Eigenflow

GUIDE · 13 NOVEMBRE 2026

Un projet tiré d'un blog d'ingénierie

Les quatre blogs, nos prompts Claude Code testés sur Chaos Monkey, le plan du week-end, le modèle de README et le message exact à l'auteur

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté BLOG, voici le guide. Il contient ce que promet la fin de la vidéo : les quatre blogs d'ingénierie, avec des articles récents qui se prêtent à une petite version, les prompts Claude Code de notre test Chaos Monkey tels qu'on les a lancés, un plan pour le week-end, un modèle de README et le message exact à envoyer à l'auteur, en français et en anglais. À la fin, la vidéo est reprise phrase par phrase.

Blogs, articles et dépôts GitHub relus le 13 novembre 2026. Le test a été lancé le même soir avec Claude Opus 5.5. On n'a aucun lien avec Netflix, Anthropic, OpenAI ou Meta.

À savoir avant de lire

- Les quatre blogs publient toujours. Derniers articles relevés : le 9 novembre 2026 chez Netflix, le 13 novembre chez Anthropic, le 2 novembre chez OpenAI, le 13 novembre chez Meta.
- L'article sur Chaos Monkey date du 19 juillet 2011, et ses deux auteurs ont quitté Netflix. Pour ton message, prends un article de moins d'un an, signé.
- Notre test : trois prompts, 85 secondes de calcul pour Claude, 0,33 $ au prix de l'API, rien facturé (abonnement). Un seul service tué faisait tomber tout le clone : 0 requête réussie sur 20. Après la correction écrite par Claude, 20 sur 20, sur six lancers.
- Le code a été écrit par Claude et lancé seulement sur notre Mac. Le vrai Chaos Monkey de Netflix n'a pas été lancé : il demande Spinnaker, leur outil de déploiement.
- Sur GitHub, la recherche « netflix clone » donne 102 964 dépôts (13 novembre 2026). Un projet tiré d'un article précis, avec l'article en lien, sort du lot. C'est notre avis, pas une mesure.
- Personne ne publie de chiffre sur le nombre d'ingénieurs qui répondent. Écris un seul message, et ne demande pas de poste dedans.

1. Les quatre blogs, et quoi y prendre

Pour chaque blog : l'adresse, ce qu'on y trouve, où sont les noms des auteurs, et trois articles récents qui se prêtent à une petite version. Les idées de petite version sont les nôtres et n'ont pas été testées. Seul Chaos Monkey l'a été (partie 2).

Netflix TechBlog

netflixtechblog.com, hébergé sur Medium. On y lit le streaming, les données, les recommandations et la plateforme interne de Netflix. Les noms des auteurs sont en tête d'article, souvent reliés à leur profil LinkedIn : c'était le cas sur les trois articles qu'on a vérifiés.

- « Making Failure Diagnosis Reliable with Pensive AI », 26 octobre 2026, six auteurs. Les jours de pointe, environ 100 000 tâches de données échouent chez Netflix, et quelqu'un doit fouiller les journaux pour trouver la cause. Petite version : un script qui lit les journaux de tâches ratées et dit si la faute vient du code ou de l'infrastructure, jugé sur 20 journaux que tu as classés toi-même.
- « Beyond API Compatibility: Building a Tool to Measure the Blast Radius of Library Changes », 23 octobre 2026, Roberto Perez Alcolea. Leur outil, Quasar, montre où une fonction d'une bibliothèque partagée est encore utilisée, avant qu'on la change. Petite version : un outil en ligne de commande qui, pour une fonction donnée, liste ses appels dans cinq dépôts open source qui utilisent la bibliothèque.
- « Fact-Checking at Scale: Building a More Accurate and Efficient Synopsis Factuality System », 9 novembre 2026, cinq auteurs. Vérifier qu'un résumé de série ne raconte rien de faux. Petite version : un vérificateur qui découpe un résumé en affirmations et confronte chacune au texte source, testé sur dix résumés où tu as glissé des erreurs.

Anthropic, Engineering

anthropic.com/engineering, 29 articles au 13 novembre 2026. La page présente l'équipe qui construit l'infrastructure de Claude et partage « what they learn about building, evaluating, and deploying AI agents at scale ». Les articles s'ouvrent à l'adresse anthropic.com/research/…, et les auteurs sont en bas, rubrique « Acknowledgements », après « Written by ».

- « The components of agent memory », 6 novembre 2026, Lance Martin. Comment un agent garde ce qu'il apprend d'une session à l'autre, et le « dreaming » : relire ses journaux entre deux sessions pour réorganiser sa mémoire. Petite version : un agent en ligne de commande avec un fichier mémoire et une commande qui le réécrit à partir des journaux, mesuré sur dix questions avant et après.
- « Demystifying evals for AI agents », 9 janvier 2026, Mikaela Grace, Jeremy Hadfield, Rodrigo Olivares et Jiri De Jonghe. Comment évaluer un agent, et la différence entre pass@k (au moins une réussite sur k essais) et pass^k (k réussites sur k essais). Petite version : un banc d'évaluation pour un petit agent, chaque tâche lancée cinq fois, les deux scores calculés.
- « Designing AI-resistant technical evaluations », 21 janvier 2026, Tristan Hume. L'histoire de l'exercice à faire chez soi qui a servi à recruter des dizaines d'ingénieurs en optimisation chez Anthropic, et que Claude a fini par battre. L'exercice d'origine est en ligne : anthropics/original_performance_takehome. L'article le dit en toutes lettres : si tu descends sous 1 487 cycles, écris à performance-recruiting@anthropic.com avec ton code et un CV. L'offre date de janvier 2026, relis l'article avant d'envoyer quoi que ce soit.

