12

PROJET 2 • POINT DE THÉORIE • SÉANCE 3

Programmer avec un agent

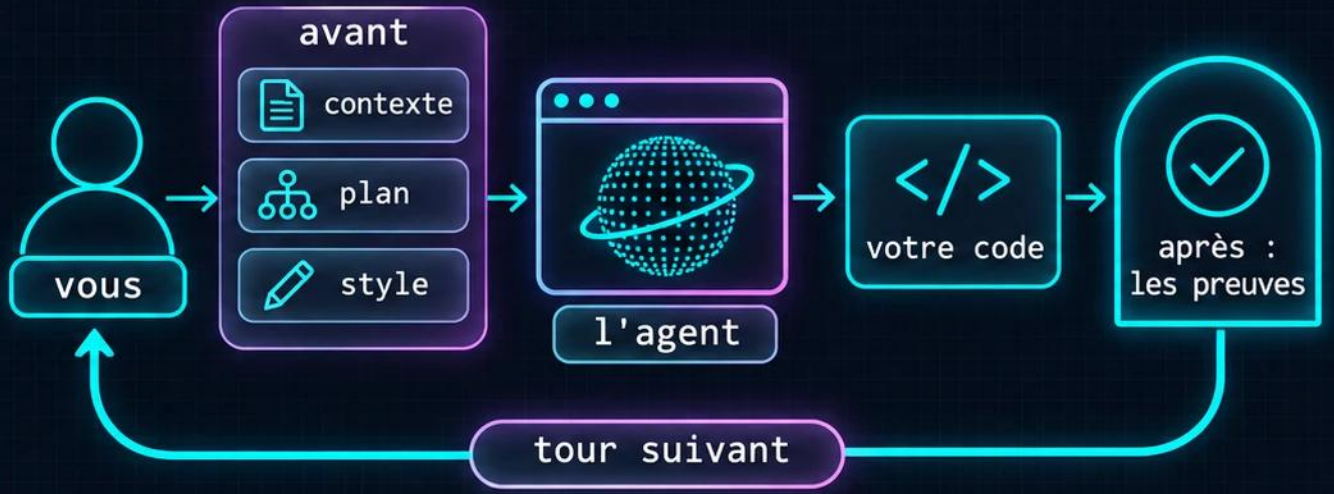
Thibault Six • IFOSUP Wavre • 2026-2027



Un agent écrit vite, mais la qualité dépend de ce que vous lui donnez avant et vérifiez après

Question : un agent écrit en quelques minutes ce qui vous prendrait une journée. Qu'est-ce qui reste votre travail ?

- **Le diriger :** sans contexte, sans plan et sans style imposé, il produit vite un code moyen, ou à côté du besoin
- **Tout vérifier :** un agent se trompe avec assurance
- **Maintenir ce qu'il écrit :** plus de code à lire, à tester et à corriger

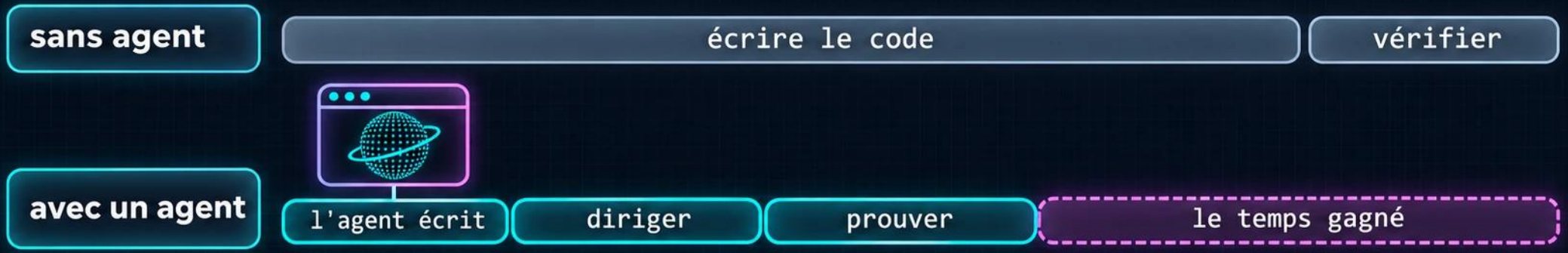


Une seule demande ne suffit jamais. Vous travaillez par tours.

Écrire le code n'est plus la partie longue du travail, et le temps gagné sert à viser plus haut

Question : si le code s'écrit en quelques minutes, à quoi sert le temps gagné ?

