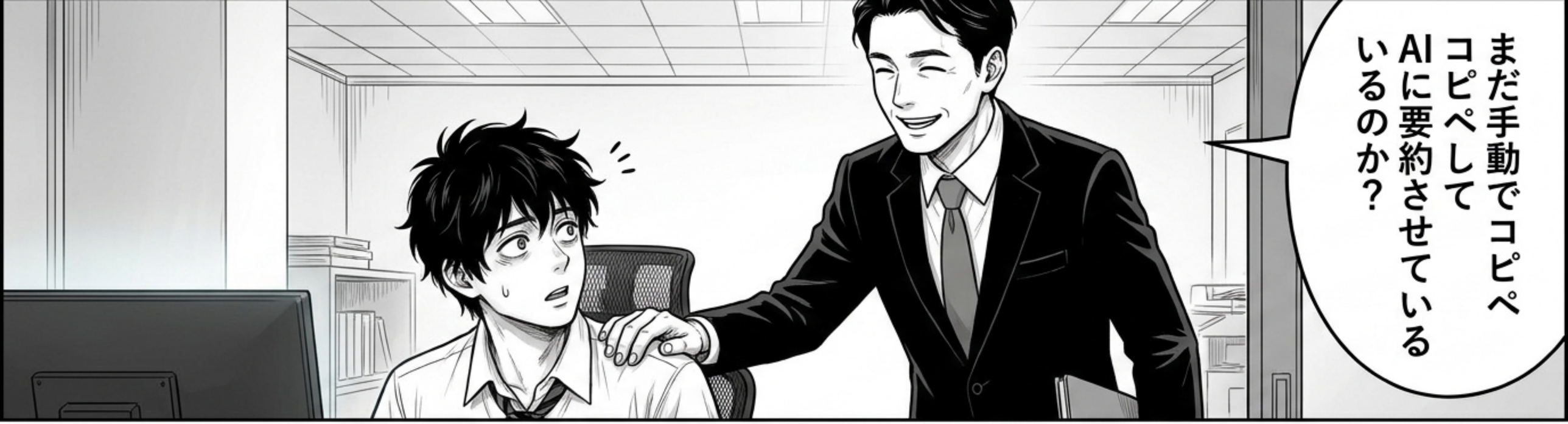

Terminal
ターミナルに常駐する自律型エージェントとの協働プロセス
Claude Codeが変える、エンジニアの新しい日常
「伝わる」Gitコミット
メッセージを自動生成する
Gemini Notebook


よし、
複雑なロジックの
実装終わった…！
git commit -m ""
実装直後の「言語化」。
これが意外と重い
認知的負荷になる。
うぅ…
自分が書いたコード
何を変更したか
簡潔にまとめる気力が
気力残ていない…
Gemini Notebook

