>RunAtLoad</key>
  <true/>
  <key>EnvironmentVariables</key>
  <dict>
    <key>PATH</key>

<string>/Users/TOI/.local/bin:/opt/homebrew/bin:/usr/local/bin:/usr/bin:/bin:/us
r/sbin:/sbin</string>
    <key>LANG</key>
    <string>fr_FR.UTF-8</string>
  </dict>
  <key>StandardOutPath</key>
  <string>/Users/TOI/fabrique/launchd.out.log</string>
  <key>StandardErrorPath</key>
  <string>/Users/TOI/fabrique/launchd.err.log</string>
</dict>
</plist>
```

Pour l'activer, lancer un passage tout de suite, puis l'arrêter :

À COLLER

```
plutil -lint ~/Library/LaunchAgents/fr.toi.fabrique.plist
launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/fr.toi.fabrique.plist
launchctl kickstart gui/$(id -u)/fr.toi.fabrique
launchctl bootout gui/$(id -u)/fr.toi.fabrique
```

N'ajoute pas `-k` à `kickstart` : il couperait la production en cours. Le manuel de launchd (macOS 26.6) le dit : si le Mac dort à l'heure d'un passage, ou si le passage précédent tourne encore, ce passage est sauté. Tu n'auras donc jamais deux productions en même temps, mais Mac en veille, rien ne démarre. `caffeinate -i` dans le lanceur empêche la veille pendant une production, pas celle du capot fermé.

Étape 6 : la validation par bouton

Crée ton bot avec BotFather sur Telegram (commande /newbot), écris-lui un message, et range son jeton et ton identifiant de conversation dans `~/.config/fabrique/telegram.json`, lisible par toi seul (`chmod 600`), jamais dans le code. `boutons.py`, lancé chaque minute par une deuxième tâche launchd (`StartInterval` à 60), envoie chaque vidéo prête avec trois boutons et lit ceux que tu touches. Seul ton compte peut valider.

À COLLER

```
#!/usr/bin/env python3
"""Envoie chaque vidéo prête sur Telegram avec trois boutons, puis lit les
boutons touchés. Lancé chaque minute."""
import fcntl, json, os, subprocess, urllib.parse, urllib.request
from pathlib import Path

BASE = Path(__file__).resolve().parent
FILE = BASE / "file.json"
CONF = json.loads((Path.home() / ".config/fabrique/telegram.json").read_text())
# {"token": "...", "chat_id": 123456789}
API = f"https://api.telegram.org/bot{CONF['token']}"
POSITION = BASE / "telegram-position.txt"
ACTIONS = {"valider": "valide", "revoir": "revoir", "rejeter": "rejete"}


def appel(methode, **params):
    data = urllib.parse.urlencode({k: json.dumps(v) if isinstance(v, (dict,
list)) else v
                                   for k, v in params.items()}).encode()
    with urllib.request.urlopen(f"{API}/{methode}", data=data, timeout=60) as r:
        return json.load(r)["result"]


with open(BASE / "file.lock", "w") as verrou:
    fcntl.flock(verrou, fcntl.LOCK_EX)  # le même verrou que lanceur.py
    etat = json.loads(FILE.read_text())

    # 1. les boutons touchés depuis le dernier passage
    position = int(POSITION.read_text()) if POSITION.exists() else 0
    for u in appel("getUpdates", offset=position, timeout=0, allowed_updates=
["callback_query"]):
        POSITION.write_text(str(u["update_id"] + 1))
        q = u.get("callback_query")
        if not q or q["from"]["id"] != CONF["chat_id"]:
            continue  # personne d'autre que toi ne valide
        action, _, ident = q.get("data", "").partition(":")
        for e in etat["elements"]:
            if e["id"] == ident and e["statut"] == "envoye" and action in
ACTIONS:
                e["statut"] = ACTIONS[action]
        appel("answerCallbackQuery", callback_query_id=q["id"], text="Noté")
```

À COLLER

```
# 2. chaque vidéo prête part une seule fois (50 Mo au plus pour un bot)
    for e in etat["elements"]:
        if e["statut"] == "pret" and e.get("video"):
            boutons = {"inline_keyboard": [[{"text": t, "callback_data": f"{a}:
{e['id']}"}
                                            for t, a in (("Valider", "valider"),
("À revoir", "revoir"),
                                                         ("Rejeter",
"rejeter"))]]}
            subprocess.run(["curl", "-sf", f"{API}/sendVideo",
                            "--form-string", f"chat_id={CONF['chat_id']}",
                            "-F", f"video=@{e['video']}",
                            "--form-string", f"caption={e.get('message') or
e['id']}",
                            "--form-string", f"reply_markup=
{json.dumps(boutons)}"],
                           check=True, stdout=subprocess.DEVNULL)
            e["statut"] = "envoye"

    tmp = FILE.with_suffix(".tmp")
    tmp.write_text(json.dumps(etat, ensure_ascii=False, indent=1))
    os.replace(tmp, FILE)
```

Un bot ne peut envoyer qu'une vidéo de 50 Mo au plus (doc de la Bot API). Au-delà de 45 secondes environ, notre version de publication les dépasse : fais une version d'envoi plus légère, à 3,6 Mbit/s.

À COLLER

