Givery, Inc.

HANDS-ON GUIDE / DAY 2 ・ 2026 年 5 月版

ハンズオンガイド Day 2

運用に耐える形に仕上げ、別テーマで再演し、
進化を追う情報源を確定する

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	9:00 - 17:00（実働 400 分、昼休憩 60 分、午前午後の休憩各 10 分）
題材	Day 1 の成果物に Hooks / Cron / MCP / DESIGN.md を載せる
版数	v2.0（2026 年 5 月 3 日）

本書は受講者の手元で開く資料です。Day 1 の成果物を起点に、運用面の磨き込みと別テーマでの再演を行う 7 時間分の手順を集約しています。

SECTION 01 ・ OVERVIEW

1. 本日のゴール

Day 1 で作った α（案件ヘルスチェックダッシュボード）に、Hooks / Cron / MCP / DESIGN.md を載せて運用に耐える形に仕上げます。後半 100 分は α 完成度向上 / 自社業務 / 別テーマの 3 択で再演し、最後に公式情報源探索でキャッチアップ習慣の素を作ります。

形式	オンサイト ／ 7 時間 ／ 実働 400 分
環境前提	Bash 4 以上、cron / launchd 登録権限、npx / uvx 通信可能（NG 時は講師デモへ切替）
主要ツール	Claude Code CLI、VSCode、Hooks（PreToolUse / PostToolUse / Stop）、cron、MCP（filesystem / sqlite）

1.1 Day 2 の到達点

1. Day 1 の Subagent 群（risk-extractor / code-reviewer / security-auditor）を並列で同時に動かす運用が見えている
2. Hooks（PreToolUse / PostToolUse / Stop）でセッションに自動化線が引けている
3. cron / launchd / タスクスケジューラで非対話 Claude Code の定期実行を仕掛けている
4. MCP（filesystem / sqlite）で外部データを Claude から直接参照できている
5. DESIGN.md でフロント画面を磨き、配色・余白・コンポーネント方針を 1 ファイルで管理できている
6. Day 1 成果物の完成度向上 / 自社業務 / 別テーマ再挑戦のいずれかで 100 分の仕上げ実装を完走
7. Anthropic 公式情報源を 3 つ以上ブックマークし、月 1 のキャッチアップ習慣をカレンダーに登録している

SECTION 02 · AGENDA

2. タイムテーブル

開始	終了	区分	内容
09:00	09:20	S05	Exercise 1：Day 1 リフレクション（提出物の振り返り）
09:20	10:10	S06	Exercise 2：ローカル Agent 並列操作（50 分）
10:10	10:20	休憩	10 分
10:20	11:10	S07	Exercise 3：Hooks 自動化（50 分）
11:10	11:50	S08	Exercise 4：Cron 定期実行（40 分）
11:50	12:30	S09	Exercise 5：MCP 実践（40 分）
12:30	13:30	昼休憩	60 分
13:30	14:00	S10	Exercise 6 前半：DESIGN.md 導入（30 分）
14:00	15:40	S11	Exercise 6 後半：仕上げ実装（A / B / C 選択、100 分）
15:40	15:50	休憩	10 分
15:50	16:20	S12	Exercise 7 前半：成果共有・デモ（1 人 3 分）
16:20	16:50	S13	Exercise 7 後半：公式情報源探索
16:50	17:00	クロージング	研修全体の振り返り

SECTION 03

3. ハンズオン前提とデモ切替の見方

社内環境の制約により、各自の手元で実行できない演習があります。各演習の冒頭「環境前提」を最初に読み、自分の環境で実行可能か判断してください。実行できない場合は講師の画面共有でデモ進行に切り替わります。手順書は完成形まで記載されているため、持ち帰り検証用にも使えます。

演習	ハンズオン or デモの判断ポイント
Exercise 3（Hooks）	Bash（macOS / WSL2 / Git Bash）が動く、`chmod +x` できる権限
Exercise 4（Cron）	cron 編集権限 / launchd 登録権限 / タスクスケジューラ登録権限
Exercise 5（MCP）	npx と uvx（または pipx）が外部レジストリへ通信可能

