ORIENTATION / 座学編

# オリエンテーション 座学編
# ハーネスの有無で結果がどれだけ変わるかを測る一日

午前の座学で扱う内容をまとめた読み物です。生成AIの現在地とClaudeファミリーの使い分けを押さえたうえで、VSCode の Claude Code 拡張の画面と設定を一通り見ます。最後に、CLAUDE.md・docs・Skill・Subagent・Hook・rules が「どの層で何をするのか」を1枚の地図にして、午後の演習3本につなぎます。演習の手順はハンズオンガイドにあります。

2026年8月17日(月) 9:00-18:00・オンライン(Zoom)

主演習3本（Lv1 / Lv2 / Lv3）＋ミニ演習7本

GitHub Copilot 経験あり・Claude Code 未経験の開発エンジニア向け

## この読み物の構成

5-6章 **Claude Code の基礎**

拡張の画面・権限モード・ネットワーク要件と、モデル・文脈・コマンド。

7章 **ハーネス3段階の地図**

Lv1 / Lv2 / Lv3 の定義と、午後の演習3本との対応。

8章 **出典一覧**

本文で挙げたURLをまとめて置いています。

ORIENTATION 00

# この資料の読み方

本文の数値は 2026年7月30日に一次情報を開いて確認したものです。研修当日は8月17日で、3週間近い差があります。数値を使う場面では、この差を前提にした読み方をしてください。

## 00.1 時点情報の扱い

- 数値には「2026年7月30日時点」のように確認時点を添えています。時点の書いていない数値はこの資料には載せていません
- モデルの版数（Opus 5、Sonnet 5 などの番号）を書いているのは3章と4章だけです。そこでも時点表記を必ず付けています
- 5章・6章の操作説明と、ハンズオンガイド・事前セットアップガイドには版数を書いていません。当日パネルの `/model` を開いて実機の一覧を見ます
- 一次情報で裏が取れていない項目は「未確認」と書き、断定していません。当日の実機確認で確定させます

> **週次で動くものを手順書に焼き込まない**
>
> Claude Code は週ごとに機能が増えます。公式の週次ダイジェストで見ると、Week 29（2026年7月13〜17日）でスクリーンリーダーモードと /fork、Week 28（7月6〜10日）でデスクトップアプリの内蔵ブラウザと /doctor が入りました。手順書をバージョン番号や画面の細部に縛ると、数週間で書き直しになります。この資料は「どこを見れば分かるか」を書く方針で、画面の並び順に依存する説明を避けています。

## 00.2 出典の見方

各章の末尾に SOURCES の欄があります。本文の数値と主張は、そこに並ぶURLで確認できます。ベンチマークの数値については、次の2点を本文でも明記しています。

| 確認すること | この資料での書き方 |
| --- | --- |
| **誰のハーネスで測った値か** | 共通ハーネスの公式リーダーボード値と、ベンダーが自社ハーネスで測って発表した値を、同じ表に混ぜていません |
| **計測方法が公開されているか** | ベンダー発表の顧客事例は、計測方法が非開示であることを本文に書いています |

操作の説明は、Windows と Mac を必ず併記しています。キー操作（`Ctrl+Shift+X` / `Cmd+Shift+X`）、パス（`%USERPROFILE%\.claude\` / `~/.claude/`）、Python の呼び出し名（`py` と `python3`）の3つが対象です。

## 00.3 理解度チェック

### Q0

**この資料に載っている数値を、当日そのまま断定して話してよいかを判断する基準はどれですか。**

- **A** 出典URLが付いていれば、確認時点は気にしなくてよい
- **B** 公式ドキュメントが出所であれば、確認時点は不要である
- **C** 出典URLと確認時点の両方が付いていて、研修前週の再確認が済んでいる
- **D** 3か月以内に確認された数値であれば、再確認は不要である

**正解と解説を開く**

正解は C です。

本文の数値は 2026年7月30日に確認したものです。研修当日は8月17日で、その間に新しいモデルの公開やリーダーボードの更新が入ります。出典と時点の両方が揃っていて、研修前

週（8月10〜14日）の再確認を通ったものだけを断定して扱います。それ以外は「7月30日時点ではこうでした」と時点を添えて話します。

ORIENTATION 01

# 本日のゴールと到達点

この研修は Claude Code の操作説明で終わりません。ハーネス（Claude にあらかじめ前提と手順を渡しておく仕組みのこと。CLAUDE.md・docs・Skill・Hook などをまとめてこう呼びます）が有る環境と無い環境で、同じ指示の結果がどう変わるかを自分の手で測るところまで行きます。

## 01.1 到達点4つ

到達点 01

### 往復を自分で回せる

VSCode の Claude Code 拡張で、指示 → 生成 → 差分の確認 → 修正 の往復を自分で回せる状態になります。

到達点 02

### 差を測った経験を持つ

ハーネスが有る環境と無い環境の差を、自分の手で測った経験を持ちます。演習1で有る側、演習2で無い側を体験します。

到達点 03

### ハーネスを自分で足せる

CLAUDE.md を2箇所に書く、コーディング規約を置く、Hook を自然言語から作らせる、成功した手順を Skill にする、公式Skill に提案させて導入する。この5つを1回ずつ通します。

到達点 04

### 自社で始められる

自社のリポジトリで同じことを始められる状態になります。持ち帰るのは手順書ではなく、ハーネスを育てる順序です。

到達したかどうかは、各演習の README にある OK基準で判定します。演習の途中で理解度チェックを挟み、思い違いをその場で拾います。

## 01.2 一日の分数の内訳

進行は時刻ではなく分数で管理します。講義・演習・準備の合計は420分で、そこに休憩と昼休憩と予備を足すと540分（9時間）です。

| 区分 | 中身 | 分数 |
| --- | --- | --- |
| 講義 | 講師紹介、イントロ、座学4章、振り返り | [136min] |
| 準備 | 環境立ち上げ、公式Skill のコピー、個人設定のバックアップ、フォルダの開き直し4回 | [38min] |
| ミニ演習 | M1 から M7 の7本 | [106min] |
| 主演習 | 演習1 Lv1、演習2 Lv2（前半・後半）、演習3 Lv3 | [140min] |
| 小計 | ここまでで420分 | [420min] |
| 休憩ほか | 定着・休憩 [10min] × 4回、昼休憩 [60min]、予備 [20min] | [120min] |

座学は午前に集中します。午後は演習とミニ演習が交互に入り、最後に振り返りと個人設定の後始末をします。休憩で区切った連続稼働の最長は86分です。

## 01.3 演習ごとにフォルダを開き直す運用

本日は、演習を切り替えるたびに VSCode の「ファイル」から「フォルダーを開く」でその演習フォルダを開き直します。配布フォルダの一番上を開いたままにはしません。

> **開き直す理由**
>
> 演習2は「プロジェクト側のハーネスが無い状態」を体験する回です。配布フォルダのルートを開いたままだと、ルートに置いた設定が効いてしまい「無い」と言えなくなります。加えて、/init の対象がその演習の CLAUDE.md になること、統合ターミナルの作業ディレクトリが演習フォルダになること、参考解の _reference_完成形/ と hints/ がワークスペースの外になって答えが文脈に混ざらないこと。この4つが揃うのが開き直す形です。開き直すと会話も新しく始まります。前の演習の続きにはなりません。

**開き直しても消えないもの**

午前中に ~/.claude/CLAUDE.md と ~/.claude/settings.json へ書いた設定は、どのフォルダを開いても効き続けます。個人設定（ユーザースコープ）とプロジェクト設定は別の層です。演習2で「ハーネスが無い」と言っているのはプロジェクト側だけで、個人設定は動いています。ここを混ぜると「設定が消えた」と誤解します。

## 01.4 理解度チェック

### Q1

**研修が終わったあと、自分のリポジトリで最初に手を付ける内容として、本日の設計に沿うのはどれですか。**

- **A** VSCode で自分のリポジトリを開き、/init で CLAUDE.md を1枚作る
- **B** Skill・Subagent・Hook・rules を一式そろえてから使い始める
- **C** 本日と同じ演習フォルダを社内に配り、全員に一度やってもらう
- **D** モデルと Effort の組み合わせを総当たりで試して、最適な組を先に決める

**正解と解説を開く**

正解は A です。

本日持ち帰るのは手順書ではなく、ハーネスを育てる順序です。まず CLAUDE.md を1枚置き、次に効いた規約を1つだけ Hook に上げる、という順で広げます。演習3で見る Lv3 の構成は、そろえるだけで手間がかかります。最初からその形を目指すと作りきる前に止まるので、順序として推奨していません。

