Givery, Inc.

ORIENTATION · 2026 年 5 月版

ClaudeCode研修オリエンテーション

PwC コンサルティング合同会社様向け（2 日間 × 2 回）

提供	Givery 株式会社 / 講師 安田 光喜
配布先	PwC コンサルティング合同会社 御中
所要	S01 オープニング 30 分 + S02 ベストプラクティス講義 90 分
題材	案件ヘルスチェックダッシュボード（FastAPI + React + TypeScript）
版数	v2.0（2026 年 5 月 3 日）

研修当日の S01〜S02（座学）で投影・配布される資料です。手順書はハンズオンガイド Day 1 / Day 2 を別途参照してください。

1. 本研修の位置付けと 2 日間ゴール

Anthropic のリサーチャー Sholto Douglas は 2026 年 2 月の Lenny's Newsletter で「Anthropic 内部のコードの 90〜95% を Claude Code が書いている」と語り、Boris Cherny は同じ場で「GitHub 上の全コミットの 4% が Claude Code 由来」と数字を出しました。研修当日いきなりターミナルを開く前に、本オリエンで「Claude Code とは何か／どこへ向かっているか／何を真似ればいいか」を 120 分で揃えます。

本研修の目的は使い方の暗記ではなく、自分の案件に持ち帰って **Claude Code をチームの基盤として組み込む判断軸を作ること**です。Boris 自身が公言している「verification loop で品質が 2-3x になる」「git worktrees で並列セッションが最大の生産性アンロック」を、本研修の案件ヘルスチェックダッシュボードで実機体験します。

形式　2 日間 × 2 回（計 4 日間）／ オンサイト

対象　Claude Code の基本操作経験あり、ベストプラクティスを深く学びたい SE

題材　案件ヘルスチェックダッシュボード（架空クライアント マツカゼ製薬の海外子会社向け CRM 刷新案件）

技術　FastAPI + SQLite + React 18 + TypeScript 5 + Vite + VSCode

講師　安田 光喜（Givery）

1.1 受講者の前提

- 業務で Claude Code を使った経験がある
- VSCode / Git / Python のいずれかは触ったことがある
- 自社チームのプロジェクト構造を、平均的なエンジニアと同程度に把握している

1.2 本日扱う / 扱わない

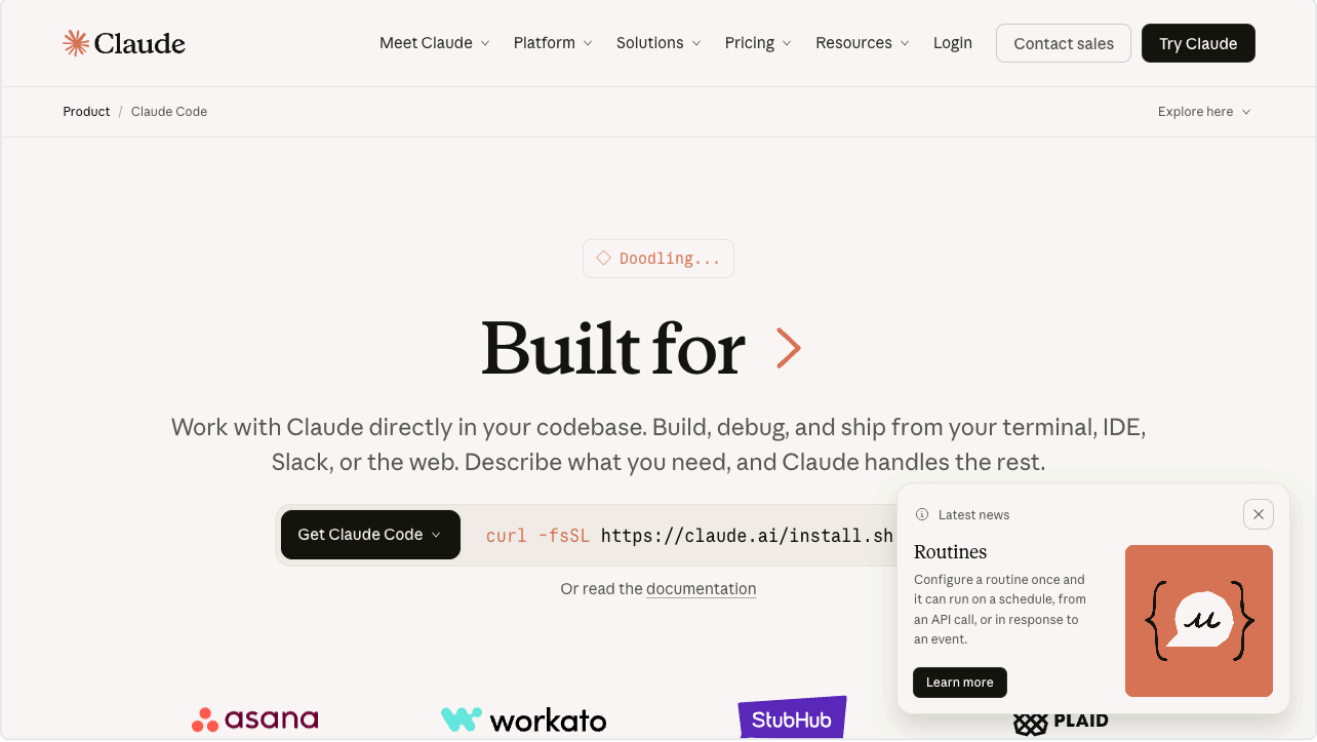
扱う	扱わない
CLAUDE.md / Skills / Subagents / Hooks / MCP の設計と使い分け、コスト管理、レビュー基盤化、自律開発	Claude Code の基礎操作（事前セットアップガイド参照）、LLM そのものの仕組み、他 LLM サービスとの比較

> **前提：架空案件で進めます**
>
> 題材は架空クライアント「株式会社マツカゼ製薬」の海外子会社向け統合 CRM 刷新案件（社内コード MTKZ-CRM-2026）です。実在の企業・人物とは無関係。実機密データは扱いません。

SECTION 02

2. Claude Code、Claude.ai、API、Claude Desktop — どれを使うか

「Claude」と一括りで呼ばれていますが、Anthropic が出しているのは別々の製品です。本研修中の「これは Claude.ai と何が違うのか」が混ざらないよう、最初に整理します。



Claude Code 公式トップ — claude.com/product/claude-code("Built for developers")

2.1 Anthropic の主要 4 プロダクト

プロダクト	形態	主な使いどころ	本研修での扱い
Claude Code	ターミナル CLI(`claude` コマンド)/ VSCode 拡張	プロジェクト全体を読ませて編集・実行・テストさせるエージェント。CLAUDE.md・Skills・Subagents・Hooks・MCP を組み合わせる	**主役**
Claude.ai	Web / モバイルアプリ	会話ベースのチャット。Projects に資料を常駐、Artifacts で右ペインに成果物を育てる。コード周辺の「設計・要件・調査」に強い	補助的に使用(仕様整理・調査)
Claude Desktop	Mac / Windows ネイティブ	Claude.ai のデスクトップ版。MCP サーバー接続のサンプル実装が最初に動いた場所。GitHub・Slack・社内 DB と直結したい時の入口	触れない(接続概念だけ Day 2 で)
Claude API	HTTP API(Messages API)	自社プロダクトに Claude を組み込む。本研修の受講者層が直接触ることは多くないが、Claude Code は内部でこの API を呼んでいる	触れない(仕組みだけ言及)

