TRAINING

生成AIプログラミング入門編

VS Code と Claude Code で手を動かす

研修教材　全240分

Givery

本資料の二次転載禁止について

本資料は受講者ご本人の学習用です。
無断での二次転載と社外配布はご遠慮ください。

本資料の著作権は株式会社ギブリーに帰属します。
受講者ご本人の復習と、業務での実践に使ってください。
社外への配布、SNSや社内ポータルへの掲載はご遠慮ください。
講義の録画と録音も、事前の許可なく行わないでください。
引用が必要な場合は、事務局を通じてご相談ください。

要確認 / 要記載
配布資料に関する問い合わせ先として、事務局の窓口を記載します。

講師紹介

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

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。

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。
#ClaudeCode #Codex #Cursor #Antigravity
#全国統一生成AI活用技能試験S1
#生成AIパスポート　#G検定

本日のゴール

240分で業務アプリを1本作り、Javaのバグを2件直し、テストを緑にして帰ります。

1本

業務アプリを作る

課題A 一覧と検索と詳細の3画面

2件

バグを直す

課題B サービス層の不具合

```
BUILD SUCCESS
```

テストを通す

課題B ./mvnw test の実行結果

> 作ったものは持ち帰れます。配布フォルダごと手元に残るので、
> 明日の業務でそのまま開いて続きができます。

本日の流れ

240分を6区切りで進めます。手を動かす時間が140分です。

- S01 オリエンと支援の段階遷移 [20min]
- S02 座学とデモ [40min]
- **S03 演習A 素の状態からハーネスまで** [70min]
- **S04 演習B Skillとバグ修正** [70min]
- まとめとAI設計デモ [20min]
- S05 振り返りと質疑 [20min]

進め方とルール

詰まったら手を挙げてください。
進めなくなる前に拾います。

- **発表はありません**
  書いたものを全員の前で読み上げる時間は取りません。
- **ペア作業はありません**
  手元のPCで、一人ずつ進めます。
- **hints は詰まってから開きます**
  5分ほど考えて進まないときに、該当ステップの hints を見てください。
- **Git は使いません**
  配布ZIPを展開したフォルダの中だけで完結します。
- **気づきは memo.md に書きます**
  各ステップの記録先を1つにまとめておくと、あとで見返せます。