# 講師紹介

本日は2名で進めます。進行とサポートの役割は当日の状況で分担します。詰まったところは遠慮なく声を掛けてください。

## 02.1 安田 光喜

**Givery 人的資本インテリジェンス部門講師・生成AIエバンジェリスト**

### 安田 光喜

Mitsuki Yasuda

**得意領域**

生成AIをはじめとした世界最先端トレンド／生成AIを用いた新規事業の開発、運営、補助／アプリケーション開発・自動化プログラム環境構築／教育（大学講師、研修、スクール運営など）

**実績**

生成AI研修／大手企業、外資、教育機関（小中高大）でのITトレンド講演／自治体と連携した地域教育実績／上場企業社員対象のリスキリング支援

**メッセージ**

みなさんがこれから生成AIをパートナーに新しい時代を生きていけるように、本日は全力でサポートしていきます。なにかご不明点などありましたら、講師陣になんでも気軽にご質問ください。

公立はこだて未来大学 大学院（メディアデザイン領域）を卒業し、街づくり会社での複合商業ビルの立ち上げにおける情報インフラ整備やデザイン全般業務の責任として関与。その後デザイン事務所を立ち上げ、飲食店を中心としたデザイン業務や美術館展示のアートコンテンツ開発などを行う。また、地域ブランディングにも注力し、市主催の企画運営業務を行なった。2018年10月「小中学生向けのプログラミング教育」に注目し、函館市で初めての小中学生向けプログラミングスクール「D-SCHOOL北海道」を立ち上げて運営。その後北海道のドラッグストア「サッポロドラッグストアー」からスクールの買収を受け、教育を専門とするグループ会社「株式会社シーラクンス」にて技術開発責任者として運営。スクール設計・運営、20以上の自治体連携（イベントや出張教室運営）、大人向けプログラミングスクールメンターなどさまざまな「次世代IT教育」に取り組む。Dify公式資料の日本語翻訳や、SecondMeアンバサダーなど生成AI最新トレンドに注力。

## 02.2 松原 孝司

株式会社ベクターデザイン 取締役／ジェネラリスト系エンジニア

# 松原 孝司

Koji Matsubara

### 得意領域

組み込み、Android、Web、インフラ、IoTを横断した開発／新技術を素早く理解し、実務へ取り入れる技術調査／新規事業やプロジェクトの「0.1を2にする」推進／エンジニアと非エンジニアをつなぐ橋渡し

### 実績

ソフトウェア開発の実務に基づく生成AI活用支援

### メッセージ

生成AIも、正解を教えてくれる魔法の道具ではなく、一緒に考え、試し、成果を磨いていくパートナーだと思っています。20年以上のエンジニア経験を生かし、皆さんと同じ目線で手を動かしながらサポートします。疑問や気づきがあれば、ぜひ気軽に共有してください。

東芝グループのソフトウェアエンジニアとしてキャリアをスタートし、携帯電話向け組み込みソフトウェアや車載用ポータブルナビゲーションシステムの開発に従事。約10年の勤務後、東大生・東工大生が立ち上げたジョイントベンチャーの設立に参画し、AI・機械学習につながる技術を活用したSNS投稿の自動フィルタリングシステムの開発・事業化を担当する。その後、複数のスタートアップで経験を重ね、NewsPicksのAndroidアプリ開発を行ない、エンジニア兼プレイングマネージャーとして携わる。現在は株式会社ベクターデザインの取締役として、気象観測・防災IoTシステムなどに技術面から関わる一方、複数のスタートアップでもエンジニアとして活動。幅広い技術領域を理解し、エンジニアとデザイナー、編集、営業など異なる職種の間をつなぐジェネラリスト型エンジニア。事業やプロジェクトの「0.1を2にする」段階を前進させることを得意としている。

# 生成AI最新トレンド

市場の話ではなく、皆さまの手元の仕事がどう変わったかの話をします。数値はすべて 2026年7月30日に一次情報で確認したもので、出典は章末に並べています。この章と次の章だけ、モデルの版数を時点表記付きで出します。

## 03.1 数値を読む前の注意

ベンチマークの数値は、モデルの性能だけで決まりません。測る側の実装を直しただけで数値が動きます。Terminal-Bench は 2.0 から 2.1 へ上がるときにベンチマーク側のバグを89問中28問修正しましたが、それだけで同じモデル構成のスコアが12.1ポイント動きました（2026年5月6日）。
この章では、共通ハーネスで測った公式リーダーボードの値と、各社が自社ハーネスで測って発表した値を、同じ表に混ぜていません。数値を引用するときは、どちらの種類かを合わせて見てください。

## 03.2 開発者の仕事がどう変わったか

| 0〜20% | 19% | 66% |
| --- | --- | --- |
| 丸ごと任せられるタスクの割合 | 熟練開発者の作業時間が長くなった幅 | 不満の1位「ほぼ正しいが少し違う」 |

Anthropic の Societal Impacts チームの調査では、開発者は自分の仕事の約60%でAIを使う一方、完全に任せられる（fully delegate）タスクは0〜20%にとどまると回答しています（2026年1月21日発行）。同じレポートに、AI支援で行われた作業の約27%は「AIがなければやらなかった作業」だという内部調査もあります。ダッシュボードの作成、探索的な調査、優先度が低くて放置されていた小さな不具合などです。
体感と実測はずれます。METR が熟練オープンソース開発者16名・246課題で行ったランダム化比較試験では、AIを使うと作業時間が19%長くなりました（信頼区間 +2%〜+39%、2025年7月10日）。事前予想は「24%速くなる」で、実測後も本人たちは「20%速くなった」と回答しています。2026年2月24日の続報では初回参加者が -18%、新規参加者47名が -4% で、いずれも信頼区間が0をまたぎました。METR 自身がこれを確定値として扱わず、実験設計の変更を表明しています。
Stack Overflow の2025年調査（AI設問の回答33,662件）では、開発者の84%がAIツールを使うか使う予定と答えました（前年76%）。精度を信頼するのは33%、信頼しないのは46%

です。不満の1位は「ほぼ正しいが少し違う解答」で66%、2位は「AI生成コードのデバッグに時間がかかる」で45%でした。

> **3つの数字がつながる先**
>
> 使うのは当たり前になった。ただし体感は当てにならない。任せきれない部分と、惜しい出力の後始末が新しい仕事になった。本日の演習で扱う差分の確認と承認、CLAUDE.md、権限モード、レビューの分業は、この3つの数字に対する打ち手として置いています。

## 03.3 モデルの現在地とコスト

Artificial Analysis の知能指数（2026年7月28日時点）では、上位が Claude Opus 5 (max) 61、Claude Fable 5 60、GPT-5.6 Sol (max) 59 と並び、上位3モデルの差は2ポイントです。同じ表の1タスク当たり平均コストは DeepSeek V4 Pro の0.04ドルから Claude Fable 5 の2.75ドルまで開き、スコア差17ポイントに対してコスト差は約69倍になります。性能で選ぶ段階は終わっていて、実際の判断はコストと速度で決まります。
コーディング性能は、評価軸が変わると順位も変わります。OpenAI が自社発表ページに掲載した比較表（2026年7月9日、各社が自社ハーネスで測った値）では、SWE-Bench Pro が Claude Mythos 5 80.3%、Claude Fable 5 80%、Claude Opus 4.8 69.2%、Grok 4.5 64.7%、GPT-5.6 Sol 64.6% の順です。同じ資料の Terminal-Bench 2.1 では GPT-5.6 Sol Ultra 91.9%、Sol 88.8%、Claude Mythos 5 88% と入れ替わります。

> **同じベンチマーク名でも数字が違う**
>
> SWE-bench Verified の公式リーダーボードは全モデルを同一ハーネス（mini-SWE-agent v2、500問）で走らせており、最高は76.80%です（2026年2月17日時点）。同じベンチマーク名で、ベンダーは90%台を自己申告しています。実務規模の改修を扱う SWE-bench Pro の公開セット（731問）では最高が61.5%です（2026年7月28日時点）。「SWE-bench で何%」という言い方だけでは比較になりません。

