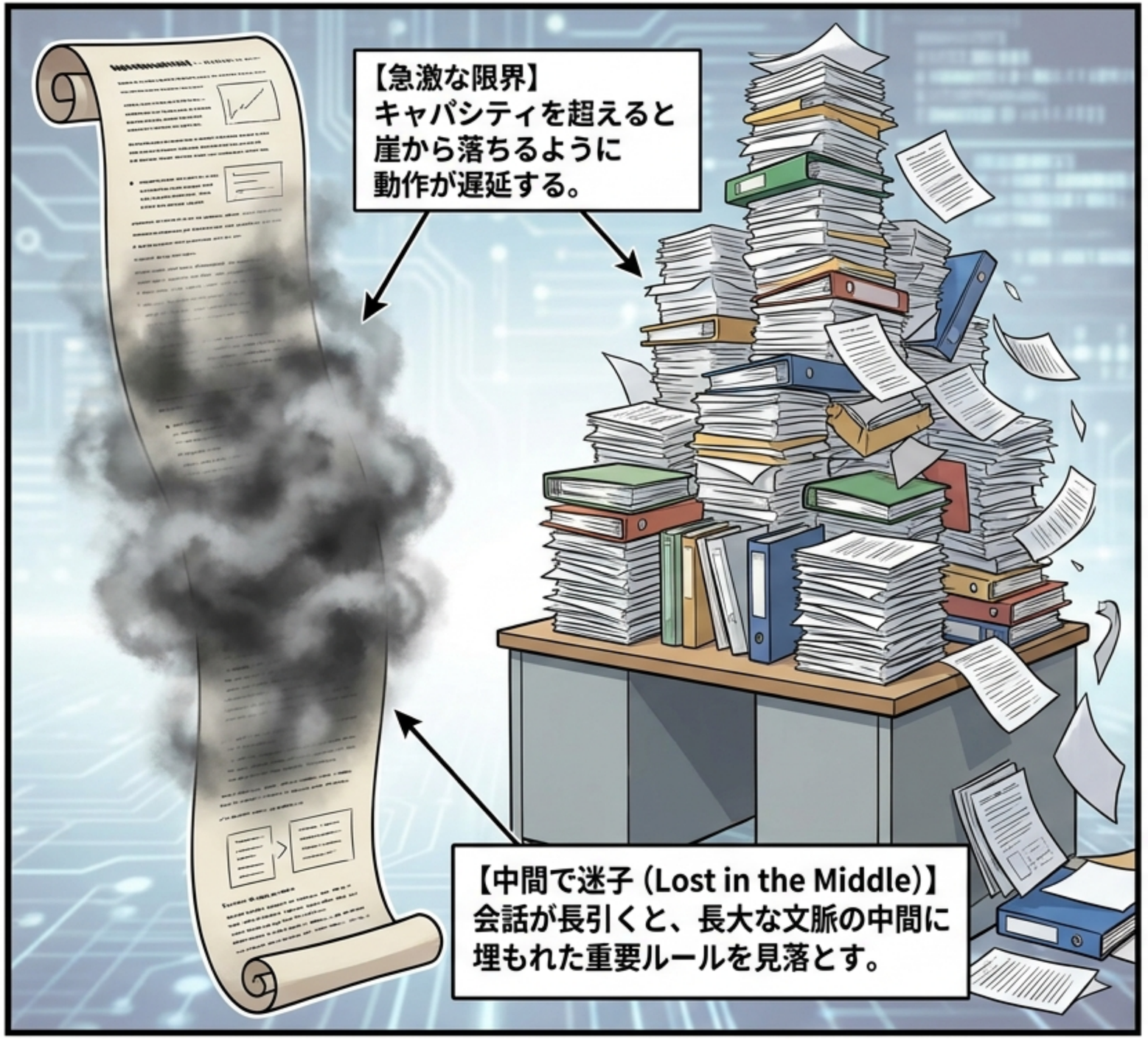
【完全ガイド】Claudeサブエージェントで爆速並行作業！



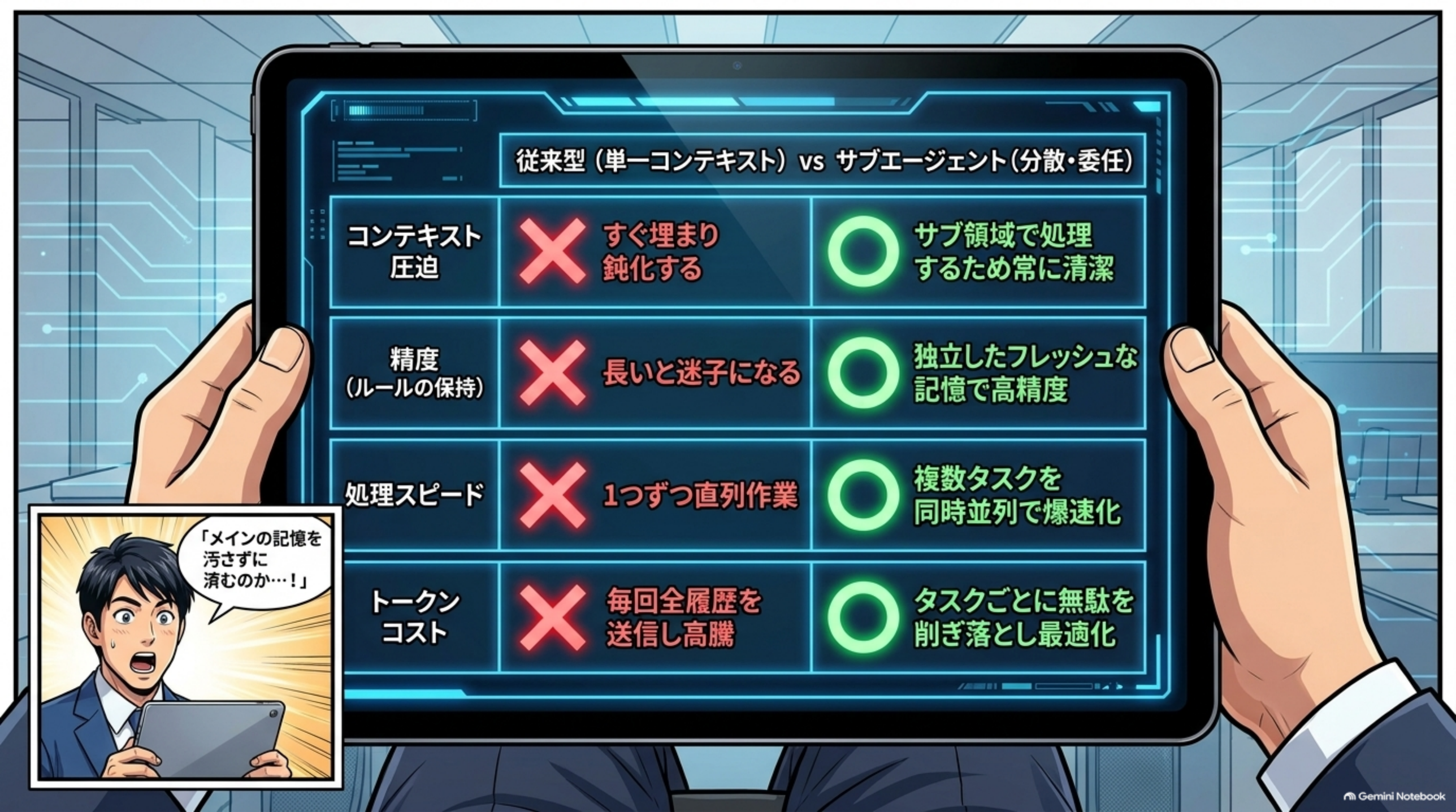
Gemini Notebook



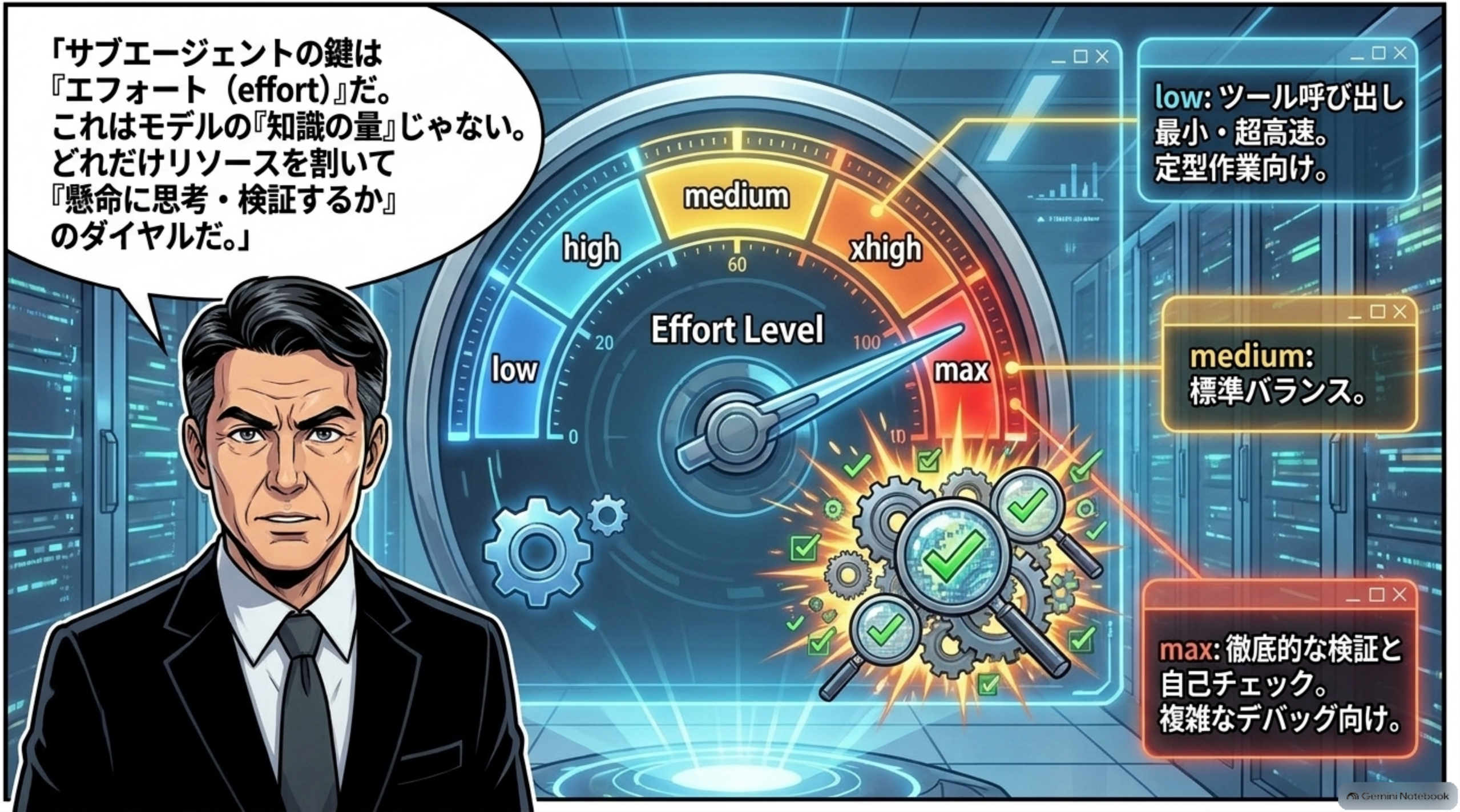


Gemini Notebook


「解決策は
『サブエージェント』への
へのタスク委任だ！」
中間プロセスは
個室（サブ）で完結！
メインデスクには
「要約結果」だけ
を持ち帰る！
＝ メインコンテキストを
常にクリーンに保つ！


