ORION AI 活用 LT ｜ 2026.11.19

pstackについて

寺地 龍樹 ｜ @nocode_teraji



AI エージェントのフルスタックエンジニアをやってます



寺地 龍樹

@nocode_teraji

出身	和歌山
趣味	筋トレ、スポーツ全般、ゲーム
経歴	ERP ベンダーで SE・開発 → ソフトバンクで AI エージェント開発
個人でやっていること	ゲーム開発（大富豪ロワイヤル）、OSS へのコントリビュート
pstack 歴	約 1 か月（2026年10月〜、Claude Code で）

pstack は、AI エージェントが賢く動くようになるスキル集

生まれ方：AI の失敗を、1 つずつスキルに

最初からプラグインを作るつもりはなく、見かけた失敗のパターンをスキルにしていったら、まとまった

中身：AI 向けのスキル集（プラグイン）

本人が Cursor で毎日使っている手順と判断の基準を、そのまま入れてある。無料（MIT）

2026年8月の 1 か月で本番に出した PR

2,500 件

pstack を使って（本人の発表より）

作ったのは、Grokbotのエンジニアのローレンタンさん



Lauren Tan

@poteto（ポテト）

Netflix	テックリード、エンジニアリングマネージャー
Meta	React コアチームで、React Compiler を担当
Cursor	毎日使うスキルを作り、pstack として公開
いま	SpaceXでGrokbotを開発
最近の講演	AI Engineer CODE 2026「All Code Is Slop (Except Mine)」（11/11）

Cursor でも Claude Code でも。自分のエージェントにも入れられる

Cursor

もともと Cursor のプラグインとして公開されている

Claude Code

プラグインとして、そのまま入れられる（私はこちら）

自分のエージェント

pstack を渡して「自分のエージェント用にカスタマイズして」と頼めば入る。私は Muse にも入れている

入れ方は、最後に紹介します

WHY GREAT ｜ 何がすごいの？

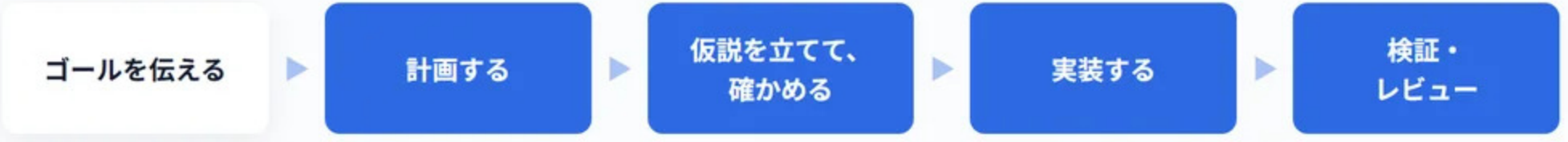
長いタスクを、
めちゃくちゃ賢くやりきる

いちばんわかりやすい、pstack のすごさ

ふつうの AI エージェントは、長いタスクの途中で崩れる

抜け漏れる	前に伝えたことが、途中で抜け落ちる
止まる	長い作業の途中で、手が止まってしまう
指示どおりに動かない	最初に頼んだことと、違うものができる
品質が低い	動くけれど、雑なところが混ざる
道がそれる	関係のない方向へ、進んでいってしまう

計画して、確かめてから作り、最後にもう一度確かめる



青＝AI　白＝人

実装は、AI が AI に指示する

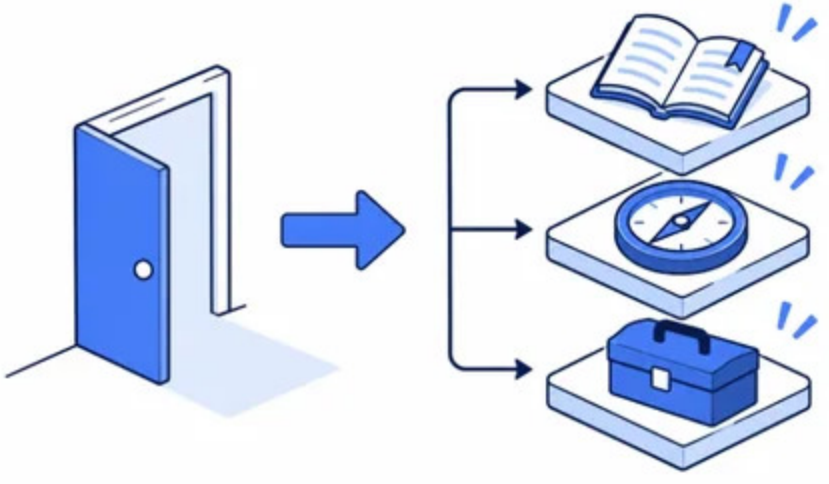
メインのモデルが指揮役になり、サブエージェントを起動して作業を任せる。終わったら、指揮役が結果を確かめる