OpenAI, rubrique Ingénierie

openai.com/news/engineering. Depuis la France, la page s'affiche en français, avec des titres traduits (openai.com/fr-FR/news/engineering). Son sous-titre : « Stories about the technology and builders at OpenAI. » Leur flux RSS compte 23 articles classés Ingénierie, le plus ancien du 30 octobre 2025. Les noms sont sous le titre (« Par … ») ou en bas (« Auteurs »).

- « Scaling Git at OpenAI » (« Le passage à l'échelle de Git chez OpenAI »), 2 novembre 2026, Aaron Friel, Taylor Blau et Ted Nyman. Pando, leur nouveau stockage pour Git, pensé pour beaucoup de push en même temps, ceux des ingénieurs et ceux de leurs agents. Petite

version : un banc qui lance 20 push simultanés vers un dépôt local, sur 20 branches, et mesure les refus et le temps, avant et après une file d'attente que tu ajoutes.
- « Core dump epidemiology: fixing an 18-year-old bug » (« Épidémiologie des core dumps : corriger un bug vieux de 18 ans »), 30 juin 2026, Nathan Bronson. Au lieu d'étudier quelques plantages à la loupe, l'équipe les a tous classés, comme une épidémie : une partie venait d'une machine défectueuse, l'autre d'un bug dans la bibliothèque libunwind. Petite version : un script qui regroupe des rapports de plantage par machine et par signature de pile, et retrouve la machine malade que tu as cachée dans des données fictives.
- « Building self-improving tax agents with Codex » (« Créer des agents fiscaux auto-améliorants avec Codex »), 27 mai 2026, quatre auteurs d'OpenAI et de Thrive Holdings. Chaque correction d'un comptable devient un cas de test, et l'agent s'améliore contre ces tests. Petite version : un petit agent qui trie des notes de frais, où chaque correction que tu fais s'ajoute à un fichier de tests relancé à chaque changement.

Engineering at Meta

engineering.fb.com, plusieurs articles par semaine. Le site se décrit comme « a technical news resource for engineers interested in how we solve large-scale technical challenges at Meta », et chaque page affiche des offres d'emploi, rubrique « Available Positions ». Les noms sont en tête d'article, chacun relié à une page auteur.

- « Catching Bugs Before They Land: JiTTesting's Human-on-the-Loop Approach at Full Scale », 29 octobre 2026, cinq auteurs. Avant qu'une modification soit fusionnée, un modèle écrit un test qui montre ce qu'elle change dans le comportement du code. Petite version : un script qui prend un diff de ton projet, demande à Claude un test qui passe avant la modification et échoue après, et le lance sur les deux versions.
- « PrivacyLib: Privacy Controls at the Data Access Boundary », 27 octobre 2026, cinq auteurs. Une bibliothèque qui vérifie, au moment où une donnée est lue, que cet usage est permis, y compris quand c'est un agent IA qui la lit. Petite version : une petite bibliothèque Python qui enveloppe l'accès à une base, vérifie une règle par champ et par usage, note tout et bloque ce qui n'est pas permis.
- « Your Algorithm: Giving People More Ways to Personalize Facebook Recommendations », 10 novembre 2026, huit auteurs. Les gens écrivent en phrases ce qu'ils veulent voir, et les recommandations suivent. Petite version : des recommandations de films sur un jeu de données public, où une phrase de l'utilisateur, traduite en poids par Claude, change le classement.

Des blogs d'ingénierie français

Actifs au 13 novembre 2026, avec la date de leur dernier article. Tous les titres relevés sont en anglais.

- Doctolib, medium.com/doctolib : « From npm to pnpm: making frontend CI faster at Doctolib », 26 octobre 2026.
- Alan, medium.com/alan : « Hopper: we built an internal coding agent. Now 80% of Alan uses it. », 15 octobre 2026.
- Bedrock Streaming, l'équipe technique derrière M6+, tech.bedrockstreaming.com : « Migrating the blog from Jekyll to Astro », 14 octobre 2026, et « Hosting the 2026 World Cup on M6+ », 25 août 2026.

- leboncoin, medium.com/leboncoin-tech-blog : « AI can generate tests, but can it guarantee their reliability? », 13 octobre 2026.
- Qonto, medium.com/qonto-way, dernier article le 8 octobre 2026, et BlaBlaCar, medium.com/blablacar, le 9 septembre 2026.

Si tu vises un stage en France, regarde-les en premier : ce sont des entreprises françaises.

