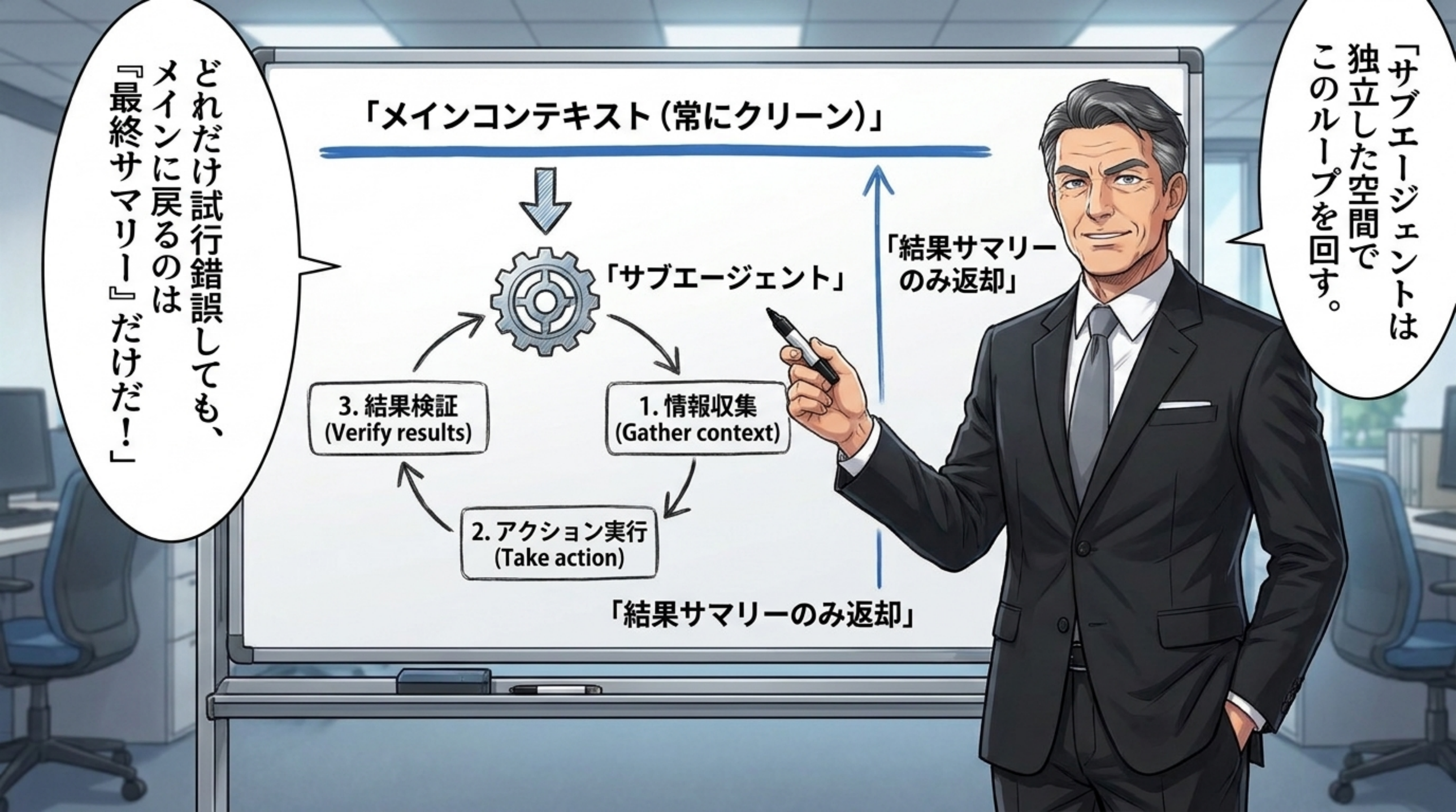