## 03.4 ツールと規格の動き

| 時点 | 動き |
| --- | --- |
| 2026-06-01 適用 | GitHub Copilot の課金が使用量ベースに変わり、premium request units は GitHub AI Credits に置き換わりました。消費はモデルごとの公開API単価に基づいて計算されます。コード補完と Next Edit Suggestions は AI Credits を消費しません。1 AI credit は0.01ドルで、月あたりのクレジットはプラン料金と同額です（Pro 10ドル、Pro+ 39ドル、Business 19ドル、Enterprise 39ドル） |
| 2026-07-07 | オープンウェイトモデル Kimi K2.7 Code が Copilot のモデルピッカーで選べるようになりました。Business と Enterprise では既定でオフで、管理者がポリシーを有効化しないと選択できません |
| 2026-07-28 | MCP 仕様 2026-07-28 が公開されました。双方向ステートフルなプロトコルから、リクエストごとに完結する形と、リクエストごとの機能ネゴシエーションへ変わり、サーバーをサーバーレスやエッジに置けるようになりました |
| 2026-07-29 | Visual Studio Code 1.131 が公開されました。実行中サブエージェントのモデル・経過時間・実行中ツールを、会話を開かずに確認できます |

MCP については、Anthropic が同仕様の認可を OAuth 2.0 と OIDC の実運用構成に合わせ、Entra や Okta とつながる形にしたと説明しています。MCP の SDK ダウンロードは月4億件を超え、1年で4倍になりました。Claude のコネクタディレクトリには950を超える MCP サーバーが載っています（2026年7月28日時点）。

## 03.5 国内の開発現場

| 事例 | 内容 | 数値の種別 |
| --- | --- | --- |
| NEC | Anthropic と戦略的協業を開始し、NECグループの約3万人に Claude を展開。社内に Center of Excellence を設け、Claude Code を業務で使うAIネイティブなエンジニアリング組織を作ると説明しています（2026年4月24日） | 展開規模 |
| 楽天 | 1,250万行・複数言語の大規模OSSである vLLM への活性ベクトル抽出手法の実装を Claude Code に任せ、1回の実行で7時間の自律作業を経て完了。参照実装との数値一致は99.9%（2026年1月21日） | Anthropic レポート内の記載。計測方法は非開示 |

| 事例 | 内容 | 数値の種別 |
| --- | --- | --- |
| GMOデザインワン | 2025年11月26日より全エンジニアに Claude Code を導入。作業時間を約50%短縮することを目標として掲げています | 目標値であり実績ではありません |

削減した時間の行き先も測られています。パーソル総合研究所の調査（2026年3月13日）では、生成AI利用者のうち業務削減時間が生じたのは25.4%で、削減幅はタスクレベルで平均16.7%（週26.4分）でした。浮いた時間を仕事に再投下した人は61.2%で、その再投下先の75.4%は日常業務です。

> **この章の結論**
>
> 順位表を見て社内の標準ツールを決めても、測り方が変わると結論が変わります。自社リポジトリの実タスクを3本決めて、モデルを変えて同じプロンプトで流し、通ったか・何回やり直したか・かかった時間を記録する。この型のほうが実務に効きます。本日の演習3本は、その練習台として作っています。

## 03.6 理解度チェック

### Q2

**ベンチマークの順位表を見て、自社で使うモデルを決めようとしています。この章の立場に最も近いのはどれですか。**

- A 共通ハーネスで測った公式リーダーボードの値だけを見れば、順位はそのまま使える
- B ベンダー発表値のほうが新しいモデルを含むので、そちらを優先して判断する
- C 複数の順位表で共通して上位に入るモデルを選べば、それで判断できる
- D 順位表は候補を絞るまでにとどめ、自社の実タスクを3本決めて同じプロンプトで測る

**正解と解説を開く**

正解は D です。

同じ SWE-bench Verified でも、共通ハーネスの最高が76.80%（2026年2月17日時点）に対して、ベンダーは同じ名前で90%台を自己申告しています。Terminal-Bench では、ベンチマーク側のバグを28問修正しただけでスコアが12.1ポイント動きました（2026年5月6日）。順位表は測り方に強く依存するので、候補を絞る材料としては使えても、最後の判断は自社のタスクで測ります。

SOURCES


- Anthropic: 2026 Agentic Coding Trends Report（2026-01-21）
- METR: 熟練開発者のランダム化比較試験（2025-07-10）
- METR: 続報と実験設計の変更（2026-02-24）
- Stack Overflow Developer Survey 2025: AI（2025年調査）
- Artificial Analysis: モデル一覧と1タスク単価（2026-07-28 時点）
- SWE-bench 公式リーダーボード（2026-02-17 時点）
- SWE-bench Pro 公開セット リーダーボード（2026-07-28 時点）
- OpenAI: GPT-5.6 発表ページ（SWE-Bench Pro / Terminal-Bench 2.1 の比較表、2026-07-09）
- Terminal-Bench 2.1 のリリースノート（2026-05-06）
- GitHub: Copilot の使用量ベース課金への移行（2026-04-27 発表 / 2026-06-01 適用）
- GitHub Docs: Copilot のモデル別単価と AI Credits
- GitHub Changelog: Kimi K2.7 Code が Business / Enterprise でも選択可能に（2026-07-07）
- Model Context Protocol 仕様 2026-07-28
- Anthropic: MCP 2026-07-28 の Claude への反映（2026-07-28）
- Visual Studio Code リリースノート（1.131、2026-07-29）
- Anthropic: NEC との戦略的協業（2026-04-24）
- GMOデザインワン: 全エンジニアへの Claude Code 導入（2025-11-26）
- パーソル総合研究所: 生成AIによる業務削減時間の調査（2026-03-13）

# Claudeファミリーの現在地

版数を覚える章ではありません。どの軸で選ぶかを持ち帰る章です。表の縦軸をモデル名にすると版数の暗記になるので、軸のほうを縦に置いています。数値はすべて 2026年7月30日時点で、当日は変わっている前提で見てください。

## 04.1 現行ラインナップ

**1,000,000**
主力3モデルのコンテキストウィンドウ（トークン）

**128k**
同じ3モデルの最大出力トークン

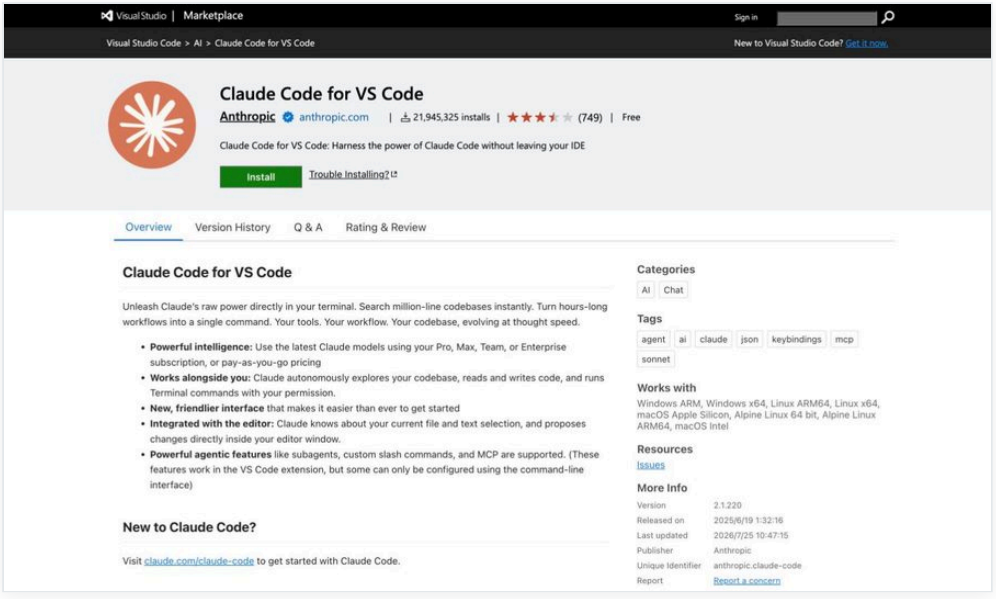
| モデル | 公式の位置づけ | 入力 / 出力（100万トークンあたり） | コンテキスト / 最大出力 | 相対レイテンシ |
|---|---|---|---|---|
| Claude Fable 5 | 長時間稼働エージェント向けの次世代知能。最も高性能な一般提供モデルと表記 | 10ドル / 50ドル | 1M / 128k | Slower |
| Claude Opus 5 | 複雑なエージェント型コーディングと業務向け | 5ドル / 25ドル | 1M / 128k | Moderate |
| Claude Sonnet 5 | 速度と知能の組み合わせが最良 | 2ドル / 10ドル（2026年8月31日までの導入価格。以降は3ドル / 15ドル） | 1M / 128k | Fast |
| Claude Haiku 4.5 | フロンティア近傍の知能を持つ最速モデル | 1ドル / 5ドル | 200k / 64k | Fastest |