従来型（単一コンテキスト） vs サブエージェント（分散・委任）
コンテキスト圧迫
すぐ埋まり鈍化する
サブ領域で処理するため常に清潔
精度（ルールの保持）
長いと迷子になる
独立したフレッシュな記憶で高精度
処理スピード
1つずつ直列作業
複数タスクを同時並列で爆速化
トークンコスト
毎回全履歴を送信し高騰
タスクごとに無駄を削ぎ落とし最適化
「メインの記憶を汚さずに済むのか…！」
Gemini Notebook


「うわっ、70%超え!?
重いわけだ…」
Context Usage:
142,800 / 200,000 tokens
(71.4%)
「まずは今の使用量を
確認し、掃除しろ。」
/context
CLICK!
Enter
「よし、`/compact`で
中間ログだけを圧縮！
まっさらにしたい時は
`/clear`ですね！」


「サブエージェントの鍵は
『エフォート（effort）』だ。
これはモデルの『知識の量』じゃない。
どれだけリソースを割いて
『懸命に思考・検証するか』
のダイヤルだ。」
low
high
medium
xhigh
max
0
20
60
100
Effort Level
low: ツール呼び出し
最小・超高速。
定型作業向け。
medium:
標準バランス。
max: 徹底的な検証と
自己チェック。
複雑なデバッグ向け。

