完全ガイド：Claude Codeで Gitコミット操作をマスターする

～ 初心者でも迷わない実践手順書 ～



主人公（26歳・エンジニア）：
Git操作に苦手意識を持つ若手。

上司（50歳・マネージャー）：
豊富な経験と最新AIツールを
使いこなすベテラン。

Gemini Notebook



初心者の壁：「変更内容の把握不足」と「コミットメッセージ作成の苦痛」。

Gemini Notebook

[作業ディレクトリの変更]

（自動解析）

[Claude Code]

（提案＆承認）

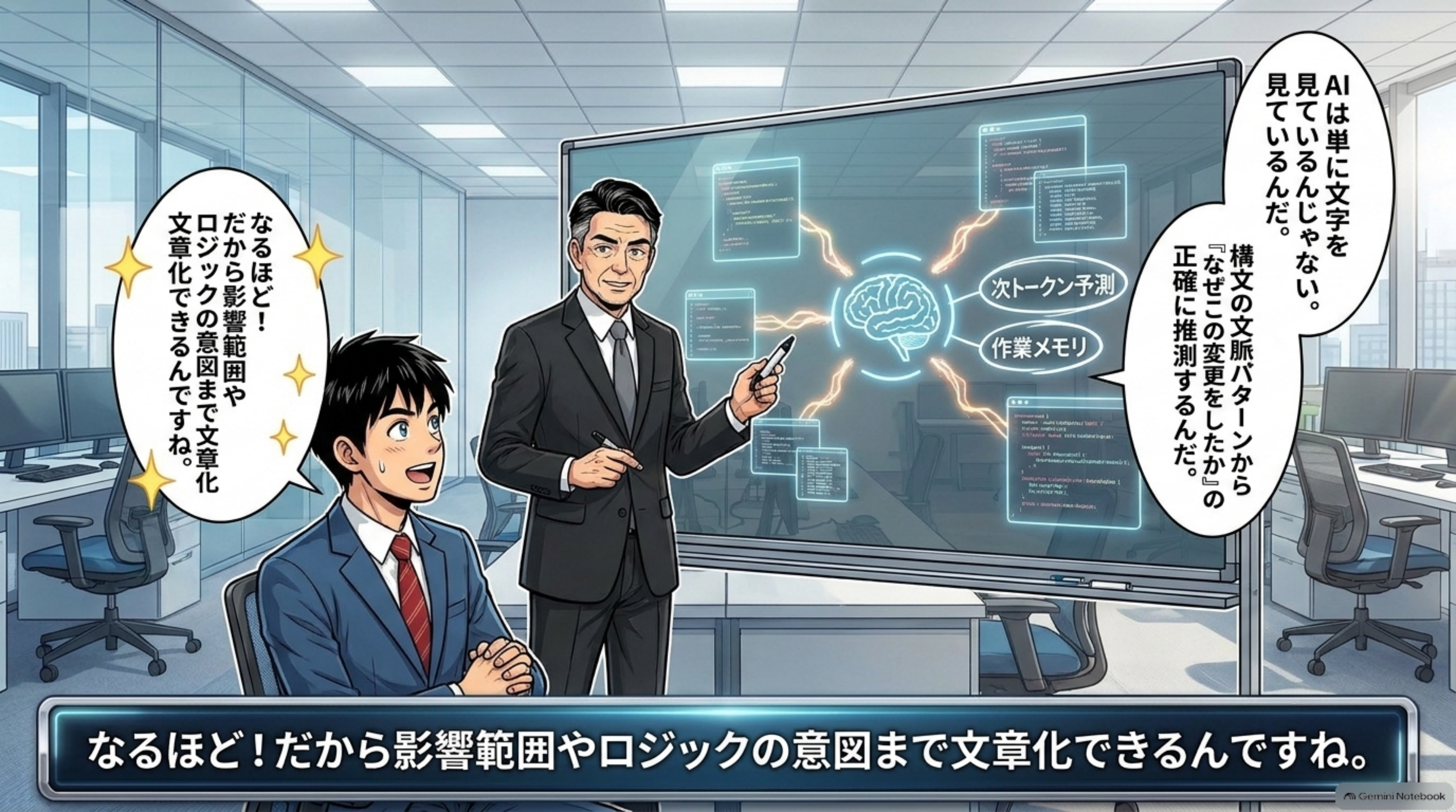
[明確なコミットメッセージ]