```
ffmpeg -i publication.mp4 -c:v libx264 -b:v 3600k -maxrate 3600k -bufsize 7200k
-c:a aac -b:a 128k -movflags +faststart envoi.mp4
```

Ce modèle s'arrête à `revoir` : pour une reprise, remets l'élément en `a_faire` avec ta remarque dans `note`, à la main ou avec une deuxième consigne.

Étape 7 : la publication

L'API d'Instagram n'a pas d'heure de publication : elle publie quand tu l'appelles. La programmation, c'est ton script qui attend l'heure. Il ne doit publier que les éléments au statut `valide` et doit relire la doc de Meta avant de s'écrire. Ce qu'il te faut :

- un compte Instagram professionnel et une appli Meta avec Instagram Login, et les permissions `instagram_business_basic` et `instagram_business_content_publish` ;
- la vidéo à une adresse publique : Meta va la chercher (la doc : « the media must be hosted on a publicly accessible server at the time of the attempt ») ;
- trois appels : créer le conteneur (`POST /<IG_ID>/media` avec `media_type=REELS` et `video_url`), attendre que `status_code` vaille `FINISHED`, puis publier (`POST /<IG_ID>/media_publish` avec `creation_id`) ;
- `is_ai_generated=true` à la création du conteneur si ta vidéo a une voix ou des images de synthèse ;

- un endroit qui tourne à l'heure dite : ton Mac allumé suffit pour commencer. Chez nous, un petit serveur et une tâche toutes les 5 minutes, pour que la sortie ne dépende pas du Mac.

Meta limite à 100 publications par l'API sur 24 heures glissantes (une autre section de la même page dit 50). Avec trois reels par jour, tu en es loin.

Étape 8 : mesurer, et relancer en essai

Relève les chiffres de chaque reel par l'API (`GET /<MEDIA_ID>/insights`, métriques `views`, `reach`, `saved`, `shares`, `ig_reels_avg_watch_time`, `reels_skip_rate`), une fois par jour pendant une semaine. Pour les reels d'essai, la création du conteneur prend `trial_params` avec `graduation_strategy` : `MANUAL` (tu le partages toi-même dans l'appli) ou `SS_PERFORMANCE` (Instagram le partage tout seul s'il marche bien). Ne renvoie jamais le même fichier : change l'accroche, la musique, le cadrage.

---

4. Les garde-fous

- **Rien ne part sans validation.** Seul ton bouton met une vidéo en `valide`, et la publication ne lit que ce statut. La consigne interdit à Claude de publier, programmer, écrire à quelqu'un, créer un compte ou faire un commit.
- **Jamais d'arrêt de processus par son nom.** Le 14 novembre, une de nos sessions qui arrêtait son propre script par son nom a tué deux productions voisines, deux fois de suite. On arrête seulement ce qu'on a lancé, par son numéro : c'est ce que fait le lanceur.
- **Un budget de requêtes de voix.** Notre quota Gemini : 100 requêtes par jour et par modèle à notre palier, remis à zéro à minuit heure du Pacifique d'après la doc de Google, soit 9 h à Paris en hiver. D'où 25 requêtes au plus par reel. La file compte ce qui reste ; à court, elle passe sur une voix de secours ou met la vidéo en attente de voix.
- **Ne jamais citer l'original.** Ni son nom, ni ses vues, ni ses images à l'écran. La vidéo de référence te dit qu'un format ou un sujet marche, elle ne te donne pas ton contenu.
- **Le droit d'auteur.** L'article L122-4 du Code de la propriété intellectuelle rend illicite la reproduction d'une œuvre sans l'accord de son auteur, et précise qu'il en est de même pour la traduction et l'adaptation. L'idée est libre, la forme ne l'est pas : plus ta version s'écarte des phrases de la référence, avec tes exemples, tes preuves et tes chiffres, plus tu es à l'abri.
- **La collecte.** Les conditions d'utilisation d'Instagram (en vigueur depuis le 1er janvier 2025) interdisent la collecte d'informations « de manière automatisée sans notre autorisation expresse », que tu sois connecté ou non. Lis-les avant toute veille automatique. La voie officielle pour lire les posts d'autres comptes professionnels est la Business Discovery de l'API (avec Facebook Login, vues, j'aime et commentaires de chaque post) ; nous ne l'avons pas encore testée. Ne connecte jamais ton propre compte à un outil de collecte.
- **Les droits des musiques.** Un titre connu dans ton fichier peut faire couper le son par Instagram. Garde toujours un master sans musique. La voie sûre : une musique sous licence CC0 ou CC BY, avec le crédit dans la légende. Par Instagram Login, l'API ne pose pas de musique de la bibliothèque d'Instagram.