IN SHORT | つまり？

優秀なエンジニアが
無意識にやっていることを、
スキルにした

中身は「手順・判断・道具」の３層

扉が /poteto-mode。その先に、手順書・コンパス・道具箱がある



扉 ＝ 入口の /poteto-mode。依頼を読んで、どれを使うかを選ぶ

23 **Playbook ＝ 本**
作業ごとの手順書

24 **Principle ＝ コンパス**
迷ったときの判断の基準

34 **Skill ＝ 道具箱**
単体でも呼べる道具

使うべきなのは全員。
ただし、長いタスクのときに

向いている：長い・大きいタスク

- ✓ バグを直す（再現 → 原因 → 修正 → 確認）
- ✓ しくみや原因を調べる（調査だけ、も頼める）
- ✓ 機能を足す・作りを整理し直す
- ✓ 遅いところを、測ってから速くする
- ✓ 寝ている間に、長い作業を進めてもらう

向いていない：小さいタスク

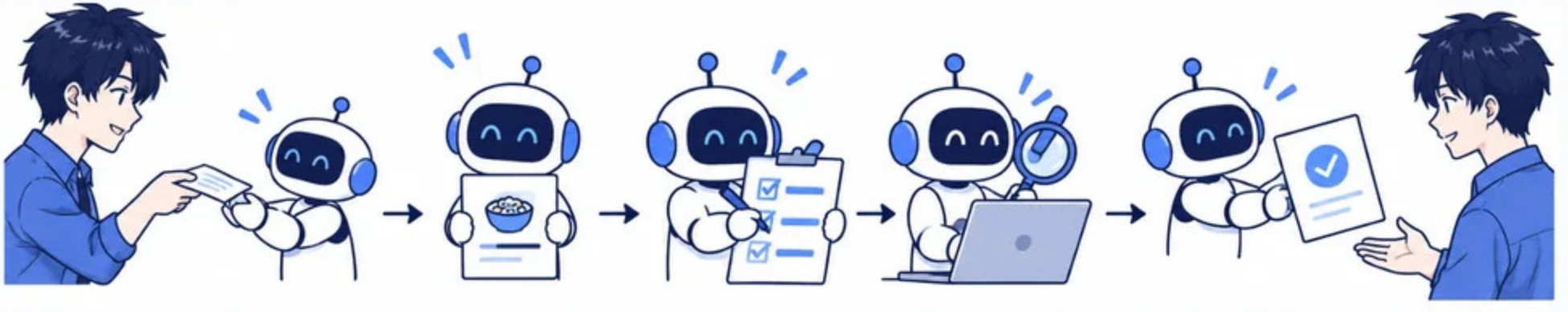
1 行だけの修正なら、ふつうに頼むほうが速い。

手順がしっかりしているぶん、小さな作業には重すぎることがある。

迷ったら、/poteto-mode に「ゴール」を伝える

/poteto-mode

依頼	このスライドを作るスキルを作りたい
ゴール	私が「いい感じ」と思う会社紹介資料を集めて、「こうなったら完成」と見せた



流行りの「Human out of the loop」を、そのまま形にしたスキル

これまで

人がループの中にいる

AI が作るたびに、人が確かめて、指示を出し直す

▶

Human out of the loop

AI が AI を改善していく

計画・実装・検証・手直しを、AI が自分で回す。人は介入しない

最初は、人が結果を評価して AI を「育てる」

① 結果を自分で確かめる

最後のアウトプットは、やっぱり人が見る

② 良かったか、だめだったかを評価する

1 回まわした結果がどうだったかを、自分で判断する

③ 仕組みを設計し直す

評価をもとに、スキルやテストを整えていく



ローレンさんは、この「設計」にいちばん時間をかけている

注意書きを 1 行足す前に、コードで「間違えられない」ようにする

① コードで縛る
API 設計・アーキテクチャ・lint・型・テスト

② 検証で確かめる
実行して確かめる・形式検証

③ 文章で伝える
ルール・スキル・AGENTS.md