Comment choisir ton article

- Moins d'un an, et signé par des personnes nommées.
- Un mécanisme que tu peux refaire sur ton ordinateur, sans leurs données ni leurs serveurs, en quelques centaines de lignes. Notre test en fait un peu plus de 200.
- Une mesure simple qui prouve que ta version marche, comme notre 0 sur 20 puis 20 sur 20.
- Un sujet proche du poste que tu vises.
- Si tu ne peux pas résumer le problème de l'article en deux phrases après deux lectures, prends-en un autre.

---

2. Notre test Chaos Monkey : les prompts tels quels

Le montage

Le 13 novembre 2026, entre 22 h 03 et 22 h 06, heure de Paris. Claude Opus 5.5, lancé par claude -p (Claude Code en ligne de commande, sur abonnement), dans un dossier vide, sans fichier CLAUDE.md, avec seulement les outils de fichiers : lire, écrire, modifier, chercher. Claude ne pouvait lancer aucune commande. C'est nous qui avons démarré l'appli et lancé le Chaos Monkey, en local, sur 127.0.0.1. Trois prompts, un par passe, dans le même dossier.

La commande exacte, si tu veux refaire le test à l'identique. Elle vient de notre version de Claude Code, les options peuvent différer chez toi :

À COLLER

```
claude -p --safe-mode --restricted --model claude-opus-5-5 --output-format json
--tools Read,Write,Edit,Glob --permission-mode acceptEdits
```

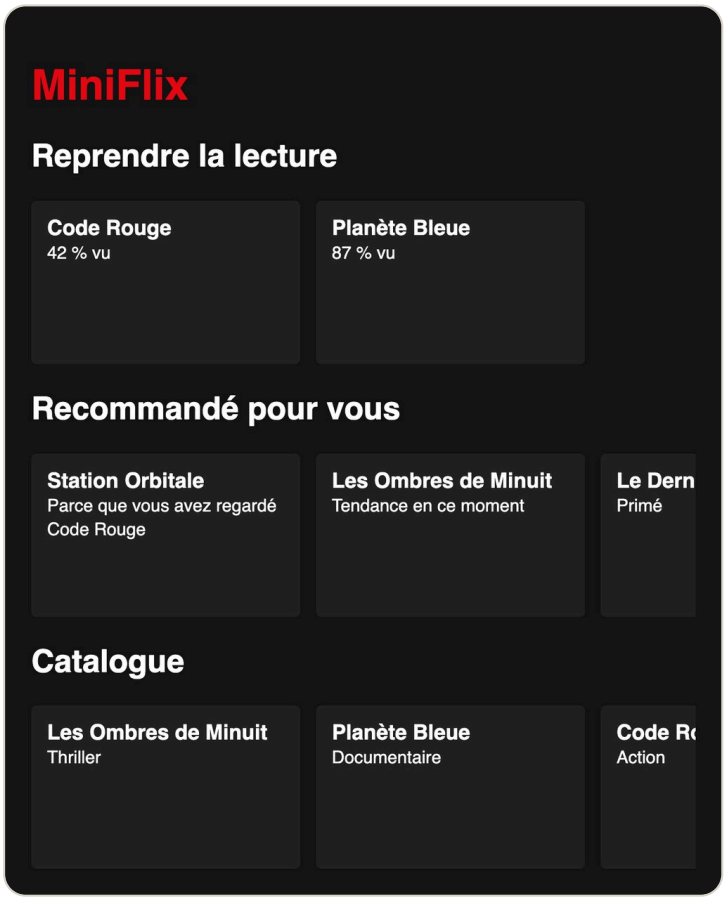
Dans Claude Code en mode normal, ouvre un dossier vide et colle les prompts un par un. Claude pourra lancer les commandes lui-même, après t'avoir demandé la permission.

Prompt 1 : le clone

> **À COLLER**
>
> ```
> Construis un mini clone de Netflix en Python, bibliothèque standard seulement. Quatre services HTTP sur 127.0.0.1 : accueil (port 8000), catalogue (8001), recommandations (8002), reprise de lecture (8003). L'accueil appelle les trois autres et renvoie une page HTML avec les trois rangées. Un script start.sh lance les quatre en arrière-plan et écrit le PID de chacun dans pids/<service>.pid. Fais la version la plus rapide possible, comme un projet de portfolio fait en une soirée : aucune gestion d'erreur ni délai d'attente, on testera la résilience après.
> ```

Pourquoi « aucune gestion d'erreur » : on voulait le clone typique, fait en une soirée, pour voir ce que le singe allait casser.

En 27,5 secondes, Claude a rendu MiniFlix : quatre fichiers Python de 20 à 46 lignes et le script start.sh. Il a prévenu : « J'ai écrit les fichiers, mais je ne les ai pas exécutés : je n'ai pas de shell dans cette session. Le code n'est donc pas testé. » Et il a annoncé la faiblesse avant qu'on la cherche : « Si un service est arrêté, l'appel depuis l'accueil lève une exception et la page ne se charge pas ».



