

La Gueznet IA n°9

Les prompts ! · Octobre 2026

Les **36 prompts** trouvés dans la veille du mois, sortis de 15 publications, traduits en français et prêts à coller. Chacun avec le lien de son post d'origine.

36 prompts · 15 publications · 11 annonces creuses

Traduction intégrale, crochets compris · Aucun prompt corrigé ni réécrit

Comment lire ce catalogue

Les crochets sont des trous. Tout ce qui est entre crochets est à remplacer par votre situation réelle. Un prompt collé avec ses crochets intacts produit une réponse sur un cas imaginaire.

Je n'ai rien amélioré au passage. Un prompt mal fichu reste mal fichu. C'est une information sur son auteur. Les fautes et la ponctuation sont d'origine.

Vérifiez ce qui sort. Plusieurs de ces prompts réclament au modèle des volumes de recherche, des salaires ou des marges qu'il ne connaît pas. Il produira des chiffres tout de même, et ils auront l'air sérieux.

Les garde-fous des auteurs font partie des prompts. La série n°9 précise « sur votre propre visage et celui de personne d'autre ». Lancée sur le visage d'un tiers, cette recherche est de la surveillance, et elle est illégale en France comme dans la plupart des pays européens.

Les 15 séries

#	Série	Compte	Prompts
1	Un sous-agent qui ne construit que des tableaux de bord de progression	@Voxyz_ai	1
2	Une équipe de quatre pour produire une vidéo animée de produit	@Voxyz_ai	1
3	Les notes de réalisateur pour une vidéo motion design avec Opus	@rexan_wong	1
4	Cloner un visualiseur musical et le rhabiller en Evangelion	@LuisBizarro	1
5	Le prompt d'exemple pour une première vidéo IA en local	@xkungumavishnu	1
6	Le second cerveau qui compile : les quatre prompts du système	@rvaniaaaa	4
7	Des agents « employés » : charte, clôtures, fiches de rôle et tickets de passage	@Lummox_eth	6
8	Une équipe d'agents qui travaille sans toi : les sept consignes du guide	@Argona0x	7
9	Une entreprise de douze Bots avec Jev comme couche de décision	@unicodef1wn	6
10	Le prompt qui « installe Jev » et le brief pour brancher ses agents	@0xCodila	2
11	Auditer son environnement d'agent, puis y poser Jev aux points de décision	@0xwhrrari	1
12	Le squelette de prompt pour faire coder un agent sur Jev	@Mnilax	1
13	La configuration Codex qui répartit le travail entre Luna, Jev et Sol	@sairahul1	1
14	Trouver dans son code les décisions qu'un modèle de décision pourrait prendre	@vicky_grok	1
15	Le prompt de classification à étiquettes A/B/C pour un « Jev » local	@_avichawla	2

1 Un sous-agent qui ne construit que des tableaux de bord de progression

À envoyer à Claude Code avant de laisser tourner une longue tâche en autonomie.

Mets en place un sous-agent qui ne construit que des tableaux de bord de progression :

1. Crée dashboard-builder dans ~/.claude/agents : model: opus, effort: medium, memory: user. Précharge les skills de design que j'ai installés (par exemple impeccable). Il ne peut lire et écrire que le dossier .dashboard/ et sa propre mémoire.
2. La première fois, il me demande quel style j'aime : sombre ou clair, dense ou aéré, et une couleur d'accent. Il enregistre ma réponse en mémoire et la suit à chaque fois. Choisis les panneaux en fonction de la tâche en cours ; n'utilise pas de gabarit.
3. Le tableau de bord est un seul fichier HTML qui montre les tâches et leur statut, les questions qui m'attendent avec l'action par défaut, les derniers livrables, et tout ce qui est bloqué. Utilise l'horloge réelle pour chaque heure affichée. Il s'ouvre d'un double-clic et se rafraîchit tout seul toutes les 10 secondes.
4. Ajoute une règle à ~/.claude/CLAUDE.md : pour toute tâche de plus de 5 étapes ou qui devrait durer plus de 30 minutes, fais préparer le tableau de bord par dashboard-builder avant de commencer ; mets-le à jour après chaque étape ; quand tu as besoin d'une décision de ma part, ajoute-la à la liste des questions et continue avec l'action par défaut.

Montre-moi le contenu des fichiers que tu vas créer ou modifier, puis explique tout le déroulement avec des mots qu'un enfant de 10 ans pourrait suivre. N'écris rien tant que je n'ai pas confirmé.

2 Une équipe de quatre pour produire une vidéo animée de produit

Vox (@Voxyz_ai), 705 j'aime, 27/09/2026 · X · 1 prompt

Ressource : <http://hyperframes.dev>

À envoyer à Claude Code. Les vidéos de référence se cherchent sur <http://whatships.com> selon l'auteur.

Mets en place dans ~/.claude/agents une équipe de vidéo animée que je pourrai utiliser pour chaque vidéo de produit. Si HyperFrames n'est pas installé, installe-le d'abord en suivant <http://hyperframes.dev>.

1. reference-breaker (effort: medium) : utilise ffmpeg pour extraire une image par seconde de la vidéo de référence que je te donne, assemble-les en mosaïque dans une seule image à regarder, et décompose le rythme, la typographie, les couleurs et les transitions en une liste de plans accompagnée d'un code de départ HyperFrames. Décompose seulement ; ne construis pas.
2. storyboarder (effort: medium) : à partir de la liste de plans, prépare 3 options de storyboard avec quelques images clés chacune, assemblées en mosaïque dans une seule image pour que je choisisse. Une fois mon choix fait, utilise npx hyperframes snapshot pour produire une image fixe par scène.
3. La session principale construit la vidéo : écris chaque scène en HyperFrames, et s'il n'y a pas de musique, synthétise une piste avec Python. Fais d'abord un rendu de brouillon avec npx hyperframes render --quality draft, et ne fais le rendu final qu'après mon accord.
4. film-reviewer (effort: high) : utilise ffmpeg pour extraire une image par seconde et passe le rendu en revue seconde par seconde. Liste les problèmes en P0, P1 et P2 (les P0 doivent être corrigés). Revue seulement ; ne modifie rien. La session principale corrige tous les P0 avant de me montrer.