- **La mention IA.** Voix de synthèse et images faites en code : on la coche à chaque fois, sur les trois réseaux.

- **Les secrets.** Jeton Telegram, clé Gemini, jeton Instagram : dans des fichiers lisibles par toi seul, jamais dans une consigne. Tout ce que lit Claude peut finir dans un journal.

---

5. Ce que ça demande

- **Un Mac allumé et réveillé.** Sur secteur, empêche la mise en veille automatique (l'intitulé du réglage varie selon le Mac). Capot fermé, rien ne tourne.
- **Un abonnement Claude avec Claude Code.** Sur la page des formules, lue le 21 novembre 2026 : Pro à 20 $ par mois (17 $ par mois en annuel), Max à 100 $ par mois avec 5 fois l'usage de Pro, Ultra à 200 $ par mois avec 10 fois l'usage de Pro par semaine et pas de limite sur 5 heures pour les comptes vérifiés. Prix hors taxes. La formule gratuite n'inclut pas Claude Code.
- **Une clé Gemini pour la voix**, créée dans Google AI Studio. Le modèle Gemini 3.8 Flash TTS est gratuit sur le palier gratuit (Google se sert alors de tes données pour améliorer ses produits) ; au palier payant, 9 $ le million de jetons audio jusqu'au 31 décembre 2026, soit environ 0,00225 $ pour 10 secondes, puis le double à partir du 1er janvier 2027. Les quotas par modèle se lisent dans AI Studio. Une voix conçue se garde un an après sa dernière utilisation.
- **Une appli Meta** pour l'API d'Instagram, un compte professionnel, et une adresse publique pour tes vidéos.
- **Un bot Telegram**, gratuit.
- **Les outils** : Python 3, Node, ffmpeg, Google Chrome, Whisper (sur un Mac Apple Silicon, le paquet mlx-whisper). Earshot si tu veux un avis sur ton son.

---

6. Les limites, honnêtement

- **La file rate encore des accroches.** Cette nuit même, le 152 a demandé une reprise sur la sienne. Sans ta relecture, une vidéo moyenne sortirait.
- **Le choix des sujets reste humain.** Notre veille date du 31 octobre et la file ne la refait pas seule : chaque nuit commence par les liens qu'Abyane a choisis.
- **Le temps et l'usage.** 1 h 12 à 2 h 38 par vidéo cette nuit-là, une à la fois, et environ 300 $ de calcul estimé pour cinq reels et une reprise. Anthropic écrit que les limites affichées des formules Pro, Max et Ultra « supposent un usage ordinaire et individuel » de Claude Code. Combien de reels par semaine tient chaque formule, nous ne l'avons pas mesuré : c'est à tester.
- **Les conditions d'Anthropic.** La page « Legal and compliance » de Claude Code réserve la connexion par abonnement à l'usage de Claude Code et des applications d'Anthropic, et interdit de faire passer les demandes d'autres personnes par ton abonnement. La même page laisse un utilisateur se connecter avec son propre abonnement au Claude Code non modifié. Notre lecture : une file pour ton propre compte, sur ta machine, reste dans ce cadre ; une file que tu ferais tourner pour des clients sur ton abonnement n'y est plus, et la page renvoie alors vers une clé API.
- **Le Mac.** En veille, la file s'arrête. Une production coupée repart au passage suivant, avec une partie du travail à refaire.

- **Les jetons expirent.** Celui d'Instagram au bout de 60 jours, la connexion Claude à sa date de fin. Une file qui tourne seule ne prévient pas toujours : mets-toi des rappels.
- **Les reels d'essai ne garantissent rien.** Ils montrent ta vidéo à des inconnus ; le nombre de vues reste incertain.

Sources

Toutes lues le 21 novembre 2026.

- Anthropic : Legal and compliance de Claude Code, Authentication, Configure permissions, les formules de Claude, What is the Max plan, aide de la commande `claude` (version 2.1.327).
- Google : tarifs de l'API Gemini, limites de débit, génération de la parole.
- Meta : publication de contenu, Business Discovery.
- Instagram : conditions d'utilisation, en vigueur depuis le 1er janvier 2025.
- Telegram : Bot API, version 10.5 du 22 novembre 2026 (relue le 25 novembre : limite de 50 Mo inchangée).
- Apple : manuels de launchd.plist, launchctl et caffeinate, macOS 26.6.2.
- Article L122-4 du Code de la propriété intellectuelle, version en vigueur depuis le 3 juillet 1992.
- Earshot, licence MIT.
- Nos fichiers : journal et état de la file, heures de création des fichiers du reel 155, rapport de notre veille du 1er novembre, relevés de l'API d'Instagram du 21 novembre.