MiniFlix, la page d'accueil du clone écrit par Claude, avec ses trois rangées : reprendre la lecture, recommandé pour vous, catalogue. Test du 13 novembre 2026

Prompt 2 : le Chaos Monkey

```
À COLLER

Voici un extrait de l'article de Netflix sur Chaos Monkey :

[ici, les trois paragraphes de l'article, collés en anglais, de « The cloud is
all about redundancy » à « we won’t even notice. »]

Construis un tout petit Chaos Monkey pour cette appli, dans chaos_monkey.py
(bibliothèque standard) : il tue un des services au hasard (jamais l'accueil),
en lisant pids/, puis il envoie 20 requêtes à la page d'accueil, affiche le
résultat de chacune (code HTTP ou erreur) et un bilan. L'appli doit survivre.
```

Dans notre test, l'extrait était collé en entier : 346 mots, tirés de l'article du 19 juillet 2011.

En 34,3 secondes, Claude a écrit chaos_monkey.py (77 lignes). Et il a corrigé l'accueil sans qu'on le demande, à cause de la dernière phrase du prompt. Son explication : « Dans la version d'origine, l'accueil appelait les trois services sans aucune protection. Dès que l'un d'eux tombait, chaque requête plantait et le client recevait une erreur de connexion. Le Chaos Monkey aurait donc échoué 20 fois sur 20. C'est exactement le maillon faible dont parle l'article. »

Il a aussi pris une précaution qu'on n'avait pas demandée : avant de tuer un processus, le script vérifie que son numéro correspond toujours au bon service. Sans ça, il pourrait tuer un autre programme qui aurait récupéré le même numéro.

La correction, avant et après. Les deux lignes de commentaire qui commencent par « # Passe » sont de nous, le reste est de Claude :

```
À COLLER

# Passe 1 : l'accueil appelle un service, sans délai ni repli
def fetch(port):
    with urllib.request.urlopen(f"http://127.0.0.1:{port}/") as r:
        return json.load(r)

# Passe 2 : délai d'une seconde, et None si le service ne répond pas
def fetch(port):
    """Appelle un service ; renvoie None s'il est indisponible (dégradation
gracieuse)."""
    try:
        with urllib.request.urlopen(f"http://127.0.0.1:{port}/", timeout=1) as
r:
            return json.load(r)
    except (OSError, ValueError) as e:  # connexion refusée, timeout, JSON
invalide…
        print(f"service :{port} indisponible : {e}", flush=True)
        return None
```

Quand un service ne répond pas, la fonction qui dessine sa rangée renvoie du vide, et la page s'affiche sans elle.

Ce qu'a donné le singe

On a lancé le singe sur deux copies de l'appli : celle de la passe 1, puis celle corrigée en passe 2. Sur le clone d'origine, il a tiré le catalogue. Les 20 requêtes à l'accueil ont échoué : une seule pièce tuée, et toute la page tombe. C'est le « Il a tout tué ! » de la vidéo.

Sur la version corrigée, on a relancé le singe jusqu'à ce que chaque service soit tombé au moins une fois : six lancers. La reprise de lecture a été tuée deux fois, les recommandations trois fois, le catalogue une fois. 20 sur 20 à chaque lancer, 120 requêtes réussies sur 120.

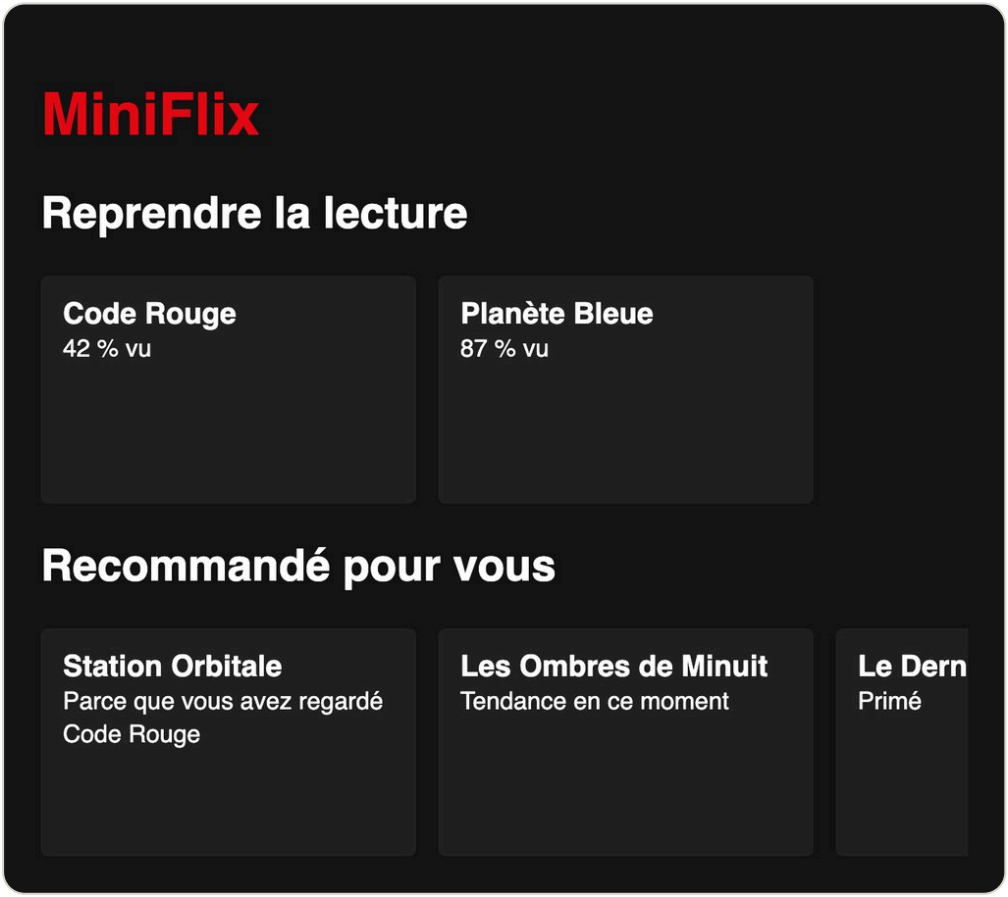
Lancer 1 · clone d'origine, tel que Claude l'a écrit · 13/11/2026 22:05:20

```
Chaos Monkey tue « catalog » (PID 94982)
  requête  1/20 : ERREUR RemoteDisconnected: Remote end closed connection without response
  requête  2/20 : ERREUR RemoteDisconnected: Remote end closed connection without response
  …  (requêtes 3 à 19, même résultat)
  requête 20/20 : ERREUR RemoteDisconnected: Remote end closed connection without response
Bilan : 0/20 succès, 20 échec(s) après la mort de « catalog ».
L'appli n'a pas survécu.
```