1M のコンテキストは既定で使えて、長文だけ割高になる料金は設定されていません。公式ドキュメントの選択指針は「どれを使うか迷ったら、複雑なエージェント型コーディングと業務向けには Claude Opus 5 から始める。使える中で最高の能力が必要な作業には Claude Fable 5 を使う」です。Haiku 4.5 だけ 200k・64k で、同じ Claude でも桁が違います。

## 04.2 使い分けの軸

| 軸 | 見るもの |
| --- | --- |
| **タスクの難しさと自律実行の長さ** | 分類・要約・整形のような一発仕事、通常の実装と調査、複数ファイルにまたがる実装や大きめのリファクタ、夜間放置の長時間エージェント。この順に上げます |
| **レイテンシの許容度** | 対話で待てる時間と、バッチで放置できる時間は別物です。公式表の相対レイテンシは Fastest / Fast / Moderate / Slower の4段です |
| **トークン消費量** | Opus 4.7 以降の世代は新しいトークナイザーを使い、同じ文章でおよそ30%多いトークン数になります。単価表だけを見て「安くなった」と判断すると外れます |
| **Effort** | 同じモデルの中でコストと精度を動かすつまみです。モデルを上げる前に Effort を上げる、という順番で検討する余地があります。操作は6章で扱います |
| **コンテキスト長** | 1M と 200k の差です。1M は既定で使えて追加料金はありません |
| **思考の扱い** | モデルごとに thinking の対応が違い、モデルを差し替えるとAPIの書き方まで変わることがあります。API を書く場面では、リリースノートの破壊的変更の欄を先に見ます |

## 04.3 3つの入口の違い




Marketplace の拡張ページです。発行元が Anthropic、Unique Identifier が anthropic.claude-code であることを確認してから入れます。

| | Claude Code | Claude アプリ | Claude API |
|---|---|---|---|
| 何をするもの | コードベースを読み、ファイルを編集し、コマンドを実行するエージェント型コーディングツール | 会話・文書・ファイル・コネクタで、人が考える作業を進める場所 | 自社プロダクトやワークフローに Claude を組み込むための開発者向けインターフェース |
| 入口 | ターミナル、VSCode 拡張、JetBrains プラグイン、デスクトップアプリ、Web、モバイル | claude.ai、デスクトップアプリ、Claude Cowork | platform.claude.com、各言語 SDK、Managed Agents |
| 課金 | Claude のサブスクリプション（Pro 以上）または Console アカウント | サブスクリプション | 従量（トークン単価） |

手元のコードに手を入れさせたいなら Claude Code、考えを整理して文書にしたいならアプリ、他の人が使う仕組みとして組み込むなら API です。Claude Code はどの入口を選んでも同じエンジンにつながり、CLAUDE.md・settings・MCP サーバーの設定はすべての入口で共有されます。本日は VSCode 拡張で進めますが、あとでターミナルやデスクトップに移っても設定は持ち越せます。
第三者プロバイダ（Amazon Bedrock、Google Cloud、Microsoft Foundry）経由での利用に対応しているのは、ターミナルの CLI と VSCode の2つです。社内で Bedrock 経由に寄せる方針がある場合は、この2つが選択肢になります。

## 04.4 他社モデルとの比較をどう扱うか

Anthropic の公式発表ページは、他社モデルの名前を出しません。Opus 5 の発表（2026年7月24日）でも比較対象は自社の Opus 4.8 / Fable 5 / Mythos 5 で、他社は「the next-best model」という表現にとどまります。「Claude が他社より何ポイント上」という形の数字は、Anthropic の一次情報からは取れません。
他社名入りの数字が要る場合は第三者のリーダーボードを使うことになりますが、性質を必ず併記します。llm-stats.com の SWE-bench Verified リーダーボード（最終更新 2026年7月30日、掲載104モデル）は、独立検証が0件で全てベンダーの自己申告です。3章で見たとおり、共通ハーネスの値と自己申告値は10ポイント以上ずれます。

> **検索結果の要約に出てくる数字を使わない**
>
> 検索スニペットには「Opus 5 が SWE-bench Verified 97.00%」といった数値が出てきますが、実際のリーダーボードページを開くと該当する値が見当たりませんでした。この資料では、ページを開いて目で確認できた値だけを使っています。

## 04.5 最新動向の追い方

| URL | 何が分かるか | 更新の粒度 |
|---|---|---|
| code.claude.com/docs/en/whats-new | Claude Code の週次ダイジェスト。各週にバージョン範囲と、動くコード例・本編ドキュメントへのリンクが付きます。まず読むならここです | 毎週 |
| code.claude.com/docs/en/changelog | 全バージョンの変更点。バグ修正や小さな挙動変更まで載ります | ほぼ毎日 |
| platform.claude.com/docs/en/release-notes/api | Claude API・各言語SDK・Console の変更。新モデル公開、ベータヘッダーの追加、400 エラーになる破壊的変更、廃止予告がここに集まります | 数日ごと |
| platform.claude.com/docs/en/about-claude/models/overview | 現行モデルの一覧表。モデルID、コンテキスト長、最大出力、価格、thinking の対応状況、知識カットオフ | モデル公開時 |
| platform.claude.com/docs/en/about-claude/model-deprecations | 非推奨・引退の日付と推奨移行先。使っているモデルIDの寿命を確認する場所です | 廃止告知時 |
| status.claude.com | 稼働状況と障害履歴。メール・SMS・Slack・Microsoft Teams・Webhook・RSS で購読できます | 障害発生時 |
| anthropic.com/news | 製品発表・研究・事例。モデル公開時の位置づけとベンチマークの説明はここが一次情報です | 随時 |

情報システム部門に共有するなら、status.claude.com の購読設定を最初に入れておくのが実務的です。障害の切り分けで「自分の環境か、サービス側か」を最初に判断できます。

## 04.6 料金の考え方

Claude Code は Pro 以上のプランに含まれます。無料プランには含まれる旨の記載がありません。既定モデルはプランによって違い、Sonnet 5 が Pro / Team Standard / Enterprise サブスクリプション席の既定、Opus 5 が Max の既定で Pro でも選べる最上位です（2026年7月30日時点）。
コストを下げたいときの順番は、モデルを安いほうへ落とす前に、Effort を下げる・プロンプトキャッシュを効かせるの2つを先に検討します。プロンプトキャッシュは書き込みが基本入力価格の1.25倍（5分間有効）または2倍（1時間有効）、読み出しが0.1倍なので、5分キャッシュなら1回読まれた時点で元が取れます。トークナイザーが世代で変わるため、単価表の比較だけでは判断できない点も合わせて押さえてください。

> **研修当日に消えるページが1つあります**
>
> 旧 Workbench（platform.claude.com/workbench）は 2026年8月17日でアクセス終了です。研修当日と同じ日付にあたるため、当日「Console の Workbench を開いて」という案内はしません。experimental prompt tools API も同日で廃止されます。

## 04.7 理解度チェック

### Q3

**出力が惜しい形で外れます。ファイルを読み飛ばしたり、テストを走らせないまま完了と言ったりします。まず動かす設定はどれですか。**

- A モデルを1段上のものへ切り替える
- B Effort を1段上げる
- C コンテキストウィンドウの大きいモデルへ切り替える
- D プロンプトの末尾に think hard と書き足す

**正解と解説を開く**

正解は B です。

公式ブログの整理では、知識が足りないときはモデルを上げ、丁寧さが足りないときは Effort を上げます。読み飛ばし・テスト未実行・確認不足は後者にあたります。モデルと Effort を同時に動かすと、どちらが効いたのか切り分けられなくなります。なお、プロンプト内でキーワードとして認識されるのは `ultrathink` だけで、`think hard` は通常の文章として扱われます。

SOURCES

- Claude Platform: モデル一覧と選択指針
- Claude Platform: 料金（トークン単価・キャッシュ・Batch API）
- Claude Platform: API リリースノート
- Claude Platform: モデルの非推奨と引退
- Anthropic: Claude Opus 5 の発表（2026-07-24）
- Claude Code 公式ドキュメント: Overview
- Claude Code 公式ドキュメント: 週次ダイジェスト
- Claude: プランと価格
- llm-stats.com: SWE-bench Verified リーダーボード（自己申告値、2026-07-30 時点）
- Claude: 稼働状況

