Claude × Codex コスト・性能管理チートシート

タスクに合うモデルとエフォート、それを仕込む場所、判断だけで済むところで LLM を呼ばない方法 ／ 2026-09 時点

Givery

タスク別のモデルとエフォート

タスク	Claude（モデル・effort）	Codex（モデル・effort）	決め手
はい・いいえや N 択の判定	ollaya の decider で生成なし。LLM なら Haiku 4.5	Luna low	選択肢で済む答えに文章は要らない
置換・整形・リネーム	Haiku 4.5（effort の設定なし）	Luna low	正解が1つで、速さが効く
探索・下読み・ログの抜き出し	Haiku 4.5 か Sonnet 5 low をサブエージェントで	Luna medium	読む量は多く、判断は浅い
要約・抽出・議事録	Sonnet 5 medium	Luna high（公式の開始点）	形が決まっている
日常のコーディング・バグ修正	Sonnet 5 high（既定）	Sol medium（既定）	量をこなす主力
長いエージェント作業・複数ファイルの実装	Opus 5.5 high〜xhigh	Sol high〜xhigh	Opus 5.5 は既定 medium。明示して上げる
設計・計画・レビューの方針決め	`opusplan`（計画は Opus、実行は Sonnet）	Sol と `plan_mode_reasoning_effort`	考えるところだけ上位にする
原因不明の不具合・難問	Fable 5.1 xhigh〜max	Astra high〜max	外した時の手戻りがいちばん高い
分けて並べられる大仕事	サブエージェント並列、`/effort ultracode`	Sol・Astra の ultra	早く終わるが、消費は人数分ふえる
顧客や受講者が読む日本語	Opus 5.5 high	Sol high	下位モデルは言い回しが崩れやすい
急ぎの対話デバッグ	Opus 5.5 と `/fast`（最大2.5倍速、$8 / $40）	`service_tier = "fast"`（API は4倍単価）	速さをお金で買う。バッチには使わない

料金（1M トークンあたり）

	Fable 5.1	Opus 5.5	Sonnet 5	Haiku 4.5	GPT-6 Astra	GPT-6 Sol	GPT-6 Luna	Jev
入力 / 出力	$10 / $50	$4 / $20	$2 / $10	$1 / $5	$10 / $50	$2 / $10	$0.10 / $0.50	$0.042 / 無料
既定 effort	high（思考は常にオン）	medium	high	なし（200K）	low	medium	medium	判定だけの商用 API

モデルとエフォートの置き場

`/model` ・ `/effort` そのセッション。Enter で既定に保存、s でこの回だけ。Desktop は Cmd+Shift+I・E。1ターンだけなら `ultrathink`

スキル（SKILL.md） フロントマターの `model: haiku` ・ `effort: low` がそのスキルの実行中だけ効く。`context: fork` で別の文脈に

サブエージェント `.claude/agents/*.md` に `model` ・ `effort` ・ `maxTurns`。調べ役や確かめ役を安く回す

`CLAUDE_CODE_SUBAGENT_MODEL` model 未指定のサブエージェントやチームメイトを一括で。本体は Opus、手足は Sonnet

`modelSettings` settings.json でモデルごとに既定の effort を持たせる。Opus 5.5 の medium を底上げ

`opusplan` プランモードは Opus、実行は Sonnet。計画だけ高いモデルに払う

Agent Teams `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1` で有効。消費は約7倍でチームメイトは Sonnet 推奨。Desktop では使えない

`availableModels` 管理設定で選べるモデルを絞る。研修 PC で上位モデルを封じる

Codex の config.toml `model` と `model_reasoning_effort` で普段の基準。計画だけ `plan_mode_reasoning_effort`。GPT-5.5 は 10/14 に退役

Codex のプロファイル `~/.codex/ci.config.toml` を置き `codex --profile ci`。夜間や CI は Luna に落とす

Codex の `[agents]` 生成するサブエージェントの既定モデルと effort。`max_concurrent_threads_per_session` で同時数に上限

仕込み例

`~/.claude/settings.json`（抜粋）

```
{
  "model": "sonnet",
  "modelSettings": {
    "claude-opus-5-5": { "effortLevel": "high" }
  },
  "fastModePerSessionOptIn": true,
  "env": { "CLAUDE_CODE_SUBAGENT_MODEL": "sonnet" }
}
```

`~/.codex/config.toml`

```
model = "gpt-6-sol"
model_reasoning_effort = "medium"
plan_mode_reasoning_effort = "high"
[agents]
default_subagent_model = "gpt-6-luna"
default_subagent_reasoning_effort = "low"
```

夜間バッチ（設定を読ませない）

```
claude -p "$PROMPT" --model sonnet \
  --setting-sources "" --strict-mcp-config \
  --tools "" --disable-slash-commands \
  --no-session-persistence \
  --output-format json --json-schema "$SCHEMA" \
  --system-prompt "$SYS"
```

判断だけ呼ぶ仕組み

ollaya 判定モデルをローカルで動かす実行環境（Apache-2.0）。choice・score・noul を確率で返し、文章は作らない。トークンも API 代もかからない

`claude mcp add -s user ollaya -- ollaya mcp`

laya Convai Innovations の判定モデル（421M）。この Mac で約0.07秒と速いが、日本語は読めず見積もり依頼を other と誤判定

decider Mapika の判定モデル（Qwen3.5 系）。日本語はこちら。この Mac で約4秒、同じメールを確率 1.00 で当てた

Jev TypeSafe AI の商用 API（9月発表）で OSS ではない。ollaya は同じ形式で答えるので呼び先を差し替えられる

AnyJev Nokia の OSS。手持ちのオープン LLM を学習なしで判定器にする

`pip install "anyjev[hf]"`

閾値の目安 choice は confidence 0.6 以上で実行。noul は 0.8 以上を yes、0.2 以下を no。その間は人か LLM に回す

任せないもの 「本日中」のような期限の急ぎ度は decider でも外した。日付は正規表現で拾う

コストと速度を下げる設定と習慣

`/clear` と `/compact` 別の作業に移るたびに `/clear`（ただ）。続けるなら `/compact` 残す観点 。要約は大きな文脈を読むぶん高い

CLAUDE.md は200行以内 手順はスキルへ。スキルの本文は呼んだ時だけ読まれる

MCP `/context` で見て、使わないサーバーは `/mcp` で切る

フックで前処理 テスト出力を FAIL 行だけにする（公式の例）。数万トークンが数百になる

キャッシュ サブスクは1時間、使用クレジット中は5分。休憩明けは全部読み直す

fast mode 途中で ON にすると、それまでの文脈を fast 単価で読み直す。使うなら最初から

`/usage` 5時間枠・週枠と、スキル・サブエージェント・MCP 別の消費%

`--max-budget-usd` `-p` で使うドルの上限。サブエージェントの分も数える

`--bare` 速く起動するがキーチェーンを読まない。サブスク認証だと api_error になった（この Mac）

Tips・注意

結果が浅いと感じたら、上位モデルに替える前に effort を1段上げてみる。

設定を読ませない `claude -p` で、1件あたり約8万トークンが約1.2万トークンに減った（この Mac の夜間採点で実測）。

Sources（公式・一次情報 / クリックで開く）

1. Model configuration
2. Subagents
3. Skills
4. Manage costs
5. Fast mode
6. CLI reference
7. Claude の料金
8. Codex のモデル
9. Codex config reference
10. OpenAI API の料金
11. ollaya
12. Laya（Hugging Face）
13. Jev の発表（MarkTechPost）
14. AnyJev