Lancer 4 · même service tué, après la correction · 13/11/2026 22:05:40

```
Chaos Monkey tue « catalog » (PID 95313)
  requête  1/20 : HTTP 200
  requête  2/20 : HTTP 200
  …  (requêtes 3 à 19, même résultat)
  requête 20/20 : HTTP 200
Bilan : 20/20 succès, 0 échec(s) après la mort de « catalog ».
L'appli a survécu.
```

Sorties réelles du Chaos Monkey, raccourcies, sans l'emoji du script. Test du 13 novembre 2026



La même page après la mort du catalogue, sur la version corrigée : la rangée Catalogue a disparu, le reste s'affiche. Test du 13 novembre 2026

Prompt 3 : la correction, préparée mais pas utilisée

On avait écrit ce prompt pour la passe suivante. Claude ayant déjà tout corrigé, il n'a pas servi. Le voici quand même : c'est celui qu'il te faut si tu veux d'abord voir ton appli tomber.

```
À COLLER

Le Chaos Monkey a tout fait tomber : voici sa sortie.

[colle ici la sortie du Chaos Monkey]

Corrige l'appli pour qu'elle survive à la perte de n'importe quel service sauf
l'accueil : délai d'attente court et repli, la dégradation gracieuse dont parle
l'article. Garde le code court.
```

Prompt 4 : le README

```
À COLLER

Écris le README.md du projet en français : ce que c'est, l'article de Netflix
qui l'a inspiré avec son lien (https://netflixtechblog.com/the-netflix-simian-
army-16e57fbab116), comment le lancer, ce que le Chaos Monkey a cassé la
première fois et ce qu'on a corrigé. Court.

Première sortie du Chaos Monkey :
[colle ici la première sortie, celle à 0/20]

Sortie après la correction :
[colle ici la sortie à 20/20]
```

En 22,9 secondes, Claude a écrit un README de 64 lignes : un tableau des quatre services, l'article avec son lien, le lancement, les deux sorties raccourcies et la correction. Il a ajouté une réserve : « je n'ai pas l'ancienne version de home.py. J'ai expliqué la panne par une exception non gérée dans le handler, qui coupait la connexion sans réponse. » Son explication était la bonne. Le modèle de la partie 4 part de ce README.

Ce que ça a coûté

Trois passes, 84,7 secondes de calcul en tout (27,5, 34,3 et 22,9). Le compteur de Claude Code indiquait 0,33 $ au prix de l'API. Rien n'a été facturé, la session passait par l'abonnement. Le code final fait 204 lignes, script de démarrage compris.

Ce que ça ne prouve pas

- Le code a été écrit par Claude et lancé seulement sur notre Mac, sur 127.0.0.1. Rien n'a tourné dans un cloud.
- Le vrai Chaos Monkey de Netflix n'a pas été lancé. Sa version actuelle, Netflix/chaosmonkey, écrite en Go, tue des machines virtuelles et des conteneurs, et demande Spinnaker. Le nôtre tue des processus sur une seule machine.
- 20 requêtes par lancer, c'est très peu. Pas de charge, pas de service lent, seulement des services morts, et aucun test automatique.

- La correction est la plus simple possible : on cache la rangée. L'article parle aussi de redondance, plusieurs copies de chaque pièce. On ne l'a pas fait.
- Chaque prompt a été lancé une fois. Si tu le relances, Claude écrira un autre code.