Chaque sous-agent a besoin de Bash et doit pouvoir regarder des images. Avant de commencer, demande-moi la vidéo de référence, le logo, les couleurs, les polices et les captures d'écran du produit. Montre-moi le contenu des fichiers que tu vas créer, puis explique le déroulement avec des mots qu'un enfant de 10 ans pourrait suivre. N'écris rien tant que je n'ai pas confirmé.

3 Les notes de réalisateur pour une vidéo motion design avec Opus

Rexan Wong (@rexan_wong), 6 726 j'aime, 26/09/2026 · [X](#) · 1 prompt

Ressource : <http://whatships.com>

Le post décrit une méthode en six étapes (vidéo de référence, HyperFrames ou Remotion, 21st.dev, trois storyboards, une image fixe par scène) sans donner de prompt complet. Seules les trois notes de l'étape 6 sont citées mot pour mot, à envoyer une par une après le premier rendu.

```
ralentis tous les zooms à 0,7x  
coupe franche ici  
travelling avant sur le bouton
```

4 Cloner un visualiseur musical et le rhabiller en Evangelion

Luis Bizarro (@LuisBizarro), 1 157 j'aime, 27/09/2026 · [X](#) · 1 prompt

Ressource : <https://github.com/bizarro/evangelion>

À lancer dans Opus 5.5 après avoir cloné le dépôt d'un autre auteur (@_mexicat). Aucune consigne technique, résultat au bout de 15 minutes selon l'auteur.

```
peux-tu créer un nouveau projet intitulé Evangelion qui reprend le même principe, mais avec une chanson  
[nom de la chanson] qui est sur mon bureau ? S'il te plaît, toute l'interface doit correspondre à cette  
recherche Google sur l'interface d'Evangelion pour la réactivité : [recherche Google]. Ajoute 10 à 15  
visualisations différentes
```

5 Le prompt d'exemple pour une première vidéo IA en local

xkungumavishnu (@xkungumavishnu), 1 455 j'aime, 14/09/2026 · [Instagram](#) · 1 prompt

Ressource : <https://pinokio.co/>

Diapo 6 du carrousel: à taper dans Wan2GP (lancé depuis Pinokio, sur localhost:7860) une fois le modèle choisi.

```
Un golden retriever qui court dans un champ de tournesols au coucher du soleil, cinématographique,  
ralenti, 4K.
```

6 Le second cerveau qui compile : les quatre prompts du système

rvaniaaa (@rvaniaaaa), 469 j'aime, 24/09/2026 · [X](#) · 4 prompts

Ressource : <https://x.com/rvaniaaaa/status/2090512486738845784>

Méthode « LLM Wiki »: un dossier [raw/](#) pour les sources, un wiki tenu par le modèle, un [CLAUDE.md](#) à la racine. L'auteur précise que la boucle quotidienne demande Claude Desktop et une offre payante.

6.1 · Ingérer une nouvelle source

Je viens d'ajouter dans mon dossier raw un nouveau fichier nommé [nom du fichier]. Lis-le, extrais les concepts et les affirmations clés, écris un article de wiki pour chaque concept majeur, relie-les aux pages apparentées déjà présentes dans mon wiki, et signale toute contradiction avec ce que j'ai déjà. Puis résume en trois phrases ce qui a changé.

6.2 · Construire son CLAUDE.md

Tu mets en place mon second cerveau. Interroge-moi une question à la fois. Demande-moi : qui je suis et ce que je fais, mes objectifs pour cette année, comment je veux que tu communique avec moi, mes forces et mes faiblesses, et mes projets en cours. Attends chaque réponse avant de passer à la suivante. Quand c'est fini, écris le tout dans un fichier nommé CLAUDE.md à la racine du coffre, organisé avec des titres clairs, pour que tu le charges automatiquement à chaque session.

6.3 · Créer un dossier de projet

Crée un dossier de projet nommé [nom du projet]. À l'intérieur, crée quatre dossiers : Inputs, Process, Outputs, Feedback. Puis écris un CLAUDE.md dans ce dossier de projet qui décrit ce qu'il est, son objectif unique, à quoi ressemble « terminé », et ton rôle précis pour m'aider à y arriver.

6.4 · Mettre en place la boucle de compilation quotidienne

Mets en place une tâche de compilation quotidienne. Vérifie mon coffre. Range au bon endroit tout ce qui est nouveau dans les dossiers Inputs et relie-le aux notes apparentées. Signale tout ce qui est devenu périmé ou qui n'a pas été mis à jour depuis plus de deux semaines. Cherche les contradictions entre les ajouts récents et les pages de wiki existantes. Écris-moi un point : ce que tu as modifié, ce que tu as relié, ce que tu as signalé, et ce que je devrais regarder aujourd'hui.

7 Des agents « employés » : charte, clôtures, fiches de rôle et tickets de passage

LummoX (@LummoX_eth), 2 427 j'aime, 27/09/2026 · X · 6 prompts

Ressource : https://x.com/LummoX_eth/status/2100288151897981255

Le post liste dix dépôts de mémoire pour agents; les blocs ci-dessous viennent de l'article cité (un guide en sept étapes pour monter une équipe d'agents sur GPT-6 Astra). Ce sont des contenus de fichiers de rôle à charger à chaque exécution, pas des prompts de conversation.

7.1 · La charte d'un agent en cinq champs (exemple : Owen, opérations de la semaine)

- Résultat : un paquet hebdomadaire des restes, prêt à relire, existe le dimanche soir.
- Sources : la boîte partners@, le dossier /partners/forms, la liste de relances dans ops/people.md.
- Contraintes : pas de courrier personnel, aucun envoi, aucune soumission, aucune suppression. Plafond de quarante minutes. Consigne les manques au lieu de deviner.
- Livrable : un dossier daté avec la note de synthèse, le brouillon de formulaire non soumis et le journal d'actions.
- Point de contrôle : tu ouvres le dossier lundi matin avant toute action sortante.