逆ピラミッド：上ほど時間をかける

useEffect を禁止
使いこなしが難しいものは、製品コードで使えなくした

またぐ import を禁止
フォルダの分け方と lint で、入れてはいけない依存を止めた

コメントを 99% 禁止
手抜きの言い訳コメントが、次の AI への「許可証」になるから

レビューするのは差分ではなく、動かした「証拠」

動かして確かめる

AI が自分でアプリを動かす。「機能マップ」があるので、どの画面のどこかで迷わない

/create-verification-skill

TLA+ で全部の順番を試す

仕組みを「模型」に書き、起こりうる順番を機械が全部試す。複数端末の通知の流れを確かめた

/create-formal-verification-skill

図にして人が読む

画面の状態の移り変わりを図にして、動画やスクショと一緒にレビューする

/statechart

ローレンさん：「トークンのほとんどは、検証に使っている」

数を増やしても、品質は落ちなかった

Grok Bot チームの先月の数字（講演での報告）

本番に出した PR

11,000件

チーム全体、1 か月で

取り消した PR の割合

約0.5%

数が増えても、低いまま

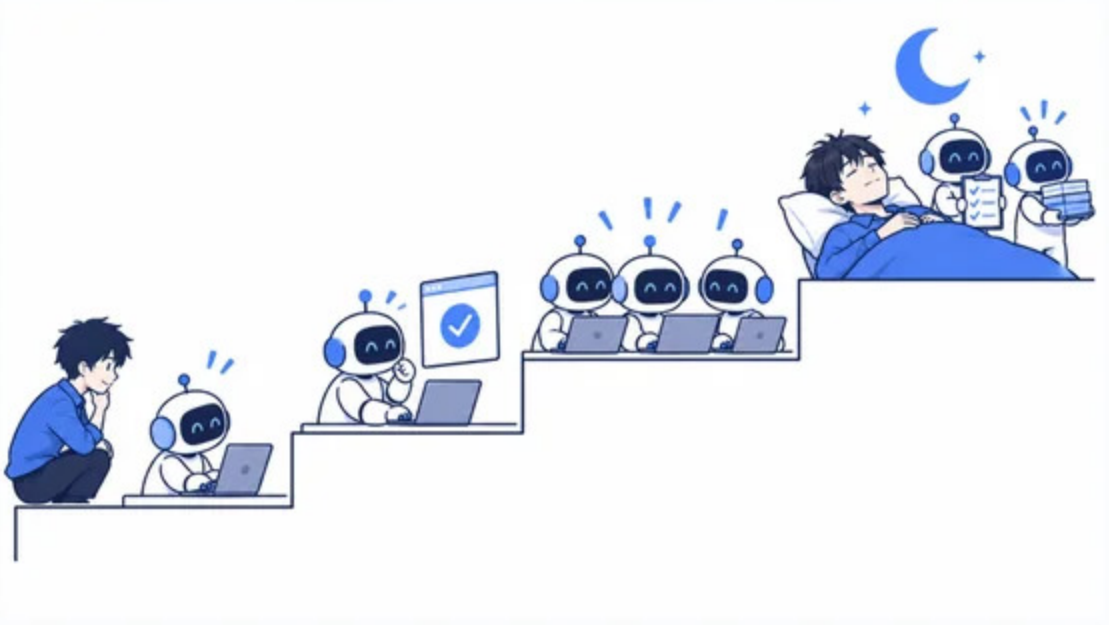
安定版のリリース

ほぼ毎日

起動や操作の速さも、公開時より改善

中身が雑なまま自動化しても、雑なコードが増えるだけ。まずはコードと検証から

目指すのは、人が介入しない全自動。(まだできていない)



STEP 1 見張る

1 体の作業を、横でずっと見る

STEP 2 確かめさせる

動かして確かめるのも、AI にやらせる

STEP 3 並べる

信頼できたら、何体も同時に走らせる

STEP 4 任せる ローレンさんはここ

朝起きると、PR がもうマージされている

入れて、モデルを選ぶだけ。Claude Code でも使える

Cursor

STEP 1　入れる

```
/add-plugin pstack
```

STEP 2　役割ごとのモデルを選ぶ

```
/setup-pstack
```

Claude Code（私はこちら）

STEP 1　入れる

```
/plugin marketplace add cursor/plugins
/plugin install pstack@cursor-plugins
```