2.2 補完型ツール(Copilot / Cursor 等)との位置関係

現場では Copilot や Cursor がすでに動いている案件が多いはずです。Claude Code はそれを置き換えるのではなく、**「エディタ補完は Copilot / Cursor、プロジェクト全体を任せるエージェントは Claude Code」**という二層運用が標準形です。Boris Cherny も「Claude Code は一つの正解という運用はない」と公言しています。

補完型

GitHub Copilot

エディタ内インライン補完が中心。タイピング中に次の数行を提案するモード。コーディング速度を底上げするが、プロジェクト全体を読み解いて複数ファイルにまたがる変更を指示するのは苦手。

エージェント補完混在

Cursor / Windsurf

Copilot 的な補完に加えて、エディタ内チャット・複数ファイル編集を持つ。VS Code フォークが多く、エディタ中心のワークフロー。Anthropic / OpenAI どちらのモデルでも動く。

エージェント特化

Claude Code

ターミナル前提。プロジェクト全体を読み込み、複数ファイル編集・テスト実行・PR 作成までを 1 ターンの依頼で進める。CLAUDE.md・Skills・Subagents・Hooks の設計で挙動が大きく変わる。

本研修のスタンス

「Copilot をやめて Claude Code にする」ではなく、**Copilot は IDE 内の補完として残し、Claude Code を新しい武器として上に積む。**これが Anthropic 自身も社内で取っている構成です。本研修は Claude Code の使い切りに 2 日間 × 2 回を集中投下します。

SECTION 03

3. Claude Code の現在地 — 数字で見る

Claude Code が世に出たのは 2025 年 2 月。それから 1 年と少しで、Anthropic 自身が「コードの大半を Claude Code が書いている」と言える状態になりました。1 日 0.5 回のペースで仕様が動いている、と思って研修を受けてください。資料の動作と最新版の動作が違っても驚かないこと。

4%

GitHub 全コミット中の Claude Code 由来比率

Boris Cherny on Lenny's Newsletter (2026-02)

90-95%

Anthropic 社内コードのうち Claude Code が書いた比率

Mike Krieger (CPO), Lenny's Newsletter (2026-02)

176

2025 年内の Claude Code アップデート数

Anthropic Engineering (2025-12)

3.1 主要マイルストーン (公式 Changelog ベース)

2024-11-25

MCP (Model Context Protocol) 公開

外部ツール (Jira / GitHub / Slack / DB 等) と AI を繋ぐオープン標準として Anthropic が公開。Claude Desktop が最初の対応クライアント。Claude Code 単体の話ではないが、後の MCP 統合の起点になる。

出典:anthropic.com/news/model-context-protocol

2025-02-24

Claude Code 研究プレビュー公開 (Claude 3.7 Sonnet と同時)

ターミナルベースのエージェント環境として登場。CLAUDE.md でプロジェクトコンテキストを宣言する仕組みも初日から搭載。Boris Cherny いわく「Anthropic API を学ぶ最も安い方法は CLI を作ることだった」。

出典:anthropic.com/news/claude-3-7-sonnet

2025-05-22

Claude Code 一般提供 (Claude 4 と同時)

研究プレビューを経て GA。Opus 4 / Sonnet 4 と組み合わせ、長尺リファクタリングや複数ファイルにまたがる実装を任せられる水準に。楽天は Opus 4 で 7 時間連続自律リファクタリングをやらせる事例を公表。

出典:anthropic.com/news/claude-4

2025-08-12

Sonnet 4 が 1M トークンコンテキストに対応

関連ファイル・設計資料・ログを束ねて扱いやすくなった。同時に「context rot」が課題として顕在化し、Anthropic 自身が 9 月に "Effective context engineering for AI agents" を公開する流れに繋がる。

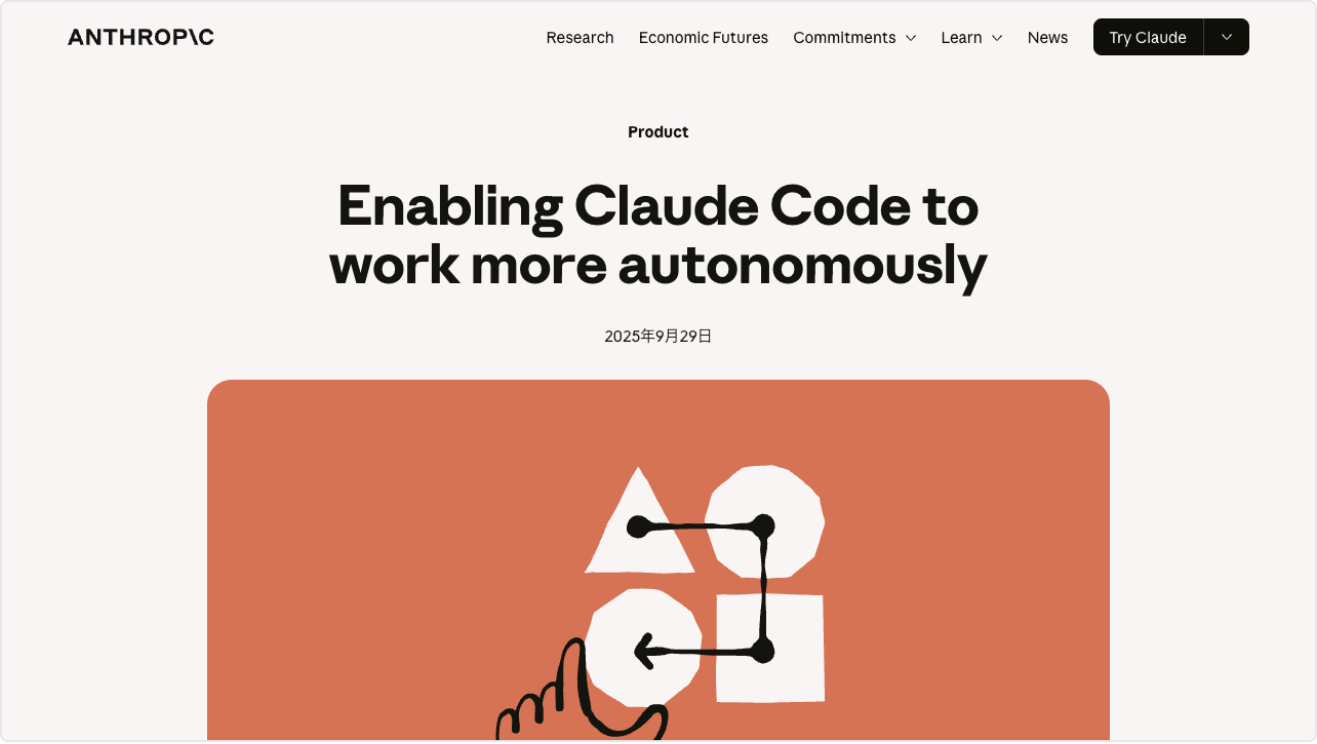
出典:anthropic.com/news/1m-context

2025-09-29

Claude Code 2.0:Subagents / Plan Mode / Hooks / Checkpoints (Sonnet 4.5 と同時)

本研修の主要演習で扱う機能群が揃った日。Subagent で文脈を切り分け、Plan Mode で先に計画を立てさせ、Hooks で自動実行ポイントを刺し、Checkpoint で安全に巻き戻す。"more autonomous" (より自律的) という Anthropic 公式の方向性が打ち出された。