複雑な初期設定は不要。リポジトリ直下で起動するだけで、Git履歴（diff）を直接読み取る。

Gemini Notebook


AIは単に文字を
見ているんじゃない。
見ているんだ。
構文の文脈パターンから
『なぜこの変更をしたか』の
正確に推測するんだ。
次トークン予測
作業メモリ
なるほど！
だから影響範囲や
ロジックの意図まで文章化
文章化できるんですね。
なるほど！だから影響範囲やロジックの意図まで文章化できるんですね。

【Before (人間)】	説明
git commit -m "fix bugs" 	どの機能のバグか全く不明… 過去ログ調査が困難
【After (Claude Code)】	**結果**
git commit -m "fix(auth): correct token validation logic in login handler"	影響範囲（auth）と修正内容（トークン検証ロジック）が一目で把握可能！

これならチームのコードレビューも、将来のバグトラキングも劇的に速くなるぞ。

これならチームのコードレビューも、将来のバグトラッキングも劇的に速くなるぞ。

Gemini Notebook



基本：

変更点を確認して、適切なコミットメッセージを作成してください。

応用：

git diffを確認し、Conventional Commitsの形式に従って、日本語で簡潔なメッセージを作成してください。

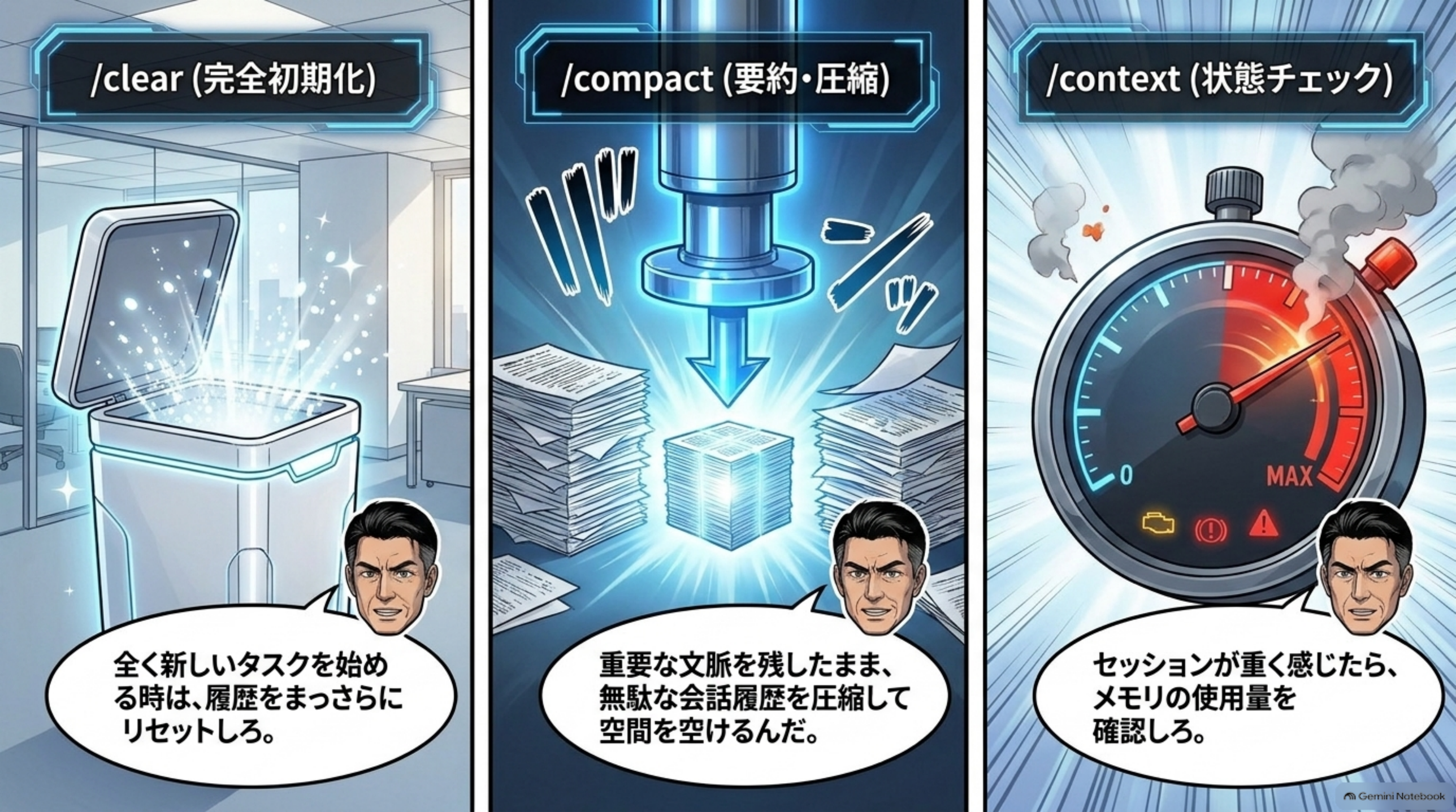
応用 (チケット紐付け)：

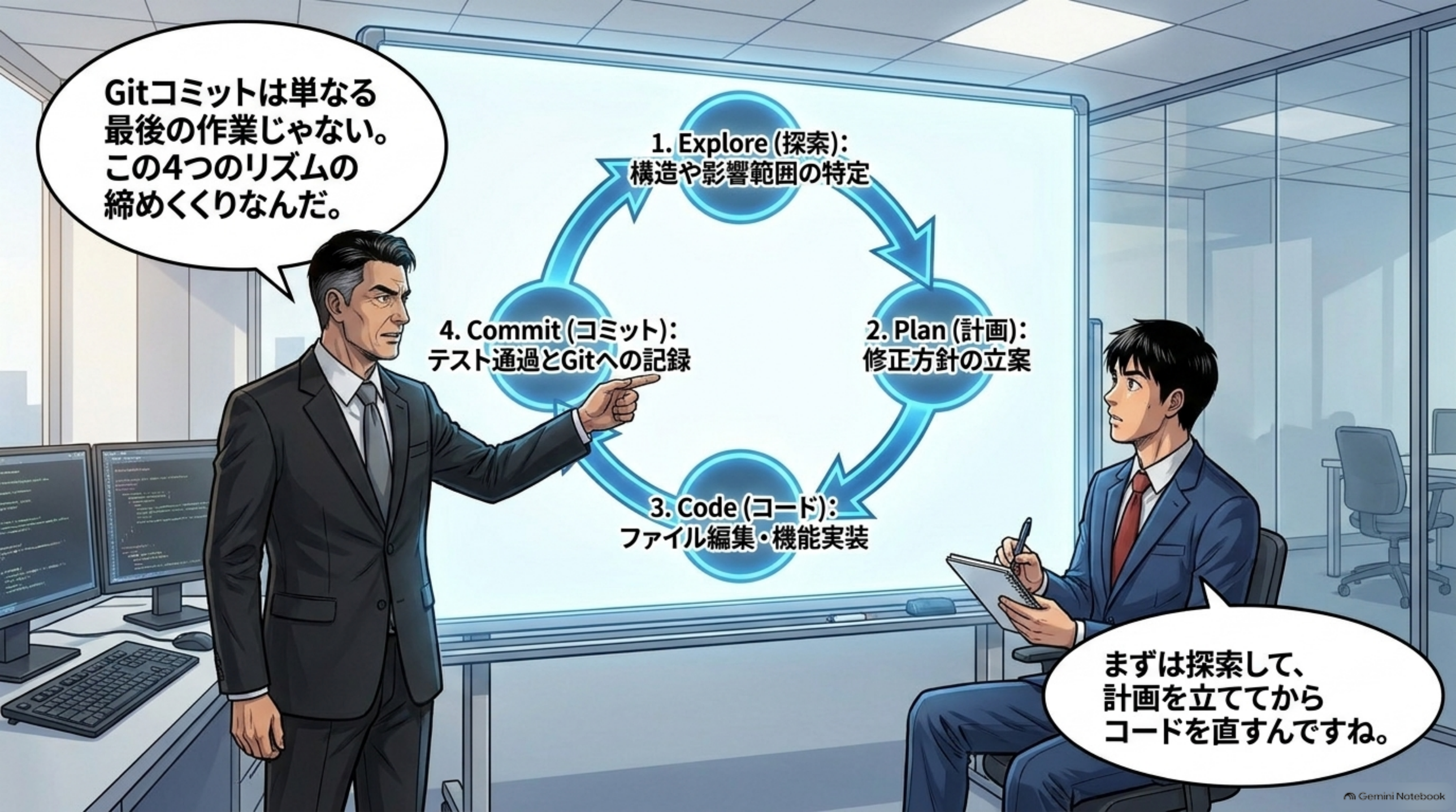
頭に [#DEV-1024] を付与して作成してください。

Gemini Notebook


Anthropicの調査によれば、
AIとのやり取りの85.7%で反復的
フィードバックが行われており、
これが品質を最大化する。
追記指示の例
もう少し簡潔にして、1行で
収めてください。
エラーメッセージの追加についても
触れてください。
待て。
AIの初稿をそのま
受け入れるるな。
対話（反復）
を重ねて
洗練させるんだ。
ゴッ
Gemini Notebook


AIの綺麗な出力を見ると、
人間の推論を疑う行動 (-3.1%)、
事実確認 (-3.7%)、
コンテキスト欠落の指摘 (-5.2%)
が著しく低下する。
AIが綺麗な出力 (アーティファクト)
を出すと、人間は『完成している』
と思い込み、事実確認を怠る。
これが【アーティファクト効果】だ。
承認ボタン(y)を押す前に、
必ず自分の目で git diff を
確認するんだ！
Gemini Notebook


/clear (完全初期化)
全く新しいタスクを始める時は、履歴をまっさらにリセットしろ。
/compact (要約・圧縮)
重要な文脈を残したまま、無駄な会話履歴を圧縮して空間を空けるんだ。
/context (状態チェック)
0
MAX
セッションが重く感じたら、メモリの使用量を確認しろ。
Gemini Notebook


Gitコミットは単なる
最後の作業じゃない。
この4つのリズムの
締めくくりなんだ。
1. Explore (探索):
構造や影響範囲の特定
2. Plan (計画):
修正方針の立案
3. Code (コード):
ファイル編集・機能実装
4. Commit (コミット):
テスト通過とGitへの記録
まずは探索して、
計画を立ててから
コードを直すんですね。
Gemini Notebook


初心者のうちは、Plan Modeで
手順を確認し、Manual Modeで
1歩ずつ進めば、
予期せぬコード書き換えや
事故は絶対に起きない。
操作モードの比較
1
Manual Mode:
毎ステップ承認
（デフォルト推奨 / 厳密な確認）
2
Plan Mode:
事前に修正計画を提示
（複数ファイルの変更・安全性確保）
3
Auto-accept:
自動承認（プロトタイピング用）
Plan Mode
&
Manual Mode
Gemini Notebook