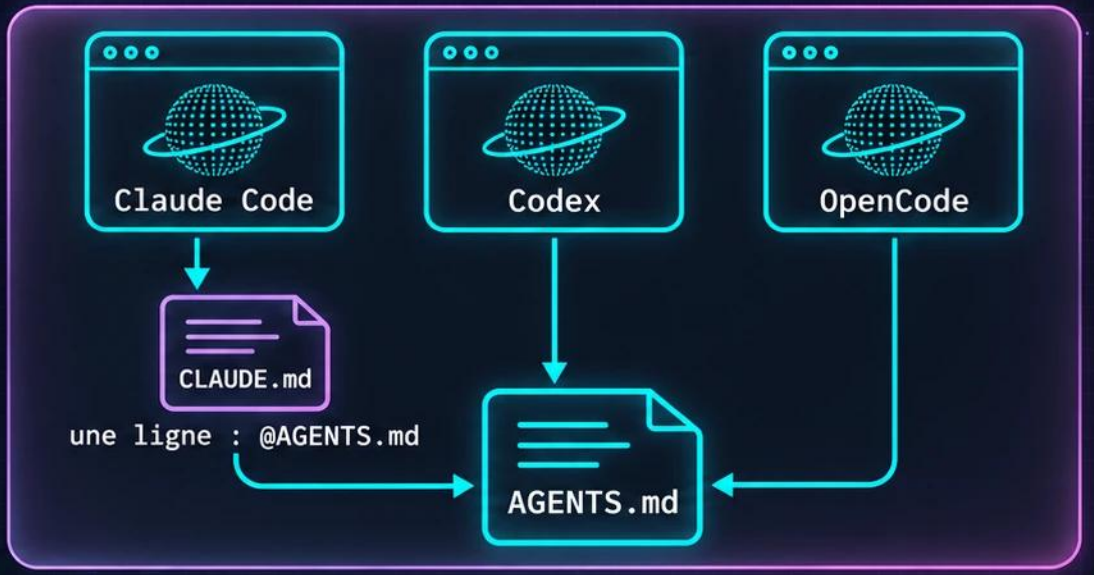
- **Essayer plus de choses :** un prototype coûte une heure, vous pouvez en faire trois et garder le meilleur
- **Être plus ambitieux :** une fonctionnalité écartée faute de temps redevient faisable
- **Soigner ce qui passait à la trappe :** les tests, les cas limites, l'accessibilité, la documentation
- **Sortir de votre stack,** tant que vous savez vérifier le résultat



Trois agents peuvent tourner sur votre poste, et tous lisent AGENTS.md, les règles du dossier

Question : rappel du chapitre « Un agent dans votre application ». Quand le modèle veut agir, que fait-il ?

Agent	Modèle	Compte
Claude Code (app Claude, onglet Code)	Claude	abonnement payant
Codex (app ChatGPT de bureau)	GPT	ChatGPT, gratuit ou payant
OpenCode	au choix, dont des modèles gratuits	aucun