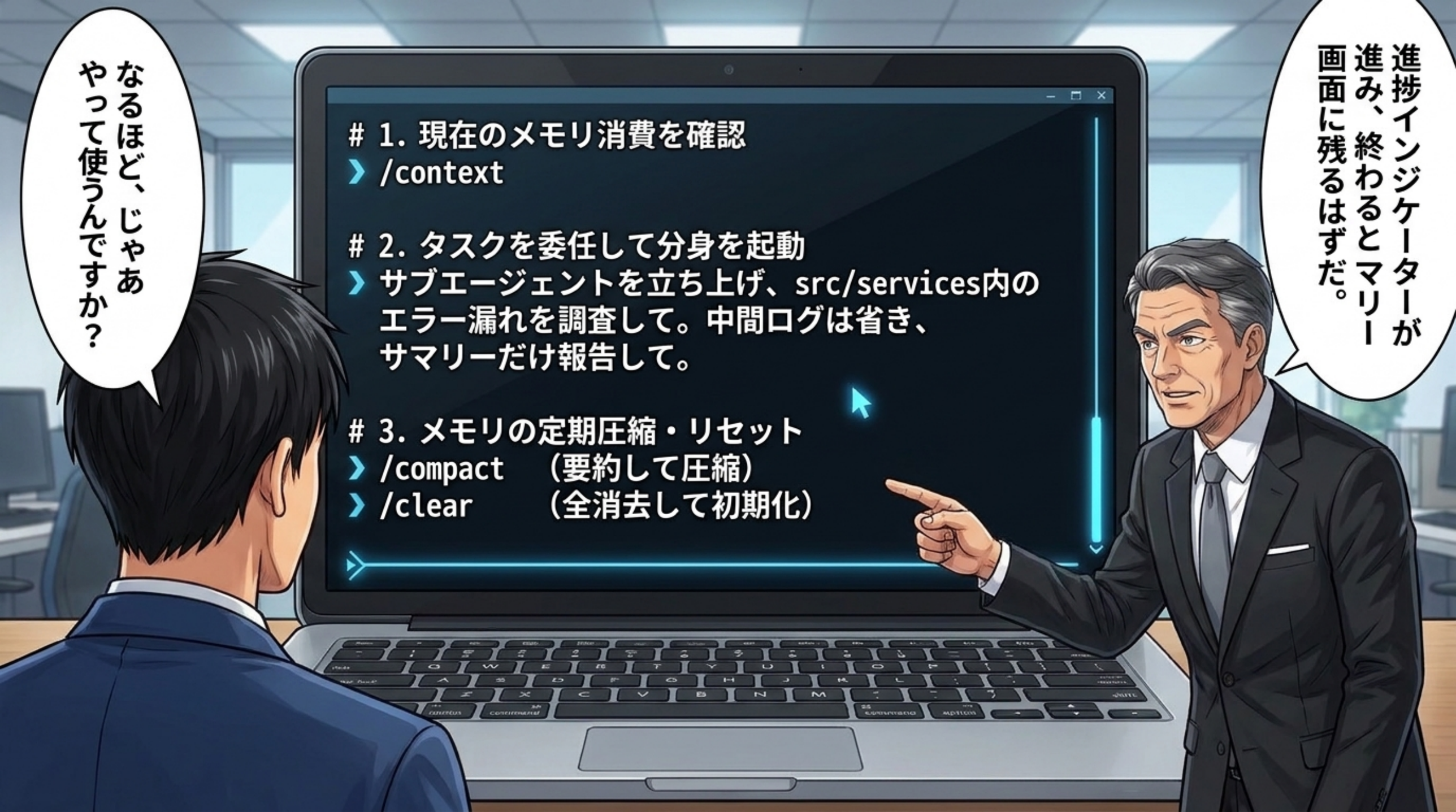