7.2 · Les cinq clôtures : ce que l'agent ne termine jamais sans accord

- Envoyer. Aucun mail, aucun message privé, aucun commentaire sur un ticket, aucun « petit ping » depuis l'adresse de l'agent.
- Dépenser. Aucun passage en caisse, aucun paiement de facture, aucune montée en gamme, aucun siège ajouté.
- Publier. Aucun post, aucune mise en ligne, aucune page visible des clients, aucun « je mets juste la bio à jour ».
- Supprimer. Pas de corbeille, pas d'archivage qui vaut suppression, pas d'écrasement de l'unique copie.
- Juridique. Aucune acceptation de contrat, aucune signature, aucun changement de politique, aucun envoi de quoi que ce soit qui ressemble à un conseil que tu voudrais faire relire par un avocat.

7.3 · Les cinq fiches de rôle de l'équipe

Mara · prospects

- Possède : la liste des prospects entrants, tenue honnête.
- Résultat : chaque nouveau nom de la semaine est soit qualifié, soit rejeté avec une raison, soit en attente d'une information manquante.
- Définition de terminé : leads/AAAA-MM-JJ.csv plus une courte note d'exceptions.
- Clôture : elle n'écrit pas au prospect. Elle ne marque pas une affaire comme gagnée.

Cole · prise de contact

- Possède : la relance et le premier contact, en brouillon.
- Résultat : chaque personne qui a besoin d'un rappel a un message prêt dans les brouillons, rattaché à la liste de Mara ou au journal d'Owen.
- Définition de terminé : des brouillons dans le dossier Brouillons de la boîte mail, plus une file d'envoi pour toi.
- Clôture : envoyer.

Rina · fichiers

- Possède : les fichiers que d'autres attendent.
- Résultat : la présentation, la capture d'écran, l'export, l'image recadrée, le paquet demandé par le formulaire.
- Définition de terminé : les fichiers dans assets/AAAA-MM-JJ/ avec des noms qu'un inconnu peut lire.
- Clôture : publier. Elle ne met en ligne ni page visible des clients ni post public.

Vince · boîte de réception

- Possède : ce qui arrive.
- Résultat : la boîte de travail est triée dans les chemins sources que le reste de l'équipe a le droit de voir. Courrier personnel intact. Le bruit est consigné, pas transféré comme du travail.
- Définition de terminé : libellés ou dossiers appliqués, une note quotidienne des arrivées, une liste des éléments qui demandent un humain.
- Clôture : envoyer, supprimer. Il classe. Il ne range pas en détruisant.

Owen · opérations de la semaine

- Possède : le paquet des restes que tu connais déjà.
- Résultat : la semaine tient dans un seul dossier qu'un responsable peut ouvrir et comprendre.
- Définition de terminé : la note de synthèse, le brouillon de formulaire, le journal d'actions, les liens vers ce que Mara, Cole et Rina ont déjà terminé.
- Clôture : envoyer, dépenser, publier, supprimer, juridique. Owen emballe, il ne signe pas.

7.4 · La fiche en quatre lignes (exemple : Cole)

Nom : Cole.
Possède : les brouillons de prise de contact pour la liste de relances en cours.
Définition de terminé : un brouillon par personne, dans les brouillons, avec en tête la ligne source de Mara ou d'Owen.
Clôture : envoyer.

7.5 · La description de rôle pour les passages de relais (format, puis exemple de Cole)

À inclure, dans cet ordre :

1. Ce que je possède.
2. Ce que je reçois, de qui, dans quel chemin.
3. Ce que j'émet, vers qui, dans quel chemin.
4. Ce que je ne fais jamais (la clôture, plus le travail du voisin).
5. Ce que je fais quand ce qu'on me transmet est incomplet.

Exemple, Cole :

Reçoit leads/AAAA-MM-JJ.csv de Mara et ops/AAAA-MM-JJ/action-log.md d'Owen. Émet des brouillons dans le dossier Brouillons de la boîte mail et un send-queue.md pour l'humain. N'envoie jamais. N'invente jamais un prospect que Mara a rejeté. Si une ligne n'a pas de ligne source, il consigne le manque et saute cette ligne.

7.6 · Le ticket de passage de relais (mêmes champs à chaque fois)

- Nom du flux de travail.
- Étape.
- Responsable.
- Chemin de ce qui est transmis.
- Objectif du groupe.
- Blocages.
- Responsable suivant.

8 Une équipe d'agents qui travaille sans toi : les sept consignes du guide

Argona (@Argona0x), 470 j'aime, 24/09/2026 · X · 7 prompts

Ressource : <https://x.com/Argona0x/status/2091898304900571501>

Les sept encadrés de l'article cité (un guide en sept étapes pour monter une équipe de Grok Bots), vides au premier passage. Les deux blocs de variables d'environnement de l'étape 4 (brancher Kimi K3) sont de la configuration, ils ne sont pas repris.

8.1 · La première tâche : lire un document à ta place

Résume ce document en cinq puces. Liste chaque date, chaque décision et chaque question ouverte dans une section à part. Cite la page ou la section pour chaque élément. Ne modifie pas le fichier source.

8.2 · La charte en quatre phrases

Tu possèdes tout ce qui arrive sous forme de document ou de formulaire. Lis ce qui arrive, extrais chaque date, chaque décision et chaque question ouverte, et tiens une seule liste courante de ce qui est dû et pour quand, le plus proche en premier. Cite la page ou la section pour tout ce que tu affirmes. Ne soumetts jamais un formulaire, n'envoie jamais une réponse en mon nom et n'accepte jamais de conditions sans me demander d'abord.

8.3 · La fiche d'embauche en cinq lignes

Nom : [un seul mot, celui que tu diras à voix haute]
Possède : [le résultat que personne d'autre dans l'équipe ne possède]
Travaille à partir de : [les fichiers, les sites et les comptes qu'il peut utiliser]
Commence par : [la tâche du jour, en une phrase]
Jamais : [la seule chose pour laquelle il doit te demander d'abord]

8.4 · La ligne d'ouverture d'un groupe

À vous trois, préparez mon dossier de candidature pour vendredi. Chacun de vous porte une étape : comprendre ce que le formulaire demande réellement, remplir tout ce qui peut l'être à partir de mes fichiers, puis écrire la lettre d'accompagnement en dernier. Mettez chaque pièce terminée dans un seul dossier et dites-moi comment s'appelle ce dossier. Ne soumettez rien.