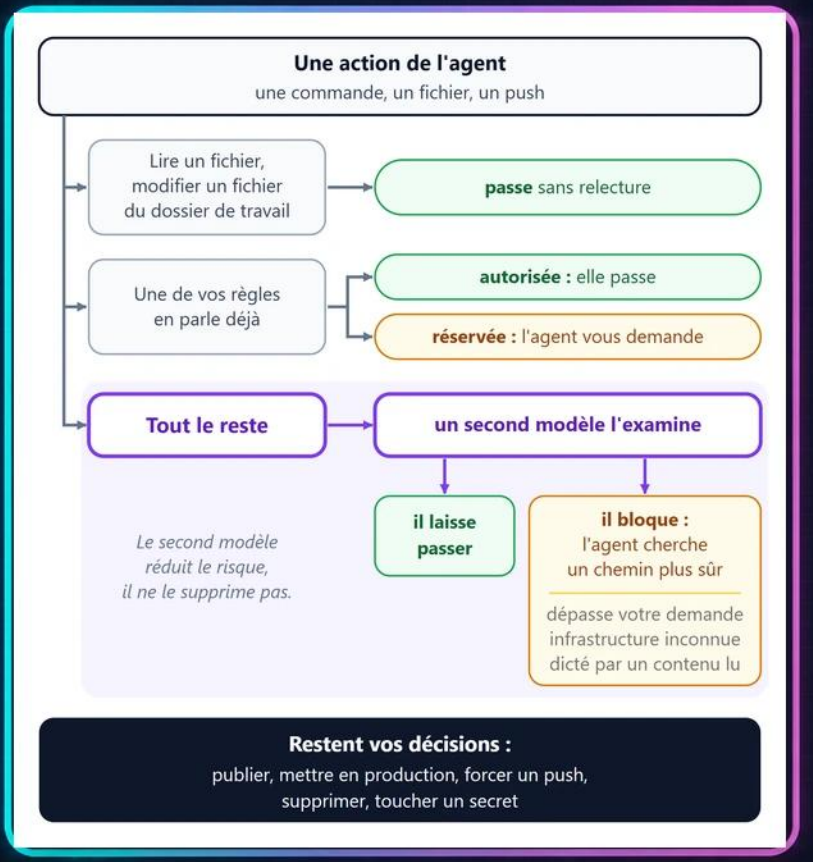


Le modèle demande un outil, un programme l'exécute et lui rend le résultat. Cette boucle ne change pas, mais ses outils sont maintenant votre terminal, vos fichiers, git et le web. Certains modèles gratuits peuvent utiliser vos données pour s'améliorer. Ne leur confiez ni donnée personnelle ni secret.

En mode automatique, un second modèle examine les actions qui vous auraient demandé un accord

Question : un agent vous demande votre accord cent fois par soirée. Que finissez-vous par faire ?

À force, vous finissez par approuver sans lire. Claude Code appelle ce mode « auto mode », sous la zone de saisie de l'app. Codex l'appelle « Auto-review ». Claude Code bloque par défaut un script téléchargé (`curl | bash`), un push forcé, un déploiement en production. Un push simple passe, même sur la branche principale. Travaillez donc par PR.



ce qui passe, ce que vos règles décident, ce que le second modèle examine

Chaque accident d'agents sans règles s'évite par une règle simple, écrite à l'avance

Question : quatre agents en lancent seize. Combien de temps dure alors un quota d'utilisation prévu pour plusieurs heures ?

Ce qui arrive		La règle qui l'évite
⚠ Quatre agents en lancent seize, et le quota de plusieurs heures part en dix minutes	→	✓ Un agent lancé par un autre agent n'en lance aucun
⚠ Un agent « nettoie » un dossier et efface les fichiers d'un agent voisin	→	✓ Un dossier de sortie par agent, et aucune suppression
⚠ Un formateur de code lancé sans chemin reformate les fichiers d'un autre agent	→	✓ Un formateur ne tourne que sur les fichiers de l'agent qui le lance
⚠ Un modèle invente des données pour réussir son propre contrôle	→	✓ Relire chaque diff, la liste des lignes changées

Un formateur de code est l'outil qui remet les fichiers en forme.

Le second modèle laisse passer des actions de trop, donc l'irréversible reste votre décision

Le mode automatique réduit le risque, il ne le supprime pas. Publier, mettre en production, forcer un push, supprimer, manipuler un secret : ces actions attendent votre accord, donné en clair dans la conversation. Dites exactement ce que vous autorisez.

17 %

des 52 actions réelles où un agent en faisait trop sont passées malgré le second modèle