注意：環境制約は講師に申告

事前確認で OK が取れていれば手を動かしてください。NG なら読み物として後追い。当日の朝に状況が変わった場合は、各演習開始前に講師に申告してください。

SECTION 04

4. Day 2 のデータフロー

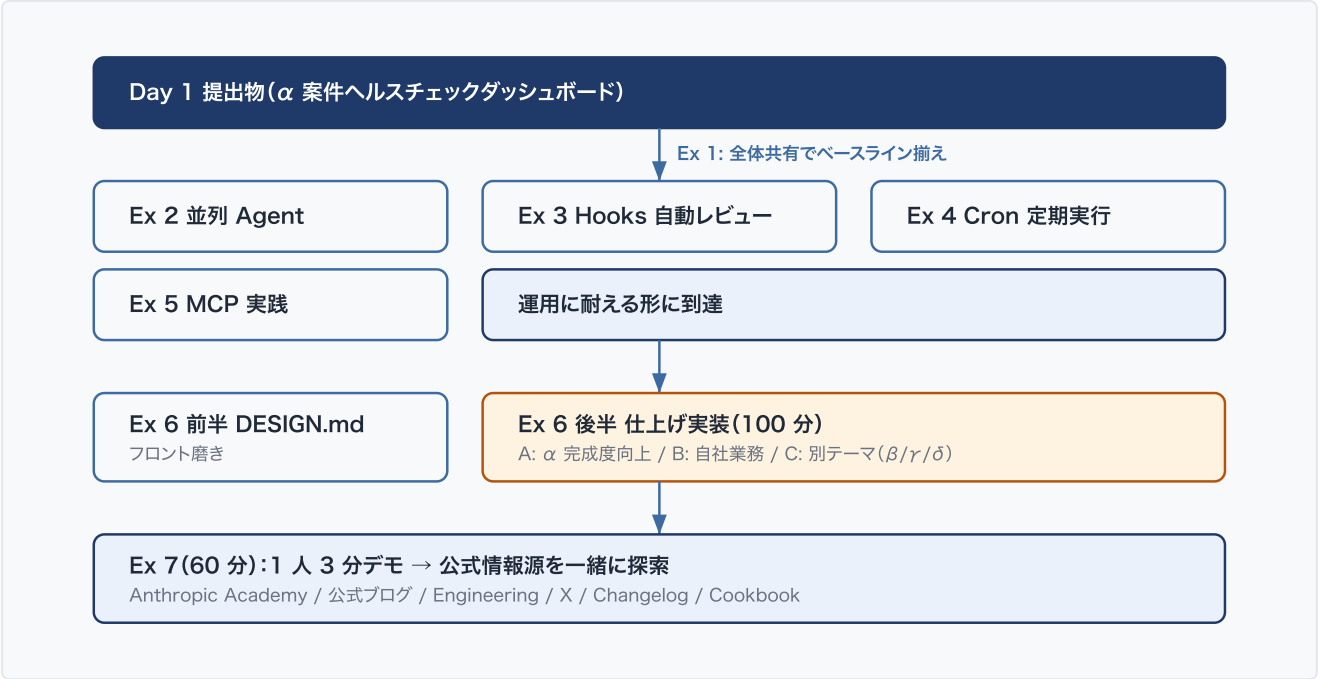

図 1：Day 2 のデータフロー。Day 1 成果を起点に運用面を磨き、選択肢で再演し、情報源で締める

DAY 2 ・ EXERCISES

5. 演習詳細

EXERCISE 1 ／ 20 分　Day 1 リフレクション

詳細手順　`exercises/day2/exercise1-reflection.md`

やること

Day 1 提出物を全員で共有し、Day 2 のベースラインを揃えます。「印象に残ったプロンプト 3 つ」を持ち寄り、講師から代表 3〜5 名にコメント。Day 1 で書いた CLAUDE.md / Skills / Subagents の差分を全員で見比べ、Day 2 で発展させる方向を決めます。

確認ポイント

- Exercise 1 の素朴 → 追記後の差分を 5 つ以上書き出した提出物があるか
- `/review-changes` と `/security-scan` で違反を CRITICAL 検出できた人の比率
- 3 モデル比較（Haiku / Sonnet / Opus）の表が完成しているか
- Exercise 7 自律開発で動く成果物まで到達した人の比率

