Givery, Inc.

SEMINAR LECTURE · 2026 年 5 月版

エージェンティックエンジニアリング

パーソルクロステクノロジー株式会社様 CA企画 ／ 技術トレンド解説 第 1 回

提供	Givery 株式会社
配布先	パーソルクロステクノロジー株式会社 御中
日時	2026 年 5 月 21 日(木) 19:00–21:00
形式	ZOOM オンライン／定員 500 名
題材	log-review-agent — Claude Code でログ解析エージェント
版数	v2.0（2026-05-09）

自律的に動く AI エージェントを設計・構築・運用するための考え方と技術を、概念整理 60 分とハンズオン 45 分で扱います。本書は座学パートの配布用資料で、セミナー後の読み返しに使ってください。

SECTION 01

1. 本セミナーの位置付けと到達点

エージェントの周辺には似た言葉が増えすぎました。プロンプトエンジニアリング、コンテキストエンジニアリング、ハーネス、オーケストレーション、MCP、Hooks、Skills、サブエージェント。整理せずに実装に進むと、設計のたびに迷います。本セミナーは概念地図を 30 分で揃え、残り 70 分を Before デモとハンズオンに使います。

シリーズ位置付け	技術トレンド解説 AI シリーズ第 1 回
前回	2026 年 2 月『生成 AI 最前線』(概況とユースケース)
今回の踏み込み	AI エージェントの設計・実装・制御
本日の到達点	概念地図を持って実装まで行く

1.1 受講者の前提

- 業務でコードを書くエンジニア(IT 派遣スタッフ含む)
- LLM API 利用経験がある、またはエージェント設計に関心がある
- VS Code と Claude Code(または GitHub Copilot)が動かせる
- ターミナルの基本操作経験がある

1.2 今夜の流れ

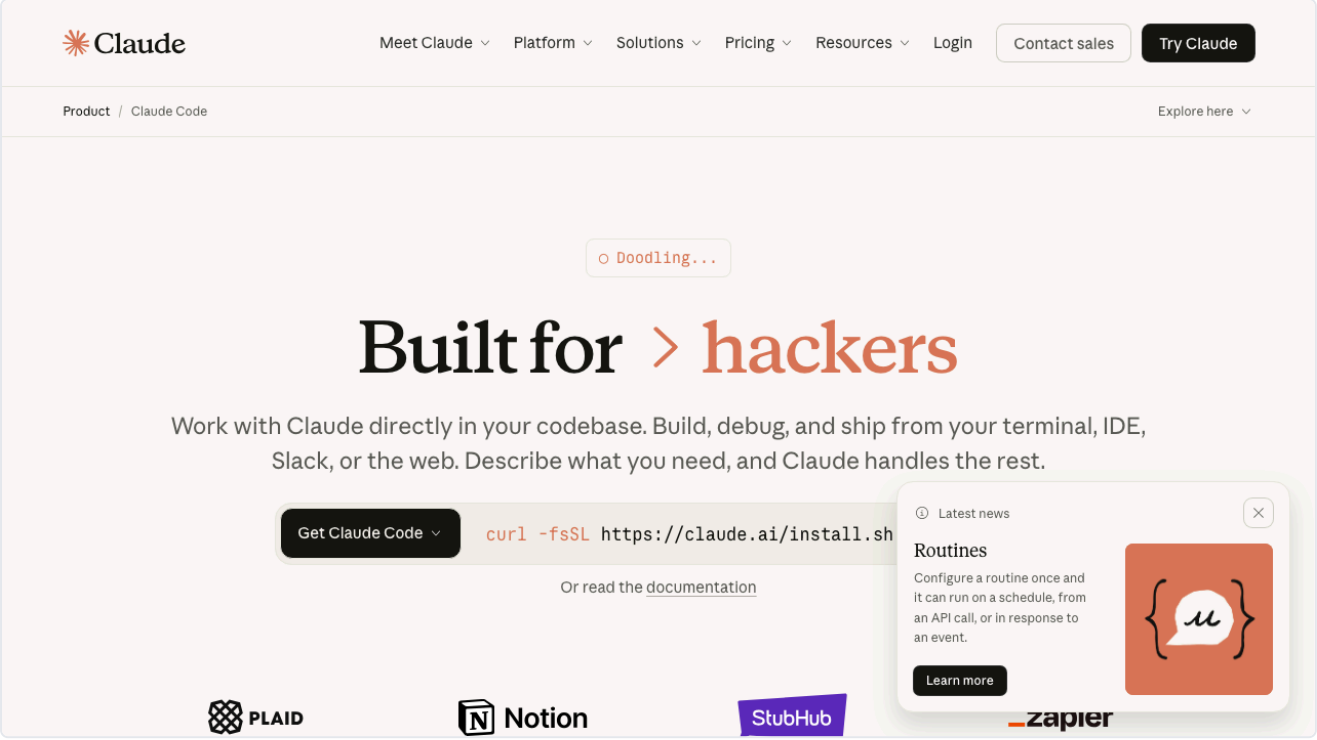
時刻	枠	内容	形式
19:00–19:10	[10min]	オープニング、定義と全体地図	座学
19:10–19:40	[30min]	概念整理、相違点と関係	座学
19:40–19:55	[15min]	Before デモ、素のエージェントの限界	講師ライブデモ
19:55–20:40	[45min]	ハンズオン、Hooks / Skills / サブエージェント	受講者作業
20:40–21:00	[20min]	振り返り、次回予告、Q&A	座学

> **この資料の使い方**
> セミナー中はスライド投影を見ながら聞いてください。本書はセミナー後に手元で読み返すための配布資料として作っています。出典 URL を本文と末尾の双方に置いています。

SECTION 02

2. エージェンティックエンジニアリングとは — 数字で見る現在地

「エージェンティックエンジニアリング」は、自律的に動く AI エージェントを設計・構築・運用するための工学のまとまりです。プロンプト 1 発で結果を引き出す時代から、ツールを呼び、状態を保ち、長時間走り続けるシステムを作る時代に移りました。生産性インパクトは、もう「期待値」ではなく「実測値」のフェーズに入っています。