vous

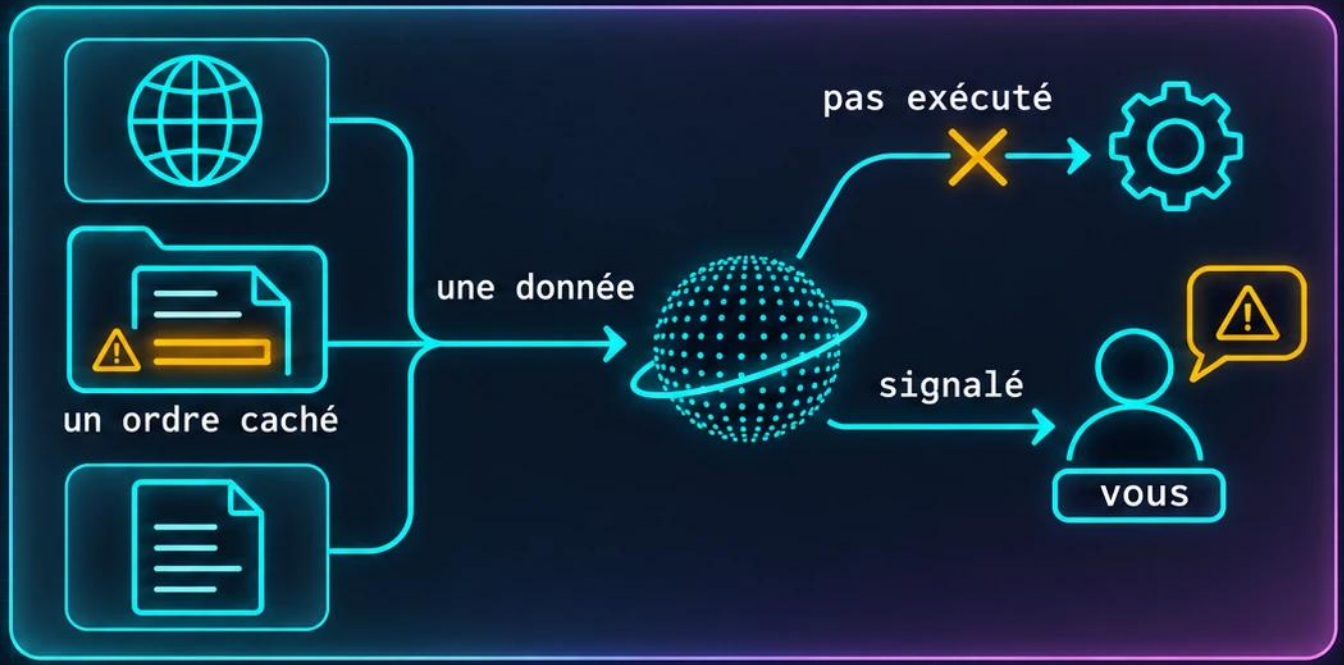
~~vas-y~~

oui, pousse la branche fix-login

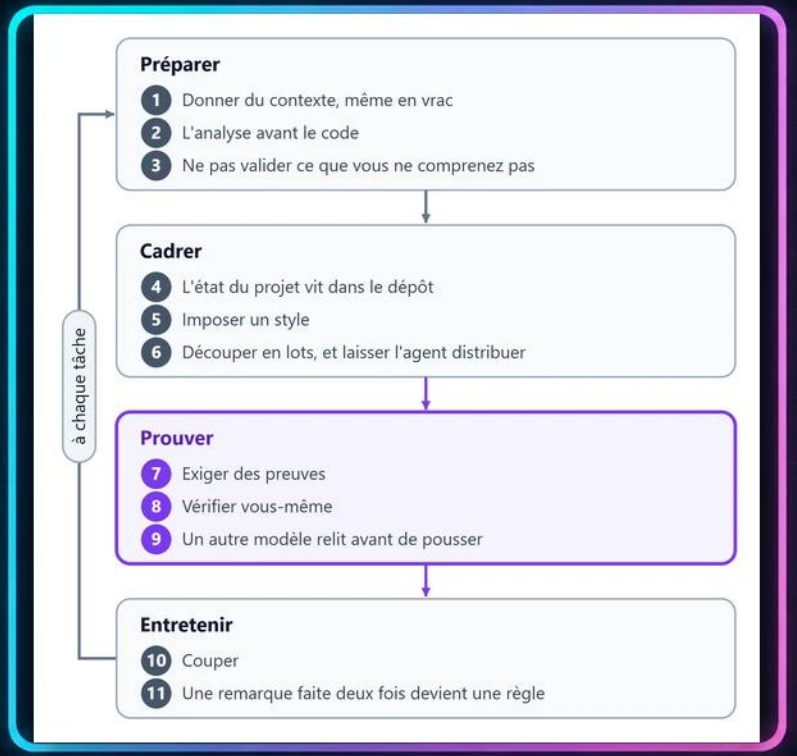
Ce que l'agent lit est une donnée, pas un ordre, et cette règle s'écrit dans votre `AGENTS.md`

Question : une page que l'agent lit contient un ordre écrit pour lui. Que doit-il en faire ?

Une page web, un dépôt cloné ou un skill, une consigne prête à l'emploi, peuvent contenir des ordres écrits pour les agents qui les liront. C'est une *injection de prompt*. Écrivez la règle dans votre `AGENTS.md`, avec la consigne de signaler toute instruction trouvée dans un contenu lu.



Onze étapes en quatre temps, préparer, cadrer, prouver, entretenir, reviennent à chaque tâche



onze étapes, quatre temps, un retour au début

Elles tournent en boucle. Vous y repassez à chaque tâche, de la première demande à la dernière remarque.

Votre première demande donne le contexte en vrac et réclame une analyse, pas encore du code

Question : avant la première ligne de code, que demandez-vous à l'agent ?

Donnez le besoin, la taille réelle (« une trentaine d'utilisateurs »), la stack et un projet à imiter. Puis : « Commençons par un audit et un plan de travail. » Répondez à ses questions ligne par ligne, avec > :

votre réponse, sous son message

```
2. Peut-on mettre une session en pause ?
> non, plus simple. On peut seulement abandonner.

6. Simple page web, ou application installable ?
> simple page web. Le reste est hors périmètre.
```

Une réponse n'est pas claire ? Dites-le, et attendez l'explication avant de valider.