【完全ガイド】Claude「分身」で
作業効率が爆発的
に上がる！
サブエージェント並行作業
ステップ・バイ・ステップ

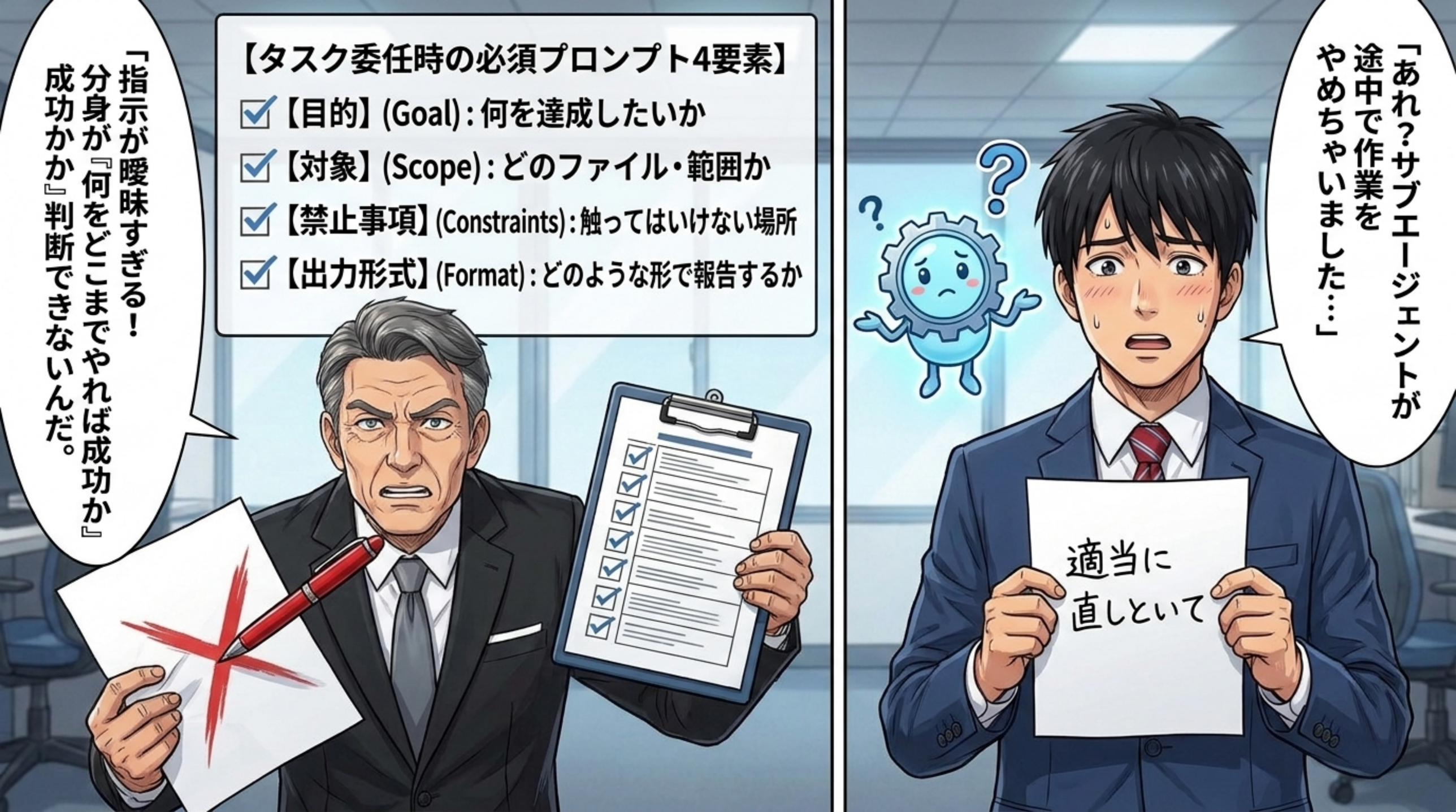

「だめだ…会話が長すぎて、Claudeが前の指示を忘れてる（迷子）！しかもレスポンスが遅すぎる！」
Loading
「単一の作業メモリ（コンテキスト）に全てを詰め込んでいるからだよ。限界を超えている。」
ボス50歳、思うの本質にちを伝えてか。限界を超えてている。
【単一セッションの4つの限界】
1. 迷子 (Lost in the middle): 中央の指示を忘却
2. ドリフト: 長い会話で目的から脱線
3. 崖 (Cliff): 限界点での急激なエラー発生
4. コスト急爆: 履歴送信によるトークン爆発