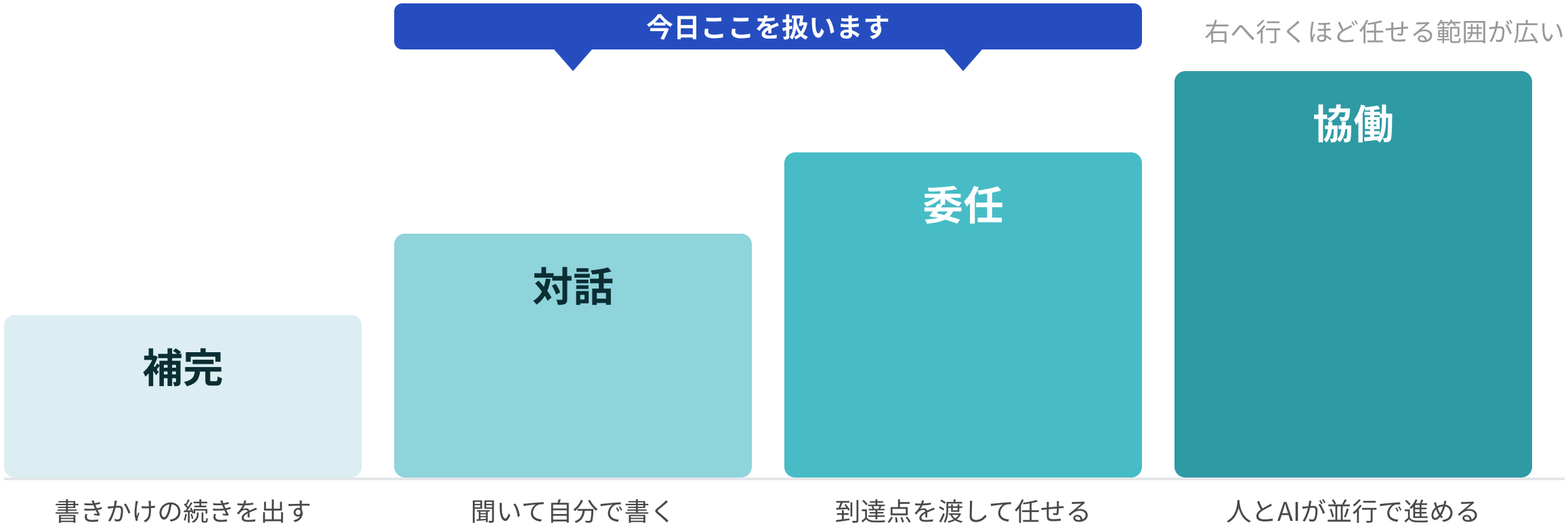
01

コーディング支援の現在地

補完から協働まで、AIに任せる範囲がどこまで広がったかを整理します

支援の段階遷移

補完から協働まで4段あります。変わるのは道具ではなく、AIに任せる範囲です。



今日扱う範囲

今日は対話と委任の境目を扱います。

手順を渡すか、到達点を渡すか。その差を演習で確かめます。

補完は日々の入力補助として、協働はチーム運用の話として、別の機会に扱います。

質問

AIが出すものと人が決めるもの

AIは案を速く出しますが、採否を決めるのは人です。

AIが出すもの

- 動くコードの案を数分で
- 抜けの指摘と別案の提示
- 手順書やテストの下書き
- 知らない書き方の説明

人が決めるもの

- どの案を採用するか
- 業務ルールに合うかの判断
- 直さない範囲の線引き
- 本番に出してよいかの承認

直近1週間で、あなたが「採用しない」と判断した場面はありましたか。

クイズ

Q1 段階遷移

Q. 手順ではなく到達点を伝える使い方は、4段のどれに当たりますか。

- **A** 補完。書きかけの行の続きを出してもらう使い方
- **B** 対話。分からない箇所を聞いて説明を受ける使い方
- **C** 委任。到達したい状態を渡し、進め方は任せる使い方
- **D** 協働。人が確認せずAIどうしで進める使い方

正解は次のページで確認します。まず自分で考えてみてください。

解説

正解はC

委任は、手順ではなく到達点を渡す使い方です。

C **正解 委任。到達したい状態を渡し、進め方は任せる使い方**
どう直すかを指定せず、満たすべき状態だけを伝えます。

A 補完。書きかけの行の続きを出してもらう使い方
行単位の補助で、任せる範囲がいちばん狭い段階です。

B 対話。分からない箇所を聞いて説明を受ける使い方
説明を受けて自分で書くので、作業は手元に残ります。

D 協働。人が確認せずAIどうしで進める使い方
人の確認を外す話ではありません。採否は今日も人が決めます。

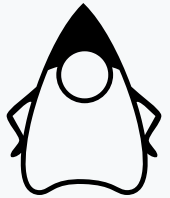
実務でのポイント
到達点で伝えると途中の判断まで任せられます。演習Aでその差を確かめます。