8.5 · La ligne de passage de relais, à coller à la fin de chaque description

Tout ce qui n'est pas à moi va à celui qui le possède : les noms à Mara, les lettres à Cole, les images à Rina, les réponses à Vince, les chiffres à Owen.

8.6 · La phrase de planification

Chaque jour de semaine à 7 h 30, lance la méthode Inbox sweep sur le courrier de la nuit. Publie une seule liste dans cette conversation : ce qui attend une réponse de ma part, ce que tu as déjà rédigé en brouillon, et ce qui peut attendre. N'envoie rien. Si la boîte mail ne se charge pas, dis-le et arrête-toi au lieu de réutiliser la liste d'hier.

8.7 · Le déclencheur sur événement

Quand un message dans #deadlines contient une date et le mot « due », ajoute-le à ma liste courante avec la date, le lien et la personne qui a demandé. Publie ici la liste mise à jour. Ne réponds jamais dans ce canal sans me demander d'abord.

9 Une entreprise de douze Bots avec Jev comme couche de décision

unicode (@unicode1wn), 183 j'aime, 23/09/2026 · X · 6 prompts

Ressource : <https://x.com/unicode1wn/status/2102375891875581966>

Blocs tirés du long article cité par le post (Grok Bot + Jev). Ce prompt demande à un assistant de préparer l'usage de Jev; il n'installe rien à lui seul, Jev est une API payante.

9.1 · Consigne à ajouter au Bot « Projects Manager »

Tu es la porte d'entrée de cette entreprise.

Tu portes la coordination, pas l'exécution spécialisée.

Pour chaque demande :

1. Convertis ma demande en un seul objectif clair.
2. Repère les informations qui manquent.
3. Repère quel spécialiste doit porter chaque partie.
4. Délègue le travail spécialisé au lieu de le faire toi-même.
5. Suis le travail terminé, les preuves, les blocages et le travail restant.
6. Ne reviens vers moi que lorsque :
 - l'objectif est atteint,
 - une approbation est requise,
 - ou le système est réellement bloqué.

Ne prétends jamais qu'un travail délégué a été fait. Ne marque jamais comme terminé un objectif non trivial sans preuve. Ne reprends jamais le travail d'un spécialiste au seul motif que ça irait plus vite de le faire toi-même.

9.2 · Le prompt donné à tinkabot pour construire le plugin de décision

Construis un Agent Plugin appelé « Jev Decision Layer »

Utilise l'API officielle de TypeSafe.

Authentification :

- clé d'API en Bearer
- stocke-la dans TYPESAFE_API_KEY
- ne code jamais le secret en dur et ne le journalise jamais

L'API TypeSafe expose des décisions System One.

Crée ces quatre outils :

1. jev_route_worker

Entrée :

- objective
- completed_work
- blockers
- available_workers, avec une courte description de poste pour chacun

Utilise une question Choice.

Renvoie :

- selected_worker
- probabilities
- confidence

2. jev_check_research

Entrée :

- claim
- evidence
- source_quality
- known_conflicts

Utilise un Choice :

- accept
- verify_more
- reject

Renvoie les probabilités et la confiance.

3. jev_review_completion

Entrée :

- original_objective
- required_outputs
- completed_work
- verification
- known_gaps

Utilise un Choice :

- complete
- verify_more
- incomplete

4. jev_guard_action

Entrée :

- proposed_action
- target
- side_effects
- reversibility
- existing_approval_policy

Utilise un Choice :

- allow
- confirm
- human_review
- deny

Ne laisse aucun outil exécuter l'action qui en résulte.

Le plugin ne fait que renvoyer des décisions.

Ajoute un skill court qui explique quand les agents doivent utiliser chaque outil.

Teste chaque outil en local avec un exemple sans risque avant de l'empaqueter

9.3 · Une tâche d'exemple pour éprouver le routage

Recherche les plus gros changements de la semaine dans les agents IA, transforme les résultats en un article long pour X, crée les visuels d'accompagnement et prépare une prospection vers les entreprises d'IA concernées. Ne publie rien et n'envoie rien.

9.4 · La règle « pas de preuve, pas de terminé »

Avant de déclarer terminé un objectif non trivial :

1. Reformule l'objectif d'origine.
2. Liste chaque livrable exigé.
3. Montre où se trouve chaque livrable.
4. Montre ce qui a réellement été vérifié.
5. Liste chaque lacune connue.
6. Lance jev_review_completion.

Si COMPLETE :

renvoie le dossier final.

Si VERIFY_MORE :

effectue les contrôles demandés.

Si INCOMPLETE :

identifie le responsable manquant et continue.

Ne retire jamais une lacune connue dans le seul but de faire passer la revue.

9.5 · Les cinq routines de départ

Veille du matin

Cooper

Chaque matin :

- trouve les changements importants côté IA et concurrents
- joins les sources
- sépare les faits confirmés des signaux faibles
- fais passer les affirmations importantes par le contrôle de recherche
- ne renvoie que ce qui mérite l'attention de l'équipe

Chaîne de contenu

Writing Bot + SEO & AEO Desk

Quand une recherche vérifiée est disponible :

- trouve l'angle le plus fort
- prépare le brief de l'article
- rédige le brouillon
- lance le contrôle de fin
- arrête-toi avant la publication

Préparation de la prospection

Outbound Prospecting

Quand une nouvelle campagne existe :

- recherche les entreprises qui correspondent
- construis le contexte
- rédige les premiers messages
- arrête-toi à l'approbation

Boucles ouvertes

GTM Loop Closer

Chaque jour :

- inspecte les promesses et les relances en attente
- montre les preuves
- prépare les mises à jour nécessaires
- n'envoie rien

Revue de l'entreprise

Projects Manager

Une fois par jour :

- ce qui est terminé
- ce qui est bloqué
- ce qui attend une approbation
- ce qui tourne en ce moment
- ce qui a échoué
- ce qui demande une décision humaine

9.6 · La tâche de test qui force les passages de relais

Trouve les 10 évolutions les plus importantes des sept derniers jours dans les agents IA. Vérifie chaque affirmation importante. Transforme la recherche en un article long pour X, un brief SEO, un brief visuel et une liste d'entreprises concernées avec cinq brouillons de prospection personnalisés. Ne publie rien et n'envoie rien. Ne rends le dossier que lorsque chaque livrable exigé existe et a passé la vérification.