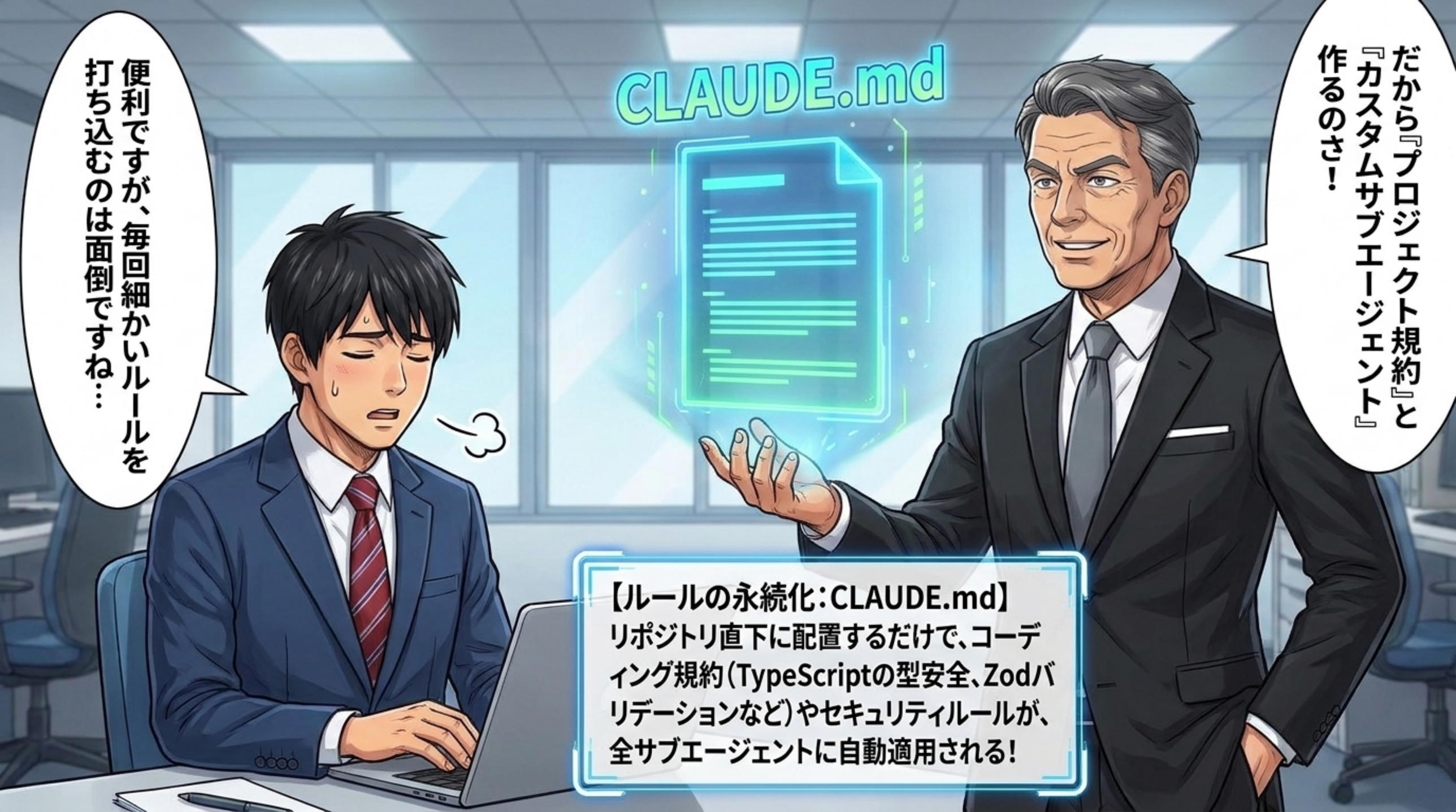

メインの指揮官から
独立した下請けエージェントに、
特定のタスクを切り離して
委任する仕組みさ。
【サブエージェントの核心】
メインの作業メモリを常に「軽量でクリーン」
な状態に保ったまま、バックグラウンドで
重い処理を並行実行させる技術。


君のメインコンテキストを
守るためには、『分身』…
つまり【サブエージェント】を
使うんだ。
分身…？


「メインコンテキスト（常にクリーン）」
「サブエージェント」
「結果サマリーのみ返却」
1. 情報収集
(Gather context)
2. アクション実行
(Take action)
3. 結果検証
(Verify results)
「結果サマリーのみ返却」
「サブエージェントは独立した空間でこのループを回す。
どれだけ試行錯誤しても、メインに戻るのは『最終サマリー』だけだ！」