本研修で触る4つ

編集はVS Code、AIはClaude、
題材はJavaとSpring Bootです。

VS Code
編集画面

Claude
対話とコード生成



Java 17
課題Bの実行環境

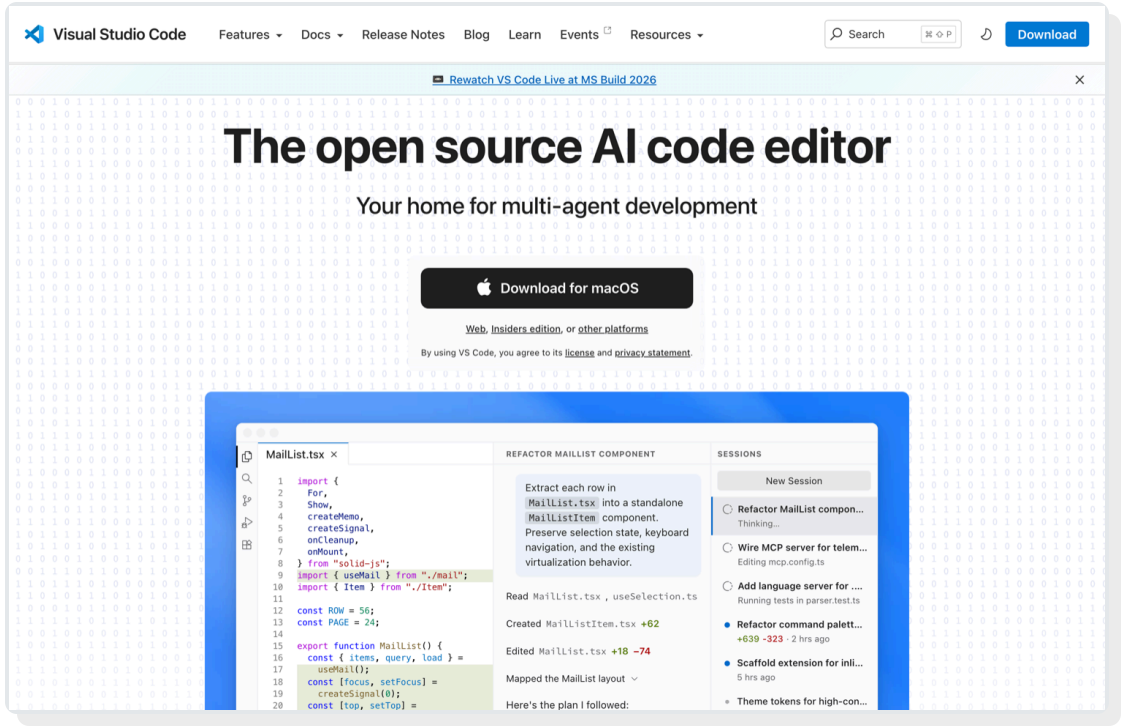
Spring Boot
課題Bのアプリ基盤

この4つ以外に、当日その場で入れていただくものはありません。

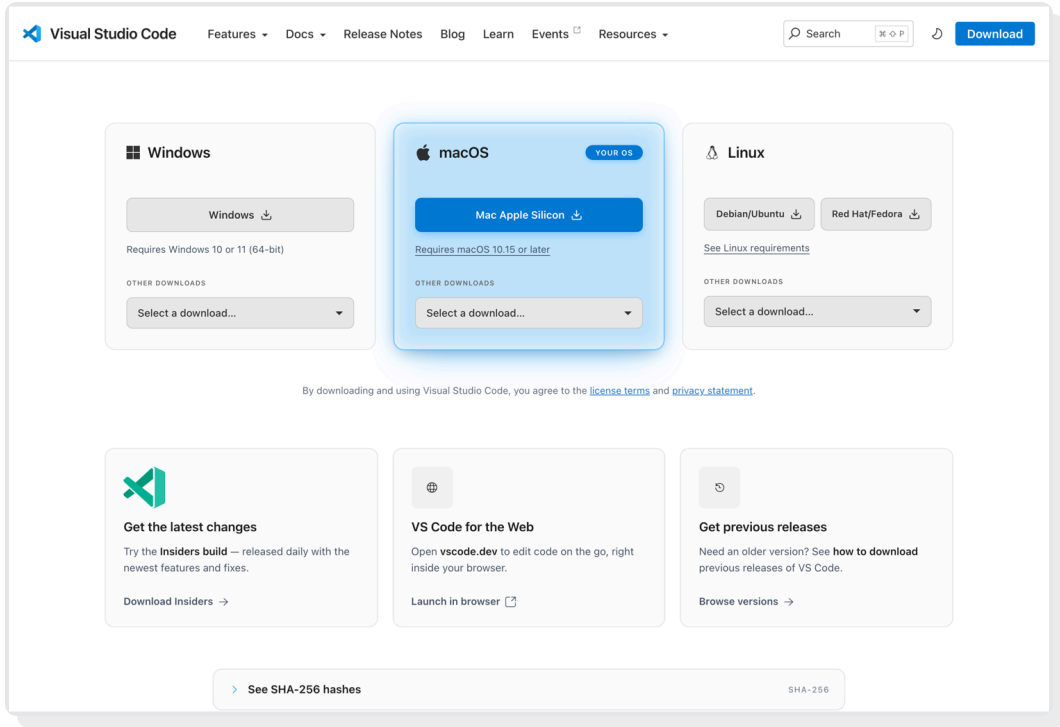
※ 各ロゴは各社の商標です。

編集画面はVS Code

VS Codeは公式サイトからOSを選んで入手します。



公式サイトの Download ボタン



OSに合う版を選ぶ画面

会場の貸出PCには導入済みです。自分のPCで進める方だけ、前日までに入れてください。

環境の前提

当日はOS・JDK 17・ネットワーク・モデル指定の4点が揃っている前提で進めます。

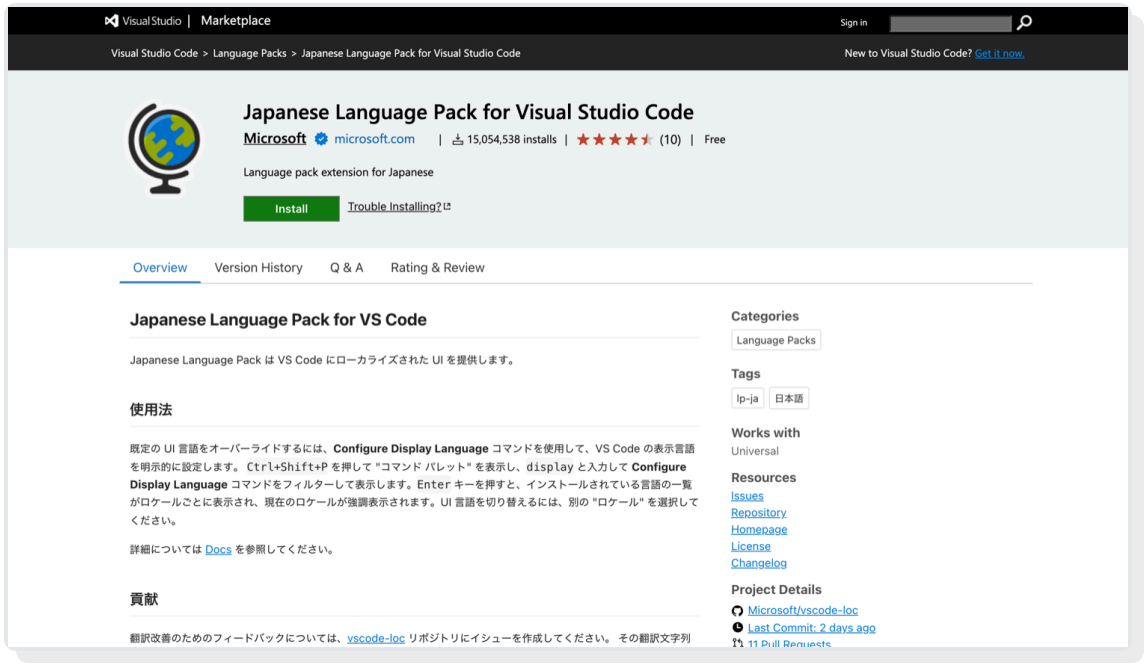
項目	必要な状態	確認方法
OS	Windows 11 または macOS	設定画面のバージョン表示
JDK	Java 17（Temurin）が導入済み	`java -version`
ネットワーク	社内ネットワークから接続できる	AIの応答が返ってくる
モデル指定	事務局が配布した設定のまま	各自の変更は不要