10 Le prompt qui « installe Jev » et le brief pour brancher ses agents

codila (@0xCodila), 2 165 j'aime, 23/09/2026 · X · 2 prompts

Ressource : <https://x.com/0xCodila/status/2100984487802708306>

Ce prompt demande à un assistant de préparer l'usage de Jev; il n'installe rien à lui seul, Jev est une API payante. Il fait aussi cloner un dépôt tiers ([Bodila51/muse-jev-playbook](#)), distinct du dépôt officiel de TypeSafe: à lire avant de laisser un agent l'exécuter.

10.1 · Installer Jev dans son agent (prompt de la capture)

Je veux que tu installes Jev de TypeSafe AI dans l'agent IA ou l'environnement LLM que j'utilise en ce moment, puis que tu rendes Jev disponible pour les prochaines tâches de ce projet. Je dois pouvoir écrire des prompts de tâche normaux. Tu appelles Jev quand sa décision peut améliorer l'action suivante, sans que j'aie à mentionner Jev.

Fais l'installation jusqu'au bout et vérifie-la. Déroule les étapes ci-dessous ; ne t'arrête pas après m'avoir donné un plan.

****1. Comprendre mon travail****

Lis la conversation et le contexte du projet auxquels tu as réellement accès. Repère trois situations précises de mon travail où Jev pourrait aider à décider quoi faire avant que le travail coûteux de l'agent ne commence. Donne un court exemple pour chacune. Ne prétends pas avoir accès à d'autres conversations ou à des fichiers que tu ne vois pas.

****2. Vérifier cet environnement****

Avant de lancer la moindre commande d'installation, identifie l'environnement exact dans lequel tu tournes : Claude Code, Codex, Grok Bot, Muse, Cursor, un autre agent de code, ou un simple chat LLM.

Affiche :

`AGENT : agent détecté`

`PROJET : chemin du projet détecté / indisponible`

`OS : OS détecté / inconnu`

`CAPACITÉS : terminal / accès aux fichiers / réseau / instructions persistantes / indisponible`

Ne vérifie que les méthodes d'installation et les commandes qui concernent l'agent et le système d'exploitation détectés. Ne teste pas Claude Code, Codex, macOS, `npx` ou d'autres environnements sans rapport, sauf s'ils sont réellement nécessaires.

Si tu as accès au terminal et aux fichiers, poursuis l'installation. Si tu ne peux pas modifier le système de l'utilisateur, ne prétends pas installer Jev : termine l'analyse et fournis la passation exacte la plus courte dont un agent compatible a besoin. Préserve les instructions et la configuration existantes du projet.

****3. Utiliser des sources vérifiées****

Consulte les instructions officielles actuelles de TypeSafe :

- <https://docs.typesafe.ai/introduction/quickstart>
- <https://docs.typesafe.ai/agent-skill>
- <https://github.com/typesafe-ai/skills>

Pour le routeur de décision fonctionnel, examine :

- <https://github.com/Bodila51/muse-jev-playbook>
- <https://github.com/Bodila51/muse-jev-playbook/blob/main/QUICKSTART.md>

Le skill officiel de TypeSafe t'apprend l'API. Le dépôt `muse-jev-playbook` fournit le code de routage, des recettes et une politique d'agent. Traite-les comme des composants séparés.

****4. Installer et tester****

Suis le QUICKSTART actuel du dépôt dans un dossier de projet approprié. Sur un système de type Unix, sa mise en place commence par :

```
`git clone https://github.com/Bodila51/muse-jev-playbook.git`
```

```
`cd muse-jev-playbook`  
`python3 -m venv .env`  
`.env/bin/pip install -r requirements.txt`  
`cp config.example.yaml config.yaml`
```

Adapte les commandes à ce système d'exploitation si nécessaire. Lis les fichiers concernés avant de modifier la configuration. Installe le skill officiel de TypeSafe avec la méthode prise en charge par cet agent ; utilise une seule méthode d'installation, pas de copies en double.

Lance d'abord l'essai à blanc hors ligne du dépôt. Guide-moi ensuite pour créer une clé d'API sur <https://console.typesafe.ai/keys>. Dis-moi exactement comment définir `TYPESAFE\_API\_KEY` de façon privée dans cet environnement. Ne me demande jamais de coller la clé dans le chat, n'affiche jamais sa valeur, ne la commite jamais. Une fois la clé disponible, lance la vérification CLI en réel du QUICKSTART et vérifie que Jev a renvoyé une vraie réponse.

Jev renvoie des réponses typées `Choice`, `Score` et `Noul`. Le routeur les traduit en actions. Ne confonds pas un import du SDK ou un appel brut réussi à l'API avec une intégration d'agent qui fonctionne.

****5. Rendre l'usage futur naturel****

Ajoute une instruction concise, limitée au projet, dans le fichier d'instructions persistantes pris en charge par l'agent détecté : `CLAUDE.md` pour Claude Code, `AGENTS.md` pour Codex, ou le fichier équivalent de règles, de mémoire, de skill, d'instructions de projet ou de configuration pour Grok Bot, Muse, Cursor ou un autre agent. Préserve son contenu actuel.