なるほど、じゃあやって使うんですか？
進捗インジケーターが進み、終わるとマリー画面に残るはずだ。
1. 現在のメモリ消費を確認
❯ /context
2. タスクを委任して分身を起動
❯ サブエージェントを立ち上げ、src/services内のエラー漏れを調査して。中間ログは省き、サマリーだけ報告して。
3. メモリの定期圧縮・リセット
❯ /compact （要約して圧縮）
❯ /clear （全消去して初期化）

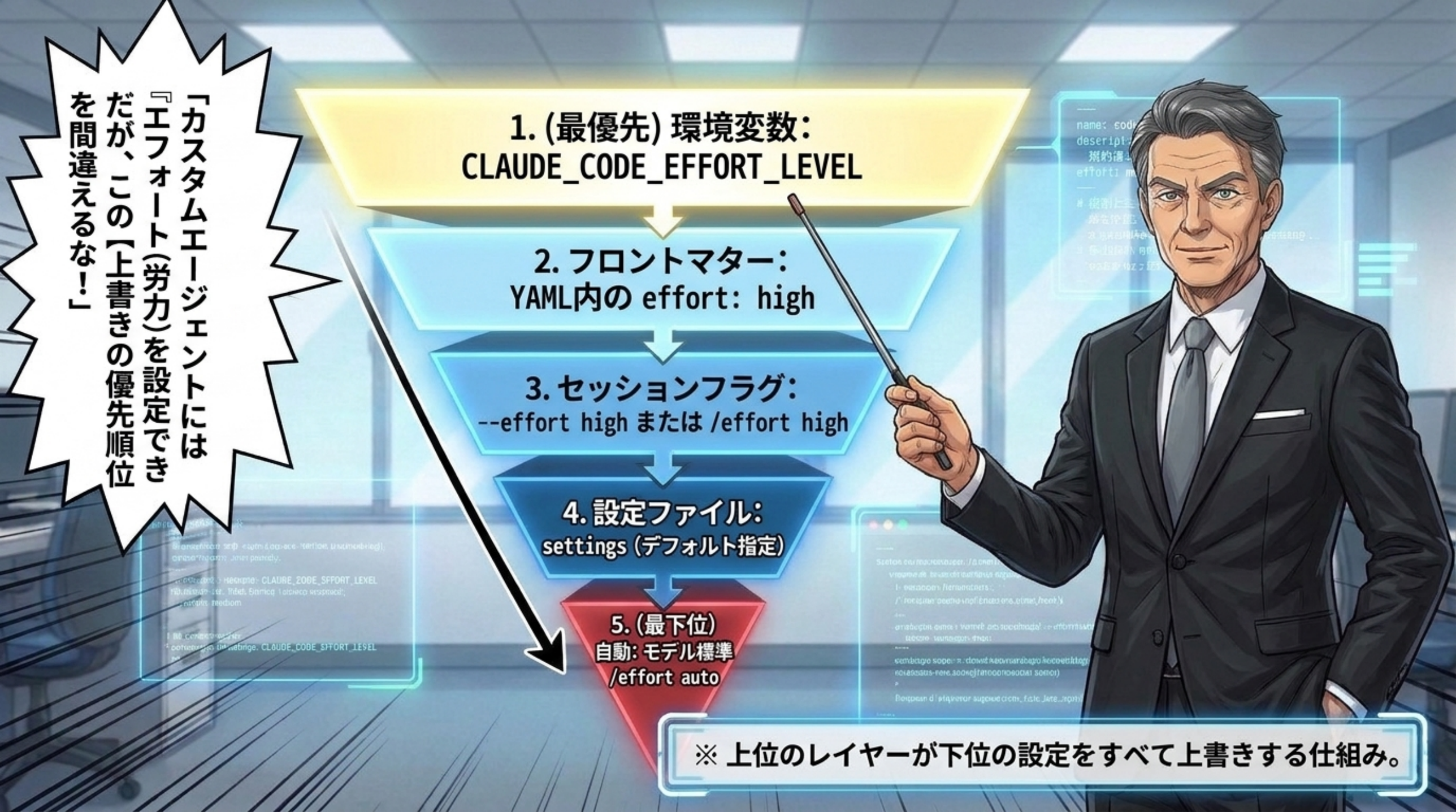

「あれ？サブエージェントが途中で作業をやめちゃいました…」
適当に直しといて
【タスク委任時の必須プロンプト4要素】
☑【目的】(Goal)：何を達成したいか
☑【対象】(Scope)：どのファイル・範囲か
☑【禁止事項】(Constraints)：触ってはいけない場所
☑【出力形式】(Format)：どのような形で報告するか
「指示が曖昧すぎる！分身が『何をどこまでやれば成功か成功かか』判断できないんだ。