Un agent estime son travail en heures d'humain, alors qu'il va dix à vingt fois plus vite

Question : le plan de l'agent annonce six heures de travail. Combien de temps va-t-il mettre ?

Dans l'exemple que vous suivrez en atelier, le plan annonçait six heures. L'agent a tout construit en vingt-quatre minutes. Ne coupez donc pas une fonctionnalité parce qu'elle est annoncée à une journée.



Une nouvelle session ne se souvient de rien, alors l'agent relit l'état du projet dans le dépôt

Question : demain, vous ouvrez une nouvelle session sur un autre ordinateur. Que sait l'agent de votre travail d'aujourd'hui ?

Au début de chaque session, l'agent relit un fichier d'état (où on en est, la prochaine action), les tickets ouverts, le journal et les décisions. Vous reprenez ainsi le travail depuis un autre ordinateur, juste en clonant le dépôt.



Sans style imposé, l'agent code dans le style moyen du web, alors imposez le vôtre en chiffres

Ce que vous imposez	Un exemple
Des limites chiffrées	méthodes de 15 lignes au plus, composants Vue de 100 lignes au plus
La langue	code en anglais, textes affichés en français
Une référence	« code comme le feraient les créateurs de Laravel »

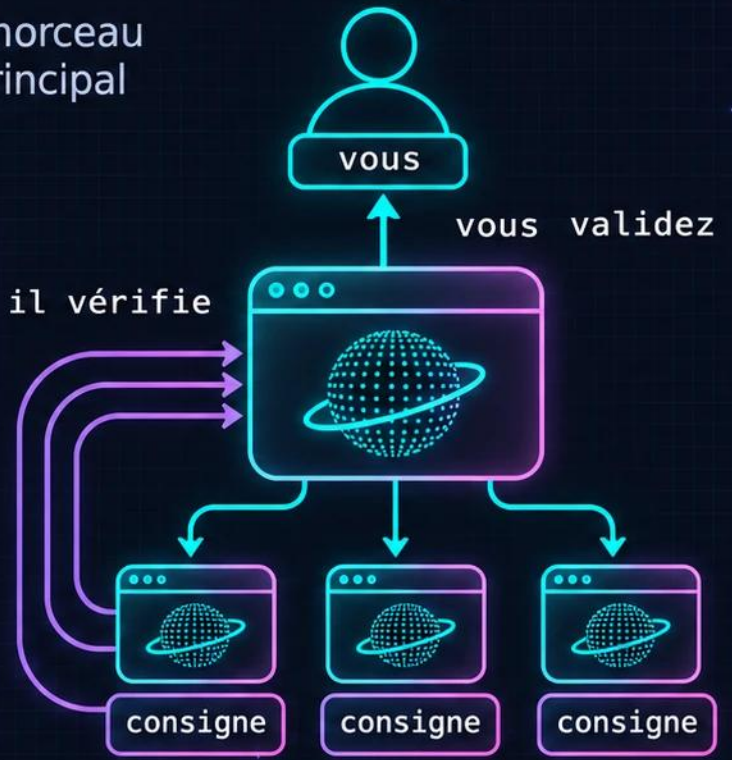
Donnez aussi la stack et ses versions, et un fichier voisin à imiter.

Sur un gros travail, l'agent principal découpe en lots et confie chaque lot à un sous-agent

Vous ne parlez qu'à un seul agent, l'agent principal. Un lot est un morceau qui se vérifie seul. Un sous-agent est un autre agent que l'agent principal lance lui-même, avec une consigne qu'il écrit.

qui fait quoi

vous	ce que vous voulez
l'agent principal	le plan, les tickets, les lots
des sous-agents	un lot chacun, avec sa consigne
l'agent principal	il vérifie leur travail et demande des corrections
vous	vous vérifiez, puis vous corrigez ou vous validez



Le travail se prouve par les vérifications, par votre essai à la main et par un autre modèle

Question : pourquoi le modèle qui a écrit le code ne suffit-il pas à le relire ?

Lint, types, tests et build verts avant tout commit, et aucune sortie inventée. Refaites ensuite le parcours sur le vrai matériel, le téléphone ou l'écran de la salle. Avant chaque push, un autre modèle relit en lecture seule et rend dix constats au plus. Jamais un modèle seul ne juge son propre travail.