.claude/subagents/doc-writer.md

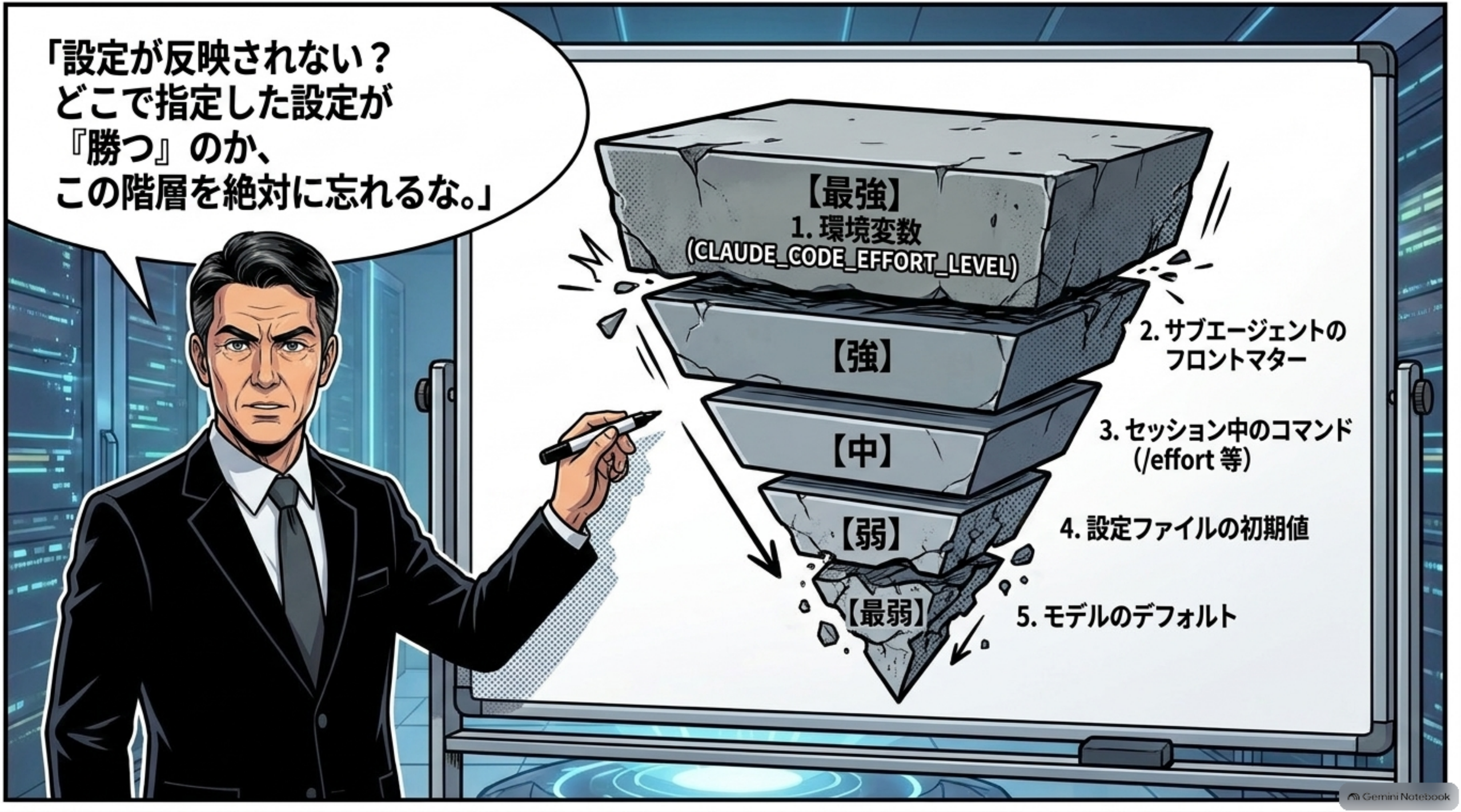
.claude > subagents > doc-writer.md

```
1  ---
2  name: doc-writer
3  effort: low
4  ---
5
6  # 役割
7  あなたはPythonコードのドキュメント作成
   専門エージェントです。
8  思考プロセスを省き、迅速に出力してくださ
   い。
```