出典:anthropic.com/news/enabling-claude-code-to-work-more-autonomously



Anthropic 公式ブログ "Enabling Claude Code to work more autonomously"(2025-09)— Subagents / Hooks / Plan Mode / Checkpoints の方向性が打ち出された

2025-10

Skills 機能(β)

指示・スクリプト・リソースをフォルダ単位で束ね、必要時に Claude が動的にロードする仕組み。PowerPoint / Excel / Word / PDF 用の公式 Skill が初日に出荷。本研修 Day 1 では risk-classifier / transcript-extractor 等の自社 Skill を自作する。

出典:anthropic.com/engineering

2026-04-16

Claude Opus 4.7 / xhigh effort / /ultrareview(v2.1.111)

最高難度のエージェントコーディングに最適化された Opus 4.7、思考予算をコマンドで持ち上げる "xhigh" effort、PR レビュー特化の `/ultrareview` サブコマンドを同時搭載。本研修開始の直前に出た最新版。

出典:code.claude.com/docs/en/changelog

2026-04-29

v2.1.123:Bedrock service tier、PR URL resume、MCP 改善

本研修当日に最も近いバージョン。AWS Bedrock 経由のサービスティア選択、PR URL からの会話再開、MCP の `alwaysLoad` オプションなどが入っている。

出典:code.claude.com/docs/en/changelog

今後の方針 — Anthropic と Boris の見立て

Anthropic は 2025 年 9 月のブログで Claude Code を *"more autonomous"*(より自律的)に向かわせると明言しました。Subagent を増やし、Hooks で自動化ポイントを増やし、Checkpoint で安全に巻き戻せるようにする方向です。Boris Cherny は Y Combinator のポッドキャスト(2026 年 2 月)で、もっと根本的なスタンスを語っています。**「At Anthropic, we don't build for the model of today, we build for the model of six months from now.」**(今日のモデル向けには作らない。半年後のモデル向けに作る。)今日の Claude Code を仕様書としてではなく、半年後にも崩れないように使う設計の練習として、本研修を受けてください。

SECTION 04

4. 設計思想 — 創業者 Boris Cherny は何を考えているか

Claude Code を作った Boris Cherny は Anthropic の Head of Claude Code（Eng Manager）で、X では @bcherny。インタビュー・X 投稿・公式 Blog で発言が多い人物です。本研修中の判断に効く 4 つの言葉を引用します。

> **Boris Cherny** @bcherny · Head of Claude Code
>
> *“There is no one correct way to use Claude Code: we intentionally build it in a way that you can use it, customize it, and hack it however you like. Each person on the Claude Code team uses it very differently.*
>
> 出典：@bcherny X 投稿（2025 年 12 月）／ チーム内ですら全員使い方が違う、を Anthropic 自身が公言。「これが正解です」を持ち帰らないでください。

> **Boris Cherny** @bcherny · Head of Claude Code
>
> *“My setup might be surprisingly vanilla! Claude Code works great out of the box, so I personally don't customize it much.*
>
> 出典：@bcherny X 投稿／ 創業者本人がほぼ素のまま使っている。研修受講後に「色々設定しないと動かない」と感じたら、自分の設定が増えすぎていないかを疑ってください。

> **Boris Cherny** @bcherny · Y Combinator Lightcone Podcast
>
> *“At Anthropic, we don't build for the model of today, we build for the model of six months from now.*
>
> 出典：Y Combinator Lightcone Podcast（2026 年 2 月）／ 半年後のモデルでも崩れないように、運用を組む。今日のプロンプト技法を暗記しても賞味期限が短い、という宣言。

> **Boris Cherny** @bcherny · "the most important tip"
>
> *“Give Claude a way to verify its work. If Claude has that feedback loop, it will 2-3x the quality of the final result.*
>
> 出典：Boris Cherny X 投稿／ Claude に作業させたら、その結果を Claude 自身がチェックできる仕掛け（テスト・lint・型チェック・PR コメントの自動レビュー）を必ず置く。Boris 自身が "the most important tip" と公言。

4.1 この思想がコードに落ちると、何が起きるか

Boris 自身が X や howborisusesclaudecode.com（2026 年 3 月）で公開している運用は、**「Claude Code を魔法ではなく、インフラとして扱う」**に集約されます。具体的には次の 4 点。

原則 1

Verification loop が品質を 2-3x にする

Claude に作業させたら、その結果を Claude 自身がチェックできる仕掛けを必ず置く。テスト、lint、型チェック、PR コメントの自動レビュー。Day 1 Exercise 4-5 で code-reviewer / security-auditor Subagent を作って、この loop を実装する。

原則 2

git worktrees で 3-5 並列セッション

1 つの作業 = 1 つの worktree = 1 つの Claude セッション。複数を並列で走らせ、各タスクの文脈を完全分離する。Boris の言葉では「the single biggest productivity unlock」。Day 2 で並列 Agent 操作として体験する。

原則 3

CLAUDE.md・Hooks・Permissions で固める

memory file（CLAUDE.md）でプロジェクト規約、Hooks で commit 前の自動レビュー、permission で危険コマンドをブロック。インフラを仕込んでから本人は素で使う。Day 1 Exercise 1 と Day 2 Exercise 3 で扱う。

原則 4

自動化ポイントを早めに刺す

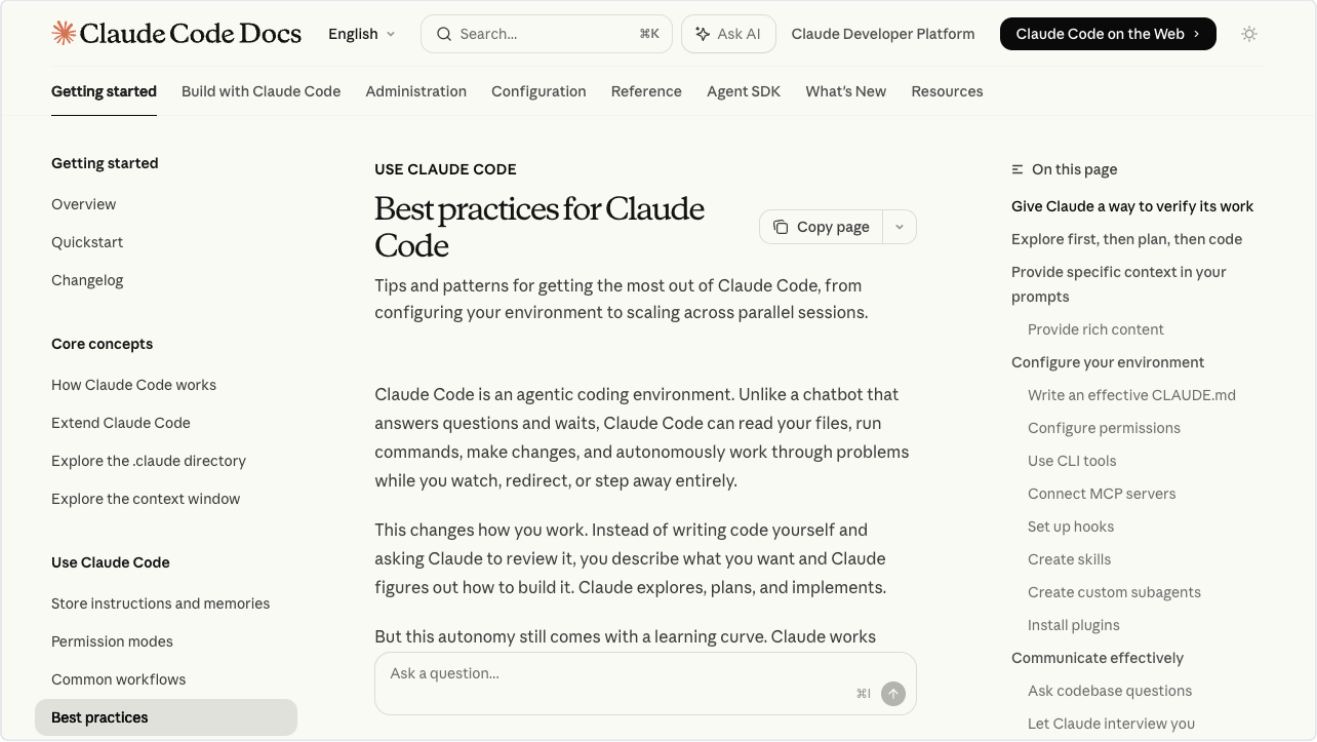
Hooks（PreToolUse / PostToolUse / Stop）と Cron で「Claude が自動で動く時間」を増やす。「コードを書く時間」より「Claude が回せる仕組みを作る時間」を優先する。Day 2 Exercise 3-4 で実装する。