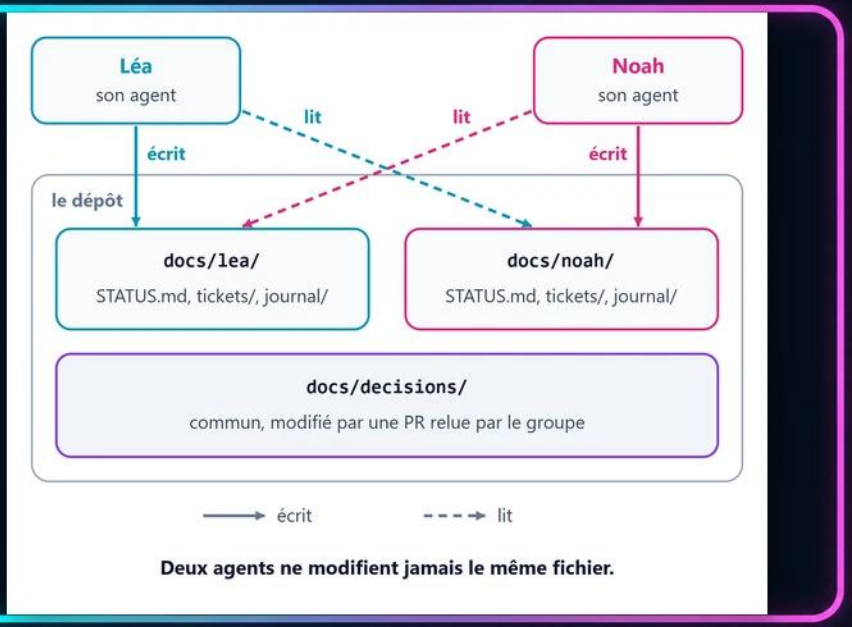
Vous coupez ce que l'agent ajoute, et une remarque faite deux fois devient une règle écrite

Un agent ajoute volontiers. Vous retirez : « trop compliqué, retire les redirections ». Quand vous répétez une remarque, demandez : « fais une procédure dans le AGENTS.md pour que tous les agents respectent la règle ». La session suivante et l'agent de votre coéquipier la liront.



Chaque membre tient son état dans son dossier, et les décisions communes passent par une PR

Question : deux agents veulent modifier le même fichier d'état. Que prévoit la règle ?



chaque agent écrit chez son membre et lit les autres

Une branche par tâche et une PR, comme au projet 1. Votre dossier rejoint main avec votre PR, et les autres y lisent votre état. Lors d'un essai avec un petit modèle gratuit, l'agent a refusé de corriger le fichier d'état d'une coéquipière. Il a proposé de le lui dire ou d'écrire une décision par PR.

Demandez une version plus profonde de la tâche, pas une version plus grande

Question : vous écrivez « agis comme un expert ». Qu'est-ce que cette phrase change au travail de l'agent ?

1. **Le but avant la tâche**
2. **Les méthodes des experts, pas leur costume**
3. **Un objectif précis et difficile**
4. **Plusieurs métiers sur le même travail**
5. **Un juge extérieur, jamais l'agent qui note son propre travail**

une demande écrite avec ces conseils

Avant de coder l'écran des réservations : comment s'y prendrait un développeur expérimenté de l'équipe Laravel ? De quelles données partirait-il, quels tests écrirait-il d'abord, et quelle mesure prouverait que l'écran est prêt ?

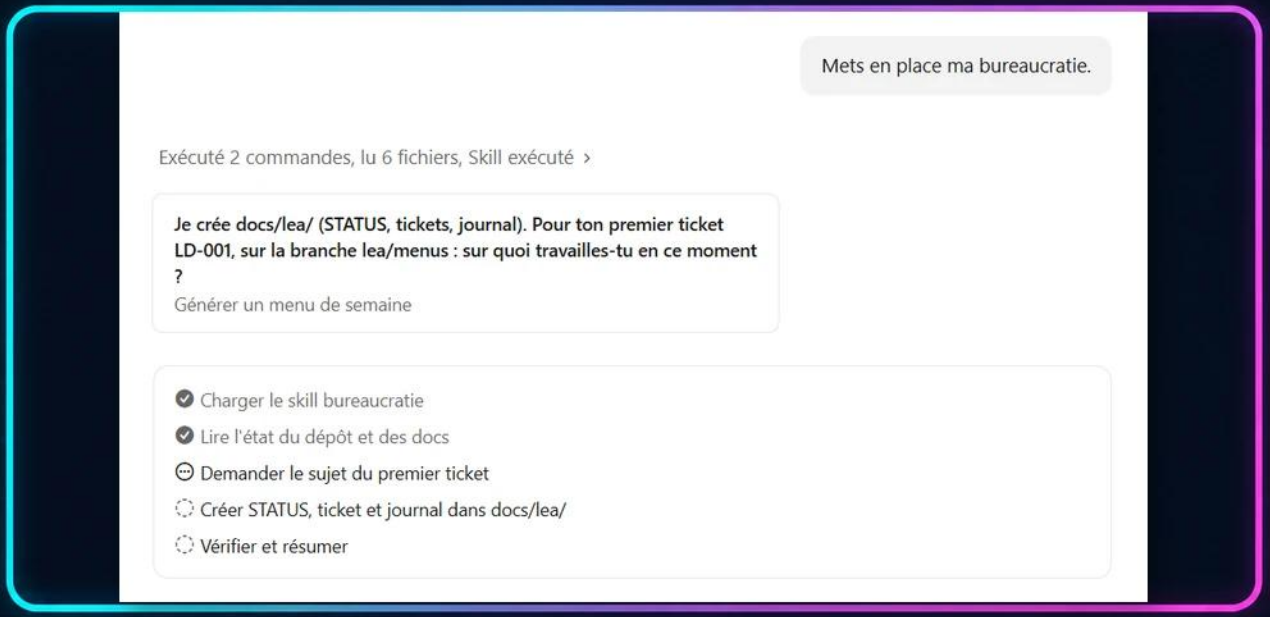
Propose, ne code pas encore.