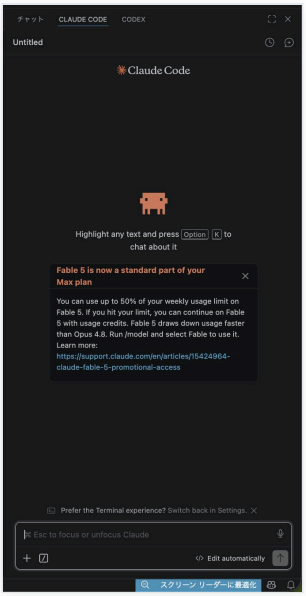
ORIENTATION 05

# Claude Code の基礎A 画面と位置づけ

ここからは操作の話です。本日の主軸は VSCode の Claude Code 拡張（拡張ID `anthropic.claude-code`、発行元 Anthropic）で、ターミナルで `claude` と打つ CLI は使いませ

ん。公式ドキュメントは拡張を「VSCode に組み込まれたネイティブのグラフィカルインターフェース」と説明し、VSCode で使う場合の推奨手段としています。

## 05.1 起動口と画面の部位




VSCode の右側に Claude Code のパネルを開いた状態です。入力欄の左下にある + が Add context、その右がモードインジケータです。

VSCode 全体を通じて、Spark アイコンが Claude Code を意味します。起動口は4つあります。

- エディタ右上の Spark アイコン。最も早い経路です。ファイルを開いているときだけ出ます
- アクティビティバー（左サイドバー）の Spark アイコン。セッション一覧が開きます。このアイコンは常に表示されます
- コマンドパレット（Windows `Ctrl+Shift+P` / Mac `Cmd+Shift+P`）から「Claude Code」
- ウィンドウ右下のステータスバーの `Claude Code`。ファイルを開いていなくても使えます

フォーカスの行き来は Windows `Ctrl+Esc` / Mac `Cmd+Esc` です。macOS Tahoe 以降はシステムの Game Overlay が `Cmd+Esc` を先に奪うため効かないことがあります。アイコンのクリックを第一手順にしてください。

| 画面の部位 | 場所 | できること |
| --- | --- | --- |
| プロンプト入力欄 | パネル下部 | 指示の入力。`Shift+Enter` で送信せず改行を足します |
| モードインジケータ | 入力欄の下部 | 現在の権限モードの表示。クリックで切り替えます |

| 画面の部位 | 場所 | できること |
| --- | --- | --- |
| **Context usage** | 入力欄まわり | コンテキストの残量。表示は「% of context remaining until auto-compact.」です |
| **選択範囲インジケータ** | 入力欄のフッター | エディタで選択している行数。クリックで Claude に見せるかどうかを切り替えます。目に斜線が入ったアイコンは隠れている状態です |
| **添付ファイル** | 入力欄 | 添付として追加したファイルが並びます。X で文脈から外れます |
| **コマンドメニュー** | 入力欄で / | ファイルの添付、モデルの切り替え、使用量などが並びます。Customize セクションから MCP サーバー・フック・メモリ・権限・プラグインへ入れます |
| **セッション履歴** | パネル上部の Session history | 会話履歴の一覧。キーワード検索と期間で辿れます。クリックすると全メッセージ付きで再開します |
| **差分ビュー** | エディタ領域 | 編集提案を元の内容と並べて表示します |
| **Account & Usage** | コマンドメニュー | アカウント情報と使用量の上限。使用量の内訳も見られます |

パネルの置き場所は3通りです。右サイドバー（コードを書きながら常に見える配置）、左サイドバー、エディタ領域のタブ。パネルのタブかタイトルバーをドラッグして移動します。

## 05.2 差分の確認と承認

Claude がファイルを編集しようとすると、元の内容と提案内容を並べて表示し、許可を求めます。ここでできることは3つです。受け入れる、却下する、代わりにどうしてほしいかを伝える。

> **差分ビューの中で直してから受け入れられます**
>
> 差分ビューの中で提案内容を直接編集してから受け入れると、Claude には「あなたが手を入れた」ことが伝わります。元の提案どおりのファイルになっていると Claude が思い込まないための仕組みです。惜しい出力を毎回突き返すより、その場で直すほうが速い場面があります。

## 05.3 権限モード

権限モードは、プロンプト入力欄の下部にあるモードインジケータをクリックして切り替えます。チャットで「Plan モードにして」と頼んで変えるものではありません。本日扱うのは次の3つです。

| パネルのラベル | 設定値 | 確認なしで動く範囲 |
|---|---|---|
| Manual | default | 読み取りだけ。既定はこれです |
| Plan | plan | 読み取り中心。編集の前に計画を出します |
| Edit automatically | acceptEdits | 読み取り、ファイル編集、mkdir touch mv cp などのファイル操作コマンド |

新しい会話の既定モードは VSCode 設定の claudeCode.initialPermissionMode で決まります（既定値は default）。このほかに Auto と Bypass permissions がありますが、本日は使いません。Bypass permissions は拡張設定の Allow dangerously skip permissions を有効にしないと出ません。有効にしないでください。

> **Shift+Tab は拡張の操作ではありません**
>
> Shift+Tab によるモードの循環は CLI 側の操作です。公式の権限モードのページはインターフェースごとにタブを分けており、VSCode タブに書かれているのは「セッション中はプロンプト入力欄の下部のモードインジケータをクリックする」だけです。ネット記事で Shift+Tab を見かけても、拡張では効きません。

どのモードでも、いくつかのパスへの書き込みは自動承認されません。.git .vscode .idea .claude などが保護対象です。.claude/ への書き込みで出る確認には「Yes, and allow Claude to edit its own settings for this session」という選択肢があり、選ぶとそのセッション中は再確認なしになります。

## 05.4 プランモードで計画に注文を付ける

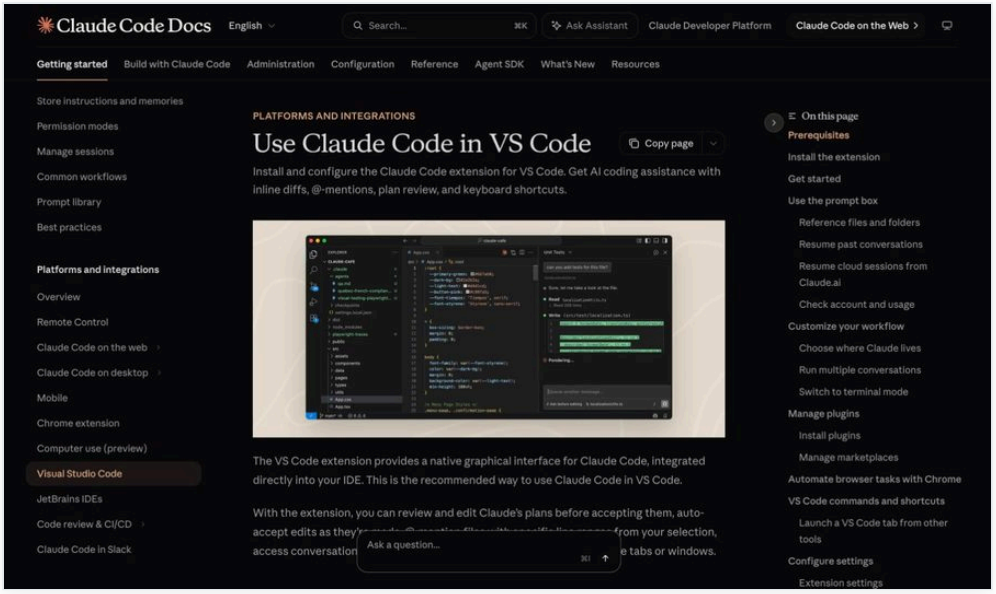
Plan モードでは、Claude が何をするかを説明し、変更を加える前に承認を待ちます。ファイルは読み、探索のためのシェルコマンドは動かし、計画を書きますが、ソースは編集しません。

拡張固有の動きとして、VSCode は計画を独立した Markdown ドキュメントとして自動で開きます。そこにインラインコメントを書き込むと、Claude が作業を始める前に計画そのものへ注文を付けられます。計画ができあがると進め方を聞かれるので、本日は「Yes, manually approve edits」を選び、編集を1件ずつ確認する流れを標準にします。

## 05.5 ワークスペースの信頼

VSCode が Restricted Mode（制限モード）だと拡張は動きません。フォルダを開いたときに出る確認で、信頼する側を選んでください。本日は演習ごとにフォルダを開き直すので、この確認も演習ごとに出ます。
Spark アイコンがエディタ右上に見えないときは、公式が挙げている5点を順に確認します。ファイルを開いているか、VSCode が 1.94.0 以上か（Help から About）、コマンドパレットの `Developer: Reload Window` で再読み込みしたか、他のAI拡張（Cline、Continue など）を一時的に無効にしたか、ワークスペースを信頼しているか。