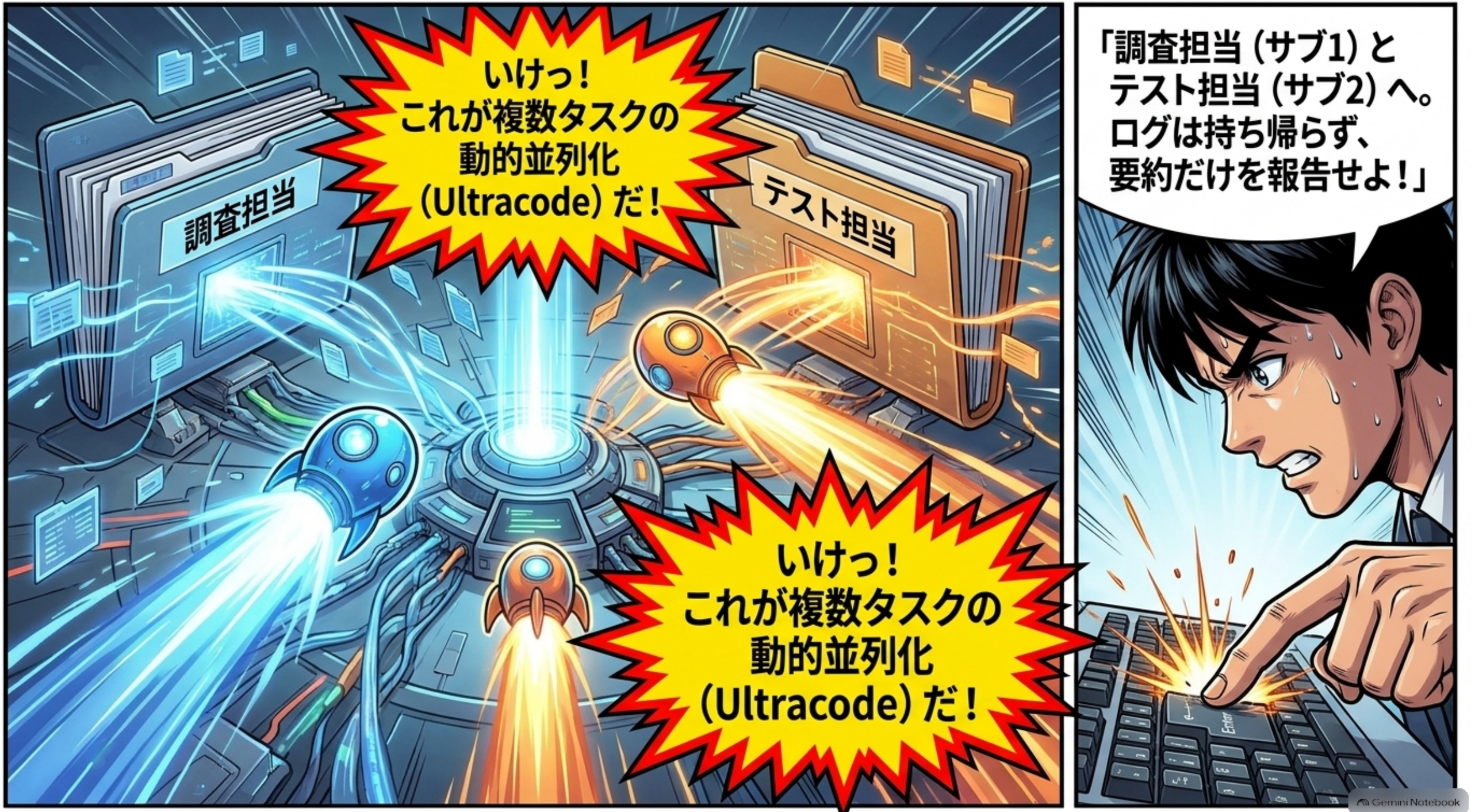
「フロントマターで指定すれば、
呼び出された瞬間だけ
この低コスト設定で動く！」