出典：@bcherny の Claude Code Tips スレッド（2025 年 12 月）／ howborisusesclaudecode.com ／ Lenny's Newsletter "What happens after coding is solved"（2026 年 2 月）

SECTION 05

5. Anthropic 公式ベストプラクティス — 研修中に必ず体験する 5 つ

Anthropic は 2025 年 12 月に *"How Anthropic teams use Claude Code"*（公式 PDF）を公開しました。社内 10 部門が何をどう使っているかが書いてあり、共通項を 5 つに集約できます。本研修ハンズオンはこの 5 つを最低でも 1 回ずつ体験する設計です。



Claude Code Best Practices（公式 Docs） — code.claude.com/docs/en/best-practices

01 CLAUDE.md でプロジェクト前提を 1 ファイルに集約する

リポジトリ直下の `CLAUDE.md` に、コーディング規約・アーキテクチャ・ビルド手順・禁止事項を書く。Claude Code は起動時に自動読込するため、毎回プロンプトで前提を説明し直す必要がなくなる。**本研修 Day 1 Exercise 1** で「素朴プロンプト → CLAUDE.md → 仕様駆動」の 3 段階で生成品質の差を体験する。

02 Verification loop（検証ループ）を必ず置く

Claude に作業させたら、Claude 自身が検証できる仕掛け（テスト・lint・型チェック・PR コメントの自動レビュー）を必ず通す。Boris Cherny 自身が "the most important tip" と公言、品質が 2-3x になる。**Day 1 Exercise 4 では code-reviewer Subagent**、Exercise 5 では `/security-scan` Custom Command で verification loop を体験。

03 Skill / Custom Command で「自社の判断基準」を装着する

判断ロジックを Skill に切り出すと、AI への指示で毎回説明せずチームで再利用できる。決まった手順は Custom Command に。「素プロンプトで毎回説明する」を卒業する分岐点。**Day 1 Exercise 2 で risk-classifier / transcript-extractor Skill**、Exercise 3 で `/analyze-transcript` Custom Command を自作する。

04 Subagent で文脈を分離する（並列前提）

大きなタスクを役割ごとに切り出して Subagent に委任。各 Subagent は独立コンテキストで動くため、親会話の文脈を汚さない。Boris の言葉では「git worktrees + 並列セッションが最大の生産性アンロック」。**Day 2 Exercise 2 で並列 Agent 操作**、code-reviewer / security-auditor / risk-extractor の役割分担まで体験する。

05 Hooks / MCP は権限設計を先に決める

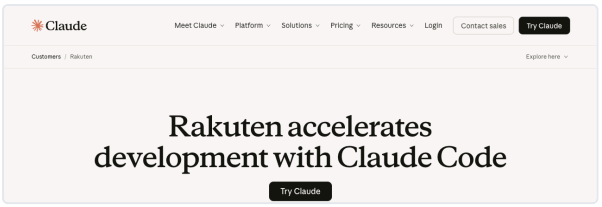
Hooks は commit 前や push 前のタイミングで自動処理を差し込む仕組み。MCP は外部システム（DB / GitHub / Slack）を AI から扱う標準。便利な反面、暴走させると本番事故になる。**Day 2 Exercise 3 で pre-commit Hook**、Exercise 5 で filesystem / sqlite MCP を自作し、必ず「読み取り専用 / 更新承認 / 監査ログ」の権限設計から入る。

出典：Claude Code Best Practices（公式 Docs code.claude.com/docs/en/best-practices）／ "How Anthropic teams use Claude Code"（公式 PDF, 2025 年 12 月）／ Effective context engineering for AI agents（Anthropic Engineering, 2025 年 9 月）

SECTION 06

6. 国内活用事例 — 「うちでやれそうか」の判断材料

日本企業での事例は 2025 年後半から急増しました。「全社導入したらどうなったか」「生産性は実際どれだけ上がったか」を、公開されている範囲で 8 件揃えています。研修後にチームへ持ち帰る話材として使ってください。



事例 1 ／ 2025

楽天グループ

7 時間 連続自律実行

vLLM の 1,250 万行コードベースに対し、Claude Opus 4 で活性化ベクトル抽出を 7 時間連続自律実装。99.9% の数値精度、人手介入は時折の指示のみ。「長尺タスクをエージェントに任せる」可能性を国内で示した最初期の事例。

出典：claude.com/customers/rakuten

事例 2 ／ 2025-06

ユニークビジョン

60% 超 が自発利用

早期導入企業の社内調査。「体感的に生産性が大きく向上した」という意見が多数。エンジニア側からの自発的な利用拡大が起きるパターン。費用補助制度を全エンジニア対象で開始。

出典：PR TIMES（uniquevision プレスリリース）

事例 3 ／ 2026-03

Gemcook

数日 で数万行のコード生成

2026 年 2 月の全社導入直後の社内記録。Cursor / Windsurf / Devin と比較検証して全社導入を決定。意思決定までのドキュメントが Zenn で公開されており、稟議資料の参考に使える。

出典：zenn.dev/gemcook

事例 4 ／ 2025-05

Box Japan — SDK 統合アプリを 30 分で完成

通常数日かかる Box SDK 統合のドキュメント生成アプリを、Claude Code と Box API の連携で約 30 分で完成。SaaS 提供企業側の活用事例として位置づけが明確。

出典：japan.box.com/blog

事例 5 ／ 2026

札幌の SIer 約 50 名 — 開発生産性 10 倍超

AI 非使用時との比較。Claude Code を活用した開発フローがチーム全体に定着。中堅 SIer 規模での「定着まで持っていけた」ケースとして参考になる。設計から開発までのフローに Claude Code を組み込む伴走支援の結果。

出典：firecracker.jp 企業導入事例まとめ

事例 6 ／ 2026-02

NRI 野村総合研究所 — 国内初 Anthropic 認定 Bedrock リセラー

設計・開発・コンサル業務に Claude Code を横展開、外販の導入支援サービスも整備。**PwC 様と同業（コンサル / SI）の事例として最も近い。**社内 Claude for Enterprise も導入。