## 05.6 CLI との違い



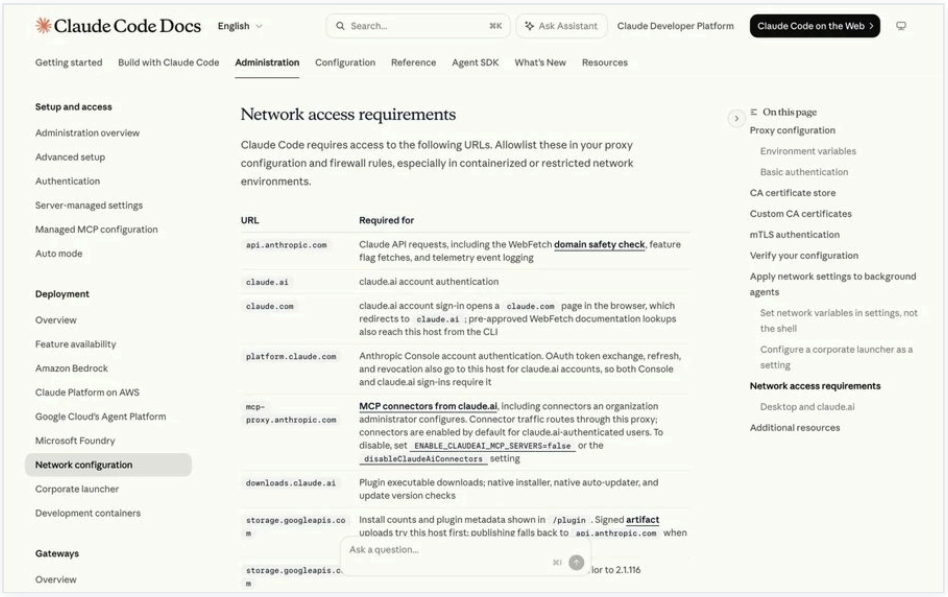
公式ドキュメントの VS Code のページです。拡張が推奨手段であることが書かれています。

| 機能 | CLI | VSCode 拡張 |
| --- | --- | --- |
| コマンドとスキル | すべて | 一部（/ を押して使えるものを確認します） |
| MCP サーバーの設定 | できます | 一部（追加は CLI、既存サーバーの管理はパネルの /mcp） |
| チェックポイント | あります | あります |

| 機能 | CLI | VSCode 拡張 |
| --- | --- | --- |
| ! の bash ショートカット | あります | ありません |
| Tab 補完 | あります | ありません |

拡張はチャットパネル用に CLI の私的なコピーを同梱していますが、それは PATH に入りません。統合ターミナルで claude と打っても起動しないのはこのためです。本日 CLI をインストールする必要はありません。
設定と会話履歴は共有されます。~/.claude/settings.json の Claude Code 設定は拡張と CLI の両方で読まれ、許可コマンド・環境変数・フック・MCP サーバーはここに書きます。拡張独自の設定は VSCode 側（Windows Ctrl+, / Mac Cmd+, から Extensions の Claude Code）にあります。

## 05.7 ネットワーク要件



公式ドキュメントに載っている許可対象ドメインの一覧です。情報システム部門へ提出する際はこの表を使います。

情報システム部門へ提出する許可ドメインは6つです。社内プロキシ環境では、この一覧を先に通してもらってください。

| ドメイン | 用途 |
| --- | --- |
| api.anthropic.com | Claude API との通信 |
| claude.ai | アカウント認証 |
| claude.com | サインイン時にブラウザで開きます |

| ドメイン | 用途 |
| --- | --- |
| platform.claude.com | サインインのトークン交換・更新。許可漏れが最も多いのがここです |
| downloads.claude.ai | インストールと自動更新 |
| marketplace.visualstudio.com | VSCode が拡張を取得するため（Microsoft 側の要件） |

疎通確認は curl -I で各ドメインを叩きます。判定は「200 が返るか」ではありません。タイムアウトや名前解決失敗にならず、何らかの HTTP ステータス行が返れば到達できています。403 や 404 でも到達の確認としては十分です。プロキシは HTTPS_PROXY / HTTP_PROXY / NO_PROXY に対応します（SOCKS は非対応）。設定は ~/.claude/settings.json の env に書くのが推奨です。

## 05.8 理解度チェック

### Q4

**拡張のパネルで権限モードを Manual から Plan へ変える操作はどれですか。**

A Shift+Tab を押してモードを循環させる

B プロンプト入力欄の下部にあるモードインジケータをクリックする

C パネルに「Plan モードにしてください」と入力して切り替えてもらう

D コマンドパレットから Claude Code の Plan Mode コマンドを実行する

正解と解説を開く

正解は B です。

Shift+Tab の循環は CLI 側の操作で、拡張にはありません。拡張では入力欄の下部のモードインジケータをクリックして Manual / Plan / Edit automatically を切り替えます。新しい会話の既定モードは VSCode 設定の `claudeCode.initialPermissionMode` で決まります。

SOURCES

- Claude Code 公式ドキュメント: VSCode で使う
- Claude Code 公式ドキュメント: 権限モード
- Claude Code 公式ドキュメント: セットアップと動作要件
- Claude Code 公式ドキュメント: ネットワーク設定
- Visual Studio Marketplace: Claude Code for VS Code

ORIENTATION 06

# Claude Code の基礎B モデル・文脈・コマンド

同じパネルの中で、出力の質とコストを動かせるつまみが2つあります。モデルと Effort です。あわせて、ファイルの渡し方4経路と、コンテキストの見方、使うスラッシュコマンドを押さえます。

## 06.1 モデルの切り替え

パネルのコマンドメニューにモデル切り替えの項目があります（説明は「Change the AI model」）。一覧は表示名と説明つきで出ます。`/model` でも同じことができ、引数なしならピッカーが開き、`/model <名前>` で直接切り替わります。ピッカーでは Enter が「切り替えて既定として保存」、s が「このセッションだけ切り替え」です。
会話にすでに出力がある状態で切り替えると確認が入ります。次の応答がキャッシュなしで履歴を読み直すためです。指定できるエイリアスは次のとおりで、指す版数はプランと時期で変わります。

| エイリアス | 挙動 |
| --- | --- |
| `default` | モデル指定を解除して、アカウント種別の推奨モデルに戻します |
| `best` | 組織が使える中で最上位のモデルを使います |
| `fable` / `opus` / `sonnet` / `haiku` | それぞれの系統の最新を使います。長時間の作業、複雑な推論、日々のコーディング、単純で速い作業の順です |

| エイリアス | 挙動 |
| --- | --- |
| `sonnet[1m]` / `opus[1m]` | 100万トークンのコンテキストウィンドウで使います |
| `opusplan` | プランモードでは opus、実行時は sonnet に自動で切り替わります |

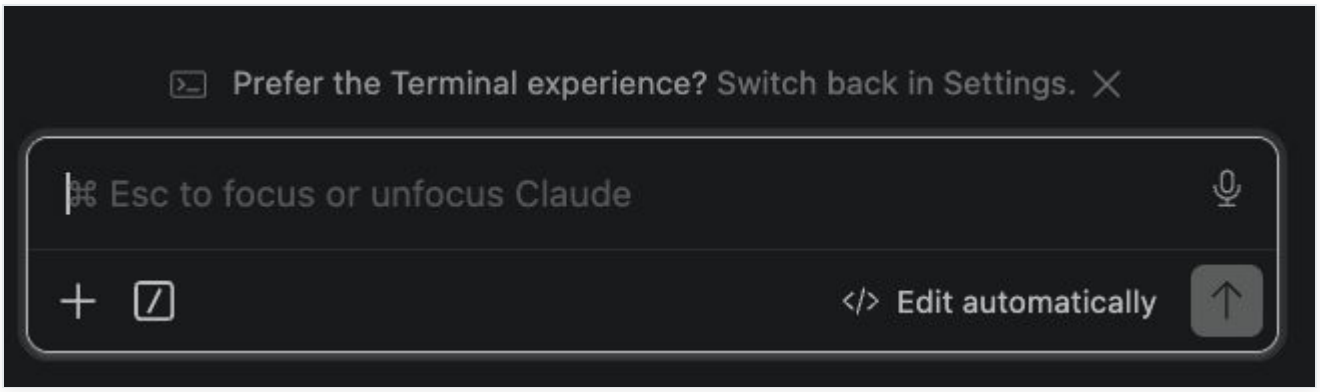
## 06.2 Effort

Effort はパネルの中で切り替えられます。権限モードのメニューに Effort の行があり、レベル表示が付きます。操作の説明文は「Click or drag to set effort level」「Click to cycle effort level」です。