Claude Code 公式トップ — claude.com/product/claude-code。本セミナー題材のハンズオンもこの CLI 上で動かす

2 日 → 5 時間
マネーフォワード
API 実装工数の削減
出典：Anthropic Customer Story / マネーフォワード（2026-01）

80%
マネーフォワード
コードのうち Claude 生成比率
出典：マネーフォワード公式プレスリリース（2026-01）

7 時間
楽天グループ
連続自律実行（精度 99.9%）
出典：claude.com/customers/rakuten

2025-09-29
Anthropic がコンテキストエンジニアリング公式ガイドを公開
出典：Anthropic Engineering Blog

2025-11-25
MCP 仕様の最新スナップショット日付
出典：modelcontextprotocol.io

2.1.x
Claude Code 安定版（async Hooks と Skills 正式提供）
出典：Claude Code Changelog

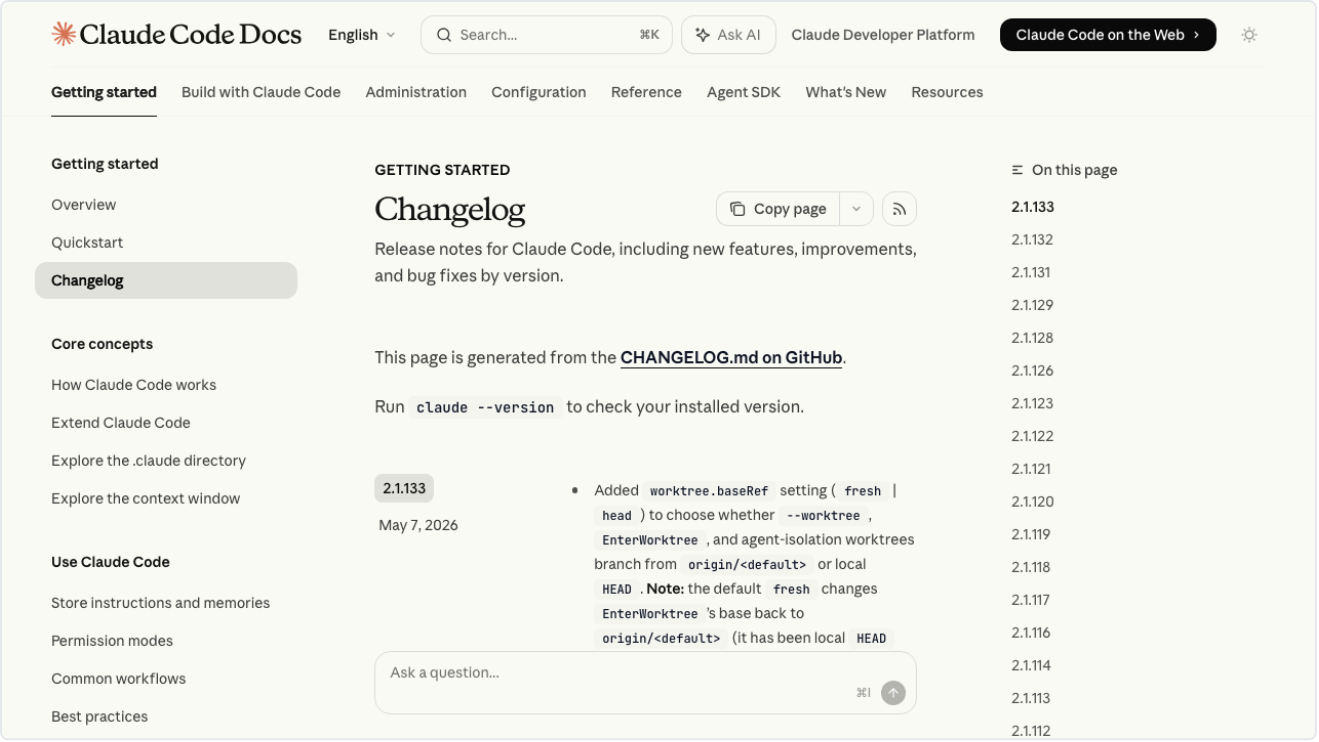
「2 日 → 5 時間」の意味

マネーフォワードのフィンテック開発現場で、API 1 本を実装する工数が 70% 削減されています。コードの 80% は Claude 生成、エンジニアの 80% が日常的に利用、1 人あたり週 7 時間の業務削減。これらの数字が公開されているのは、派遣現場で「自社にも持ち帰って試せそう」を検討する材料として珍しいレベルです。

SECTION 03

3. マイルストーン — Claude Code 1 年と少しの履歴

Claude Code が世に出たのは 2025 年 2 月。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。



Claude Code 公式 Changelog — ほぼ毎日更新される。本セミナー資料作成日と当日のバージョン差分はここで確認 — code.claude.com/docs/en/changelog

2022-10

ReAct（Yao et al.）

Thought → Action → Observation を交互に挟むパターンが論文化。エージェント設計の基礎語彙が揃う。

出典：arxiv 2210.03629

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。

出典：modelcontextprotocol.io

2025-02-24

Claude Code 研究プレビュー公開

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。

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。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。楽天は同年に Claude Code で 7 時間連続自律リファクタリングを公表（vLLM 1,250 万行コードベース）。

出典：anthropic.com/news/claude-4

2025-06

マルチエージェント研究システム公開

Anthropic Research が Orchestrator-Worker の事例を公開。lead がクエリを分解、subagent が並列で探索する設計が世に出る。

出典：anthropic.com

2025-09-29

Claude Code 2.0：Subagents / Plan Mode / Hooks / Checkpoints

本セミナーのハンズオン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" の方向性が打ち出された。

出典：anthropic.com

2025-09-29

コンテキストエンジニアリング公式化

プロンプト最適化を超えて、推論中の最適トークン集合を整える戦略として体系化。同日 Sonnet 4.5 が出たため、エージェント設計の語彙が一気に整う。

出典：anthropic.com

2025-10

Skills 機能の公開

SKILL.md と関連リソースを 1 フォルダにまとめて、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。

出典：anthropic.com

2025-11-25

MCP 仕様アップデート

日付ベースで版数管理に。Sampling / Roots / Elicitation などクライアント側機能が拡張、エンタープライズ向けゲートウェイ要件が固まる。

出典：modelcontextprotocol.io/specification

2026-04-28

Async Hooks 追加（v2.1.121）

PostToolUse / PostToolUseFailure が非同期実行に対応、duration_ms フィールドでツール実行時間を取得可能に。HTTP Hooks と組み合わせて集中監査・ポリシー検証ができるようになる。