Refaire le test chez toi

1. Ouvre Claude Code dans un dossier vide et colle le prompt 1.

2. Pour voir l'appli tomber avant de la réparer, remplace la dernière phrase du prompt 2, « L'appli doit survivre. », par « Ne modifie pas l'appli. ». On n'a pas testé cette variante.

3. Lance ./start.sh puis python3 chaos_monkey.py, et garde la sortie dans un fichier.

4. Colle cette sortie dans le prompt 3. Relance le singe jusqu'à ce que chaque service soit tombé. Entre deux lancers, arrête les quatre services (la commande est dans le README écrit par Claude) puis relance ./start.sh, sinon les ports sont déjà pris.

5. Prompt 4 pour le README, puis relis-le ligne par ligne : c'est ton nom qui sera dessus.

---

3. Le plan du week-end

Une estimation pour quelqu'un qui code déjà un peu, pas une mesure. Dans notre test, Claude a écrit le code en moins d'une minute et demie. Le week-end sert à lire, comprendre et pouvoir tout expliquer en entretien.

Vendredi soir, une demi-heure

- Parcours les quatre blogs (partie 1) et garde trois articles qui te parlent.
- Choisis-en un avec les critères de la partie 1. Note le titre, la date et les noms des auteurs.

Samedi matin : lire

- Lis l'article deux fois, la seconde avec Claude à côté pour les mots que tu ne connais pas.
- Écris cinq lignes dans un fichier de notes : le problème, le mécanisme, ce que tu vas refaire, ce que tu laisses de côté, et la mesure qui prouvera que ta version marche.
- Crée le dépôt GitHub, vide, avec un nom à toi.

Samedi après-midi : la base

- Fais construire l'appli de départ par Claude Code, l'équivalent de notre prompt 1. Demande-la simple.
- Lance-la. Lis chaque fichier, et demande à Claude d'expliquer ce que tu ne comprends pas.
- Premier commit, même si c'est moche.

Dimanche matin : le mécanisme de l'article

- Colle l'extrait utile de l'article dans le prompt, et fais construire la petite version, comme notre prompt 2.

- Lance-la et garde la sortie « avant » dans un fichier. Répare avec le prompt 3, garde la sortie « après ».
- Casse ton appli une fois à la main, pour vérifier que tu sais où regarder.

Dimanche après-midi : publier et écrire

- Le README avec le modèle de la partie 4, relu phrase par phrase.
- Pousse sur GitHub. Vérifie que le lien vers l'article marche, et que le projet se lance depuis une copie neuve du dépôt.
- Trouve l'auteur et envoie le message de la partie 5. Un seul.
- Arrête-toi là. Une deuxième version ou un autre article, ce sera pour un autre week-end.

4. Le modèle de README

Tiré du README que Claude a écrit pour MiniFlix, réorganisé pour servir à n'importe quel article. Remplace ce qui est entre crochets. GitHub affiche les ~~~ comme des blocs de code.

À COLLER

````
# [Nom du projet]

[Ce que c'est, en une phrase.] Petite version de [système], décrit par
[entreprise] dans « [titre de l'article] » ([date]).

## L'article qui l'a inspiré
[Titre de l'article](lien), par [auteurs], [date].
Gardé : [le mécanisme, en une phrase]. Laissé de côté : [ce qui demande leur
échelle ou leurs données].

## Ce que fait le projet
| Pièce | Rôle |
|---|---|
| [fichier] | [rôle] |

## Le lancer
Il faut [langage et version]. [Dépendances, ou « aucune dépendance externe ».]
~~~sh
[commande pour démarrer]
[commande pour la démo]
~~~

## Ce qui a cassé la première fois
[Ce que tu as lancé, ce qui s'est passé et pourquoi, en deux phrases.]
~~~
[sortie réelle avant, raccourcie]
~~~

## Ce que j'ai corrigé
[La correction, et pourquoi.]
~~~
[sortie réelle après]
~~~

## Les limites
- [ce qui n'a tourné qu'en local, ce qui n'a pas été testé, ce que ferait la
vraie version]

## Comment je l'ai fait
Code écrit avec Claude Code, relu et lancé par moi. [Ce que j'ai appris, ce que
je ferais ensuite.]
````

Garde de celui de Claude les sorties réelles, le tableau des pièces et la cause de la panne. Ajoute ce qu'il n'avait pas : les limites, et ta façon de travailler avec Claude.

5. Le message à l'auteur

Le message

La phrase de la vidéo, avec juste ce qu'il faut autour. Version courte, pour tenir dans une note d'invitation LinkedIn : vise moins de 200 caractères, lien compris.