揃っていない項目があれば、開始前にお声がけください。その場で一緒に直します。

日本語表示の設定

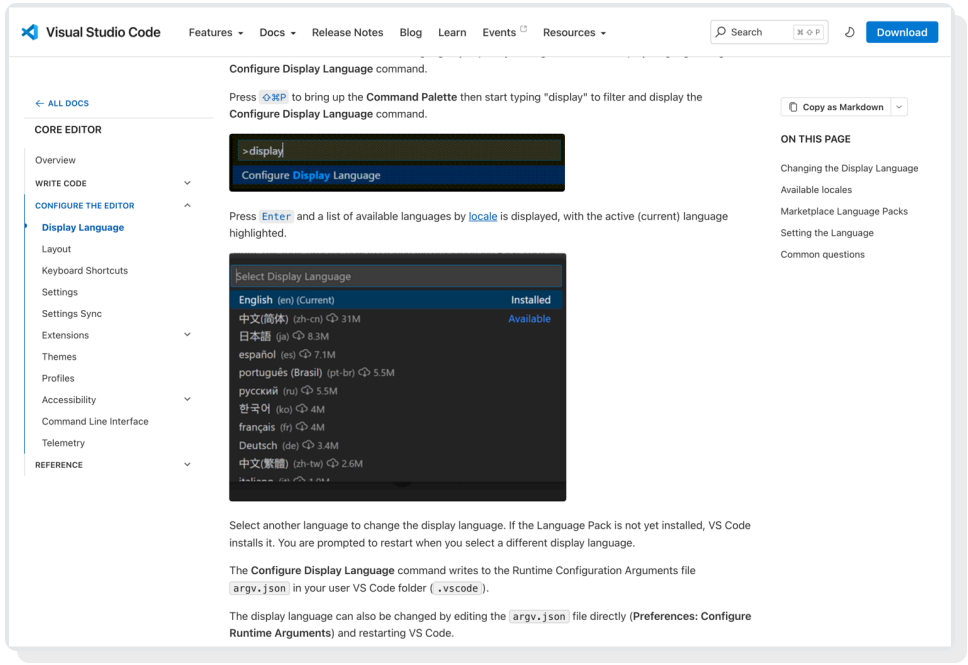
日本語パックを入れ、表示言語を切り替えておきます。

1 日本語パックを入れる



拡張機能ビューで検索して導入します

2 表示言語を日本語に切り替える



コマンドパレットから表示言語を選びます

英語表示のままでも進められます。ボタンの位置はどちらも同じです。

補足

事前セットアップの締切

事前セットアップは前日までに終わらせておいてください。

当日に環境構築から始めると、演習に使える時間がそのぶん減ります。

詰まったところは前日までに
事務局へご連絡ください。

前日までに済ませる3点

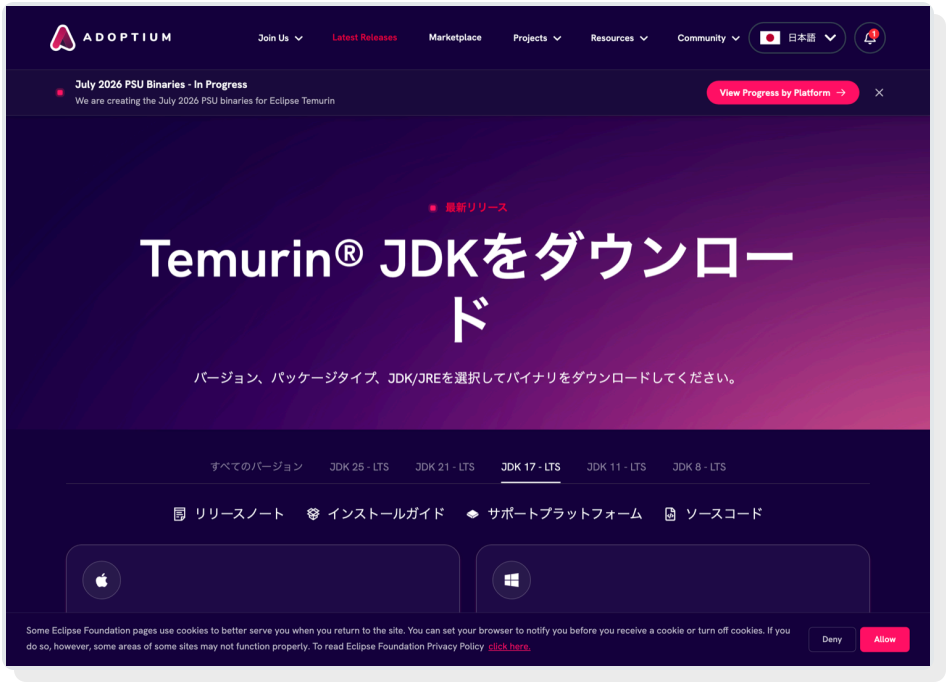
- VS Code の導入と日本語化
- Java 17 の導入（課題B用）
- 拡張機能の導入と起動確認