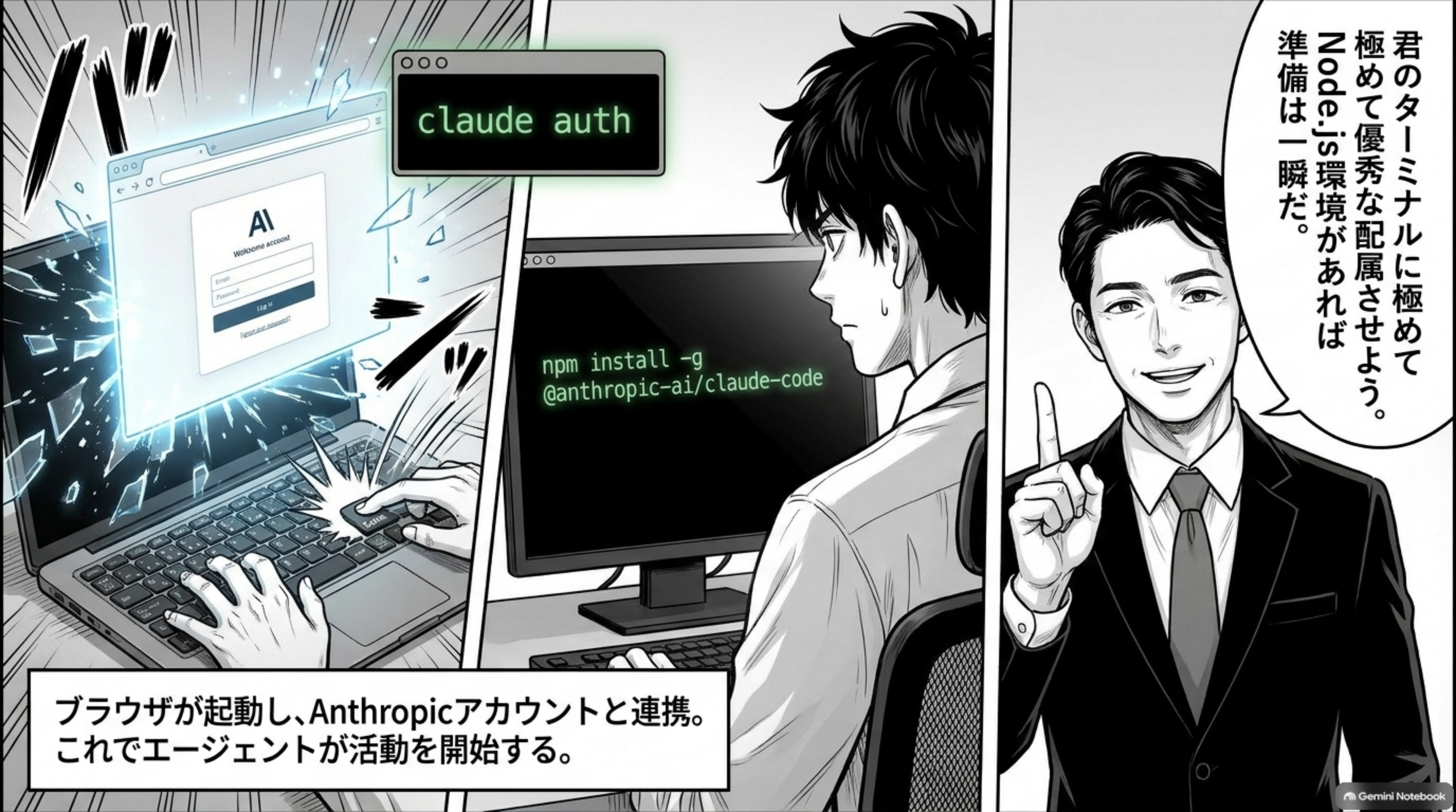


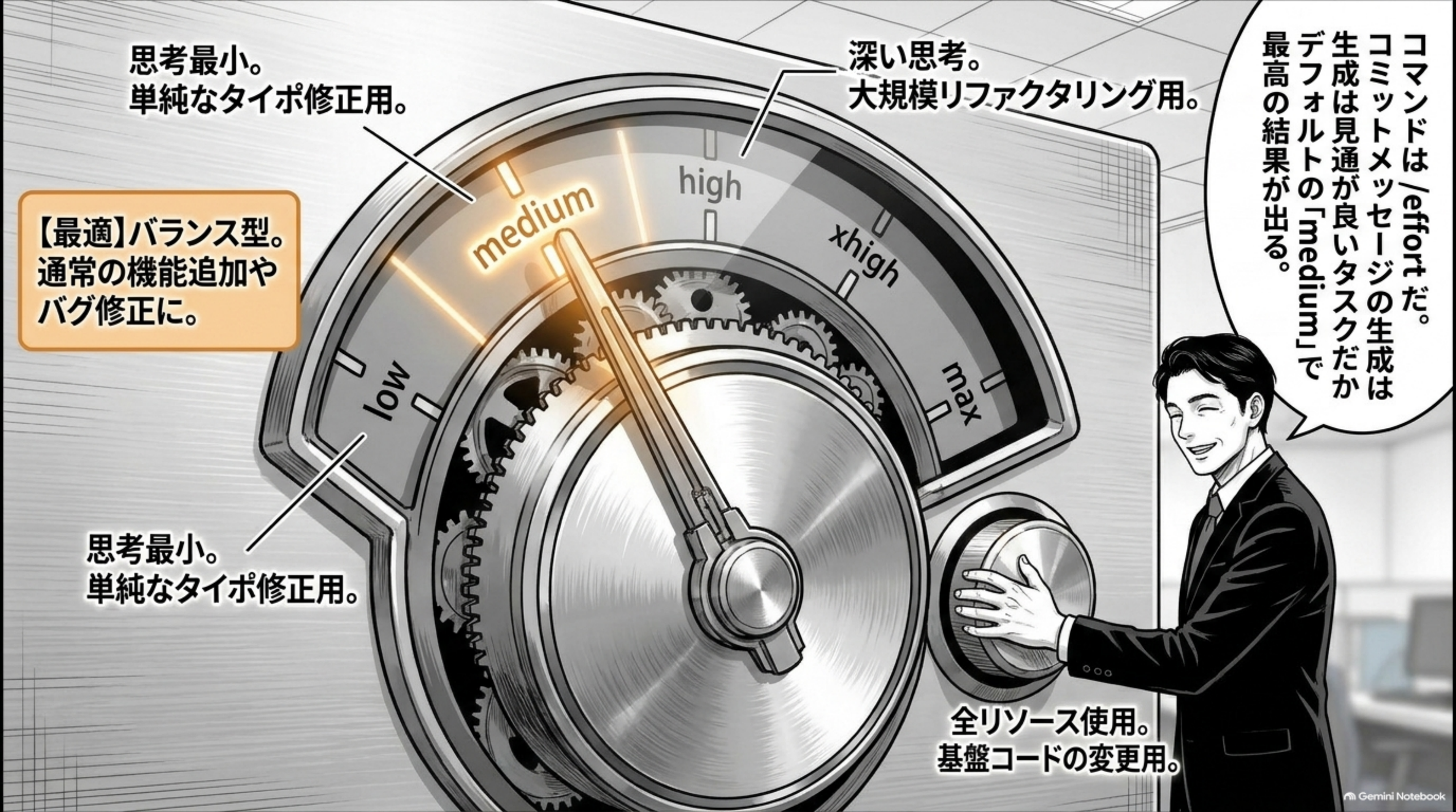
従来のチャットAI

- コンテキスト理解なし
- 変更内容をコピペして依頼
- 単なる「自動化（Automation）」ツール

Claude Code

- ターミナルに常駐
- ファイル探索・Git操作を自律代行する「エージェント」
- 共に考える「思考の拡張（Augmentation）」ツール


君のターミナルに極めて
極めて優秀な配属させよう。
Node.js環境があれば
準備は一瞬だ。
claude auth
npm install -g
@anthropic-ai/claude-code
ブラウザが起動し、Anthropicアカウントと連携。
これでエージェントが活動を開始する。


コマンドは /effort だ。
コミットメッセージの生成は
生成は見通が良いタスクだか
デフォルトの「medium」で
最高の結果が出る。
思考最小。
単純なタイポ修正用。
深い思考。
大規模リファクタリング用。
【最適】バランス型。
通常の機能追加や
バグ修正に。
low
medium
high
xhigh
max
思考最小。
単純なタイポ修正用。
全リソース使用。
基盤コードの変更用。
Gemini Notebook