L'instruction doit dire à l'agent :

« Avant une recherche web conséquente, la répétition d'une approche qui a échoué, le chargement de plusieurs outils ou skills, le lancement d'agents, le choix entre des voies d'exécution réellement différentes, ou la proposition d'une action lourde de conséquences, demande-toi si une petite décision Jev bornée changerait l'étape suivante. Si oui, construis un état compact sans secrets, appelle le routeur installé, interprète son action et poursuis la tâche d'origine. Passe-toi de Jev pour les réponses simples, les calculs déterministes, les modifications de fichiers courantes et les situations où l'appel n'apporte aucune décision utile. Respecte `bypass jev`. Garde les actions irréversibles derrière une confirmation humaine. »

Relie cette instruction à un appel d'outil que cet agent peut réellement effectuer. Ne prétends pas qu'ajouter une instruction suffit à garantir que Jev tournera. Démarre dans le mode `shadow` du dépôt, examine le journal des décisions, et explique à quel moment le passage en `active` serait justifié.

****6. Prouver que ça marche****

Teste trois tâches dans cet environnement : une question simple qui doit se passer de Jev ; une tâche conséquente avec une vraie décision de routage ; et un échec répété où le routeur peut recommander de changer d'approche. Vérifie à la fois le comportement de l'agent et les journaux du routeur. Si un test exige une nouvelle session pour vérifier que les instructions persistent, ouvres-en une si possible, ou donne-moi le test exact d'une ligne à y lancer.

Termine par un rapport compact : ce qui a été installé, où sont les fichiers, si l'appel réel à l'API a réussi, si cet agent a réellement invoqué le routeur sur une tâche adaptée, quels tests sont passés, et ce qui demande encore une action de ma part. Donne-moi trois prompts personnalisés à essayer ensuite. Ne déclare jamais l'intégration terminée sans preuve que l'agent a utilisé Jev pendant une vraie tâche.

10.2 · Brancher le routeur local sur ses agents existants (brief de l'article)

Lis le skill TypeSafe et examine les interfaces de mes agents de travail. Connecte les files d'attente JSON de chief.py aux gestionnaires de recherche et de rédaction existants. Empêche les traitements en double. Enregistre la progression après chaque action. Ajoute des limites d'appels et de dépense, une relecture en cas d'incertitude, et un contrôle de fin de tâche « le brouillon existe ». Garde la publication derrière une validation. Signale explicitement les connecteurs manquants.

11 Auditer son environnement d'agent, puis y poser Jev aux points de décision

rari (@0xwhrrari), 816 j'aime, 27/09/2026 · X · 1 prompt

Ce prompt demande à un assistant de préparer l'usage de Jev; il n'installe rien à lui seul, Jev est une API payante.

Variante du prompt de @0xCodila (n°1), avec un audit en plus: même squelette, texte différent, les deux sont donnés sans fusion.

Je veux que tu audites l'agent IA ou l'environnement LLM que j'utilise en ce moment, que tu me montres où je perds du temps, des tokens et de l'argent, puis que tu installes Jev de TypeSafe AI aux points de décision qui ont le plus de valeur. Je dois pouvoir continuer à travailler normalement. Tu n'appelles Jev que lorsque sa décision peut améliorer l'action suivante, sans que j'aie à mentionner Jev.

Fais l'audit, l'installation et la vérification jusqu'au bout. Déroule chaque étape ci-dessous ; ne t'arrête pas après m'avoir donné un plan.

****1. Comprendre mon travail****

Lis la conversation et le contexte du projet auxquels tu as réellement accès. Repère trois situations précises de mon travail où un modèle génératif coûteux ne sert qu'à choisir, noter, classer, vérifier, relancer, router ou demander une validation. Donne un court exemple pour chacune. Ne prétends pas avoir accès à d'autres conversations ou à des fichiers que tu ne vois pas.

****2. Vérifier cet environnement****

Avant de lancer des commandes d'installation, identifie l'environnement exact dans lequel tu tournes : Claude Code, Codex, Grok Bot, Cursor, un autre agent de code, ou un simple chat LLM.

`AGENT : agent détecté`

`PROJET : chemin du projet détecté / indisponible`

`OS : OS détecté / inconnu`

`CAPACITÉS : terminal / accès aux fichiers / réseau / instructions persistantes / indisponible`

Ne vérifie que les méthodes d'installation qui concernent l'agent et le système d'exploitation détectés. Ne teste pas d'environnements sans rapport, sauf nécessité.

Si tu as accès au terminal et aux fichiers, poursuis l'installation. Si tu ne peux pas modifier le système de l'utilisateur, ne prétends pas installer Jev : termine l'analyse et fournis la passation exacte la plus courte pour un agent compatible. Préserve les instructions et la configuration existantes du projet.

****3. Cartographier le gaspillage****

Examine la conversation, les fichiers du projet, les instructions, les outils, les skills, les serveurs MCP, les commandes répétées, les échecs récents et les appels de modèle coûteux que tu peux réellement voir. Construis une carte compacte du trajet du travail : demande > contexte > modèle > outils > résultat > relecture.

Trouve les points de décision répétés où Jev pourrait remplacer un appel de modèle ouvert par une réponse bornée `Choice`, `Score` ou `Noul`. Pour chaque candidat, estime la fréquence, la taille du contexte, la latence, les conséquences et la relecture humaine nécessaire.

****4. Classer les améliorations****

Renvoie un tableau avec : `DÉCISION`, `CHEMIN ACTUEL`, `PRIMITIVE JEV`, `VALEUR ATTENDUE`, `RISQUE`, `COÛT DE MISE EN ŒUVRE` et `PRIORITÉ`.

Ne retiens que les trois intégrations qui ont le plus de valeur. Privilégie les décisions bornées et répétées dont l'issue est claire. Écarte les tâches qui demandent encore de la rédaction ouverte, un raisonnement approfondi ou des règles déterministes exactes.

****5. Utiliser des sources vérifiées****

Consulte les instructions officielles actuelles de TypeSafe :

- <https://docs.typesafe.ai/introduction/quickstart>
- <https://docs.typesafe.ai/primitives>
- <https://docs.typesafe.ai/agent-skill>

- <https://github.com/typesafe-ai/skills>

Utilise la méthode d'installation prise en charge par l'agent et le système d'exploitation détectés. Préserve la configuration existante. Ne place jamais une clé d'API dans le chat, le contrôle de version, les journaux, les captures d'écran ou les fichiers générés. Demande-moi de définir `TYPESAFE\_API\_KEY` de façon privée quand c'est nécessaire.