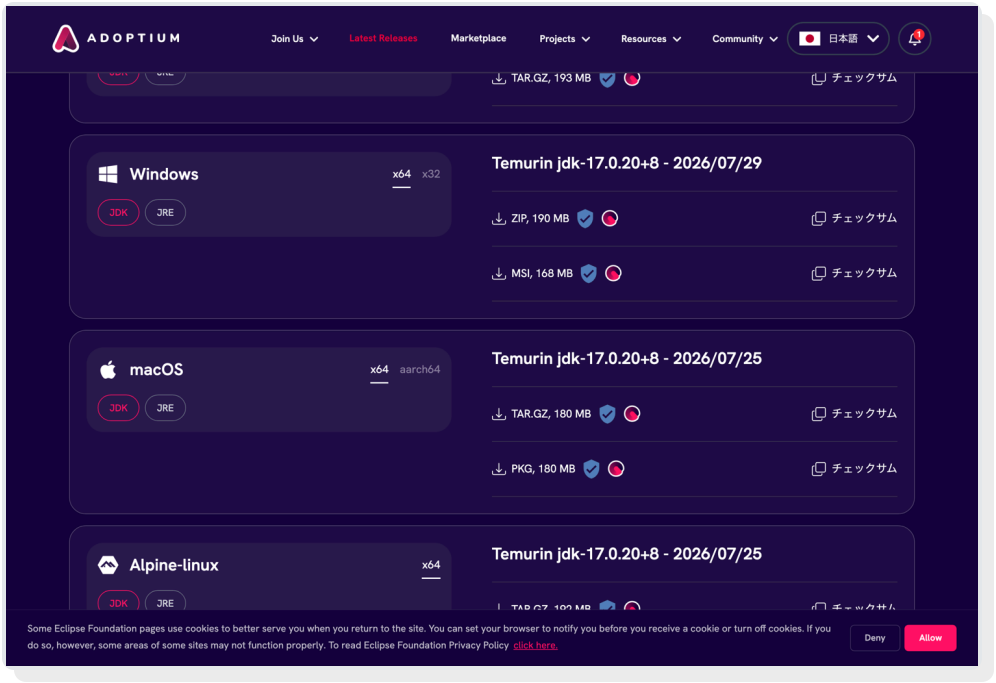
課題BにはJava 17

課題BはJava 17で動きます。

Temurin から対応する版を入手し、前日までに導入しておきます。

※ 各ロゴは各社の商標です。





Releases から Java 17 を選び、お使いのOSに合うインストーラーを入手します。

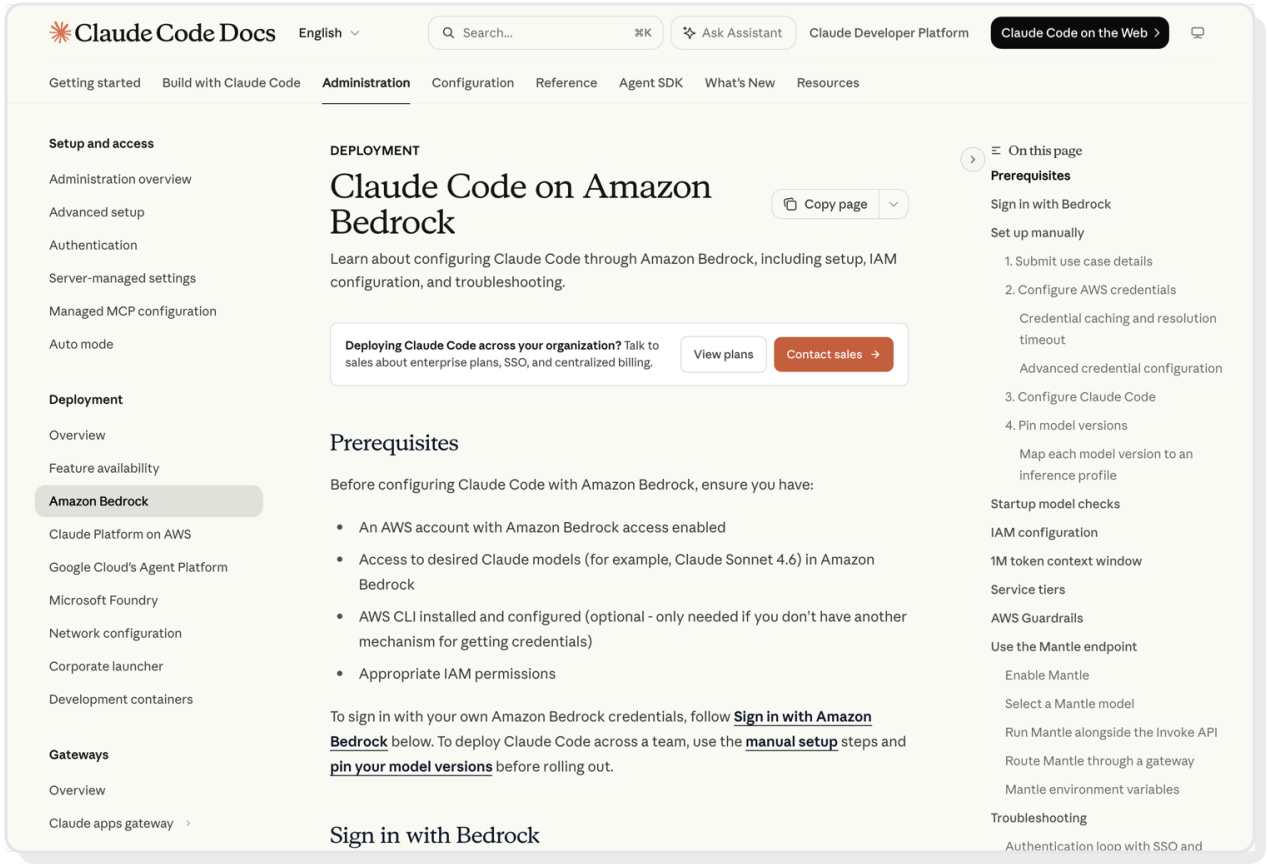
接続の経路

指示は VS Code の拡張から Amazon Bedrock を通り、Sonnet 4.6 に届きます。

入力した文は VS Code の中で完結せず、事務局が用意した AWS アカウントを経由してモデルに届きます。接続先を切り替える操作は当日ありません。

Bedrock経由の前提

Bedrock 経由では、AWS 側の
モデルアクセスの有効化が前提です。



画面: Claude Code 公式ドキュメント Amazon Bedrock ページ

1. **モデルアクセス**
   使うモデルを AWS 側で有効にしておきます。
2. **認証情報**
   AWS の資格情報が通っていることを先に確かめます。
3. **リージョン**
   有効化したリージョンを、環境変数で指定します。