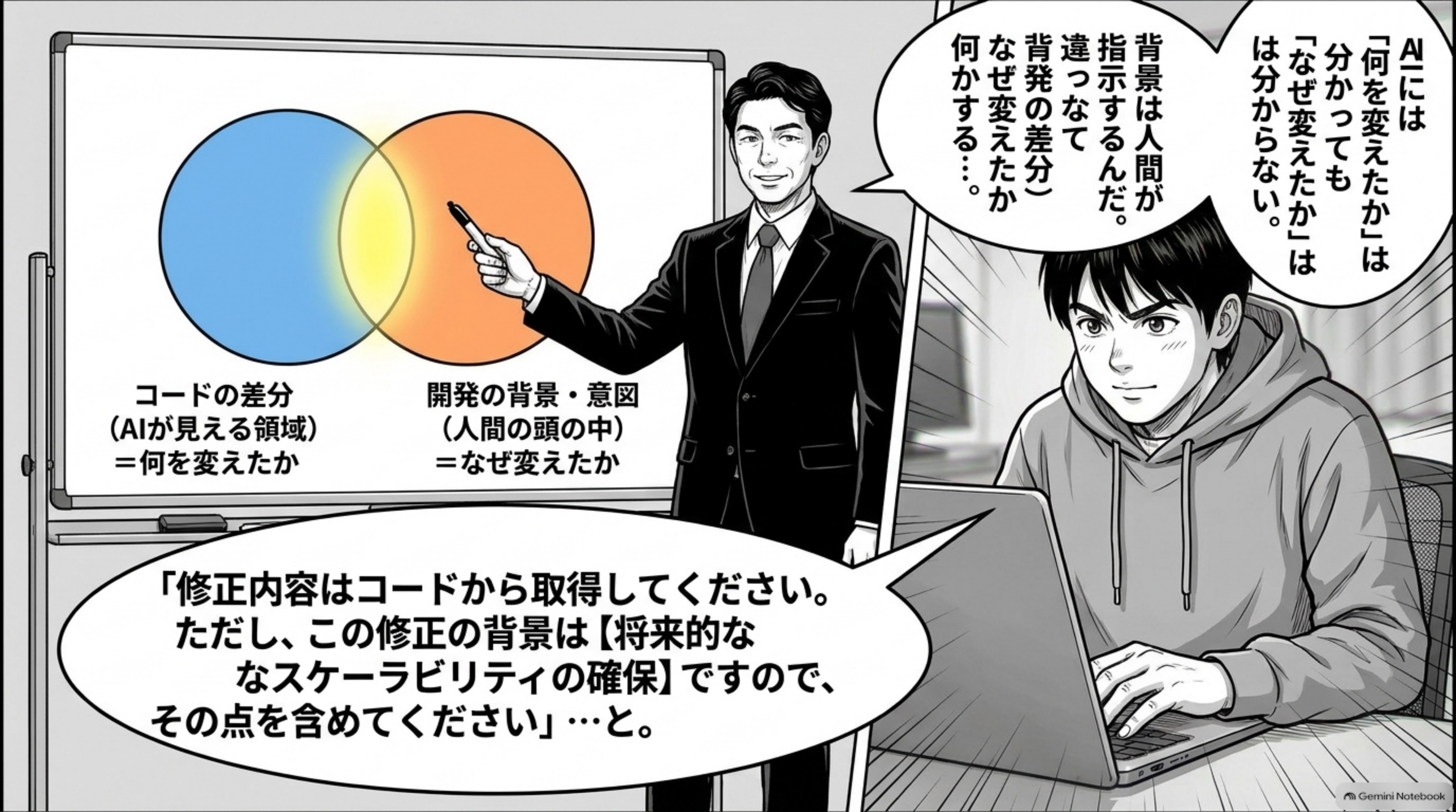

git diff --cached
指示と同時に、AIが内部で自律的にツールを使用し、
変更箇所を探索（Explore）している…！