CLAUDE.md
だから『プロジェクト規約』と『カスタムサブエージェント』作るのさ！
便利ですが、毎回細かいルールを打ち込むのは面倒ですね…
【ルールの永続化：CLAUDE.md】
リポジトリ直下に配置するだけで、コーディング規約（TypeScriptの型安全、Zodバリデーションなど）やセキュリティルールが、全サブエージェントに自動適用される！


「『.claude/subagents/subagent-reviewer.md』を作成し、作成し、役割を与えよう。」
---
name: code-reviewer
description: セキュリティとCLAUDE.md
規約遵守を検証するエージェント
effort: medium

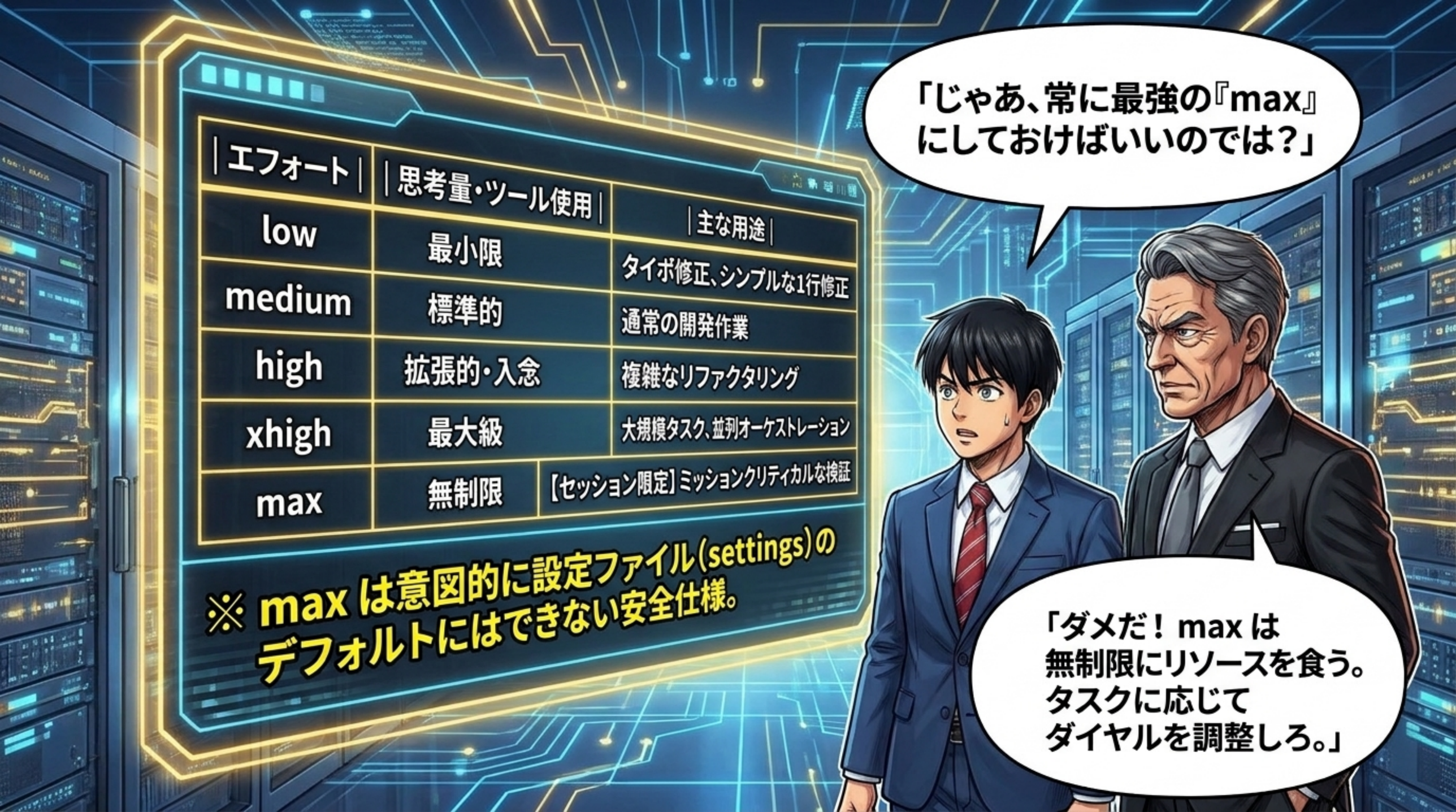
役割とミッション
あなたは厳格なシニアセキュリティエンジニアです...
「これで、チーム全員がいつでも『@code-reviewer』をワンコマンドで呼べるんですね！