D'après un article de Felix Rieseberg, chez Anthropic.

L'agent ouvre un skill quand votre demande correspond à la description du skill

Question : vingt skills sont installés dans le projet. Que lit l'agent de chacun au début de la session ?

Un skill est une consigne écrite une fois et rangée dans un dossier, avec un fichier `SKILL.md` qui commence par un nom et une description. Ici, la demande tient en une phrase, en haut à droite. L'agent l'a reconnue dans la description, a chargé le skill, puis a suivi ses étapes.



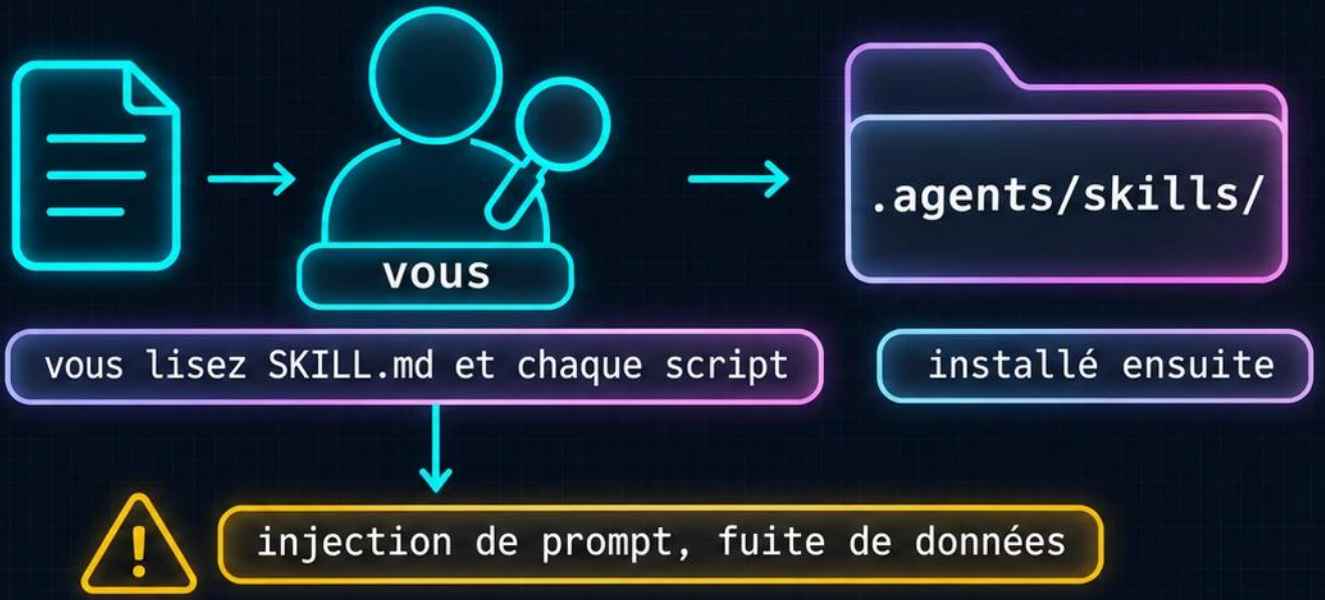
une demande en une phrase, puis le skill chargé

Un skill reçu se lit avant de s'installer, car plus d'un sur quatre contient une faille