Gemini Notebook

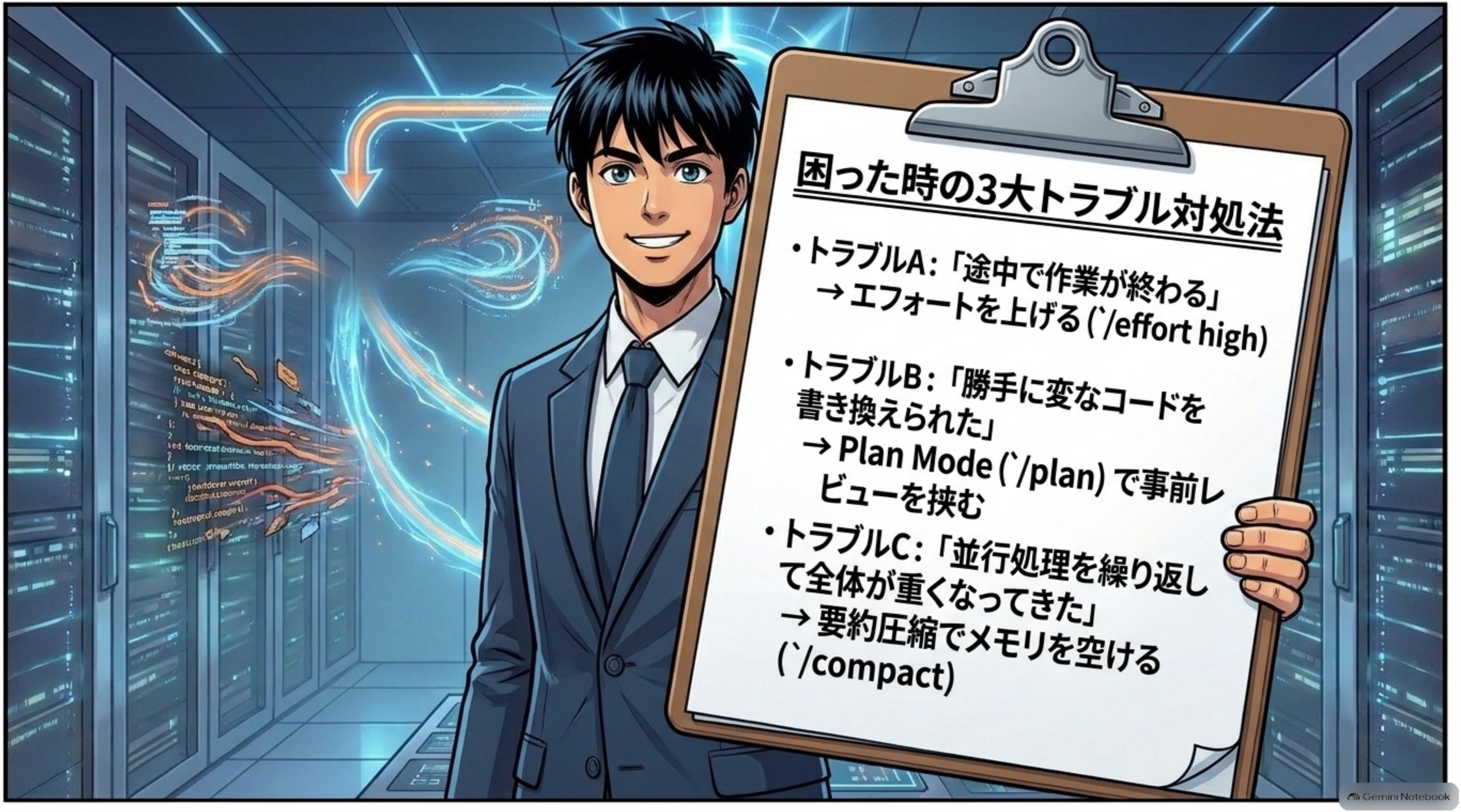

「設定が反映されない？
どこで指定した設定が
『勝つ』のか、
この階層を絶対に忘れるな。」
【最強】
1. 環境変数
(CLAUDE_CODE_EFFORT_LEVEL)
【強】
2. サブエージェントの
フロントマター
【中】
3. セッション中のコマンド
(/effort 等)
【弱】
4. 設定ファイルの初期値
【最弱】
5. モデルのデフォルト


「Current Effort: low になってる！」
$
Current Effort: low
Total Cost: $0.042
>
「あれ？　エージェントが作業を中途半端に投げ出して未完成で戻ってきました!」
「タスクの難易度に対してエフォートが低すぎる（low）のが原因だ。リソース上限で早期終了したんだ。`/effort high` に上げて再試行しろ!」