EXERCISE 2 ／ 50 分　ローカル Agent 並列操作と指示の出し方

対象ファイル　`.claude/agents/risk-extractor.md` / `code-reviewer.md` / `security-auditor.md`
詳細手順　`exercises/day2/exercise2-parallel-agents.md`
答え合わせ　`exercises/day2/answers/exercise2-parallel-agents_例.md`

やること

Day 1 で作った 3 つの Subagent を **並列で同時に動かす**運用に発展させます。Task ツール経由で複数 Subagent を起動し、結果を親で集約するパターンを業務想定で試します。並列指示の文言設計と、コンテキスト分割の感覚を磨きます。

AI への指示例（並列起動）

```
3 つの Subagent を並列で起動してください。

- Agent A（risk-extractor）：03_yamada.md（55 分・現場担当ヒアリング）から
  Risk JSON を抽出する
- Agent B（code-reviewer）：今ステージしている差分（git diff --cached）を
  CLAUDE.md と既存実装に照らしてレビューする
- Agent C（security-auditor）：backend/ 全体のハードコード秘密と
  SQL 文字列フォーマットをスキャンする

各 Agent の処理時間と検出件数を表で出力してください。
親エージェントは 3 つの結果を統合して、優先対応 5 件を選んでください。
```

期待される結果

- 3 件の Subagent が並列起動し、各々の結果が独立コンテキストで返る
- 処理時間が逐次実行と比べて短縮される（`/cost` でトークン消費比較）
- 親エージェントの結果に「優先対応 5 件」が PRIORITY 順に並ぶ
- 各 Subagent のコンテキストは親に持ち越されず、要約のみが残る

> **Boris Cherny の言葉**
>
> "Spin up 3-5 git worktrees at once, each running its own Claude session in parallel. It's the single biggest productivity unlock, and the top tip from the team."(Boris Cherny X 投稿)。本演習はその縮小版を Subagent で体験する位置づけです。本格運用では git worktrees + 並列セッションに発展させます。

EXERCISE 3 ／ 50 分　Hooks で自動化線を引く

対象ファイル　`.claude/settings.json` / `.claude/hooks/pre-tool-bash-guard.sh` / `post-edit-format.sh` / `stop-summary.sh`

詳細手順　`exercises/day2/exercise3-hooks.md`

答え合わせ　`exercises/day2/answers/exercise3-hooks_例.md`

やること

3 種類の Hook を実装します。**PreToolUse**(危険な Bash の停止)、**PostToolUse**(編集後の自動フォーマット ruff / prettier)、**Stop**(セッション終了時の差分サマリー)。Claude Code の振る舞いを「人間が忘れがちなチェックを機械に任せる」形に発展させます。

.claude/settings.json の hooks セクション例

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          { "type": "command", "command": ".claude/hooks/pre-tool-bash-guard.sh" }
        ]
      }
    ],
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          { "type": "command", "command": ".claude/hooks/post-edit-format.sh" }
        ]
      }
    ],
    "Stop": [
      {
        "hooks": [
          { "type": "command", "command": ".claude/hooks/stop-summary.sh" }
        ]
      }
    ]
  }
}
```

期待される結果

- `rm -rf /` や `git push --force` を Claude Code が実行しようとすると `exit 2` でブロック
- backend/*.py を編集すると ruff format が自動実行される
- frontend/src/**/*.tsx を編集すると prettier が自動実行される
- `/exit` で終了する時に `git diff` サマリが表示される

Hooks に載せるべき項目

Hooks は人間が忘れがちなチェックを機械に任せる場所です。逆に、人間の判断が必要なチェックを Hooks に詰め込むと、ブロックが頻発してチームが疲弊します。Hooks に任せる項目は CLAUDE.md で「Yes / No が明確に決まる」規約に限定します。

早く終わった方向け

pre-commit ではなく pre-push Hook に切り替え、ローカルコミットは速いまま、push 時にまとめて検査する設計に変えます。速度と品質ゲートの強さのトレードオフが確認できます。

EXERCISE 4 ／ 40 分　非対話 Claude Code の定期実行

対象ファイル　`.claude/commands/weekly-report.md` / `scripts/run-weekly-report.sh`

詳細手順　`exercises/day2/exercise4-cron.md`

答え合わせ　`exercises/day2/answers/exercise4-cron_例.md`

やること

Claude Code を CLI から非対話で起動し、cron / launchd / Windows タスクスケジューラに乗せます。`/weekly-report` Command を週次自動実行する運用を題材にします。

cron への登録例（毎週月曜 9:00）