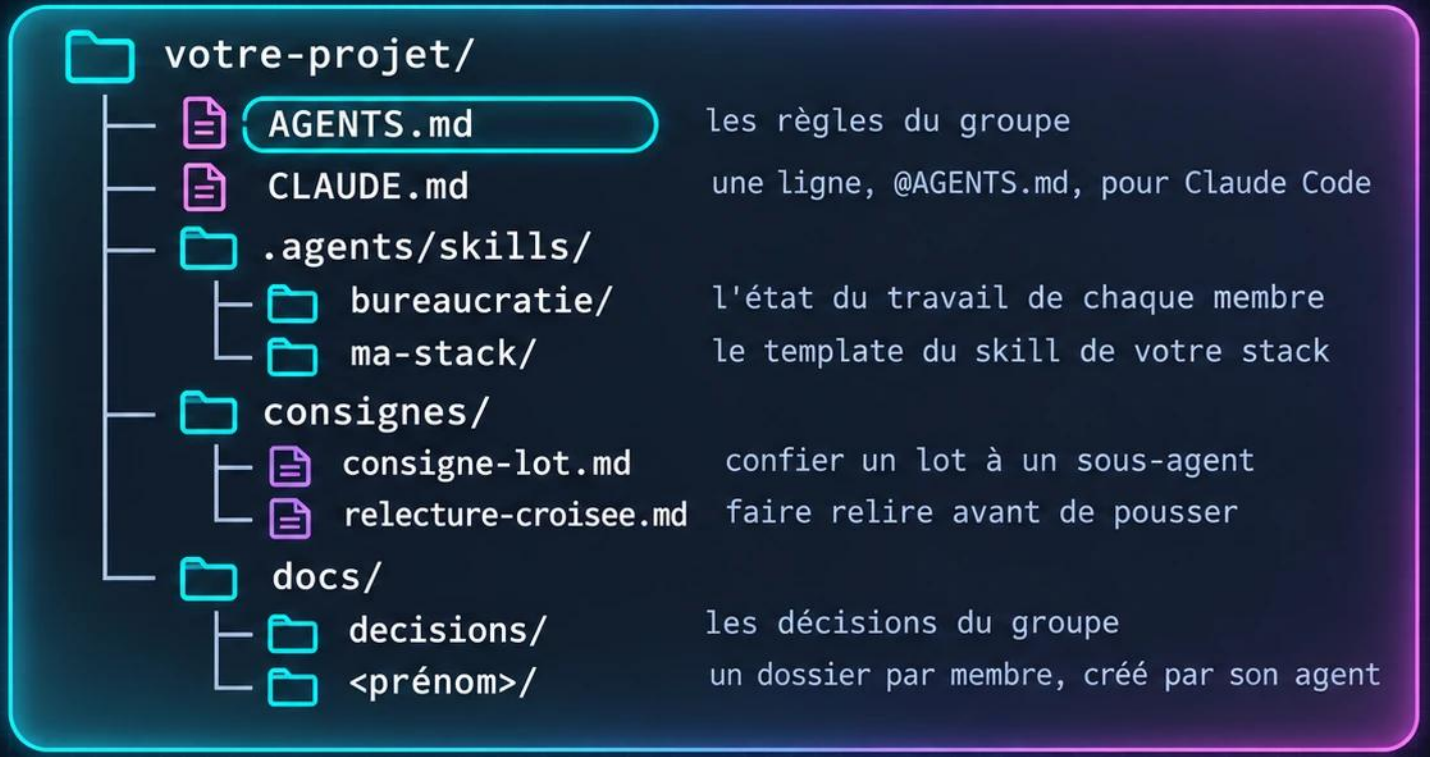
Un skill contient des instructions que l'agent suivra, et parfois des scripts qu'il lancera. Lisez `SKILL.md` et chaque script avant d'installer. Mieux, réécrivez le skill reçu à votre façon, ou écrivez le vôtre.

26,1 %

de **31 132** skills de deux grandes places de marché contiennent au moins une faille



Le kit pose dans votre dépôt les règles du groupe, deux skills et deux consignes



Tous les agents lisent **AGENTS.md** au début de chaque session. Une de ses règles : ni commit ni push sans demande, même quand le mode automatique les laisserait passer. Vous ne remplissez que **ma-stack**, en groupe. Votre agent principal remplit **consigne-lot.md** pour chaque lot qu'il distribue.

En atelier, vous menez votre application seul, puis vous posez le kit et le skill du groupe

- **Seul, environ 45 minutes** : une application de votre choix, en quatre tours, avec la méthode
- **En groupe, environ 5 minutes** : le kit dans le dépôt du projet 2
- **En groupe, environ 45 minutes** : le skill de votre stack, puis un agent qui le charge et le suit