En français :

À COLLER

```
Bonjour [prénom], j'ai lu ton article « [titre] », et j'en ai construit une
petite version : [lien GitHub]. Merci de l'avoir écrit.
```

Si tu préfères le vouvoiement : « J'ai lu votre article « [titre] », et j'en ai construit une petite version. »

En anglais, pour les quatre blogs de la partie 1 :

À COLLER

```
Hi [first name], I read your post "[title]", and I built a small version of it:
[GitHub link]. Thanks for writing it.
```

Version longue, pour un e-mail ou un message direct, en français puis en anglais :

À COLLER

```
Bonjour [prénom],

J'ai lu ton article « [titre] », et j'en ai construit une petite version : [lien
GitHub].

Ce qui m'a le plus appris : [une phrase précise, par exemple ce qui a cassé et
pourquoi].
Ce que je n'ai pas réussi à faire : [une phrase].

Merci de l'avoir écrit.
[Prénom Nom]
```

```
À COLLER

Hi [first name],

I read your post "[title]", and I built a small version of it: [GitHub link].

What taught me the most: [one specific sentence, for example what broke and
why].
What I couldn't get to work: [one sentence].

Thanks for writing it.
[Your name]
```

La ligne sur ce que tu n'as pas réussi lui donne une raison de répondre, sans que tu demandes quoi que ce soit.

Où trouver l'auteur

- Netflix : les noms en tête d'article. Sur nos trois vérifications, chaque nom menait à un profil LinkedIn.
- Meta : les noms en tête, chacun relié à une page auteur sur engineering.fb.com. Cherche ensuite le nom avec « Meta » sur LinkedIn.
- OpenAI : « Par … » sous le titre, ou « Auteurs » en bas. Sur « Scaling Git at OpenAI », les trois noms sont en haut et le bas indique seulement « OpenAI » : regarde les deux.
- Anthropic : en bas, après « Written by ». Certains articles ne donnent que des initiales (« Redesigning source control at Anthropic » est signé « Nick G. » et des initiales) : choisis-en un avec des noms complets.
- LinkedIn : une demande de connexion avec une note, ou un message si vous êtes déjà en relation. Sur un compte gratuit, la note d'invitation est courte et limitée en nombre chaque mois, d'où la version courte.
- X : seulement si la personne y parle de son travail.
- E-mail : seulement une adresse que la personne affiche elle-même, sur son site ou son profil GitHub. Ne devine pas une adresse pro.
- GitHub : si l'article renvoie à un dépôt open source, tu peux y contribuer. N'ouvre pas une issue pour dire bonjour.

Les règles

- Un seul message, à une seule personne. Pas le même texte envoyé à toute l'équipe.
- Ne demande ni poste, ni stage, ni recommandation dans ce premier message. Le lien parle pour toi.
- Pas de CV en pièce jointe.
- Pas de relance. S'il ne répond pas, ton projet reste sur GitHub, et c'est lui que tu montreras en entretien.
- S'il répond, réponds court et remercie. Tu pourras dire ce que tu cherches plus tard, si la conversation s'y prête.
- Dis la vérité sur ce que tu as fait et sur ce que Claude a écrit. Un ingénieur le verra en lisant ton code.

6. Ce que dit la vidéo, vérifié

Chaque phrase, avec son verdict et sa source, relue le 13 novembre 2026.

« Un clone de Netflix »

Le chiffre montré à l'écran est juste : 102 964 dépôts pour la recherche netflix clone sur GitHub, le 13 novembre 2026 à 22 h 14 (API de recherche de GitHub). GitHub cherche les deux mots dans le nom, la description et les sujets des dépôts : ce ne sont pas tous des clones. Avec les guillemets, « netflix clone » donne 87 338 dépôts.

« Le mien a un mode sombre. » « Netflix aussi. »

Juste. La page d'accueil de netflix.com est sur fond noir, et le clone de Claude a repris un fond presque noir.

« Chaque grande boîte écrit ses vrais problèmes. En public. »

À nuancer. Beaucoup de grandes entreprises tech ont un blog d'ingénierie, pas toutes, et elles choisissent ce qu'elles publient, souvent une fois le problème réglé. Les problèmes racontés sont bien réels : OpenAI détaille un bug de 18 ans (30 juin 2026), Anthropic publie ses incidents (17 septembre 2025, 23 avril 2026), Netflix écrit qu'environ 100 000 tâches de données échouent les jours de pointe (26 octobre 2026).

« Netflix a un blog tech. Anthropic, OpenAI et Meta aussi. »

Juste. Les quatre adresses et leurs derniers articles sont dans la partie 1.

« Pour recruter ceux qui les lisent. »

À nuancer : c'est un des buts, pas le seul. L'article de Netflix de 2011 finit par « If you’re interested in joining the fun, check out our jobs page. » Chaque page du blog de Meta affiche des offres d'emploi. Anthropic invite, dans un article de janvier 2026, à envoyer son code et un CV. Mais chaque blog se présente d'abord comme un partage de ce que l'équipe apprend.

Chaos Monkey, « un outil qui tue leurs propres serveurs exprès, pour prouver que l'appli survit »

Juste. Phrase exacte de l'article du 19 juillet 2011, signé Yury Izrailevsky et Ariel Tseitlin : « This was our philosophy when we built Chaos Monkey, a tool that randomly disables our production instances to make sure we can survive this common type of failure without any customer impact. » Les « instances » sont des machines virtuelles chez Amazon plutôt que des serveurs physiques.