補足

モデルと接続先

接続先とモデルは事務局が指定します。

各自での切り替えは要りません。
設定を書き換えると、
応答が返らなくなります。

触ってしまったときは手を挙げてください。元の値に戻します。

Claude Sonnet 4.6

※ 各ロゴは各社の商標です。

要確認 / 要記載
当日使う推論プロファイル ID と接続先リージョンの実値は、
事務局の確定後にこのページへ記載します。

呼び出し先はAWS上のBedrock

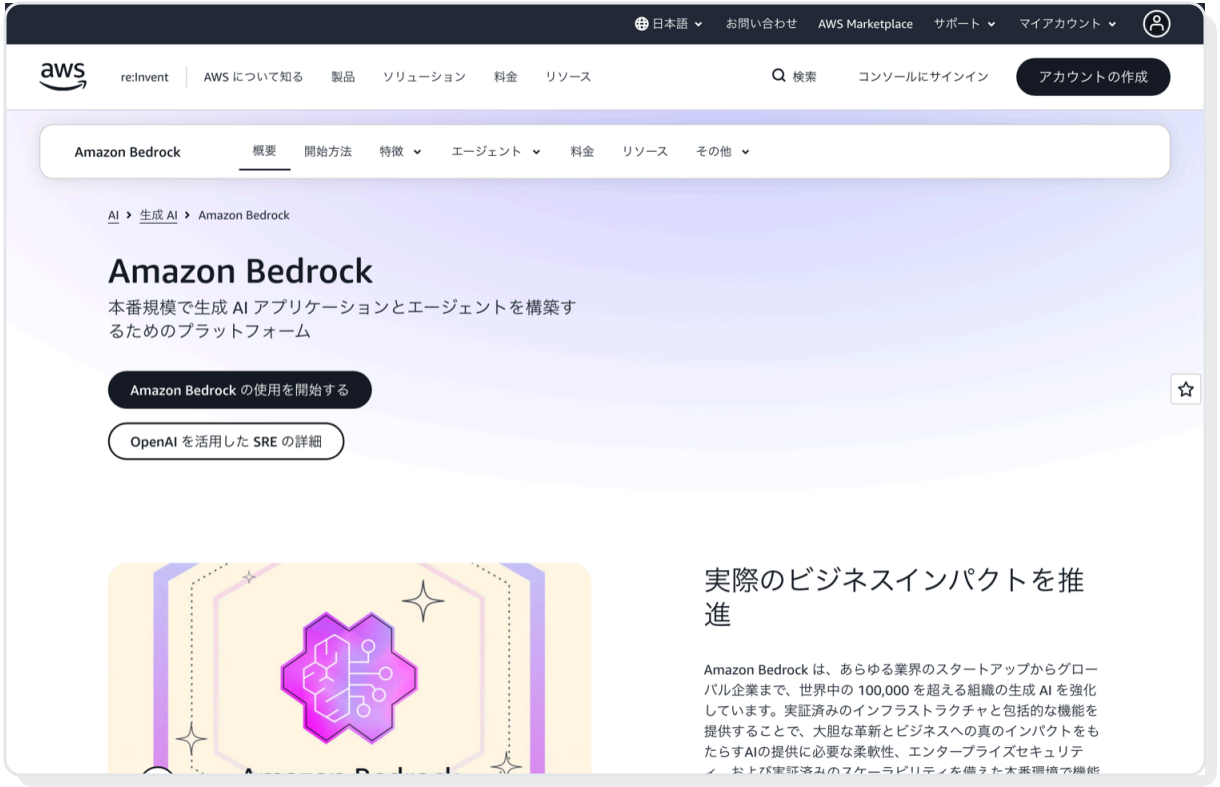
応答はAWS のアカウント内で処理されます。

経路は AWS の中

指示と応答は、事務局が用意した
AWS アカウントを通ります。

持ち出しの線引き

社外に出せない情報を貼る前に、
自社の運用ルールを確認してください。



画面: AWS の Amazon Bedrock 製品ページ

02

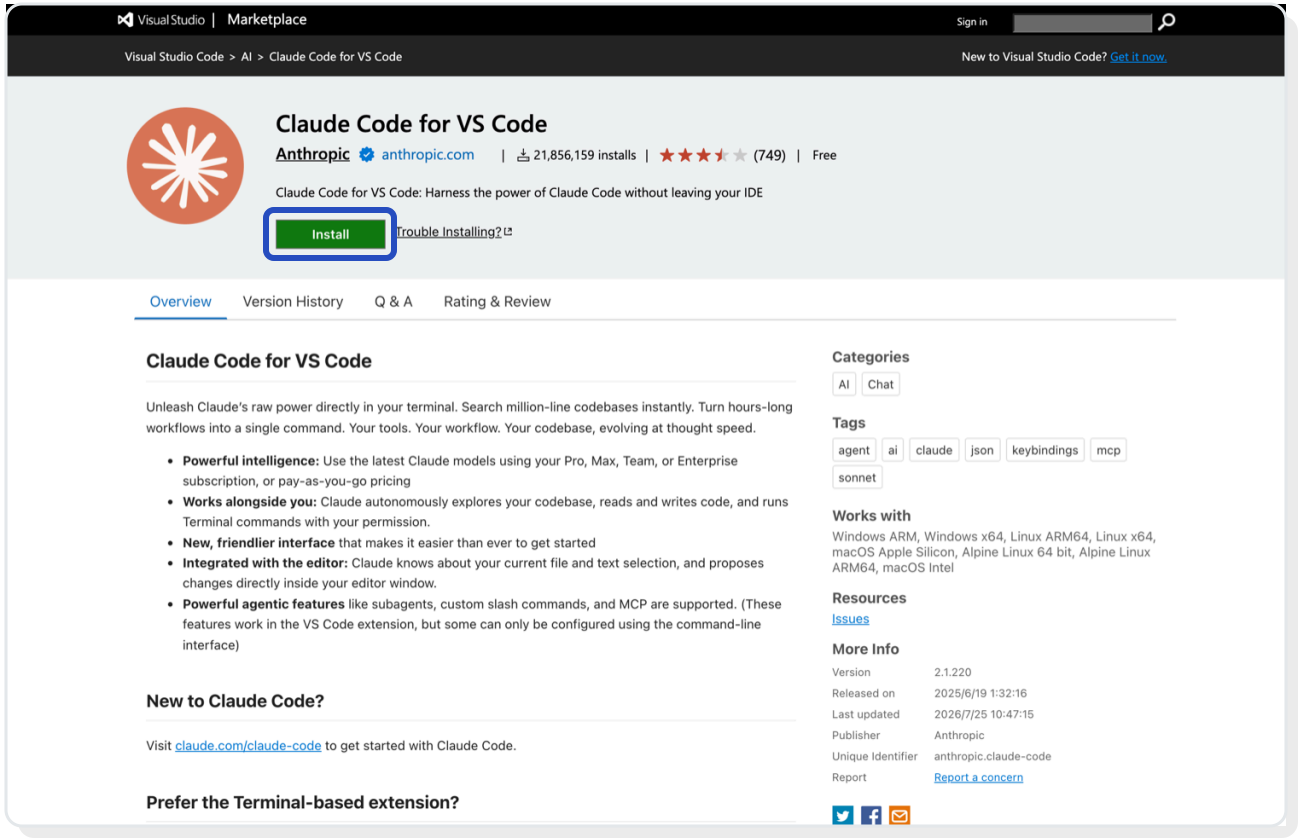
Claude Codeの基本操作