```
# crontab -e で開いて以下を追記
# 注意：% は \% でエスケープ。ファイル名に日付が入らない事故が頻発
0 9 * * 1 cd ~/pwc-cc-handson_受講者用 && \
  /usr/local/bin/claude /weekly-report \
  > docs/reports/weekly-$(date +\%Y-\%m-\%d).md 2>&1
```

期待される結果

- cron に登録した時間に `/weekly-report` が起動する
- `docs/reports/weekly-2026-MM-DD.md` が生成される
- ファイル名に日付が入っている（% エスケープが正しい）
- 社内 LLM 中継基盤の認証情報（`ANTHROPIC_BASE_URL` や基盤発行のトークン）が `.env` 経由で読まれている（cron 自体には書かない）

cron の % エスケープに注意

エスケープを忘れるとファイル名が `weekly-.md` のように日付なしで出力され、毎回上書きされます。**最初の登録テストで「毎分実行」に書き換えて 5 分待ち、ファイル名に日付が入っていることを確認してから本番設定に戻してください。**

EXERCISE 5 ／ 40 分　MCP でアプリと外部データを直結

対象ファイル　`.mcp.json` / 必要に応じて `.claude/settings.json`

詳細手順　`exercises/day2/exercise5-mcp.md`

答え合わせ　`exercises/day2/answers/exercise5-mcp_例.md`

やること

MCP(Model Context Protocol)で外部リソースを Claude Code に「公式に」見せます。`filesystem` MCP(公式リファレンス)と `sqlite` MCP(コミュニティ実装 mcp-server-sqlite)を有効化し、ヒアリング起こしの参照と SQLite ベースの Risk テーブル問い合わせを Claude 側から直接行えるようにします。

.mcp.json の例

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem",
               "./docs/source-materials"]
    },
    "sqlite": {
      "command": "uvx",
      "args": ["mcp-server-sqlite", "--db-path", "backend/data/health.db"]
    }
  }
}
```

AI への指示例

```
claude
> /mcp     # connected と出ていることを確認
> filesystem MCP 経由で 03_yamada.md を読み込み、要約してください
> sqlite MCP 経由で risks テーブルの severity=CRITICAL の件数を返してください
```

> **MCP は権限設計を先に決める**
>
> MCP で本番 DB や Jira に接続すると、Claude Code が破壊的操作を実行できる状態になります。**read-only 権限のアカウントを別に用意する、更新系操作には人間の承認を挟む**を最初から運用に組み込みます。本演習を read-only モードで設定する理由はここにあります。

> **公式 SQLite MCP は archived**
>
> 公式リファレンスサーバーのうち SQLite は `modelcontextprotocol/servers` から archived へ移動済み。本演習ではコミュニティ実装 `mcp-server-sqlite` を例として使いますが、社内本番運用では実装の保守性とライセンスを確認のうえ採否を判断してください。最新ステータスは github.com/modelcontextprotocol/servers 参照。

EXERCISE 6 ／ 130 分　DESIGN.md 導入 + 仕上げ実装(A / B / C 選択)

対象ファイル　`DESIGN.md` / `frontend/src/` 全般 / 選択肢に応じて他

詳細手順　`exercises/day2/exercise6-design-and-finish.md`

答え合わせ　`exercises/day2/answers/exercise6-design-and-finish_例.md`

前半 30 分:DESIGN.md 導入

最小版 DESIGN.md(バイブデザイン 5 語の枠だけ)を完成形相当に育て、フロントエンド画面に反映します。配色・余白・コンポーネント方針を 1 ファイルで管理する経験。

DESIGN.md に含める項目（最低限）

- バイブデザイン（5 語）：例「業務向けに堅実 / 情報密度高め / 視線誘導は左から右」
- カラーパレット（プライマリ / セカンダリ / 警告 / 成功）
- 余白・行間（base font-size、行間倍率、セクション間隔）
- コンポーネント方針（ボタン、カード、テーブル、コールアウト）
- アンチパターン（やってはいけないこと）