タスクに対する努力量
(エフォート)が足りないんだ。
/effort high や /effort max
コマンドでダイヤルを上げろ！

low
medium
high
xhigh
max
auto
low

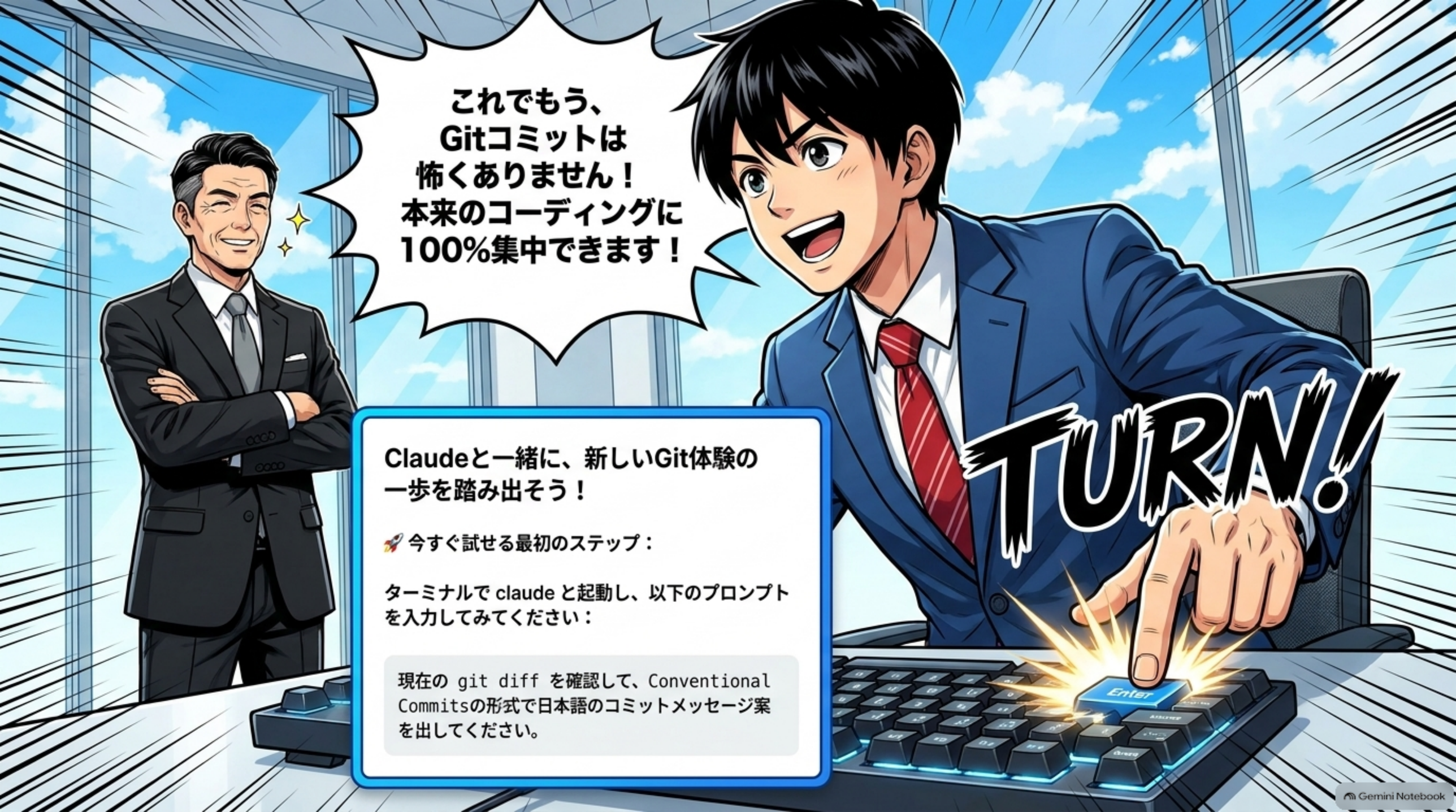
あれ？複雑なタスクの途中
でAIが作業を諦めて
戻ってきちゃいました…

知識不足で 解決できない		➡ モデル (/model) を変更
確認不足で 途中で諦める		➡ 努力量 (/effort) を変更 ※環境変数 CLAUDE_CODE_EFFORT_LEVEL も利用可能。

Gemini Notebook


毎回プロンプトで
指示するのは面倒だろ？
プロジェクト直下に
『CLAUDE.md』を
置けば、
AIが永続的な
プロジェクトメモリとして
規約を記憶する。
CLAUDE.md
1 - コミットメッセージは必ず【日本語】で記述
2 - プレフィックスには Conventional Commits 形式を使用
3 - 「なぜその変更が必要だったか」の背景理由を1行含める
『コミットして』の
一言だけで、
勝手にルール通りの
メッセージになるんだ！
Gemini Notebook


推論のドリフトでAIが暴走しないよう、CLAUDE.mdに『禁止』明記しておくことだ。
CLAUDE.md
・一括追加コマンド（git add . や git add -A）の使用は厳禁
・ステージングを行う前に、必ず対象ファイル一覧をユーザーに提示し許可を得ること
AIを使っても、Git履歴に対する説明責任（Diligence）を持つのは開発者である僕自身ですからね。
前役者
プロタ人者、26歳
Gemini Notebook


これでもう、
Gitコミットは
怖くありません！
本来のコーディングに
100%集中できます！
TURN!
Enter
Claudeと一緒に、新しいGit体験の一歩を踏み出そう！
🚀 今すぐ試せる最初のステップ：
ターミナルで claude と起動し、以下のプロンプトを入力してみてください：
現在の git diff を確認して、Conventional Commitsの形式で日本語のコミットメッセージ案を出してください。
Gemini Notebook