****6. Installer et tester****

Installe Jev une seule fois avec la méthode prise en charge. Avant de modifier la configuration, lis les fichiers concernés du projet. Crée un contrat de décision compact pour le cas d'usage qui a le plus de valeur, avec : l'état, la question exacte, le type de réponse, les critères ou les options, le routage selon la confiance, le repli, la limite de relecture humaine, les données interdites, les champs du reçu et un jeu de test réaliste.

Lance d'abord un test hors ligne ou en mode fantôme. Puis lance trois cas : un cas évident, un cas ambigu et un cas qui doit être remonté à un humain. Vérifie à la fois le comportement de l'agent et le reçu de décision. Ne confonds pas un import du SDK, une réponse simulée ou une requête brute réussie à l'API avec une intégration qui fonctionne.

****7. Rendre l'usage futur naturel****

Ajoute une instruction concise, limitée au projet, dans le fichier d'instructions persistantes pris en charge par cet agent : `CLAUDE.md` pour Claude Code, `AGENTS.md` pour Codex, ou le fichier équivalent de règles ou de configuration pour Grok Bot, Cursor ou un autre agent. Préserve son contenu actuel.

L'instruction doit dire à l'agent d'envisager une petite décision Jev bornée avant une recherche coûteuse, la répétition d'une approche qui a échoué, le chargement de plusieurs outils, le choix entre des voies réellement différentes, ou la proposition d'une action lourde de conséquences. Garde les actions irréversibles derrière une confirmation humaine. Respecte `bypass jev`.

Relie l'instruction à un appel d'outil que l'agent peut réellement effectuer. Ne prétends pas qu'ajouter du texte suffit à garantir que Jev tournera.

****8. Prouver l'amélioration****

Termine par un rapport compact : ce que tu as trouvé, ce que tu as changé, les fichiers créés ou modifiés, les tests passés, si un appel réel à Jev a réussi, l'estimation du temps et des tokens économisés, les décisions qui restent à évaluer, les instructions de retour arrière, et les deux prochaines intégrations Jev qui valent un test.

Donne-moi trois prompts personnalisés à essayer ensuite. Ne déclare jamais l'intégration terminée sans preuve que l'agent a utilisé Jev pendant une vraie tâche.

****9. Mesurer la boucle****

Fais tourner le flux de travail amélioré sur au moins cinq vraies tâches. Compare l'ancien chemin et le chemin Jev sur les tokens, la latence, le coût, les relances et les interventions humaines. Ne garde un contrat de décision que s'il apporte un gain mesurable sans masquer les échecs.

Produis un reçu de décisions hebdomadaire avec : les décisions prises, la répartition de la confiance, les remontées à un humain, les faux positifs, l'argent économisé et le prochain contrat à régler. Si les chiffres ne s'améliorent pas, reviens en arrière au lieu de forcer Jev dans le flux de travail.

12 Le squelette de prompt pour faire coder un agent sur Jev

Mnimiya (@Mnilax), 2 248 j'aime, 26/09/2026 · X · 1 prompt

Ressource : <https://github.com/typesafe-ai/skills>

Lu sur la deuxième page de notes jointe au post, à utiliser avec le skill officiel de TypeSafe installé. Ce prompt demande à un assistant de préparer l'usage de Jev; il n'installe rien à lui seul, Jev est une API payante.

Utilise le skill TypeSafe. Avant d'écrire du code :

1. lis `docs.typesafe.ai/llms.txt`, puis les pages `<SDK ou API HTTP>` et `<primitive>`
2. comportement : `<ce que l'appli affiche, sélectionne, modifie ou transmet>`
3. liste les jugements, un par question, et marque ce qui reste dans le code
4. rédige instructions + critères, avec une option « aucune correspondance » sur chaque Choice
5. pose-les tous sur `<état>` en une seule requête
6. mets les questions et les seuils dans `<fichier>`, seuils en TODO tant qu'ils ne sont pas testés sur `<jeu de cas>`

N'utilise que les champs que tu as lus sur la page de l'API.

13 La configuration Codex qui répartit le travail entre Luna, Jev et Sol

Rahul (@sairahul1), 320 j'aime, 24/09/2026 · X · 1 prompt

Ressource : <https://x.com/sairahul1/status/2102694818485096803>

Ce prompt demande à un assistant de préparer l'usage de Jev; il n'installe rien à lui seul, Jev est une API payante. À coller dans Codex, Réglages, Personnalisation. Le texte vient de l'infographie jointe au post, confirmé par le bloc de l'article.

01 Orchestration multi-modèles par défaut

Pour toute demande de génie logiciel, agis automatiquement en architecte et en orchestrateur.

Objectif : mener le travail à bien de façon fiable, avec le modèle et le niveau de raisonnement minimaux nécessaires.

02 Jev, le modèle de décision

Jev est disponible comme modèle de décision et de contrôle.

Authentification : lis TYPESAFE_API_KEY dans l'environnement. N'affiche, n'expose, ne journalise et ne commite jamais cette clé.

Utilise Jev UNIQUEMENT pour des décisions bornées :

- classer la complexité d'une tâche
- choisir le niveau de raisonnement
- continuer ou s'arrêter
- décider d'une nouvelle tentative
- décider d'une escalade
- évaluer si le travail est terminé

N'utilise PAS Jev pour :

- écrire du code
- générer des correctifs
- concevoir l'architecture
- tout ce qu'un outil déterministe peut trancher

03 Routage initial

Avant d'implémenter, classe la tâche :

SMALL : Luna, raisonnement faible

MEDIUM : Luna, raisonnement moyen

HIGH : Luna, raisonnement élevé

ESCALATE : Sol, raisonnement élevé

Prends toujours la voie la plus basse qui suffit.

04 Vérification déterministe (règle critique)

Préfère une preuve déterministe au jugement d'une IA chaque fois que c'est possible.