後半 100 分：A / B / C から 1 つ選択

選択肢	内容	向いている人
A	α（案件ヘルスチェック）の完成度向上	Day 1 の続きで作業負荷を低く完走したい人。最も完走率が高い
B	自社業務テーマの新規プロトタイプ	自社で持ち帰る前提が明確な人。要件メモ 10 行から始める
C	別テーマ（β / γ / δ）の完走	別の業務文脈で復習したい人。配布の `別テーマ素材.zip` を使う

> **選択は最初の 10 分で確定**
> 選んだ後の最初の 10 分で「ゴール」を 3 行で書き出してください。書き出さずに走り出すと 100 分が散漫になります。迷ったら A を選ぶ。

EXERCISE 7 ／ 60 分　成果共有 + 公式情報源探索

詳細手順 `exercises/day2/exercise7-share-and-keep-learning.md`

前半 30 分：1 人 3 分のデモ

受講者全員が 3 分で Exercise 6 の成果を共有。「何を作ったか」「どんな指示が効いたか」「何で詰まったか」の 3 点。録画して持ち帰り検証用に残すのも歓迎します。

後半 30 分：公式情報源を一緒に探索

講師（安田）が普段ウォッチしているソースを一緒に開きます。研修後のキャッチアップ習慣の素を作るパート。

- Anthropic 公式ブログ：anthropic.com/news
- Anthropic Engineering：anthropic.com/engineering
- Claude Code Changelog：code.claude.com/docs/en/changelog
- Anthropic Cookbook：github.com/anthropics/anthropic-cookbook
- X（@AnthropicAI / @claudeai / @bcherny / @_catwu）
- Lenny's Newsletter / Pragmatic Engineer

提出物（Day 2 終了時）

- 仕上げ成果物（A / B / C いずれか）：A は強化版スクショ、B は要件メモ + スクショ、C は完走品の README
- 公式情報源 3 つのブックマーク URL（自分のメモアプリに保存）
- 翌週・翌月・3 か月後の自分宛アクション 3 つ
- 研修全体への感想（任意）

月 1 のキャッチアップ時間をカレンダーに登録

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。最低でも「月 1 回 30 分、Changelog をまとめ読みする時間」をカレンダーに登録してから帰ります。社内チャットに「Claude Code 情報源」チャンネルを作って週 1 で投下する運用も有効です。

SECTION 06

6. Day 2 到達チェックと研修後ロードマップ

Day 2 到達チェック

- 3 つの Subagent を並列で同時に動かし、結果を親で集約できた
- PreToolUse / PostToolUse / Stop の 3 種類の Hook が動作する
- cron / launchd / タスクスケジューラに `/weekly-report` を登録した
- filesystem / sqlite MCP が `connected` 状態で、自然言語から MCP ツール呼び出しが返る
- DESIGN.md がバイブデザイン + カラーパレット + 余白 + コンポーネント方針で完成
- Exercise 6 仕上げ実装（A / B / C のいずれか）が動く状態
- 公式情報源 3 つをブックマークし、月 1 のキャッチアップ時間をカレンダーに登録

6.1 研修後 3 か月のロードマップ

直後〜 1 週間

個人で動かす最小版

- 研修中に書いた CLAUDE.md / Skills / Subagents をチームに共有
- 業務リポジトリ 1 つに `.claude/` を導入
- 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

- 業務リポで Hooks（PreToolUse: 危険コマンド停止）導入
- code-reviewer Subagent を全員配布、1 か月分の PR でテスト
- 月 1 Changelog 読みを実施

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。一気に全社展開を目指さず、対象業務を絞ったパイロット → 効果測定 → 展開判断の順で進めます。

SECTION 07 ・ FAQ

7. よくある質問（Day 2 特有）

Q1：Hook が動かないときの初動は？

`chmod +x .claude/hooks/*.sh` を確認し、改行コードを LF にして再保存。Windows 受講者は WSL2 または Git Bash で実行します。`.claude/.bak` を取ってから編集する習慣を。Hook が暴走するとセッションが回らなくなります。