起動から、送る・切る・戻すまでを一度ずつ触ります

拡張機能の入手

Claude Code はVS Code の拡張として入ります。

VS Code



Claude Code

囲みの Install を押すと、VS Code 側に拡張が入ります。

※ 各ロゴは各社の商標です。

画面: Visual Studio Marketplace の Claude Code for VS Code

起動の4手順

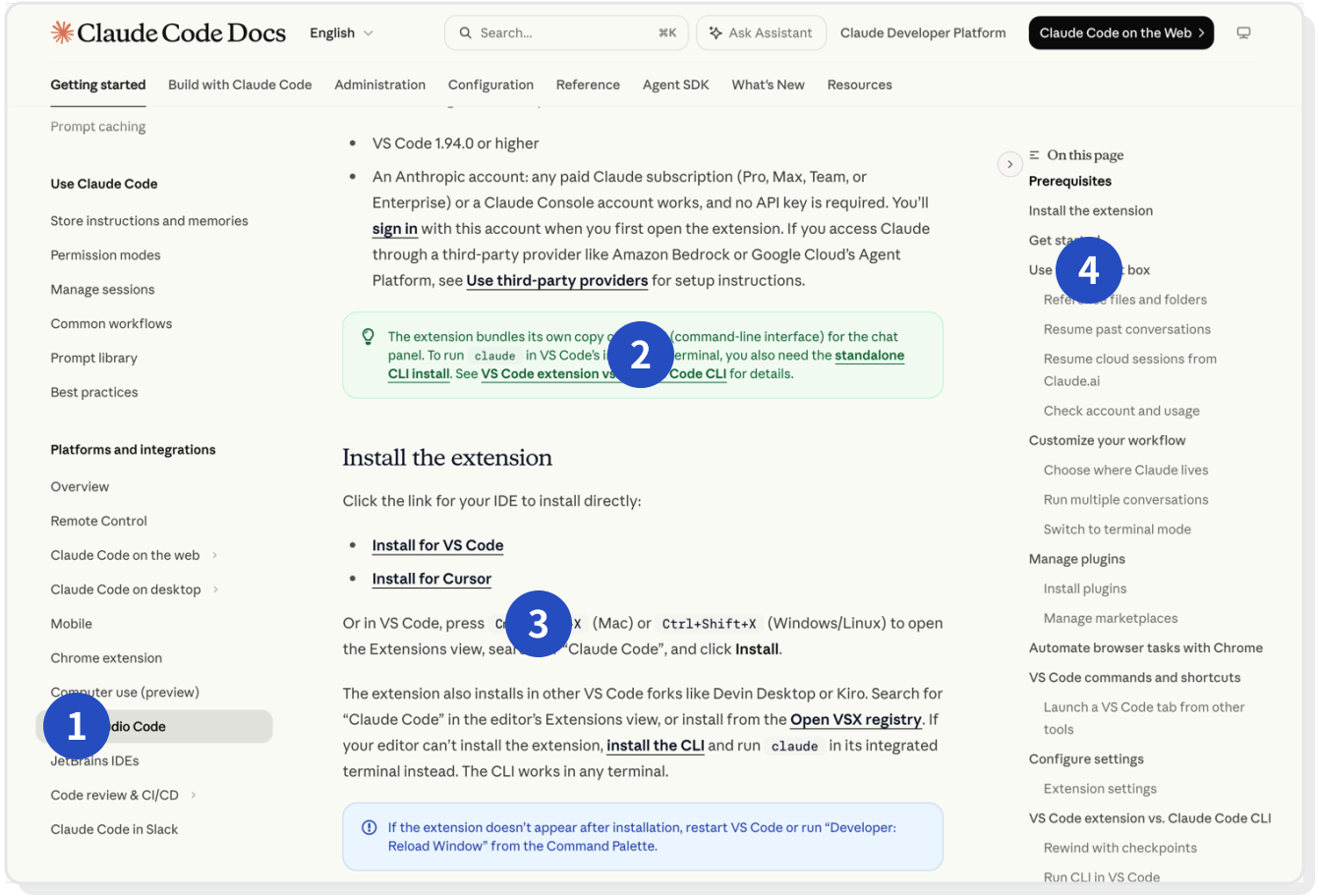
アクティビティバーのアイコンからパネルを開き、1文送るところまでが起動です。

1	**フォルダを開く**	VS Code で handson フォルダをそのまま開きます
2	**アイコンを押す**	アクティビティバーの Claude Code アイコンを押します
3	**起動を待つ**	入力欄が出るまで待ちます。初回は少し時間がかかります
4	**1文送る**	「今のフォルダに何がありますか」と送って応答を見ます

実演

パネルの位置と各部