Tests : lance les tests

Compilation : lance le compilateur

Types : lance le vérificateur de types

Lint : lance le linter

Code modifié : lance git diff

Ne demande jamais à Jev de déterminer ce qu'un logiciel peut déterminer.

05 Boucle de l'agent

Après chaque cycle d'implémentation :

- Inspecte le diff réel
- Lance les contrôles déterministes pertinents
- Rassemble des preuves concises

- Utilise Jev pour le jugement qui reste à porter
- Choisis : CONTINUE / RETRY / VERIFY / ESCALATE / COMPLETE

06 Chemin d'escalade

Luna Low > Luna Medium > Luna High > Sol High

N'escalade que si les preuves le justifient :

- les tentatives répétées échouent
- les tests continuent d'échouer
- du code sensible pour la sécurité a été modifié
- une incertitude d'architecture demeure
- la confiance de Jev est sous le seuil

N'escalade pas parce qu'un modèle plus fort est disponible.

07 Fin de tâche

N'annonce la fin que lorsque :

- le comportement demandé est implémenté
- les contrôles déterministes passent
- le diff correspond au périmètre demandé
- il ne reste aucun échec non résolu

Ne cache jamais une vérification qui a échoué.

14 Trouver dans son code les décisions qu'un modèle de décision pourrait prendre

Vikas gupta (@vicky_grok), 196 j'aime, 24/09/2026 · X · 1 prompt

Ressource : https://x.com/vicky_grok/status/2102753308108595669

Le post parle d'AX (Google); le prompt vient de l'article cité, cinq montages autour de Jev, à l'étape du journal de décisions. À lancer sur sa base de code avec n'importe quel agent, ou à coller tel quel. Il ne fait que lister des candidats, il n'installe rien.

Trouve dans cette base de code les décisions « à la Jev » : les endroits où le code doit, de façon répétée, (a) choisir parmi un petit ensemble d'options, (b) noter quelque chose selon une grille, ou (c) répondre à une question oui/non, et le fait aujourd'hui avec un appel à un LLM, une regex ou des if/else écrits à la main.

Pour chaque candidat, donne :

1. Le fichier et la fonction
2. La décision en une phrase
3. La ou les questions typées à poser (Choice / Score / Noul)
4. L'état à envoyer
5. Le nombre d'appels attendus par jour
6. Ce qui se passe quand la réponse est fausse

Liste uniquement les candidats. Ne propose pas de refactorisation.

15 Le prompt de classification à étiquettes A/B/C pour un « Jev » local

Avi Chawla (@_avichawla), 1 281 j'aime, 27/09/2026 · X · 2 prompts

Ressource : https://x.com/_avichawla/status/2101563610644496464

Tiré de l'article cité (reproduire en local le mécanisme de Jev avec SGLang et un modèle Qwen). Le prompt doit se terminer sur « Étiquette: »: le serveur ne génère rien, il lit le score du jeton suivant pour A, B et C. Chaque étiquette doit tenir en un seul jeton, et l'auteur conseille d'ajouter une option OTHER ou ESCALATE quand la liste n'est pas exhaustive. Le code Python qui interroge `/v1/score` n'est pas repris.

15.1 · La version de l'explication

Range le ticket de support dans exactement une catégorie.

Ticket :

J'ai été débité deux fois pour le même abonnement.

Étiquettes autorisées :

A = facturation

B = support technique

C = accès au compte

Renvoie uniquement l'étiquette.

Étiquette :

15.2 · Le prompt exact produit par le script

Ticket :

J'ai été débité deux fois pour le même abonnement.

Question :

Quelle catégorie correspond au ticket ?

Étiquettes autorisées :

A = facturation et paiements

B = support technique

C = accès au compte

Renvoie uniquement l'étiquette.

Étiquette :

Les annonces de prompt sans le prompt

Ces 11 publications promettaient un prompt absent de tout ce qu'elles servent. Je les ai ouvertes et j'ai cherché. C'est noté ici pour que personne n'y retourne.

Compte	Ce qui manque
Bilgin Ibryam (@bibryam) le post	Schéma comparatif LLM / Jev, aucun prompt

Compte	Ce qui manque
Mo Elgaraihy (@EngMoElgaraihy) le post	Fil en arabe sur la commande <code>/claude-api prompt-audit</code> : une commande à taper, pas un prompt
Mo Elgaraihy (@EngMoElgaraihy) le post	Fil en arabe sur les traces laissées par Windows (chemins, commandes système), aucun prompt
evolving.ai (Instagram) le post	Annonces matérielles de Meta Connect, aucun prompt (slide-13 pointée absente de l'OCR)
simplifyinai (Instagram) le post	Présentation du modèle Kev, lien promis en commentaire « AI », aucun prompt
Himanshu Hingorani (@himanshubuildss) le post	Liste de 20 ressources d'interface; les « prompts de construction » de Scrolltide sont sur le site (offre payante), aucun dans le post
furiouslytech (@furiouslytech) le post	« 700+ prompts » promis contre le commentaire « KIT »; le carrousel ne contient que des manipulations Windows, aucun prompt
@charliejhills (X) le post	16 animations faites avec Claude Code, à télécharger sur Substack; le post ne donne que la méthode en 3 étapes, aucun prompt.
@dailyprompter (IG) le post	« 700+ AI toolkit » promis contre un commentaire « YES »; les diapos ne servent que des réglages de confidentialité Windows 11.
Annatar.md (@AnnatarXBT) le post	Le post renvoie à un PDF du fondateur de Jev (non archivé) et à un article de @polydao sur le harnais Kimi K3: les deux blocs de l'article sont un <code>harness.yaml</code> et un script shell de hook, de la configuration, aucun prompt
Mr. Buzzoni (@polydao) le post	L'AGENTS.md annoncé (« quand appeler Jev, quand s'en passer ») n'est pas dans l'article. Les blocs désormais présents sont trois scripts Python de hooks et un JSON de réglages: du code. La seule consigne en langage naturel est une ligne passée à <code>claude -p</code> au milieu de <code>triage.py</code> (corriger une issue, la reproduire, ajouter un test qui échoue, ouvrir une PR en brouillon), non servie car elle vit dans le code