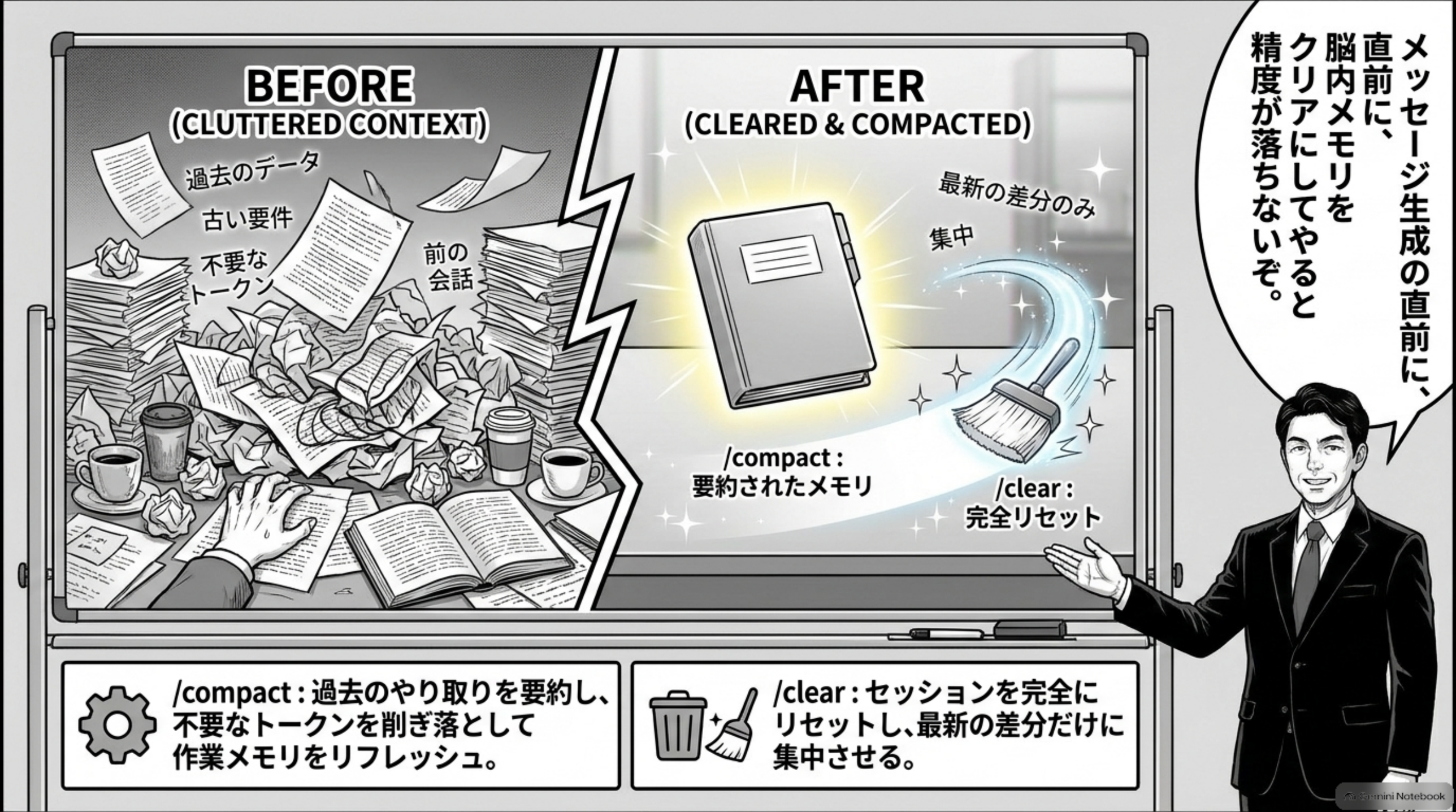

現在のステージングされた
変更内容を確認して、
Conventional Commits形式
メッセージを提案して。
提案して。
claude
Gemini Notebook