出典：nri.com/jp/news/info/20260224_1.html

事例 7 ／ 2026-04

SCSK — AWS 既存 300 社へ展開計画

Anthropic リセラー契約を結び、AI 駆動システム開発の全社標準化を宣言。SCSK クラウドサービス利用顧客に対し Claude / Claude Code 再販と伴走支援を開始。エンタープライズ展開の参考。

出典：scsk.jp/news/2026 プレスリリース

参考 ／ ANTHROPIC 自身

社内 10 部門の活用 — 検索 80% 削減等

Security / Data Infrastructure / Growth / Legal 等の社内利用を公式公開。検索 1 時間 → 10〜20 分（80% 削減）、デバッグ 3 倍速、広告生成数百件を分単位で完了。Boris 公言「エンジニアはアーキテクチャ・プロダクト思考・複数エージェントの並列オーケストレーションへ仕事の重心を移している」。

出典：claude.com/blog/how-anthropic-teams-use-claude-code

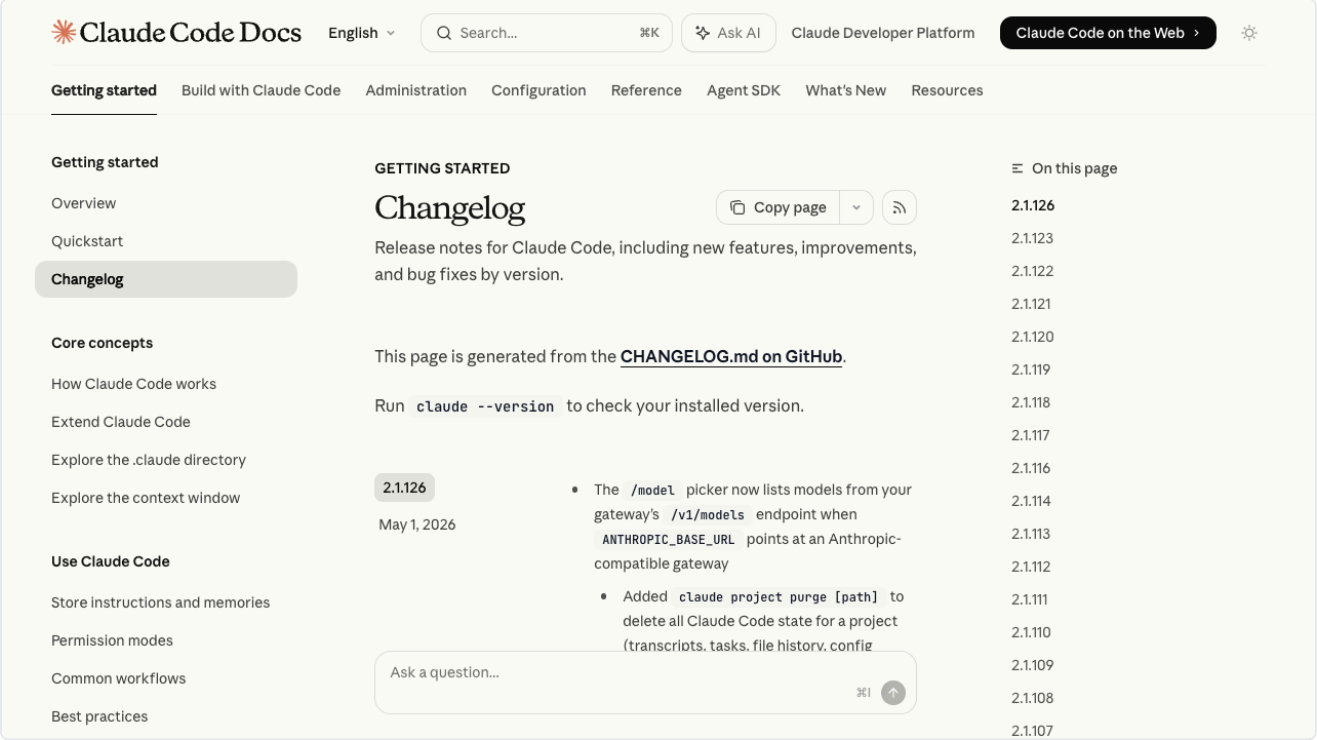
事例を見る時の注意点

「数倍の生産性」は測り方によって大きく変わる数字です。ベンチマーク前提（既存テストがあるか、対象が新規開発か保守か、レビューアの工数を含むか）を確認してから自社に当てはめてください。本研修 Day 1 Exercise 6 では、自チーム向けの効果測定 KPI 設計まで踏み込みます。

SECTION 07

7. 追うべき情報源 — 公式 5 つ・X 4 アカウント・深掘りメディア 3 つ

Claude Code は 1 日 0.5 回のペースで仕様が動きます。研修後に「最新情報をどこで追うか」を持ち帰っていただくため、講師（安田）が普段ウォッチしているソースを公開します。「とりあえずここだけ押さえる」12 個。



Claude Code Changelog — code.claude.com/docs/en/changelog（ほぼ毎日更新）

7.1 公式（5 つ）— ここが一次情報

公式 / 製品トップ

Claude Code 公式

機能の俯瞰と、最新メッセージング。新機能の名前を聞いたらまずここで概要をつかむ。

claude.com/product/claude-code

公式 / ドキュメント

Claude Code Docs

CLAUDE.md 仕様、Skills 仕様、MCP 設定、Hooks 設定、Subagents 仕様などすべて。トラブった時の最短ルート。

code.claude.com/docs

公式 / 更新履歴

Claude Code Changelog

ほぼ毎日更新。本研修当日のバージョンと教材作成時の差分をチェックするのに必須。

code.claude.com/docs/en/changelog

公式 / GITHUB

anthropics/claude-code

Issue で報告された不具合、Releases ページの詳細、コミュニティの workaround。CHANGELOG.md がここから読める。

github.com/anthropics/claude-code

公式 / 設計の解説

Anthropic Engineering Blog

"Effective context engineering"、"Building agents with the Claude Agent SDK" 等、設計思想の長文。Claude Code の使い方が変わる時の理屈はここに書かれる。