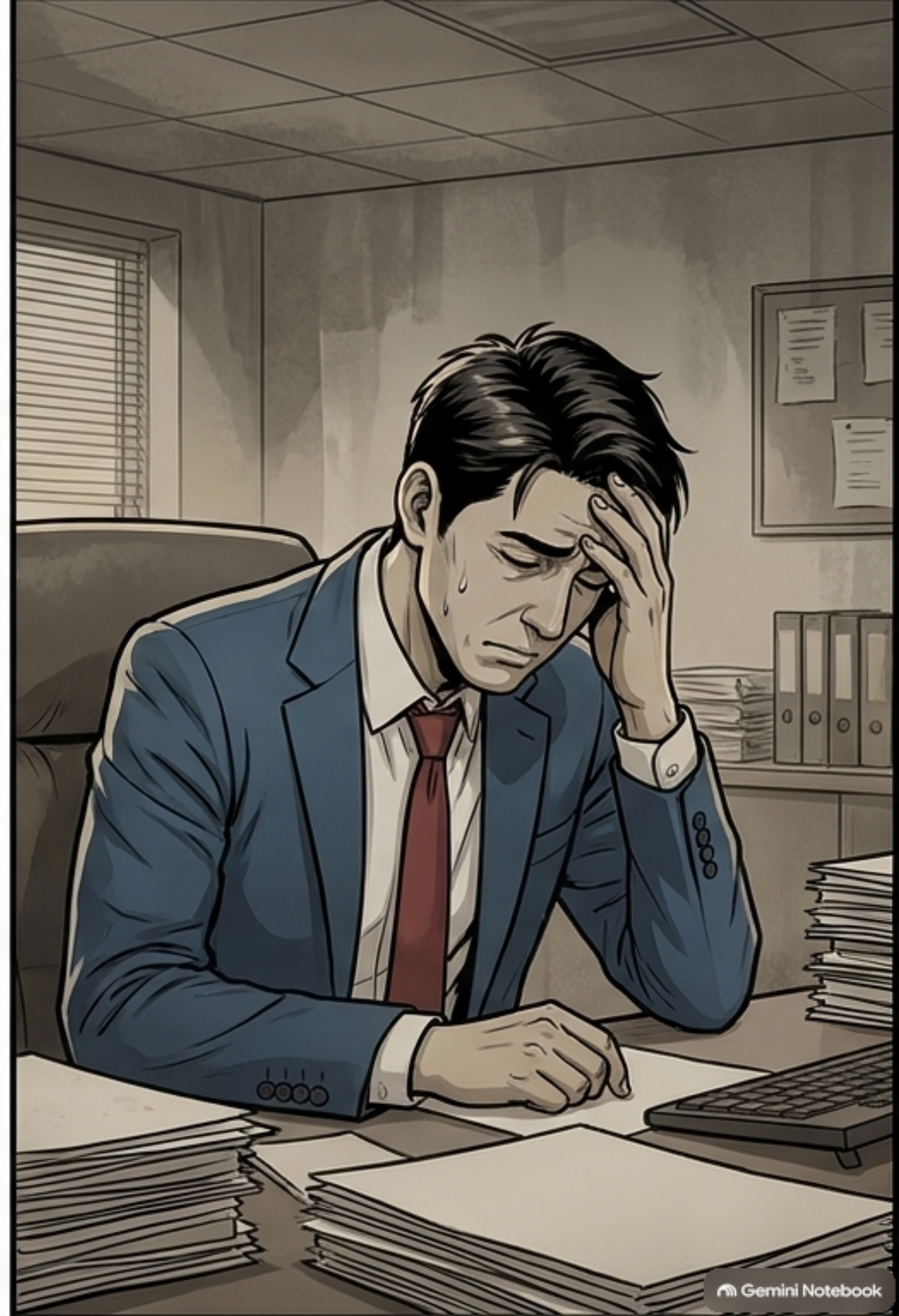

いけっ！
これが複数タスクの
動的並列化
（Ultracode）だ！
調査担当
テスト担当
いけっ！
これが複数タスクの
動的並列化
（Ultracode）だ！
「調査担当（サブ1）と
テスト担当（サブ2）へ。
ログは持ち帰らず、
要約だけを報告せよ！」
Enter