STEP 2　役割ごとのモデルを選ぶ

```
/pstack:setup-pstack
```

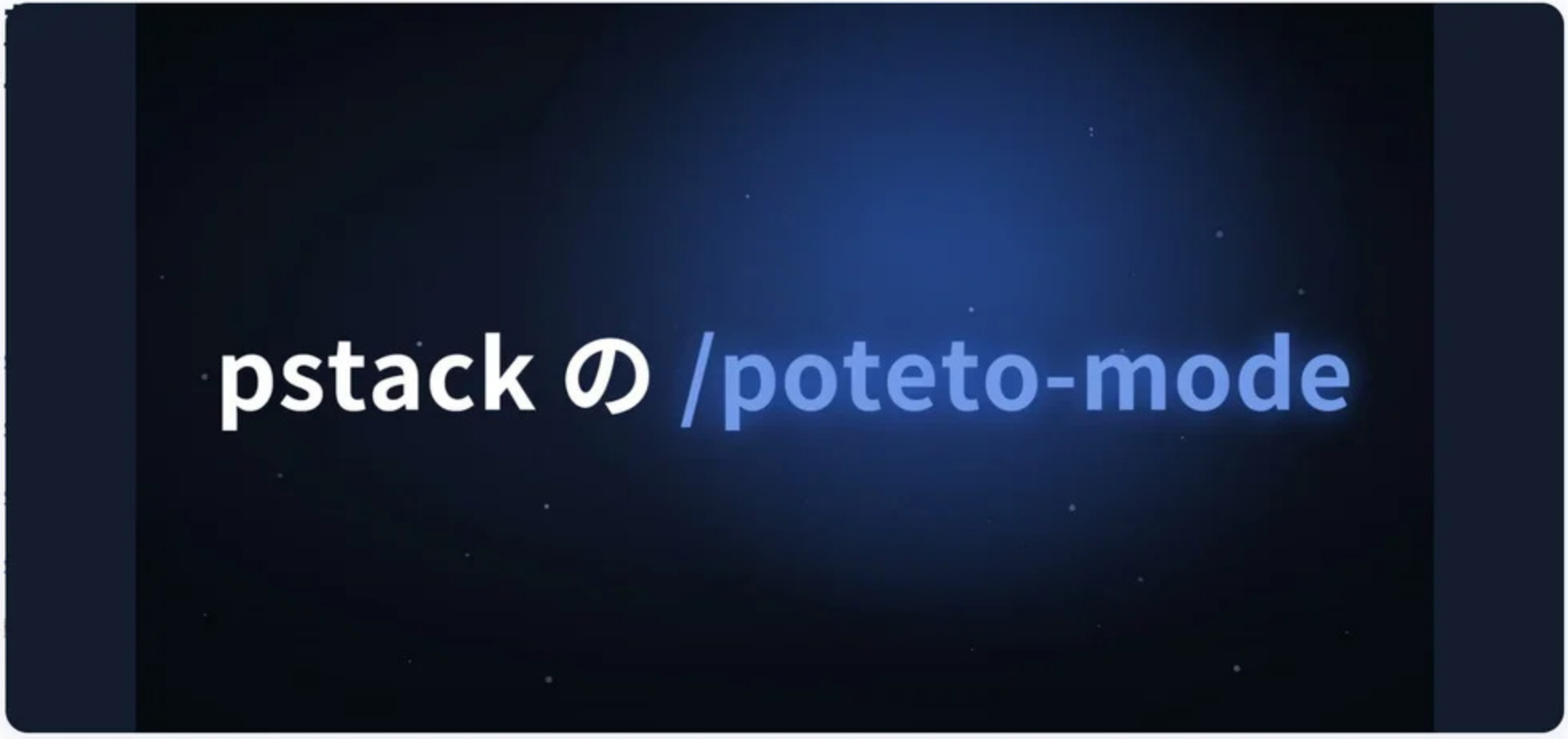
Claude Code では名前の頭に pstack: が付く（/pstack:poteto-mode）。迷ったら /poteto-help に聞けばいい

長いタスクを、AI に任せるために

01 pstack は、優秀なエンジニアの「無意識」をスキルにしたもの

02 長いタスクに使う。迷ったら /poteto-mode にゴールを伝える

03 最初は人が評価して、少しずつ任せられるように育てる


pstack の /poteto-mode

TRY IT

次の長いタスクで、/poteto-mode を試してみてください

感想・質問・連絡は X まで

@nocode_teraji



x.com/nocode_teraji

私の「たぶんこれが原因」を、動かした証拠でひっくり返された

OSS（Kubernetes の管理画面 Headlamp）で、アクセシビリティ違反を調べてもらった

私の推測

ラベルのない入力欄は、選択欄（MUI Select）の隠し入力欄では？（未確認）

poteto-mode が確かめたこと

① テストを Windows で実際に実行した（メモリ不足も自分で回避）

② 作成フォーム 41 個を全部調べて、違反ゼロを確認した

③ 原因の候補を 1 行だけ戻し、違反が出ることを再現した

結論：原因は共通部品の入力欄で、もう main で直っていた。新しい issue は立てなくてよい

「ルールを書き足す」ではなく、「確認を自動にしよう」と返ってきた

このスライドを作るスキル（slide-create）の作り方を相談した

STEP 1　聞いたこと

SKILL.md のほかに、フォルダを分けなくていい？

STEP 2　返ってきた答え

今は 1 枚で足りる。ただ、毎回手でやっている確認はスクリプトにしよう

STEP 3　できたもの

チェック用スクリプト（テスト 16 件つき）。このデッキも、それで確かめてから公開した

pstack の原則「教訓は文章ではなく、仕組みに残す」そのもの。**ローレンさんも先週の講演で、同じ話をしていた**