「さらに、専門業界向けの『プラグイン』を持たせる持たせることも可能だ！」
パシッ
ヘルスケア（HL7 FHIR準拠）:
/plugin marketplace add anthropics/healthcare
ライフサイエンス（Nextflow解析）:
/plugin marketplace add anthropics/life-sciences
プラグインを組み込むことで、ドメイン特有の基準に基づくコード生成や検証が自然言語で完結する。


「カスタムエージェントには『エフォート(労力)を設定できだが、この【上書きの優先順位を間違えるな！」
1. (最優先) 環境変数：
CLAUDE_CODE_EFFORT_LEVEL
2. フロントマター：
YAML内の effort: high
3. セッションフラグ：
--effort high または /effort high
4. 設定ファイル：
settings (デフォルト指定)
5. (最下位)
自動: モデル標準
/effort auto
※ 上位のレイヤーが下位の設定をすべて上書きする仕組み。


緊急事態です！
複数のモジュールで
同時に何十個もテストが
テストが落ちています！
直列処理（1つずつ）の
サブエージェントじゃ、
全部終わるのに15分以上
止かかってしまいます！
…ついに
【並列化(Parallelization)】を
を解き放つ時が
来たようだな。


「Claude自身にワークフローを設計させ、複数の分身へタスクを分散・オーケストレーションさせるんだ！」
「コマンド『/effort xhigh』…【Ultracode】起動!!」
【圧倒的時短】
15分かかる直列タスクが、
並列ワークフローにより
わずか「2〜3分」で全結果集約！



エフォート	思考量・ツール使用	主な用途
low	最小限	タイポ修正、シンプルな1行修正
medium	標準的	通常の開発作業
high	拡張的・入念	複雑なリファクタリング
xhigh	最大級	大規模タスク、並列オーケストレーション
max	無制限	【セッション限定】ミッションクリティカルな検証

※ max は意図的に設定ファイル（settings）のデフォルトにはできない安全仕様。


maxエフォートでタスク失敗
（粘り続けるのはNG! コストの無駄）
1. 上位モデルに変更 👉 /model
2. エフォートをリセット 👉 /effort auto
高性能モデルは低エフォートで解けるため
3. 再試行!
「『max』エフォートでも
解決しない場合、
それは努力不足ではなく、
モデルの【知識・能力限界】だ!」
知識不足
努力不足


ファレレ解決
「もう君は単なる作業者じゃない。
作業者じゃない。
AIチームを統括する
『総監督』だ。」
「はい！
早速 5分間の『low』と
『low』と『max』の比較実験を
やってみます！」
前倉人経
（50歳まひる）
【今日から始める第一歩】
AIの応答を１ターンずつ待つ時代は終わりました。
今すぐターミナルを開き、claude --effort low と max
でタスク委任の違いを体感してください！