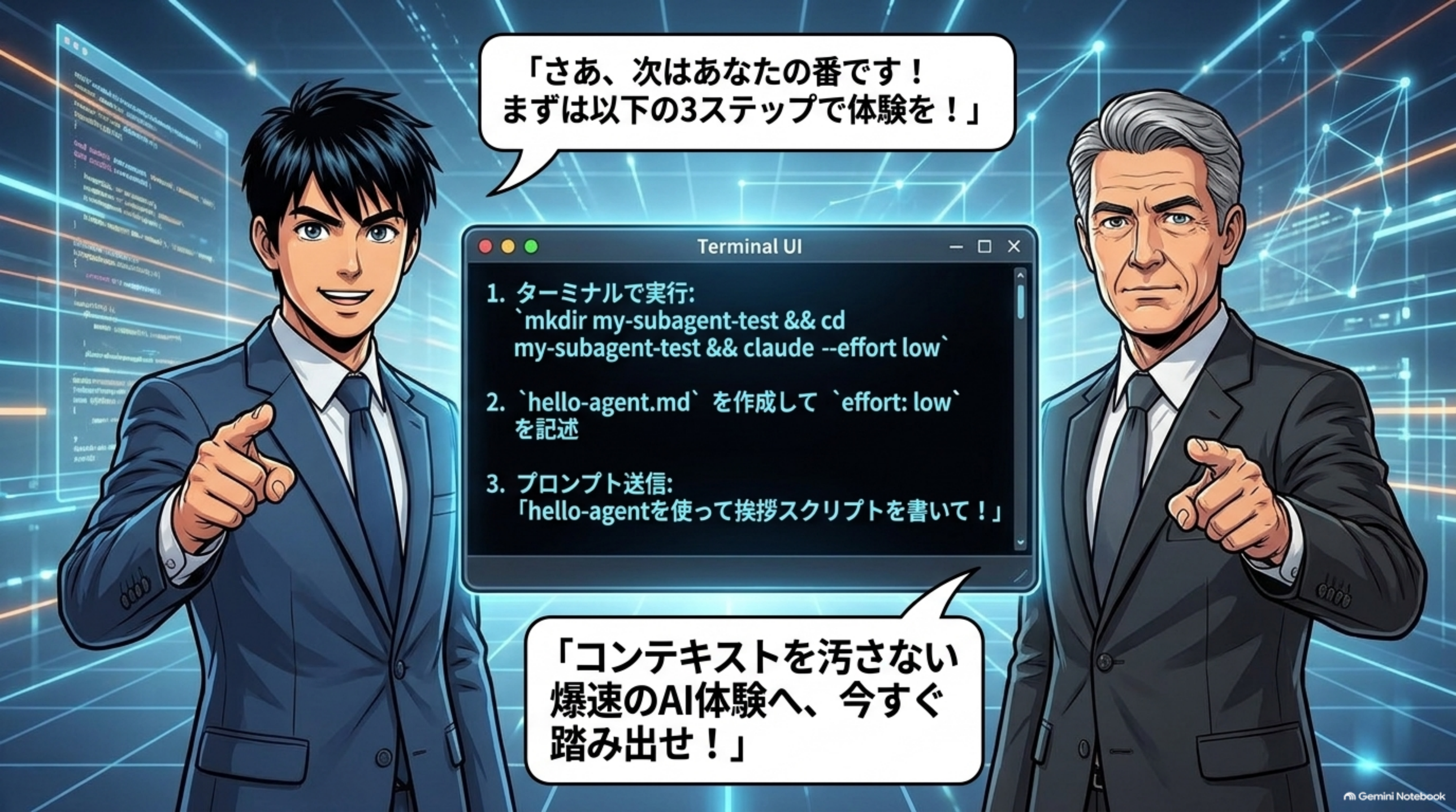

メインセッション
(クリーンなメモリ)
サブA(テスト実行)
Effort: high
膨大な中間ログ
サブB(ドキュメント作成)
Effort: low
高速整形
要約レポート
要約レポート
要約レポート
「それぞれの専門タスクを
最適なコスト(努力量)で同時実行させ、
メインメモリは100%清潔に保つ…
これがすべての知識が繋がった
究極のワークフロー!」


「自動化の暴走は防げ。
人間が手綱を握るんだ。」
【マニュアル】
実行のたびに[y/n]で
人間が承認。
最も安全で確実。
y
n
【Plan Mode / `/plan`】
コード変更前に『修正計画
(Plan)』を提示させレビューする。
脱線防止に極めて有効!
【自動承認 / auto-accept】
自動承認で爆速だが、
予期せぬ書き換えリスクあり。
要注意。
Gemini Notebook


困った時の3大トラブル対処法
・トラブルA：「途中で作業が終わる」
→ エフォートを上げる（`/effort high）
・トラブルB：「勝手に変なコードを書き換えられた」
→ Plan Mode（`/plan）で事前レビューを挟む
・トラブルC：「並行処理を繰り返して全体が重くなってきた」
→ 要約圧縮でメモリを空ける（`/compact）


「複数の作業を同時に終わらせても、コンテキストは常にクリーン！生産性が全く違います！」
「もうAIの機嫌に振り回されることはない。お前が指揮者（オーケストレーター）だ。」


Gemini Notebook


「さあ、次はあなたの番です！
まずは以下の3ステップで体験を！」
Terminal UI
1. ターミナルで実行:
`mkdir my-subagent-test && cd my-subagent-test && claude --effort low`
2. `hello-agent.md` を作成して `effort: low` を記述
3. プロンプト送信:
「hello-agentを使って挨拶スクリプトを書いて！」
「コンテキストを汚さない爆速のAI体験へ、今すぐ踏み出せ！」
Gemini Notebook