anthropic.com/engineering

公式 / ベストプラクティス

Claude Code Best Practices

Anthropic 自身がまとめた使いこなし。"How Anthropic teams use Claude Code" の PDF も併読する。

code.claude.com/docs/en/best-practices

7.2 X（Twitter）— ここでまず気づく

Anthropic 公式

モデル発表、大型機能、研究系の話題。

@AnthropicAI

Claude プロダクト公式

プロダクトの新機能、ティザー、社内 demo の流出元。

@claudeai

Boris Cherny

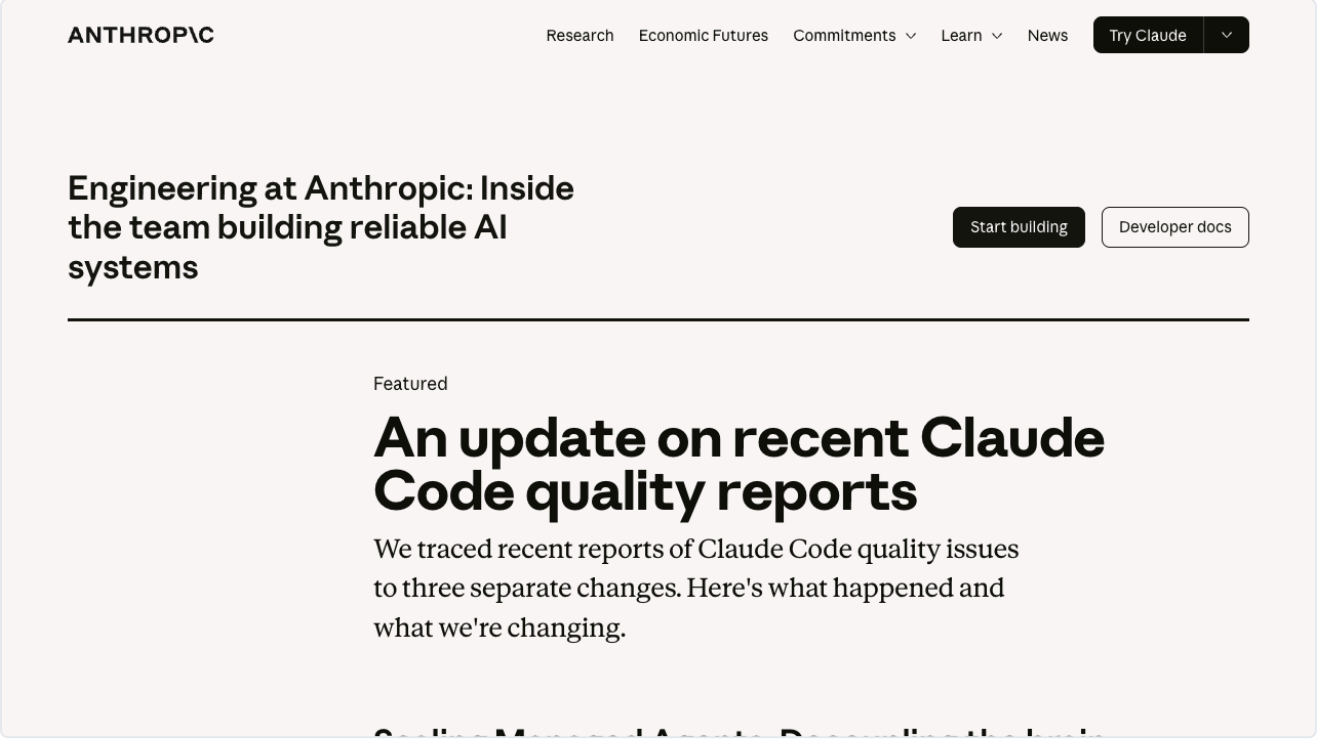
Claude Code の使い方 Tips を本人が定期的に投稿。「My setup is vanilla」シリーズが定番。

@bcherny

Cat Wu — PM

Claude Code の Product Lead。新機能のリリース予定や設計判断の解説。

@_catwu



Anthropic Engineering Blog — anthropic.com/engineering（設計思想の長文記事）

7.3 深掘りメディア（3 つ）— ここで設計思想がわかる

NEWSLETTER / PODCAST

Lenny's Newsletter — What happens after coding is solved

Boris Cherny の長尺インタビュー（2026 年 2 月）。GitHub 全コミット 4% 等の数字が出る。プロダクト原則や "coding is solved" 観の議論。

lennysnewsletter.com

NEWSLETTER / ENGINEERING

Pragmatic Engineer — Building Claude Code with Boris

Gergely Orosz による技術寄りインタビュー。CLI を選んだ理由、内部アーキテクチャの話が深い。

newsletter.pragmaticengineer.com

PODCAST

Y Combinator — Lightcone Podcast（Boris 回）

"We don't build for the model of today, we build for the model of six months from now." の出典。スタートアップ向けの設計指針として有名。

youtube / @ycombinator

7.4 コミュニティ（補助）

TIPS 集約

howborisusesclaudecode.com

Boris の使い方を 1 ページにまとめたサイト。Skill としても配布されており、コピペで真似できる構成。

変更ログ集約

claudelog.com

公式 Changelog をユーザー視点で再編集したサイト。週次でハイライトしてくれる。

SECTION 08

8. 本研修の 2 日間 — 何を作って、何を持ち帰るか

題材は架空案件「マツカゼ製薬 海外子会社 CRM 刷新（MTKZ-CRM-2026）」の **案件ヘルスチェックダッシュボード**。FastAPI + SQLite で API を作り、React + TypeScript + Vite で画面を出します。Claude Code への伝え方の練習に時間の大半を使います。

8.1 題材の構造

エンティティ	役割
Project（案件）	code / name / phase / status（GREEN/YELLOW/RED）/ owner / 開始終了日
Stakeholder（関係者）	name / role / organization / sentiment（POS/NEU/NEG）
StatusReport（週次レポート）	schedule / scope / cost / quality / team の 5 軸 status
Risk（リスク）	severity（CRITICAL/MAJOR/MINOR/INFO）× category（SCHEDULE/SCOPE/QUALITY/COST/TEAM/EXTERNAL）

初期データは Project 3 件、Stakeholder 12 件、StatusReport 18 件（直近 6 週分 × 3 案件）、Risk は手動登録 5 件のみ。残りは Day 1 のハンズオン中に `POST /api/projects/{id}/analyze` 経由でヒアリング起こしから抽出させる前提です。

8.2 Day 1 と Day 2 の構成

DAY 1 ・ 7 時間

機能を 1 つずつ投入しながら α を作る

同じ「PM 別 Risk 取得 API」を素朴プロンプト → CLAUDE.md → 仕様駆動の 3 段階で繰り返し、生成コードの品質差を可視化。続けて Skills / Subagents / Custom Command で再構築し、Verification loop（code-reviewer / security-auditor）を組み込む。最後の 120 分で自律開発演習。

- Exercise 1：CLAUDE.md（最小→規約追記）
- Exercise 2：Skills（risk-classifier / transcript-extractor）
- Exercise 3：Subagents + `/analyze-transcript` Command
- Exercise 4：code-reviewer Subagent
- Exercise 5：security-auditor Subagent
- Exercise 6：3 モデル比較（Haiku / Sonnet / Opus）
- Exercise 7：自律開発（120 分・候補 A/B/C 選択）

DAY 2 ・ 7 時間

運用に耐える形に仕上げ、別テーマで再演する

Day 1 の成果物に Hooks / Cron / MCP / DESIGN.md を載せ、運用面の磨き込み。後半 100 分は A: α 完成度向上 / B: 自社業務 / C: 別テーマ再挑戦の選択。最後に公式情報源探索でキャッチアップ習慣の素を作る。

- Exercise 1：Day 1 リフレクション
- Exercise 2：並列 Agent 操作
- Exercise 3：Hooks 自動化
- Exercise 4：Cron 定期実行
- Exercise 5：MCP 実践（filesystem / sqlite）
- Exercise 6：DESIGN.md + 仕上げ実装（A/B/C 選択）
- Exercise 7：成果共有 + 公式情報源探索

8.3 研修後に持ち帰っていただく成果物

- 育てた CLAUDE.md：プロジェクト規約、命名、設計方針、禁止事項を 1 枚に集約

- 3 つの Skill（risk-classifier / transcript-extractor / api-spec-checker）：判断ロジックを Markdown で再利用可能に
- 5 つの Custom Command：`/analyze-transcript`、`/add-endpoint`、`/review-changes`、`/security-scan`、`/weekly-report`
- 3 つの Subagent：code-reviewer / security-auditor / risk-extractor
- Hooks 設定案：commit 前自動レビューと、安全に巻き戻す手順
- MCP 接続候補と権限設計：filesystem / sqlite から始め、社内 DB / GitHub に拡張する手順
- 自社向け展開計画：1 か月 / 3 か月のフェーズ分けとアクション