出典：Claude Code Hooks

2026-04

Managed Agents 一般提供

エージェントロジックと、サンドボックスや状態管理などの実行関心事を分離するレイヤが GA。「ハーネスエンジニアリング」の運用版。

出典：InfoQ 2026/04

2026-05

本セミナー実施

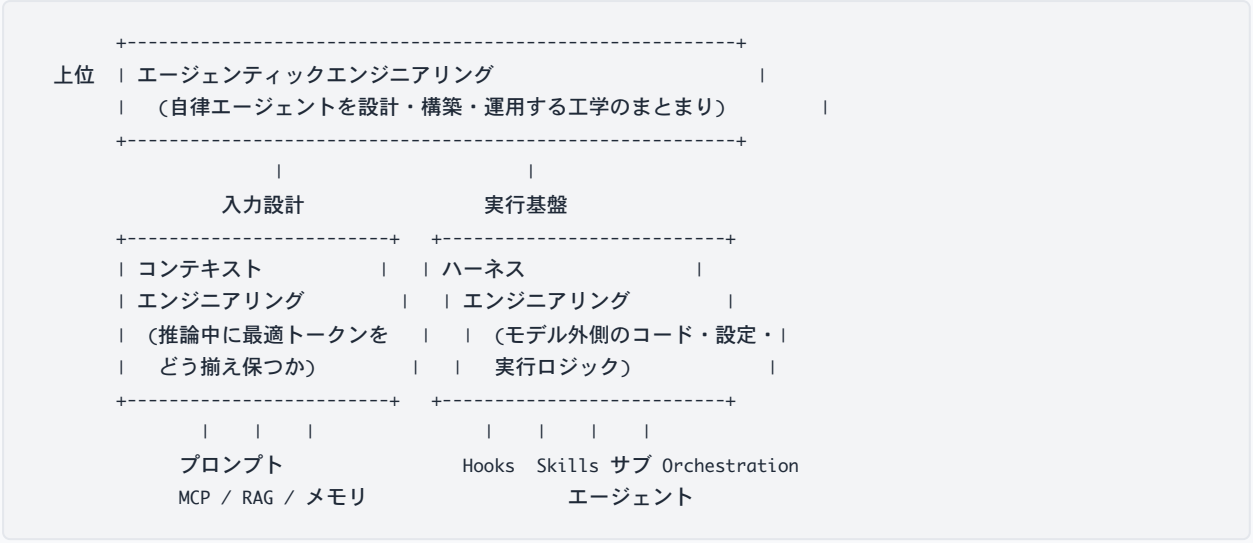
Claude Code 2.1 系で Hooks / Skills / サブエージェントが揃った状態。「概念整理 + 実装」の入口として最適なタイミング。

出典：本資料

SECTION 04

4. 概念地図 — 八つの言葉を一枚で

この章だけ持ち帰って机に貼っておけば、設計の議論で迷子にならない、を目指します。八つの言葉を「同じ地平にあるもの」と「層が違うもの」に分けます。



4.1 上下関係

エージェンティックエンジニアリングが屋根、その下に「入力設計（コンテキストエンジニアリング）」と「実行基盤の設計（ハーネスエンジニアリング）」の二本柱、さらにその下に具体技術が並ぶ、という階層で見ると整理できます。

入力設計の系

モデルに何を渡すかを決める層。プロンプトエンジニアリングが単発・静的な文字列の最適化を担い、コンテキストエンジニアリングはマルチターン・動的・全方位で「推論中のトークン全体」を扱います。MCP・RAG・ツールユース・メモリ管理は、ここに「外から流入する情報」として接続します。

実行基盤の系

モデルをどう走らせるかを決める層。Hooks がイベント駆動の強制実行、Skills が能動参照される手順書、サブエージェントが独立コンテキストの作業ユニット、オーケストレーションが複数ユニットの調停を担います。Anthropic はこの層を「ハーネス」と呼んで体系化しました。

> **覚え方**
>
> 入力設計の系は「何を入れるか」、実行基盤の系は「どう走らせるか」。設計判断で迷ったら、まず自分がどちらの系の話をしているかを切り分けると、議論が噛み合います。