Q2：cron に登録したが実行されない

絶対パスで `claude` バイナリを指定しているか確認してください。`/usr/local/bin/claude` を `which claude` で取得した値に置き換えます。`%` のエスケープ漏れも頻出（`%Y` は cron では `\%Y` と書く）。最初の登録テストで「毎分実行」にして 5 分観察してから本番に戻すのが安全です。

Q3：MCP が起動しない

`.mcp.json` の JSON 構文エラーが多いです。閉じカッコ漏れ、最後のカンマを確認。Claude Code のログ（`~/.claude/.logs/` 等、最新は公式ドキュメント参照）も合わせて確認します。

Q4：MCP を社内環境で動かせるか？

npx / uvx の通信が許可されていれば動きます。社内プロキシ越しの場合は `HTTPS_PROXY` 環境変数を設定。完全に外部通信できない環境では、自前で MCP サーバーを社内に立てる方式に切り替えます。GitHub MCP（公式リファレンス実装あり）も検討余地あり。

Q5：Cron 非対話起動の認証は？

非対話起動はブラウザログインを伴う Pro / Max サブスクリプションでは行えません。社内 LLM 中継基盤の認証情報（基盤側で発行されるトークンや `ANTHROPIC_BASE_URL` ）を `.env` から読む形にします。**Cron 自体に認証情報を書くのは避けてください。**

Q6：DESIGN.md の配色をどこまで厳密に守るか

ガイドラインを優先しつつ、業務の見やすさが勝つ場合はその場で DESIGN.md を更新してください。**DESIGN.md は変えていいドキュメント**です（理由をコメントとして残します）。

Q7：Exercise 6 で A / B / C のどれを選ぶべきか

迷ったら A（α 完成度向上）。Day 1 の続きで作業負荷が低く、最も完走率が高い選択肢です。自社で持ち帰る前提が明確な人は B、別の業務文脈で復習したい人は C。

SECTION 08 · TROUBLESHOOTING

8. 詰まったときの即応マニュアル

症状	確認	対処
Hook が動かない(pre-tool-bash-guard.sh)	`chmod +x` 済み、改行 LF	権限と改行コードを修正、再保存。`.claude/.bak` を取ってから編集する
Hook で `exit 2` が効かない	matcher の正規表現	matcher は正規表現、エスケープに注意。`"matcher": "Bash"` のように単純な単語でも OK
cron が実行されない	`which claude` で絶対パス、`%` エスケープ	`/usr/local/bin/claude` に置換、`\%Y-\%m-\%d` でエスケープ
MCP が `connected` にならない	`.mcp.json` の JSON 構文	カンマ漏れ、閉じカッコを確認。`jq . .mcp.json` で構文チェック
filesystem MCP がパスを読まない	許可ディレクトリの指定	`args` の最後に絶対パスで許可ディレクトリを指定。複数渡せる
sqlite MCP が動かない	uvx / pipx インストール、SQLite ファイルパス	`--db-path backend/data/health.db` を絶対パスに変えて再試行
定期実行で認証エラー	cron 実行時の環境変数(中継基盤の認証情報)	`.env` 読込ラッパー `scripts/run-weekly-report.sh` 経由で起動
`./mvnw clean package` が Permission denied(旧 Java 系の事故)	Windows ZIP 解凍時の実行権限	WSL2 で `chmod +x mvnw`。本研修は Java を使わないため通常該当しないが、参考

> **5 分ルール**
>
> 5 分以上自力で抜けられなければ、上の表 → 講師にエスカレ。詰まりが連鎖する方が研修時間の損失が大きい。受講者同士で相談するのも歓迎します。

参考リンク

1. Claude Code Docs:code.claude.com/docs
2. Hooks 設定:code.claude.com/docs(Hooks 章)
3. Sub-agents:code.claude.com/docs/en/sub-agents
4. Model Context Protocol — Reference Servers:github.com/modelcontextprotocol/servers
5. MCP 仕様:modelcontextprotocol.io
6. Anthropic Cookbook:github.com/anthropics/anthropic-cookbook
7. Anthropic Engineering Blog:anthropic.com/engineering

8. 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

© Givery, Inc. All Rights Reserved.