SECTION 09

9. 主要キーワード — 研修中に出てくる 12 語

本編で都度説明はしますが、ここで一度通しておくと初日の入りが軽くなります。

Project Memory

CLAUDE.md

リポジトリ直下に置くプロジェクト宣言ファイル。規約・アーキテクチャ・ビルド手順・禁則事項を書くと、全指示の前提として読み込まれる。

Agent Skills

Skill

指示・スクリプト・リソースをフォルダ単位で束ね、必要時に動的ロードする仕組み。判断ロジックの再利用に使う。

Slash Command

Custom Command

`.claude/commands/` 配下に Markdown を置くだけで自分のプロジェクト専用コマンドになる。決まった手順の呼び出しに使う。

Sub-agent

Subagent

親エージェントとは別コンテキストで走る子タスク実行系。並列実行と文脈分離が両立する。

Lifecycle Hook

Hook

commit 前・ツール実行前後など特定タイミングで自動処理を差し込む仕組み。pre-commit レビュー等。

Plan Mode

Plan Mode

実装前に Claude が計画書を作るモード。大きなタスクで先に方針を確認するのに使う。

Checkpoint / Restore

Checkpoint

Claude Code が変更前の状態を保存しておき、コマンドで以前の状態に戻せる機能。

Model Context Protocol

MCP

外部ツール（DB / GitHub / Slack）と AI を繋ぐオープン標準。読み取り専用・更新承認・監査ログの設計が前提。

Verification Loop

検証ループ

Claude の出力を Claude 自身が検証する仕組み（テスト / lint / 型 / レビュー）。Boris 公言「品質が 2-3x」。

Claude Agent SDK

Agent SDK

Claude Code の中核を SDK として切り出したもの。自社プロダクトに同等の Agent を組み込む際の入口。

DESIGN.md

DESIGN.md

UI/UX の方針を集約するドキュメント。配色・余白・コンポーネント方針を 1 ファイルで管理。

AGENTS.md

AGENTS.md

Claude Code 以外（Copilot / Cursor / Codex）と共通のガイド。`@AGENTS.md` で CLAUDE.md から取り込む。

9.1 最低限覚えておくスラッシュコマンド

コマンド	用途
`/clear`	現在の会話をクリアして新しいセッションに。コンテキストが膨らみすぎた時に使う
`/resume`	前回の会話を復元。VS Code を一度閉じても続きから再開できる
`/context`	いま読み込まれているコンテキストを表示。何が前提として渡っているか可視化
`/cost`	この会話で使ったトークン量と概算コストを確認。プラン上限が気になる時に
`/agents`	Subagent の一覧と起動状態（Day 1 Exercise 3 で使用）
`/mcp`	MCP サーバーの接続状態確認（Day 2 Exercise 5 で使用）
`/model`	モデル切替（Haiku / Sonnet / Opus）。Day 1 Exercise 6 で使用

SECTION 10

10. PwC 様の環境前提 — 動かない時のデモ切替

Claude Code は社内ネット制約下でも CLI / VSCode 拡張ともに動作する想定です。MCP サーバー起動は npx / uvx 経由のため、初回はネット接続が必要、以降はキャッシュ利用。Cron や MCP は受講者環境で動かない場合があります。

演習	動作前提	NG 時の対応
Day 1 Exercise 4-5（code-reviewer / security-auditor）	VS Code + Claude Code が起動。社内中継基盤経由で OK	講師の画面共有でデモ切替
Day 2 Exercise 3（Hooks）	Bash（macOS / WSL2 / Git Bash）が動き、`chmod +x` できる	読み物として後追い、または Day 2 後半で講師同伴
Day 2 Exercise 4（Cron）	cron / launchd / タスクスケジューラの登録権限	NG 時は講師側で動かして見せる
Day 2 Exercise 5（MCP）	npx と uvx（または pipx）が外部レジストリへ通信可能	講師側で起動済み MCP に切替

セキュリティポリシー（吉田様・佐野様確認済み前提）

業務データを社外の生成 AI サービスに直接入力することは禁止。社内中継基盤経由が必須。本研修のハンズオン題材は架空案件のため、実機密データは扱いません。Day 2 終盤に「自社で持ち帰る時の入力可否ガイドライン」を全員で言語化します。

SECTION 11

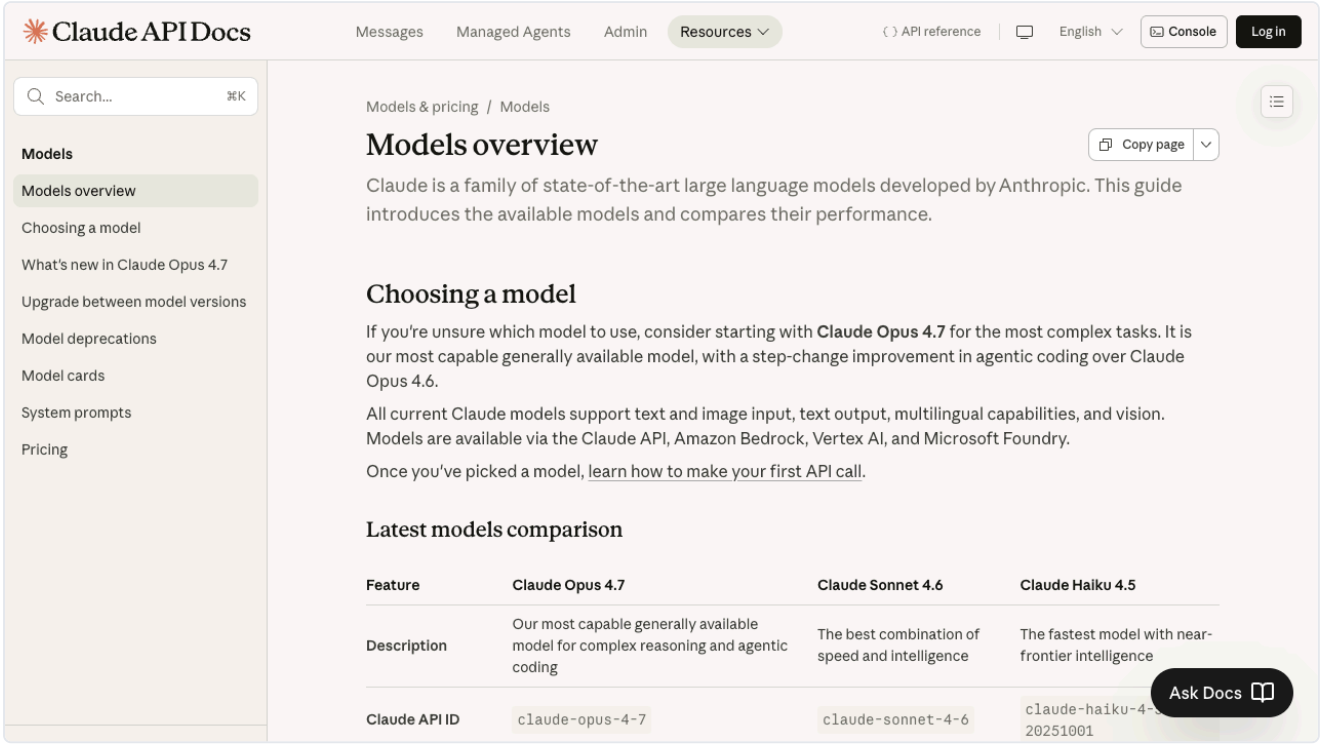
11. コスト管理 — トークン消費の負の側面

「コスト・トークン消費の負の側面を含めてほしい」という吉田様からのご要望に応える章です。研修中の各演習で実測値を記録し、Day 1 Exercise 6 で 3 モデル比較の体験を用意しています。