SECTION 05

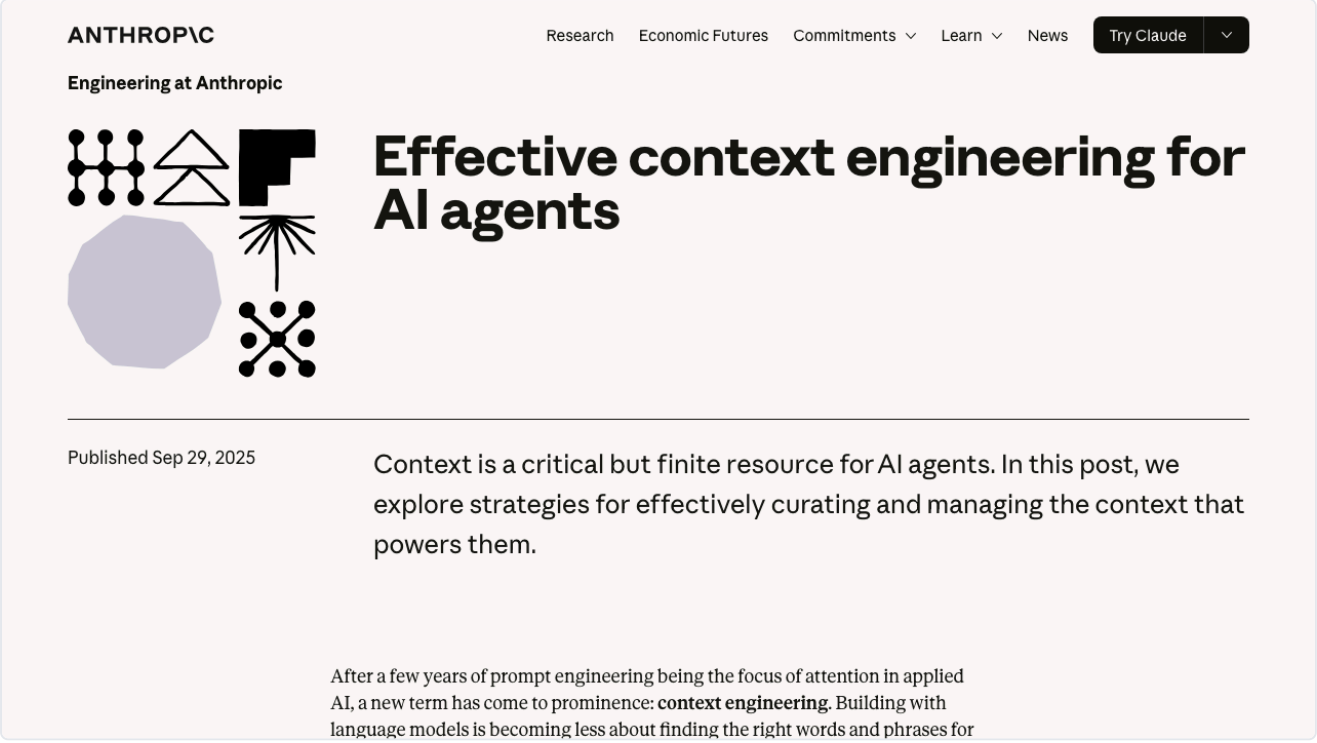
5. プロンプトエンジニアリング と コンテキストエンジニアリング

プロンプトエンジニアリングを軽く見るのは早計です。ただ、エージェント時代の中心はコンテキストエンジニアリングに移っています。違いを言葉で押さえます。

観点	プロンプトエンジニアリング	コンテキストエンジニアリング
対象	システム指示や few-shot 例など、入力テキスト本体	推論中に LLM へ到達するすべての情報。プロンプトに加え、ツール出力、MCP リソース、サブエージェントの返答、過去履歴も含む
時間軸	単発・静的	マルチターン・動的
指針	明示的指示、例示、制約、出力フォーマット	望ましい結果を最大化する、最小限の高シグナルトークン集合を維持する
主な失敗	曖昧、長すぎる、例が偏る	古い情報の混入、ノイズ蓄積、コンテキスト爆発、関連性の薄い大量出力
主担当ツール	System prompt、テンプレート、xml タグ	サブエージェントによる隔離、Skills による参照、MCP による外部リソース化、要約と圧縮

公式定義

Anthropic は「LLM 推論中に最適なトークン集合を整え維持する戦略の集合体」と定義しました。プロンプトに到達する情報以外も含むので、設計は単発の文字列ではなく時系列のフローを設計する仕事になります。出典：Anthropic Engineering Blog



Anthropic Engineering Blog "Effective context engineering for AI agents"（2025-09-29）。コンテキストエンジニアリングを公式に体系化した記事

5.1 実務での着地

毎ターンの推論で「いまの目的に対して、本当に必要なトークンだけが入っているか」を問う癖をつけます。たとえばログ全文を投げ込むのは典型的なアンチパターン。サブエージェントに読ませて要約だけ返させる、Skill にして必要時に参照する、MCP リソースとして遅延読み込みする、のいずれかに置き換えます。Before デモで講師がやって見せます。

SECTION 06

6. RAG・ツールユース・MCP の関係

三者は混ざりやすい。「外の情報をモデルに繋ぐ」点では同じだが、層が違います。

RAG

事前に作ったベクトルインデックスから関連チャンクを検索し、プロンプトに差し込む方式。検索結果はプロンプトに「混ぜる」イメージ。コンテキストエンジニアリングから見れば「入力に流す素材を取り出す手段」のひとつ。

ツールユース

モデルが関数呼び出しを通じて、検索・計算・ファイル操作を能動的に行う方式。RAG の検索もツールにできる。ツール呼び出しの結果は「Action の Observation」としてコンテキストに戻る。

MCP

ツールやリソースを「アプリ間で標準化されたプロトコル」で繋ぐ仕組み。同じ MCP サーバーを Claude / Cursor / VS Code / 各種クライアントから差し替えなく使える。バージョンは 2025-11-25 が最新。

6.1 結局どう選ぶか

専有データを毎日新しく検索したいなら RAG、その日その時の動的な操作（コード実行、API 呼び出し、ファイル整理）が要るならツールユース、複数のクライアントから同じ機能を再利用したいなら MCP、と考えると迷わない。Claude Code は MCP を一級品として扱うので、社内のデータ基盤を MCP サーバーとして整備する戦略はコスパが高い。

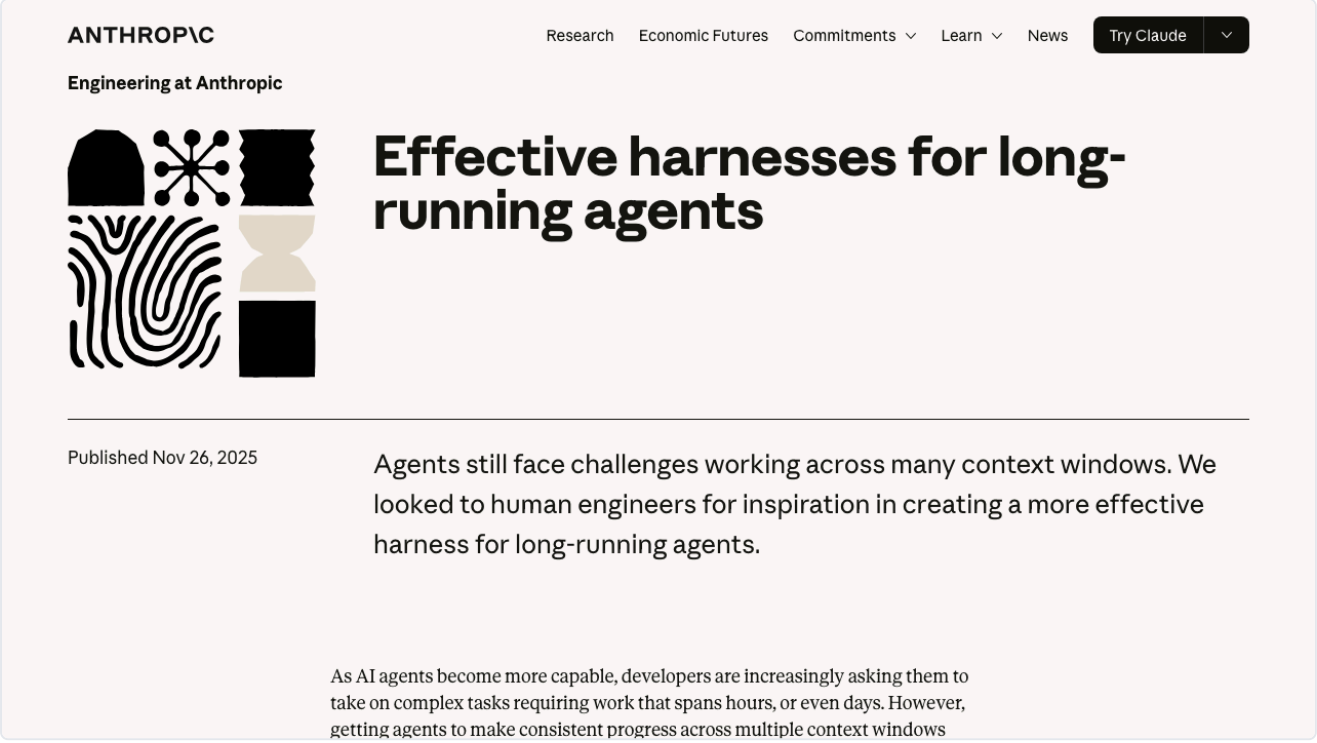
> **三者は排他ではない**
> MCP サーバーが内部で RAG をやるのは普通です。ツールユースから RAG を呼ぶのも普通です。階層を意識しつつ、混ぜて使うのが現実解です。