« Ils cassent leurs propres trucs ? Exprès ? »

Juste. En pleine journée de travail, avec des ingénieurs prêts à intervenir : « By running Chaos Monkey in the middle of a business day, in a carefully monitored environment with engineers standing by to address any problems ».

« On en construit un tout petit pour ton appli. Il tue une pièce au hasard. »

Juste, c'est notre test de la partie 2.

« Il a tout tué ! »

Juste dans notre test, à un mot près : le singe n'a tué qu'une pièce, le catalogue, et toute la page est tombée, 0 réponse sur 20.

« Excellente nouvelle ! Maintenant, tu sais quoi réparer. »

Juste, c'est l'idée de l'article : apprendre les faiblesses du système « and build automatic recovery mechanisms to deal with them ». Chez nous, après la réparation, 20 sur 20, six fois de suite.

Chaos Monkey en open source, 17 241 étoiles

Juste au 13 novembre 2026, sous licence Apache-2.0. Ce dépôt est la version 2, en Go, créé le 18 octobre 2016, qui demande Spinnaker. Le code d'origine, de 2012, est dans un dépôt archivé, Netflix/SimianArmy.

« Le nom de ceux qui l'ont écrit est sur le blog. »

À nuancer. L'article est bien signé, en bas : Yury Izrailevsky et Ariel Tseitlin. Mais il a quinze ans, et les deux ont quitté Netflix. Ariel Tseitlin est parti en 2013 chez Scale Venture Partners, où il est associé. Yury Izrailevsky est cofondateur et président de ClickHouse. Écris plutôt à l'auteur d'un article récent : ceux de 2026, sur les quatre blogs, sont signés.

« Un seul message. J'ai lu ton article, et j'en ai construit une petite version. »

Un conseil, rien à vérifier. Il rejoint celui de Haseeb Qureshi, ingénieur, dans son guide de 2016 sur sa recherche d'emploi : « Cold-email them (specific people, not companies). »

« Les ingénieurs adorent ceux qui lisent leurs articles. Ça leur arrive deux fois par an. »

Une blague, et non vérifié. On n'a trouvé aucun chiffre public sur le nombre d'ingénieurs qui répondent à ce genre de message, seulement des anecdotes de forums. Le seul appel écrit qu'on a relevé est dans l'article de 2011 : « if you have ideas, we'd love to hear them! »

« Une boîte, un projet, bien fait. »

Un avis. Les 102 964 dépôts vont dans son sens. Rien ne mesure l'effet sur un recrutement.

Le reste

« Netflix a déjà un Netflix », « Même compétence », « Garde-le. Renomme-le, avant. » : des blagues.

7. La méthode

Guide écrit le 13 novembre 2026. Blogs relus le soir même, entre 22 h 10 et 22 h 30, heure de Paris : flux RSS du Netflix TechBlog, d'Engineering at Meta et des actualités d'OpenAI, pages d'Anthropic Engineering, d'OpenAI Ingénierie et d'Engineering at Meta, puis chaque article cité, pour sa date et ses auteurs. Les pages qui refusaient la lecture automatique (Medium, openai.com) ont été ouvertes dans un navigateur piloté par script. Chiffres de GitHub relevés par son API à 22 h 14. Une copie de chaque page est gardée.

Le test : trois prompts lancés entre 22 h 03 et 22 h 06 avec Claude Opus 5.5, par claude -p, sur abonnement, dans un dossier vide, sans fichier CLAUDE.md, avec seulement les outils de fichiers. Le code a été lancé par nous, en local, sept fois : une fois sur le clone d'origine, six fois sur la version corrigée.

Pas testé : le vrai Chaos Monkey de Netflix, la variante du prompt 2 qui interdit de toucher à l'appli, les petites versions proposées pour les autres articles, l'envoi du message. On n'a trouvé aucune mesure publique du taux de réponse des ingénieurs. Les limites de LinkedIn sur les notes d'invitation n'ont pas été vérifiées dans son aide.

Ce guide décrit l'état du 13 novembre 2026. Les blogs publient chaque semaine : prends l'article le plus récent qui coche les critères de la partie 1.

Sources

- The Netflix Simian Army, Netflix TechBlog, 19 juillet 2011.
- Netflix TechBlog et son flux RSS, lus le 13 novembre 2026.
- Anthropic Engineering, The components of agent memory, Demystifying evals for AI agents, Designing AI-resistant technical evaluations, Redesigning source control at Anthropic.
- OpenAI, Ingénierie, Scaling Git at OpenAI, Core dump epidemiology, Building self-improving tax agents with Codex.
- Engineering at Meta, JiTTesting, PrivacyLib, Your Algorithm.
- Netflix/chaosmonkey et Netflix/SimianArmy sur GitHub, chiffres par l'API de GitHub le 13 novembre 2026 ; l'aide de GitHub sur la recherche de dépôts.
- Les auteurs de 2011 aujourd'hui : ClickHouse, Our story et Scale Venture Partners, Ariel Tseitlin, lues le 13 novembre 2026.
- How to Break Into the Tech Industry, Haseeb Qureshi, 7 mai 2016.