11.1 3 モデルの料金と使い分け（2026 年 5 月時点）

モデル	モデル ID	入力 / 出力料金（per MTok）	コンテキスト	使い分け
Haiku 4.5	`claude-haiku-4-5`	$1 / $5	200K	単純置換、フォーマット、CI の前段ふるい、要約
Sonnet 4.6	`claude-sonnet-4-6`	$3 / $15	1M	常用。コーディング、レビュー、分析
Opus 4.7	`claude-opus-4-7`	$5 / $25	1M	重要 PR レビュー、大規模リファクタ、複数ファイル横断（新 tokenizer +35%）

出典：platform.claude.com/docs/en/about-claude/pricing ／ anthropic.com/news/claude-opus-4-7



Claude Models Overview — platform.claude.com/docs/en/about-claude/models/overview

11.2 月次コスト試算とトークン消費の目安

- 1 セッション 50,000〜100,000 tokens：通常範囲
- 100,000 超：コンテキストの再構築を検討（`/clear` または `/compact`）
- 起こし全文の毎回投入は禁忌。要約版を渡す（Day 1 Exercise 3 で `/analyze-transcript` 経由化）
- Sonnet 常用 + Cron 定期実行 Haiku + 重要レビュー Opus で、業務利用は月数千円〜数万円のレンジ

注意：起こし全文を毎回投入しない

3 名のヒアリング起こし（合計 30,000 字）を毎ターン投入すると、1 セッションでトークン消費が爆発します。`/analyze-transcript` 経由で要約版を Risk テーブルに登録し、以降はテーブル照会で済ませる設計にします。input 4,000〜7,000 / output 1,500〜2,500 tokens に収まります。

11.3 上限設定と運用

- API キー単位での月次予算上限（Anthropic Console で設定可能）
- セッション単位でのトークン上限警告（`/cost` で都度確認）
- Subagent ごとのモデル固定（コスト予測性を上げる）
- 社内中継基盤経由の場合は基盤側の請求で管理

モデル選択は CLAUDE.md に書く

「日次のレビューは Sonnet、CI の前段は Haiku、重要 PR は Opus」という運用ルールを CLAUDE.md に書いておくと、チーム全員が同じ判断軸で運用できます。Day 1 Exercise 6 で実装。

SECTION 12

12. 詰まったらどう動くか — 3 手

「動かない」が起きるのは前提として研修を組んでいます。当日詰まった時の動き方を 3 段階で整理します。Boris 自身が言うように、Claude Code は「魔法ではなくインフラ」なので、問題が起きたら因果を追います。

第 1 手

エラー全文を Claude Code に読ませる

ターミナルのエラーメッセージをコピペして Claude Code に渡し、「このエラーの原因と修正方針を 3 案出して」と聞く。スクリーンショットでも可（画像読込対応）。複数案を出させてから自分で選ぶのがコツ。

第 2 手

公式 Docs / Changelog をその場で確認

仕様の認識違いが原因のことが多い。`code.claude.com/docs` と changelog を開く。教材作成日（2026 年 5 月）と当日のバージョン差で挙動が違うこともある。

第 3 手

講師に聞く、隣の人と画面共有する

5 分以上手が止まったら**必ず手を挙げて**講師（安田）に質問してください。研修時間が押すより、詰まりが連鎖する方が損失が大きい。研修当日は別途チャットも開いておきます。

SECTION 13

13. 研修後 3 か月のロードマップ案

研修後にキャッチアップが続くかどうかは、習慣の設計次第です。「個人で完結させない」「最低 2 人以上で同じ `.claude/` を運用する」が肝心。

直後〜 1 週間

個人で動かす最小版を作る

- 研修中に書いた CLAUDE.md / Skills / Subagents を社内チームに共有
- 業務リポジトリ 1 つに `.claude/` を導入（最小版で OK）
- Day 2 で確定した公式情報源 3 つに最初のアクセス

1 か月以内

チーム内で検証する

- 業務リポジトリで Hooks（PreToolUse: 危険コマンド停止）を導入
- code-reviewer Subagent をチーム全員に配布、1 か月分の PR でテスト
- 月 1 Changelog 読み（30 分）をカレンダーに登録

3 か月以内

標準作業フローに組み込む

- 2〜3 リポジトリに横展開
- Cron 自動レポートの本番稼働
- MCP（filesystem / git）を業務開発フローに統合
- 受講者間で振り返りミーティング 1 回

個人ではなくチーム展開を意識

個人で完結させると効果が限定的です。最低 2 人以上で同じ `.claude/` を運用し、レビュー会で気づきを共有する形が、業務効果が最も大きい。Day 1 Exercise 6 では、自チーム向けの効果測定 KPI 設計まで踏み込みます（KPI 設計は研修後の意思決定材料になります）。

補足：本研修のお手本リポジトリ

`pwc-cc-handson_完成形/` に、本研修で学ぶ要素が一通り組み込まれた完成形が同梱されています。Day 1 から順に育てていく前に概要を眺めるのは構いませんが、各演習を解く前に該当ファイルを開くと学習効果が下がるので、答え合わせのフェーズで開いてください。

> **ハンズオンガイド Day 1 / Day 2 へ**
>
> Day 1〜Day 2 の全演習、配布プロジェクトの構造、API 仕様、トラブルシューティング、コマンドリファレンスは別冊「ハンズオンガイド Day 1 / Day 2」に収まっています。各演習に「自分で考えるポイント」を置いてあるので、Claude Code の出力を受け流さず、判断を都度言語化してから次へ進んでください。

参考リンク

1. Claude Code 公式：claude.com/product/claude-code
2. Claude Code Docs：code.claude.com/docs
3. Claude Code Changelog：code.claude.com/docs/en/changelog
4. Claude Code Best Practices：code.claude.com/docs/en/best-practices
5. How Anthropic teams use Claude Code（公式 PDF）：claude.com/blog/how-anthropic-teams-use-claude-code
6. Anthropic Engineering Blog：anthropic.com/engineering
7. Effective context engineering for AI agents：anthropic.com/engineering
8. Enabling Claude Code to work more autonomously：anthropic.com/news/enabling-claude-code-to-work-more-autonomously
9. Claude モデル一覧（Haiku 4.5 / Sonnet 4.6 / Opus 4.7）：platform.claude.com/docs/en/about-claude/models/overview
10. Claude API Pricing：platform.claude.com/docs/en/about-claude/pricing
11. Claude Code 認証（Pro / Max サブスクリプション利用時のサインイン）：claude.ai/login
12. anthropics/claude-code（GitHub）：github.com/anthropics/claude-code
13. Model Context Protocol — Reference Servers：github.com/modelcontextprotocol/servers
14. Anthropic Cookbook：github.com/anthropics/antropic-cookbook
15. Lenny's Newsletter — What happens after coding is solved (2026-02)：lennysnewsletter.com
16. Pragmatic Engineer — Building Claude Code with Boris：newsletter.pragmaticengineer.com
17. Y Combinator Lightcone Podcast（Boris Cherny 回, 2026-02）
18. How Boris Uses Claude Code：howborisusesclaudecode.com
19. 楽天 Customer Story：claude.com/customers/rakuten
20. NRI Anthropic 拡大：nri.com/jp/news/info/20260224_1.html