SECTION 07

7. ハーネスエンジニアリング と オーケストレーション

ハーネス、聞き慣れない言葉ですが Anthropic の用法は明確です。「モデル本体ではない、コード・設定・実行ロジックすべて」を指します。これが本番 AI 開発の主戦場になりつつあります。

> 「本番 AI の課題はモデルではなくハーネスにある。モデルが知能を提供し、ハーネスが制御を提供する。」
>
> 出典：Anthropic Engineering Blog, Effective harnesses for long-running agents



Anthropic Engineering Blog "Effective harnesses for long-running agents"。本番 AI の主戦場が「ハーネス」であることを公式に体系化した記事

7.1 ハーネスを構成する要素

Hooks

ライフサイクルの特定地点で発火するイベント駆動の強制実行。PreToolUse / PostToolUse / Stop など。exit code 2 でツール呼び出しを止められる。

Skills

SKILL.md と関連リソースを 1 フォルダにまとめた、Claude が必要時に取り出す手順書。CLAUDE.md を巨大化させずに済む。

サブエージェント

独立コンテキスト・独自 system prompt・固有ツール権限を持つ作業ユニット。.claude/agents/ 配下に置く。大量出力をメインに戻さずに処理する。

オーケストレーション

サブエージェントを並列に走らせるパターン。Lead がクエリを分解し、specialized subagents が独立に探索、結果を集約。Anthropic Research の標準形。

7.2 プランニング、ReAct と Plan-and-Execute

観点	ReAct	Plan-and-Execute
順序	Thought → Action → Observation を 1 ステップずつ織り交ぜる	Planner が事前にステップ列を生成 → Executor が順次実行
適応性	観測のたびに次を決められる、応答性が高い	計画外の事象に弱い。リプランが要る
俯瞰性	全体像が見えにくく、局所最適に陥りやすい	事前の俯瞰がある、コスト効率と予測可能性で勝つ
論文	Yao et al. 2022（arxiv 2210.03629）	Plan-and-Act 系（arxiv 2503.09572 など）

Claude Code の長時間タスクは両者の折衷で動きます。内部で計画 → ReAct 的に実行 → 定期的にリプラン。短いタスクは ReAct のまま走らせる方が速い。

7.3 メモリ管理

2025〜2026 年でメモリは学術周縁から産業中核に移りました。3 層構成が定着しています。

短
Short-term Memory。現在のコンテキスト内の即時ワークスペース。容量はモデル依存。
話
Episodic Memory。会話・タスク軌跡の時系列保持。「いつ何が起きたか」の物語的文脈。
事
Semantic Memory。生経験から蒸留された要約・事実・ユーザープロファイル。
2026 年の Agentic Memory（AgeMem）系では、メモリ操作自体をツールとして公開し、エージェントが「いつ保存・取得・更新・破棄するか」を自律判断する設計が増えています。出典：arxiv 2601.01885

SECTION 08

8. Hooks・Skills・サブエージェント — 構造で見分ける

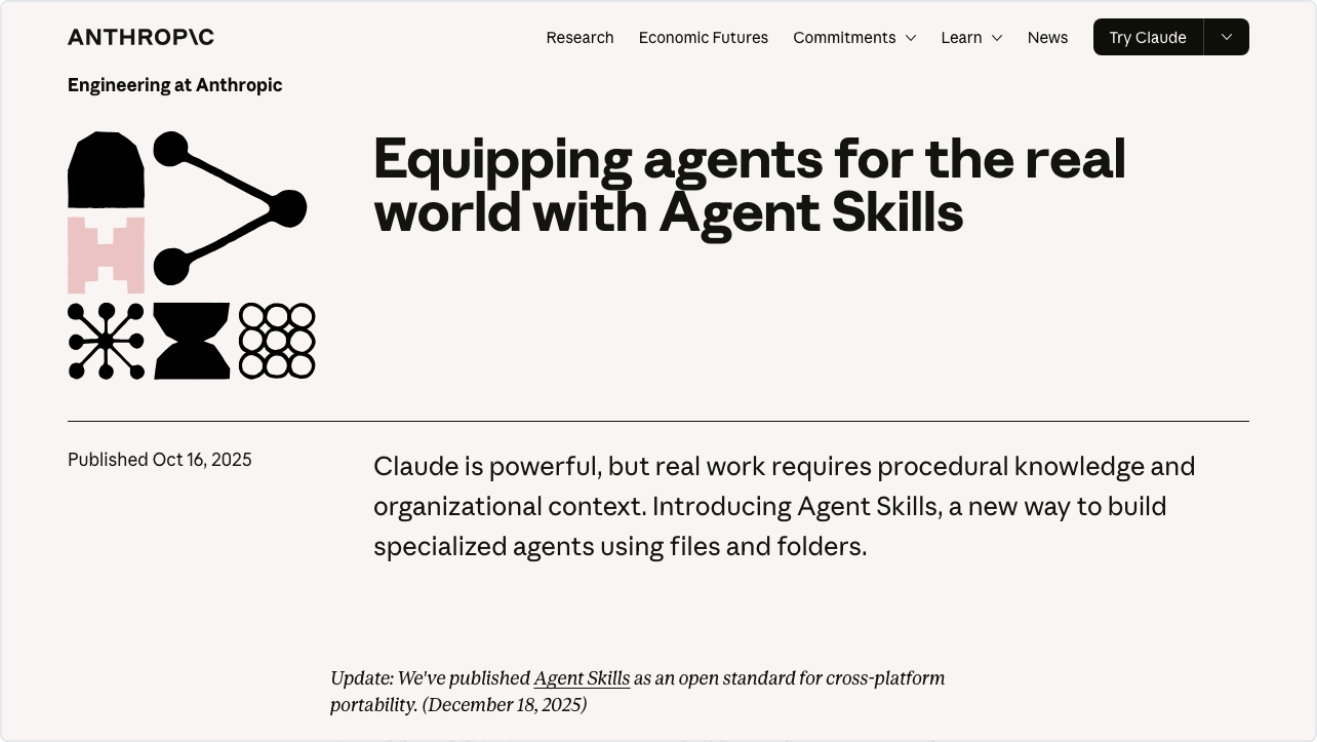
ハーネスの構成要素を、用途で見分けます。「いつ動くか」「誰が呼ぶか」「何を返すか」の三軸で並べると、混乱しません。