> **段階の数え方**
>
> パネルに出る既定の並びは low / medium / high の3つで、xhigh は条件付きで出ます。設定ファイル（~/.claude/settings.json）の effortLevel が取る値は low / medium / high / xhigh の4値です。選べる段階はモデル側の申告に従うため、モデルによって並びが変わります。ネット記事に段数の異なる説明がありますが、この資料では実機の設定スキーマと実装文字列で確認できた範囲だけを書いています。

選び方は2軸で分けると迷いません。知識が足りないならモデルを上げる。丁寧さが足りないなら Effort を上げる。公式ブログは、ファイルを読み飛ばした・テストを走らせなかった・確認をしなかった、という失敗のときに Effort を上げると書いています。両方を同時に動かすと、どちらが効いたのか分からなくなります。
プロンプトの中に書いて効くキーワードは ultrathink だけです。セッションの Effort 設定を変えずに、そのターンだけ深く考えさせます。think や think hard は通常の文章として扱われ、キーワードとしては認識されません。

## 06.3 ファイルの渡し方4経路




入力欄まわりです。左下の + が Add context、その右がモードインジケータです。ドラッグして渡すときは Shift を押しながら運びます。

| 経路 | 操作 |
| --- | --- |
| **1. @ 参照** | 入力欄で @ に続けてファイル名かフォルダ名を打ちます。あいまい一致に対応するので部分入力でも候補が出ます。フォルダは末尾にスラッシュを付けます。本日はこれを第一手順にします |
| **2. Add context** | 入力欄の左下の + です。「Add files or folders to the conversation」「Attach files from your computer」という項目が並びます |
| **3. ドラッグ＆ドロップ** | **Shift を押しながら**ファイルを入力欄へドラッグします。案内文は「Hold Shift while dragging to drop files into Claude Code」で、受け側に「Drop to attach as context」が出ます |
| **4. 範囲選択の自動共有** | エディタでテキストを選択すると、その範囲は自動で見えます。入力欄のフッターに選択行数が出ます。行範囲付きの参照を挿入するショートカットは Windows `Alt+K` / Mac `Option+K` です |

> **Shift を押さないとエディタでファイルが開くだけです**
>
> ドラッグだけではパネルに渡りません。エディタ側でファイルが開いて終わります。当日ここで詰まる方が出るはずなので、先に書いておきます。うまくいかないときは @ 参照に切り替えてください。

「ファイル名を自然文で書いて伝える」方法は、独立した機構としての記述が公式にも実装にもありません。@ のあいまい一致で拾われる形になるので、確実に渡したいときは @ を使ってください。添付は各添付の X をクリックすると文脈から外れます。

## 06.4 コンテキストと使用量の見方

コンテキストの残量はパネルに出ます。表示は Context usage で、説明は「% of context remaining until auto-compact.」です。上限に近づくと自動で圧縮が走り、「Conversation was compacted to free up context.」が出ます。満杯になってもセッションは止まりません。
使用量はコマンドメニューの Account & Usage から見ます。使用量のバーに加えて「% of your usage came from /コマンド」「MCP server」「plugin」「subagents under」という内訳が出ます。何が使用量を食っているかを自分で確認できる画面です。

> **statusLine は拡張のパネルには出ません**
>
> 設定ファイルの statusLine はフックと同じ実行系の機能で、拡張のチャットパネルには描画されません（拡張の webview 実装に参照が0件でした）。~/.claude/settings.json に書く行為自体は正しく、拡張と共有される設定に入ります。本日のミニ演習では、設定とスクリプトが正しく書けたことを確認の基準にします。モデル名・コンテキスト残量・使用量の3つは、パネルの Context usage と Account & Usage とモデル切り替えで確認できます。

## 06.5 スラッシュコマンド

入力欄で / を押すとコマンドメニューが開きます。並ぶコマンドは実行時に取得されるため、この資料に固定の一覧は書きません。当日、実機で / を開いて一覧を見ます。本日よく使う3つだけ挙げます。

| コマンド | 使いどころ |
| --- | --- |
| /init | そのプロジェクトに CLAUDE.md の雛形を作ります。本日は演習ごとに毎回実行します。既存の CLAUDE.md がある演習では、既存行を削る提案が出たら却下してください |
| /compact | 同じ会話を続けたまま、ここまでのやり取りを要約に置き換えます。/compact 認証まわりの修正に絞って のように観点を渡せます |
| /clear | コンテキストを空にして新しい会話を始めます。前の話は引き継ぎません。無関係な作業に移るときはこちらです |

この2つはコストが違います。/compact は要約するために会話全体を読むので、それ自体が大きなリクエストになります。/clear はトークンを消費しません。切り替えが目的なら

/clear のほうが安く済みます。/clear で新しいセッションが始まると、使用量の集計も0に戻ります。
このほか /context（コンテキスト使用量の可視化）、/memory（CLAUDE.md と auto memory の管理）、/permissions（承認ルール）、/status（セッション設定）、/help があります。拡張のコマンドは公式に「一部」と書かれているため、この5つが / のメニューに出るかは当日確認します。出ない場合の代替は各演習の手順に併記しています。

## 06.6 理解度チェック

### Q5

**エクスプローラからファイルをパネルへドラッグしたのに、エディタでファイルが開くだけで添付されません。原因はどれですか。**

- **A** 拡張がファイルのドラッグ＆ドロップに対応していない
- **B** ワークスペースを信頼していないため、添付が拒否されている
- **C** ファイルサイズが添付の上限を超えている
- **D** Shift を押しながらドラッグしていない

**正解と解説を開く**

正解は D です。

拡張の案内文は「Hold Shift while dragging to drop files into Claude Code」です。Shift を押さないと VSCode 側のドラッグとして扱われ、エディタでファイルが開くだけでパネルに

は渡りません。ファイルを渡す経路は、@参照、入力欄左下の Add context、Shift を押しながらのドラッグ＆ドロップ、エディタの範囲選択の自動共有の4つです。

SOURCES

- Claude Code 公式ドキュメント: モデル設定
- Anthropic ブログ: Claude Code でのモデルと Effort の選び方
- Claude Code 公式ドキュメント: コマンド一覧
- Claude Code 公式ドキュメント: コンテキストウィンドウ
- Claude Code 公式ドキュメント: コスト
- Claude Code 公式ドキュメント: ステータスライン
- Claude Code 公式ドキュメント: VSCode で使う

ORIENTATION 07

# ハーネス3段階の地図

午後の演習3本は、ハーネスの量を変えた3つの環境を順番に触る構成です。ここでは、何をハーネスと呼ぶのか、CLAUDE.md・docs・Skill・Subagent・Hook・rules がそれぞれどの層の仕事をするのかを整理します。

## 07.1 作り手のハーネスと使い手のハーネス

「ハーネス」という語は、使う人によって指すものがずれています。原義は馬具で、馬を馬車につなぎ、方向と力を伝える装備のことです。2021年に言語モデルの評価用の足回りとして lm-eval-harness に登場し、2024年にエージェント向けへ広がり、2026年に設計対象として語られるようになりました。

作り手側

### エージェントハーネス

モデルの周囲に実装される足回りです。制御ループ、ツールとメモリ、コンテキスト管理、ガードレール、検証ループ。Claude Code の内部実装がこれにあたります。皆さまが手を入れる対象ではありません。

使い手側

### ユーザーハーネス

自分の環境に合わせて、エージェントの振る舞いを制約・設定する仕組みです。CLAUDE.md、Skills、Hooks、MCP、テストや lint、ワークフロー定義。本日作るのはこちらです。

この区別を持ち込むと、記事ごとに定義が違う理由が見えます。ベンダーは制御ループの話をしていて、利用者は設定ファイルの話をしている。同じ語で別の層を指しています。

```
Harness　ハーネス（ツールと制約と検証の総体）
└── Context　コンテキスト（何を読ませるか）
    └── Prompt　プロンプト（今回の指示）

Harness ⊇ Context ⊇ Prompt
```