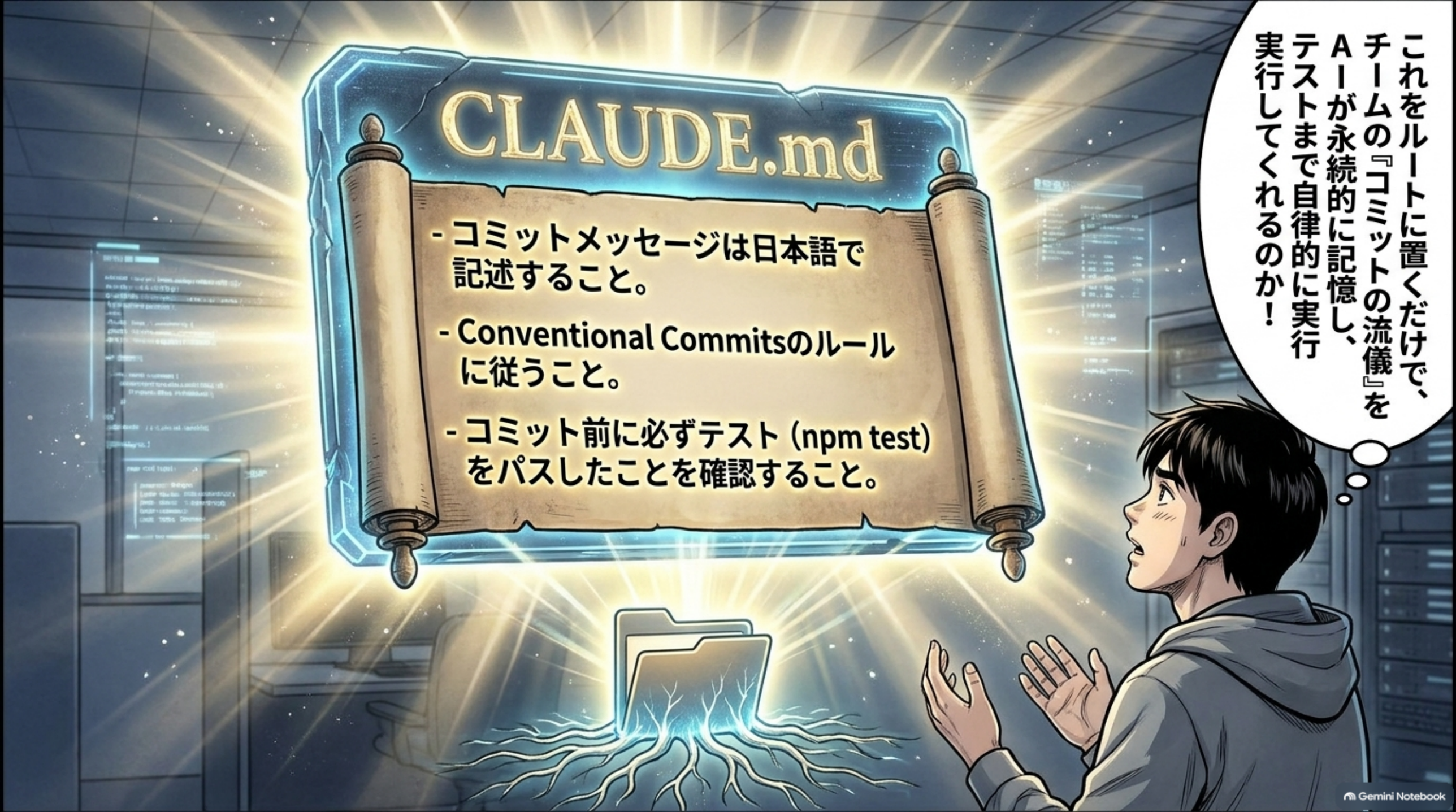
feat: ユーザー認証フローの追加

- 詳細内容は、ユーザー認証フローの追加しております。
- これは作業の確かられた場合が購入してコースに変動性を確保した。
- 変更化メッセージを維持知証するなんだよ。


...将来的なUI拡張を見据えた
データ構造の変更
あれ？
頼んでいない
「将来の拡張」のことまで
背景として書かれている…
コードにはそんな情報
ないのに！
？
次トークン予測の限界。もっともらしい言葉を続けるため、
コードにない『背景』を勝手に推測してしまうことがある。
Gemini Notebook


AIには
「何を変えたか」は
分かっても
「なぜ変えたか」は
は分からない。
背景は人間が
指示するんだ。
違って
背発の差分）
なぜ変えたか
何かする…。
コードの差分
（AIが見える領域）
＝何を変えたか
開発の背景・意図
（人間の頭の中）
＝なぜ変えたか
「修正内容はコードから取得してください。
ただし、この修正の背景は【将来的な
なスケーラビリティの確保】ですので、
その点を含めてください」…と。
Gemini Notebook


BEFORE
(CLUTTERED CONTEXT)
過去のデータ
古い要件
不要な
トークン
前の
会話
AFTER
(CLEARED & COMPACTED)
最新の差分のみ
集中
/compact :
要約されたメモリ
/clear :
完全リセット
メッセージ生成の直前に、
直前に、
脳内メモリを
クリアにしてやると
精度が落ちないぞ。
/compact : 過去のやり取りを要約し、
不要なトークンを削ぎ落として
作業メモリをリフレッシュ。
/clear : セッションを完全に
リセットし、最新の差分だけに
集中させる。


CLAUDE.md
- コミットメッセージは日本語で記述すること。
- Conventional Commitsのルールに従うこと。
- コミット前に必ずテスト（npm test）をパスしたことを確認すること。
これをルートに置くだけで、チームの『コミットの流儀』をAIが永続的に記憶し、テストまで自律的に実行してくれるのか！
Gemini Notebook

ターゲット読者による出力トーンの調整 (AI Fluency: Description)

技術レビュアー向け

指示のポイント:
技術的な詳細と影響範囲を重視。

```
export codes {
  constratcontest(() => {
    // URL: "https://src/auth/ai/downioat",
    indoxn: ℭuleaf "hetp.ai⁄context="
  };
```

fix(auth): トークン検証時の境界値エラーを修正し、401応答を正確に返すよう変更

```
// output #oot
$ githut℅ /--donatmic commits error
```

マネージャー向け

指示のポイント:
ビジネス上の価値と解決した問題を重視。

```
>_
```

fix: ログイン時に稀に発生していた接続遮断断問題を解消。ユーザー離脱を防止

誰が読むのかを明示するだけで、AIの出力は劇的に変化する。


最終的な記述内容は
私（作成者）が
全責任を
負う。
AIが生成した
したメッセージを
そセージを
使うまま使うのは
効率的だ。
だが、不正確な推論
推論がかを検証し、
最終的な説明責任を
あくまで人間である
間である君自身だ。
はい…！
ツールに依存
するのではなく、
共に品質に責任を
責任持ちます。
Gemini Notebook

「何を変えたか」だけでなく「なぜその設計判断をしたのか」を残す。
これが未来のメンテナンス性を支える礎となる。


よし、完璧な一行だ。
これなら数ヶ月後の
自分も、数ヶ月自分
チームメンバーも
助かるはず。
6:00 PM
事務的な要約はAIに。
人間は意図の確認と品質の担保に集中する。
これがAI時代のエンジニアリング。
さあ、ターミナルを開き claude と入力しよう。
/usage で確認できる微々たるコストで、
計り知れないプロジェクト履歴の価値が手に入る。