要素	いつ動くか	誰が呼ぶか	何を返すか	主用途
Hooks	イベント発火時（PreToolUse など）	ハーネス（受動）	exit code、JSON、permissionDecision	強制実行、ガード、監査
ガードレール	入出力のバリデーション時	呼び出し側コード	許可・拒否・修正	セキュリティ、コンプライアンス
Skills	関連トピックを Claude が認識した時	Claude（能動）	SKILL.md の本文と関連ファイル	手順書の遅延読込
サブエージェント	明示委譲した時	Claude or Task ツール	独立コンテキストでの実行結果（要約済み）	大量出力の隔離、専門化

8.1 Hooks の典型

```
// .claude/settings.json
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          { "type": "command",
            "command": "\"$CLAUDE_PROJECT_DIR\"/hooks/pre-bash-guard.sh" }
        ]
      }
    ]
  }
}
```

PreToolUse の hook が exit 2 を返すと、その Bash 呼び出しは止まります。stderr の内容は Claude にエラーとしてフィードバックされ、別アプローチに切り替えさせられます。「これからは X した後に Y して」という自動化は、メモリでは実現できません。Hook で実装します。



Anthropic Engineering Blog "Equipping agents for the real world with Agent Skills"。Skills の公式設計記事

8.2 Skill と SKILL.md

```
# .claude/skills/summarize-log/SKILL.md
---
name: summarize-log
description: アクセスログを読み、5xx 多発の原因を要約する
allowed-tools: Read Grep
context: fork
agent: log-reader
---
1. wc -l でサイズを確認
2. status=5\d{2} を Grep で抽出
3. 上位 3 件 + 全体傾向を 3 行
4. Markdown で返す
```

frontmatter に `context: fork` と `agent: log-reader` を入れると、本文は別コンテキストの log-reader サブエージェントで実行され、要約だけが親に戻ります。コンテキスト爆発の対症療法として効きます。

8.3 サブエージェントの定義

```
# .claude/agents/log-reader.md
---
name: log-reader
description: ログ全文を読み、異常行のみ要約して返す
tools: Read, Grep, Glob
model: haiku
---
全文は決して呼び出し元に返さない。
最終出力は 200 token 以内の Markdown 要約のみ。
```

ハンズオンで実際にこの構成を作って動かします。

SECTION 09

9. 設計思想 — 創業者・公式の言葉から

Claude Code を作った Anthropic の創業者・コアメンバーが、エージェント設計について繰り返し語っている言葉があります。本セミナーの判断軸として、4 つを引用します。

> 「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：Boris Cherny（Anthropic, Head of Claude Code）X 投稿（2025 年 12 月）／ 設計者本人が「正解はない」と公言。研修でテンプレを覚えるのではなく、自分の業務で再現する判断軸を持ち帰ることを優先します。

> 「My setup might be surprisingly vanilla! Claude Code works great out of the box, so I personally don't customize it much.」
>
> 出典：Boris Cherny X 投稿／ 創業者本人がほぼ素のまま使っている。本セミナー受講後に「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。

> 「At Anthropic, we don't build for the model of today, we build for the model of six months from now.」
>
> 出典：Boris Cherny on 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。

> 「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、"the most important tip"／ Claude に作業させたら、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。これが本セミナーの Verification loop の根拠。

9.1 この思想がコードに落ちると

Boris の運用は「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、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。本セミナーのハンズオンでは Stop Hook で監査ログを残す形で再現する。

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「the single biggest productivity unlock」。

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。本セミナーの主役。

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。

SECTION 10

10. ベストプラクティス 5 つ — 本セミナーで体験する

Anthropic は 2025 年 12 月に "How Anthropic teams use Claude Code" 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。本セミナーのハンズオンはこの 5 つのうち 3 つを直接体験します。

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。本セミナーの配布フォルダにも CLAUDE.md を入れています。

02 Verification loop 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を必ず通す。テスト・lint・型チェック・PR コメントの自動レビュー。本セミナーでは Stop Hook が監査 JSON を吐く形で再現します。

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。本セミナー Step 3 で summarize-log Skill を自作します。

04 サブエージェントで文脈を分離する

大きなタスクを役割ごとに切り出してサブエージェントに委任。各サブエージェント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。本セミナー Step 3 で log-reader サブエージェントを作って、ログ要約を委譲します。

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。本セミナー Step 2 で PreToolUse Hook で危険な Bash をブロックする設計を体験します。

出典

Claude Code Best Practices 公式ドキュメント（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（2025 年 9 月）

SECTION 11

11. 国内活用事例 — 派遣現場で持ち帰る話材

「導入事例」は数字の取り方で大きく変わります。ここでは公開ソースのある事例から、派遣エンジニアが派遣先で「うちでもやれそうか」を検討するときの話材として有用な 8 件を選びました。

事例 1 ／ 2026-01

マネーフォワード

API 実装 2 日 → 5 時間（70% 削減）、コードの約 80% を Claude 生成、エンジニアの 80% 超が導入・70% 超が日次利用、1 人あたり週 7 時間削減、オンボーディング 1 週間 → 1 日。Anthropic 公式ケーススタディに選出された日本フィンテック。引用可能な数値が最多で、派遣現場で「うちの API 実装にも使えるか」の判断材料になる。

出典：claude.com/customers/money-forward ／ マネーフォワード公式リリース

事例 2 ／ 2025

楽天グループ

1,250 万行コードベース（vLLM）で Claude Code が **7 時間連続自律実装**、99.9% の数値精度、critical errors を 97% 削減。新機能リリースのリードタイムを 24 日 → 5 日（79% 短縮）に圧縮、リリース頻度を四半期から 2 週間に。大規模レガシーコードでの長尺タスクの可能性を示した最初期の国内事例。

出典：claude.com/customers/rakuten

事例 3 ／ 2026-02

Gemcook

Web / ネイティブアプリ受託会社が全社導入。**導入から数日で数万行のコード生成。**Cursor / Windsurf / Devin と比較検証して全社導入を決定する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 ／ 2026

札幌の SIer 約 50 名

AI 非使用時との比較で **開発生産性 10 倍超。**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。地方中堅 SIer サイズの規模感は派遣現場と近い。

出典：firecracker.jp

事例 5 ／ 2026-02

malna（マルナ）

1 週間で全社導入決定、2 週間でプログラミング未経験者がコード執筆開始、業務の約 7 割を AI 置換可能と判断。定例 MTG を AI 日次サマリに置換した実例が、Hooks / Skills の応用としてセミナー本日の内容と直結する。

出典：note.com/malna_recruit

事例 6 ／ 2025-11

NRI 野村総合研究所

国内初の Anthropic 認定 Bedrock リセラー。設計・開発・コンサル業務に Claude Code を横展開し、外販の導入支援サービスも整備。金融系派遣現場で「自社経由で Claude を入れられるか」の交渉材料に。