プロンプトはコンテキストの一部で、コンテキストはハーネスの一部です。プロンプトを磨く段階から、何を読ませるかの設計へ、さらにその外側へ関心が広がっていきます。

## 07.2 CLAUDE.md を書いただけでは埋まらない3象限

ユーザーハーネスを2軸で並べると、CLAUDE.md と Skills が占めるのは右上の一角だけです。縦軸は事前・予防（ガイド）と事後・自己修正（センサー）、横軸は計算的（決定論・安価・高速）と推論的（LLMの判断）です。

| | 計算的（決定論・安価・高速） | 推論的（LLM の判断） |
|---|---|---|
| **ガイド（事前・予防）** | LSP、CLI、コードモッド | CLAUDE.md、Skills、アーキテクチャドキュメント |
| **センサー（事後・自己修正）** | linter、型チェック、テスト | AI コードレビュー、LLM-as-judge |

右上だけを埋めて満足すると、次の3つが抜けたままになります。lint や型チェックを実行ループそのものに組み込む計算的センサー。出力を別の仕組みで検証して自己修正させる検証ループ。効果をデータで示す評価と計測です。

> **CLAUDE.md は「必ず守る」仕組みではありません**
>
> CLAUDE.md は文脈を伝える層です。公式ドキュメントにも「指示が守られないとき」のトラブルシュート節があり、指示は具体的で簡潔なほど守られやすいと書かれています。裏を返すと、厳密な遵守は保証されません。必ず走らせたい判定は Hooks で書きます。本日の演習2の後半で、規約を CLAUDE.md に書いた場合と、Hook で機械判定した場合の差を見ます。

## 07.3 Lv1 / Lv2 / Lv3 の定義

| 段階 | 呼び方 | 環境の状態 | やること | 何を体で分かってもらうか |
|---|---|---|---|---|
| Lv1 | 用意されたハーネスを使う | 演習フォルダに CLAUDE.md が1枚、Skill が2本、docs が4種、`.claude/settings.json` が置かれています | `/init` を実行して読み込みを確認し、用意された Skill を呼びながら実装します | 前提が書いてあると指示が短くて済むこと。Skill は手順の型であって魔法ではないこと |
| Lv2 | プロジェクト側のハーネスが無い状態から作る | 演習フォルダに CLAUDE.md も docs も .claude もありません。午前に入れたユーザー設定は効いています | `/init` で CLAUDE.md を作り、ひな型からコーディング規約を置き、Hook を自然言語から作らせます | プロジェクトの前提が無いと毎回同じことを書かされること。1から書く手数と、ひな型を持ってくる手数の差 |
| Lv3 | 実用のパイプラインを組み合わせて回す | CLAUDE.md、`.claude/rules/` 2本、Skill 3本、Subagent 2本、Hook 1本、`permissions.deny`、docs 5種が揃っています | 用意されたパイプラインを使い分けて仕様から実装し、サブエージェントに下調べとレビューを任せ、Hook の指摘に応えます | 予防と強制と分業が同時に効くと、指示の量ではなく確認の質が変わること |

午後は演習1が Lv1、演習2が Lv2、演習3が Lv3 です。ミニ演習7本は、その間を埋める部品を1つずつ手で作る回になっています。

> **Lv3 が正解ではありません**
>
> Lv3 は作るのに手間がかかります。自社で始めるなら、まず Lv1 相当の CLAUDE.md を1枚置き、次に効いた規約を1つだけ Hook に上げる、という順です。最初から Lv3 の構成を目指すと、作りきる前に止まります。振り返りでもう一度この話をします。

## 07.4 それぞれの部品が何をする層か

| 部品 | 層 | 置き場所 |
| --- | --- | --- |
| CLAUDE.md | 予防。前提と規約を会話の開始時に読ませます。人が書く永続指示です | ~/.claude/CLAUDE.md（個人）、プロジェクト直下、サブフォルダ。作業場所に近い側が優先されます |
| auto memory | 予防。Claude が自分の気づきを書き溜めます。人が書くものではありません | 設定で有効・無効を切り替えます |
| docs | 予防。セキュリティ要件・生成AIガイドライン・コーディング規約・用語集・演習設定を、読ませたいときに読ませます | 各演習の docs/ |
| .claude/rules/ | 予防。ファイル種別ごとにルールを分けます。パス指定のルールも書けます | <プロジェクト>/.claude/rules/ |
| Skill | 予防。手順を型にして名前を付けます。/名前 で呼べます | ~/.claude/skills/<名前>/SKILL.md（個人）、<開いたフォルダ>/.claude/skills/（プロジェクト） |
| Subagent | 分業。独自のコンテキストウィンドウとツール権限を持ち、要約だけを返します。検索結果やログで本会話を埋めたくないときに使います | <プロジェクト>/.claude/agents/ |
| Hook | 強制。決められた時点でシェルコマンドや判定を必ず走らせます | ~/.claude/settings.json または <プロジェクト>/.claude/settings.json |

| 部品 | 層 | 置き場所 |
| --- | --- | --- |
| permissions | 強制。触らせたくない場所への操作を止めます | 同上 |

旧来のカスタムコマンド（.claude/commands/<名前>.md）は Skills に統合されました。.claude/commands/deploy.md と .claude/skills/deploy/SKILL.md はどちらも同じ /deploy を作り、既存の .claude/commands/ もそのまま動きます。

## 07.5 理解度チェック

### Q6

**CLAUDE.md にコーディング規約を書きました。それでも規約違反のコードが生成されることがあります。この章の整理で、次に足す層はどれですか。**

A CLAUDE.md の記述をより詳しく、長く書き直す

B Skill を増やして、手順をより細かく決める

C lint や規約チェックを実行ループに組み込む計算的センサー

D サブエージェントを増やして、作業を分業させる

**正解と解説を開く**

正解は C です。

CLAUDE.md と Skill は「事前・予防」かつ「推論的」の一角にあり、守られるかどうかはモデルの判断に委ねられます。必ず走らせたい判定は、Hooks から決定論的なチェックを呼ぶ

形にします。抜けやすいのは計算的センサー、検証ループ、評価と計測の3つで、本日の演習2の後半と演習3でその2つ目までを触ります。

SOURCES

- r.kagaya: ハーネスエンジニアリングの定義（Zenn、2026-04-23）
- Claude Code 公式ドキュメント: メモリ（CLAUDE.md と auto memory、.claude/rules/）
- Claude Code 公式ドキュメント: Skills
- Claude Code 公式ドキュメント: Hooks
- Claude Code 公式ドキュメント: サブエージェント
- Claude Code 公式ドキュメント: プラグインリファレンス（skills-directory plugins）

ORIENTATION 08

# 出典一覧

本文で挙げたURLを、用途別にまとめています。研修後に自分で確認するときの入口として使ってください。数値の確認時点はすべて 2026年7月30日です。

## 08.1 Claude Code の公式ドキュメント

- Overview（機能・対応環境・MCP）
- VSCode で使う（画面・起動口・ショートカット・CLI との違い）
- セットアップと動作要件（Windows はネイティブ、WSL2 不要）
- サインイン
- ネットワーク設定（許可ドメインとプロキシ）
- 権限モードと保護パス
- モデル設定とエイリアス
- コマンド一覧
- コンテキストウィンドウと自動コンパクト
- コスト
- ステータスライン
- メモリ（CLAUDE.md、auto memory、.claude/rules/）
- Skills
- Hooks
- サブエージェント
- プラグインリファレンス
- 週次ダイジェスト
- 変更履歴

## 08.2 Claude Platform とプラン

- モデル一覧と選択指針
- 料金（トークン単価・キャッシュ・Batch API）
- モデルの非推奨と引退
- API リリースノート

プランと価格

稼働状況

Anthropic ニュース

Claude Opus 5 の発表（2026-07-24）

## 08.3 トレンドの数値

Anthropic: 2026 Agentic Coding Trends Report

METR: 熟練開発者のランダム化比較試験

METR: 続報と実験設計の変更

Stack Overflow Developer Survey 2025: AI

Artificial Analysis: モデル一覧と1タスク単価

SWE-bench 公式リーダーボード

SWE-bench Pro 公開セット

llm-stats.com: SWE-bench Verified（自己申告値）

OpenAI: GPT-5.6 発表ページ

Terminal-Bench 2.1 のリリースノート

パーソル総合研究所: 生成AIによる業務削減時間

Anthropic: NEC との戦略的協業

GMOデザインワン: 全エンジニアへの Claude Code 導入

## 08.4 周辺ツールと規格

Visual Studio Marketplace: Claude Code for VS Code

Visual Studio Code リリースノート

GitHub: Copilot の使用量ベース課金への移行

GitHub Docs: Copilot のモデル別単価と AI Credits

GitHub Changelog: Kimi K2.7 Code の提供拡大

Model Context Protocol 仕様 2026-07-28

Anthropic: MCP 2026-07-28 の Claude への反映

r.kagaya: ハーネスエンジニアリングの定義