出典：nri.com

事例 7 ／ 2026

メルカリ — 運用面のリファレンス

Claude Code を **企業で安全に使うための 5 つのセキュリティ制限ポリシー**を全社配布。Jamf / Intune 経由で macOS / Windows / Linux に一括設定する実装が公開されている。派遣先で「セキュリティ的に入れられない」と言われたときに出せる対策テンプレ。

出典：speakerdeck.com/hi120ki

参考 ／ ANTHROPIC 自身

Anthropic 社内 10 部門の活用

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。作っている本人たちがどう使っているかの一次情報。

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

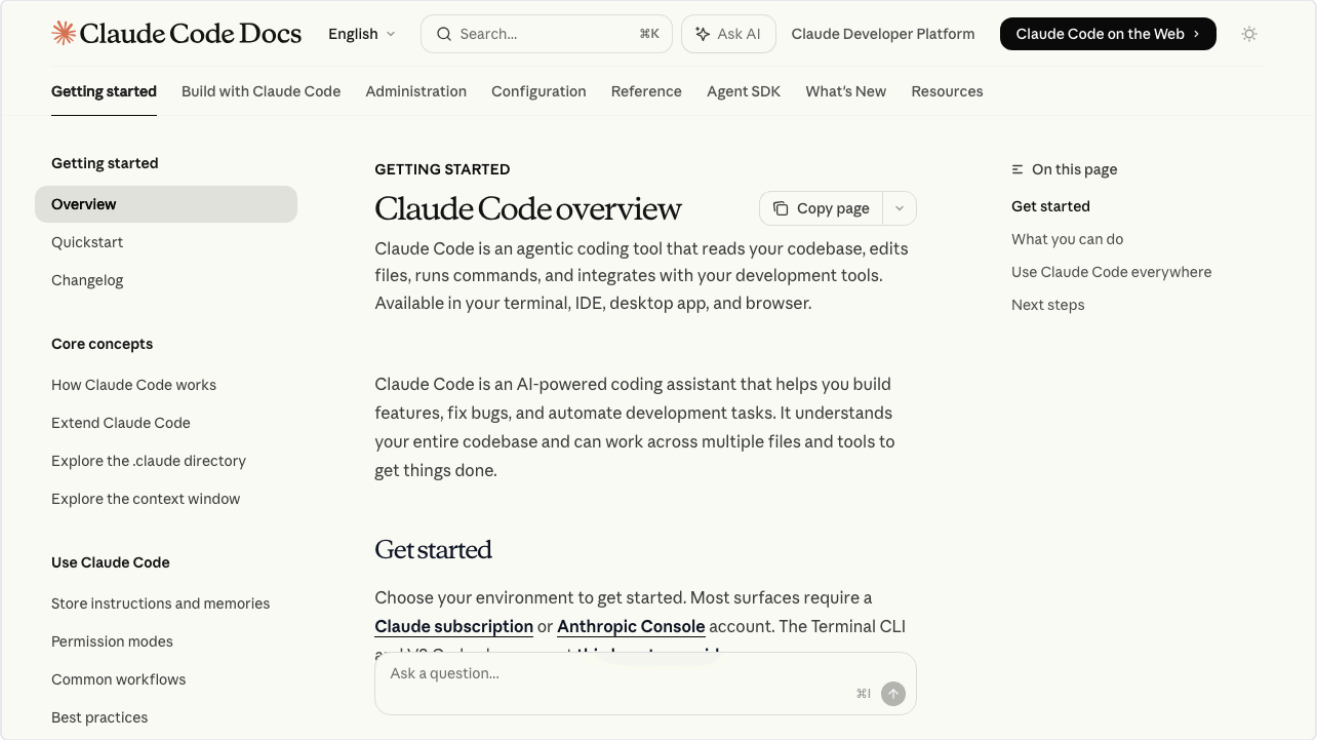
事例を見るときの注意点

「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、レビュアーの工数を含むか）を確認してから自社に当てはめてください。本セミナーで扱う Hooks / Skills / サブエージェントは、「便利」だけでなく「監査・統制」の側面が強いことに注目してください。

SECTION 12

12. 追うべき情報源 — 公式 5・X 4・深掘り 3

Claude Code は 1 日 0.5 回のペースで仕様が動きます。セミナー後に「最新情報をどこで追うか」を持ち帰るため、講師がウォッチしているソースを公開します。



Claude Code 公式ドキュメント — Hooks / Skills / Subagents / Settings / MCP すべての一次情報 — code.claude.com/docs

12.1 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。資料作成時のバージョンと当日のバージョンの差分をチェックするのに必須。

code.claude.com/docs/en/changelog

公式 / 設計の解説

Anthropic Engineering Blog

"Effective context engineering"、"Effective harnesses for long-running agents"、"Equipping agents with Skills" 等。設計思想の長文。

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

12.2 X（Twitter）— ここでまず気づく

@AnthropicAI

モデル発表、大型機能、研究系の話題。

@claudeai

プロダクトの新機能、ティザー、社内 demo の流出元。

@bcherny

Boris Cherny。Claude Code の Tips を本人が定期投稿。「My setup is vanilla」が定番。

@_catwu

Cat Wu。Claude Code Product Lead。新機能リリース予定や設計判断の解説。

12.3 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。プロダクト原則と "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。

youtube / @ycombinator

SECTION 13

13. Before デモで何を見るか

講師が 15 分かけて、素の Claude Code に「sample-logs/access.log を読んで 5xx 多発の原因を要約して」と投げます。便利だが 3 つの痛みが同時に出ます。これが after との比較基準になります。

痛み 1、コンテキスト爆発

素の Claude Code は Read を直接叩いてログ 105 行を会話に取り込む。`/cost` でトークン消費が出る。10 万行のログで同じことをすれば一発で compact が走る。

痛み 2、危険コマンドが素通りしうる

「ログを一旦全消去して再生成テストして」と頼むと permission prompt は出るが、Yes 一発で `rm -rf sample-logs` が走る位置に来る。判断が人手依存になっている。

痛み 3、監査ゼロ

セッション後、何のツールが何回呼ばれたかを集計する仕組みが標準ではない。transcript jsonl はあるが、運用ダッシュボードに繋ぐには毎回パースが要る。

After で何が変わるか

同じ依頼を Hooks + Skills + サブエージェント入りプロジェクトで投げると、サブエージェント側でログを読み、要約だけが親に返ります。Hook が監査ログを `.claude/audit/summary-*.json` に書き、危険な Bash は PreToolUse で deny されます。45 分でこの差を作れる、というのが本セミナーの主張です。

SECTION 14

14. コスト管理とモデル選び

Claude Code は使うほど API コストが効いてきます。社内導入の壁としてよく挙がる項目。本セミナーのハンズオンでも haiku をデフォルトに採用しています。

14.1 主要モデル（2026 年 5 月時点）

モデル	用途	強み	弱み
Claude Haiku 4.5	要約、分類、軽量タスク	低コスト、高速、サブエージェントの委譲先に最適	長尺の設計タスクには物足りない
Claude Sonnet 4.6	主力。日常のコーディング	性能とコストのバランス、本セミナーのデモは Sonnet で実演	超長尺タスクは Opus
Claude Opus 4.7	長尺リファクタ、複雑な設計	長時間自律実行、xhigh effort などの追加機能	コストが高め、ハンズオンには不要

14.2 コストを下げる定石

01
サブエージェントは **haiku をデフォルト**に。重い読み取り・要約タスクは haiku で十分。
02
Skill の **context: fork** でログ全文を親に渡さない。トークン消費が桁で違う。
03
/cost コマンドでセッション中のトークン内訳を意識する。長時間セッションでは `/compact` も使う。
04
定型タスクは **Custom Command** 化して、システムプロンプト分のトークンを共有する。

> **出典**
> Claude Code Pricing と Models Overview（docs.anthropic.com）／ Anthropic Pricing（anthropic.com/pricing）

SECTION 15

15. 主要キーワード — セミナー中に出てくる 12 語

本編で都度説明していますが、ここで一度通しておくと振り返りがしやすくなります。

プロンプトエンジニアリング	LLM への入力テキスト本体を設計する手法。単発・静的。
コンテキストエンジニアリング	推論中に LLM へ到達するトークン全体を整え維持する戦略。マルチターン・動的。
ハーネスエンジニアリング	モデル外側のコード・設定・実行ロジックの設計。Hooks / Skills / サブエージェント / オーケストレーションを内包。
オーケストレーション	複数のエージェントを調停して動かす設計パターン。Lead-Worker が代表。
MCP	Model Context Protocol。LLM アプリと外部リソースを繋ぐ JSON-RPC 2.0 ベースの標準プロトコル。
RAG	Retrieval Augmented Generation。検索結果をプロンプトに混ぜる方式。
ツールユース	モデルが関数呼び出しを通じて外部操作を行う仕組み。
Hooks	ライフサイクル特定地点で発火する強制実行。Claude Code の Hooks は exit code 2 でツールをブロックできる。
Skills	SKILL.md と関連リソースで構成される、Claude が必要時に参照する手順書。
サブエージェント	独立コンテキストを持つ作業ユニット。.claude/agents/ 配下に置く。
ReAct	Thought → Action → Observation を 1 ステップずつ織り交ぜるエージェントパターン。
Plan-and-Execute	Planner が事前にステップ列を作り、Executor が順次実行するパターン。

SECTION 16

16. 振り返りと 30 日ロードマップ

最後に、今夜扱った概念と実装の対応を 1 枚に圧縮します。これだけ持ち帰れば、現場で設計判断を組み立てられます。

概念	本日の触れ方	明日からの一手
コンテキストエンジニアリング	SECTION 05 で定義、Before デモで爆発を体感	毎ターンのトークン内訳を意識する習慣をつける
ハーネスエンジニアリング	SECTION 07 で全体構造、ハンズオンで Hooks / Skills / サブエージェントを実装	自分の業務エージェントに Hook を 1 つ入れる
MCP	SECTION 06 で位置付け	社内データを MCP サーバー化する POC を 1 件
サブエージェント	ハンズオンで log-reader を作って委譲を体験	大量出力タスクを 1 件、サブエージェントに切り出す
ReAct / Plan-and-Execute	SECTION 07 で違い	長時間タスクで「リプラン地点」を明示する設計を試す
コスト管理	SECTION 14 で haiku 優先のルール	サブエージェントの model 指定を haiku にする

16.1 30 日ロードマップ

D1
当日。本セミナーで作った log-review-agent を保存。Skills と Hooks の構造を覚える。
D7
1 週間後。派遣先のログを 1 種類選び、log-review-agent の sample-logs を置き換えて再現。エンドポイントだけ調整すれば動くはず。
D14
2 週間後。Hook を 1 つ追加する。具体例として「セッション終了時に audit JSON を Slack Webhook に POST」が現実的。
D21
3 週間後。Skill を 1 つ自作する。派遣先の業務でよく書くプロンプトを SKILL.md にまとめる。
D30
1 ヶ月後。チーム内勉強会で 30 分発表。事例（マネーフォワード等）と自分の Skill を組み合わせて話す。

16.2 シリーズ次回（予定）

本セミナーは技術トレンド解説 AI シリーズ第 1 回です。第 2 回はマルチエージェント設計の深掘りや AI オーケストレーションの実装パターンが候補に挙がっています。CA 企画事務局からのアナウンスをお待ちください。

> **机上の理解は 1 週間で蒸発する**
> 研修受講後の 7 日間に、配布した `log-review-agent/` をもう 1 度自分の手元で動かしてください。再現できたら、自分の業務ログに置き換える。これだけで「本セミナーを取った人」と「動く形で残せた人」が分かれます。

参考リンク

- セミナー申込ページ（パーソルクロステクノロジー CA企画）
- Effective context engineering for AI agents（Anthropic）

- Effective harnesses for long-running agents（Anthropic）
- How we built our multi-agent research system（Anthropic）
- Equipping agents for the real world with Agent Skills（Anthropic）
- Enabling Claude Code to work more autonomously（Anthropic, 2025-09）
- How Anthropic teams use Claude Code（Anthropic, 2025-12）
- MCP Specification 2025-11-25
- The 2026 MCP Roadmap
- Claude Code Hooks Reference
- Claude Code Skills Reference
- Claude Code Subagents Reference
- Claude Code Settings Reference
- Claude Code Best Practices
- Claude Code Changelog
- Prompt Engineering Overview（Anthropic API Docs）
- マネーフォワード Customer Story（Anthropic 公式）
- 楽天グループ Customer Story（Anthropic 公式）
- Gemcook 全社導入レポート
- 札幌 SIer 開発生産性 10 倍超
- malna 全社導入 2 週間レポート
- NRI Anthropic 認定 Bedrock リセラー
- メルカリ Claude Code 組織配布戦略
- ReAct（arxiv 2210.03629）
- Plan-and-Act（arxiv 2503.09572）
- Agentic Memory（arxiv 2601.01885）
- Anthropic Introduces Managed Agents（InfoQ 2026/04）

エージェンティックエンジニアリング（パーソルクロステクノロジー様 CA企画）

2026 年 5 月 21 日(木) 19:00–21:00 ／ 提供：Givery 株式会社