パネルは編集画面の横に開くので、
コードを見ながら会話できます。



画面: Claude Code 公式ドキュメント Visual Studio Code ページ

1. **左ナビの現在地**
   VS Code 向けの手順
2. **チャットパネル**
   拡張だけで動きます
3. **拡張ビューの近道**
   Cmd+Shift+X で開く
4. **入力欄の使い方**
   右の目次からたどれます

最初に送る1文

最初は短い1文で、応答が返ることだけを確かめます。

パネルに送る文

今のフォルダにどんなファイルがありますか。一覧で教えてください。

返ってくる形

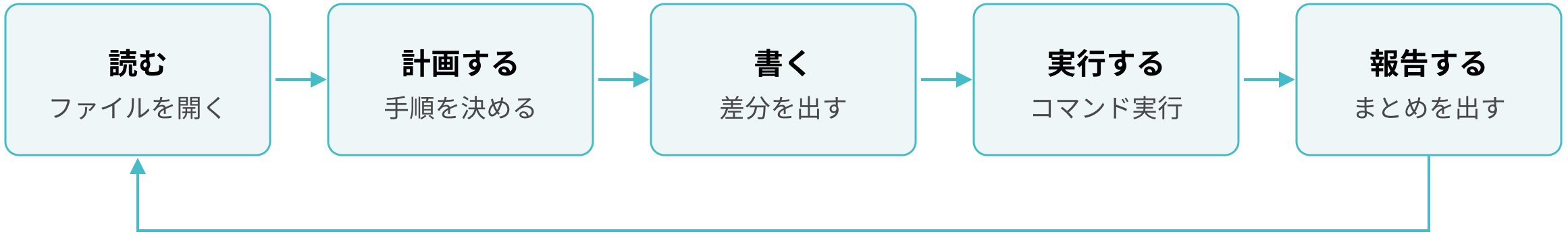
フォルダの中にあるファイルを一覧で返します。
そのあと、何をしたいかを聞き返してきます。

> いきなり本題を頼まず、まず経路が通っていることだけ確かめます。
> 返ってこないときは、パネルを開き直してから同じ文をもう一度送ります。

実演

AIの1サイクル

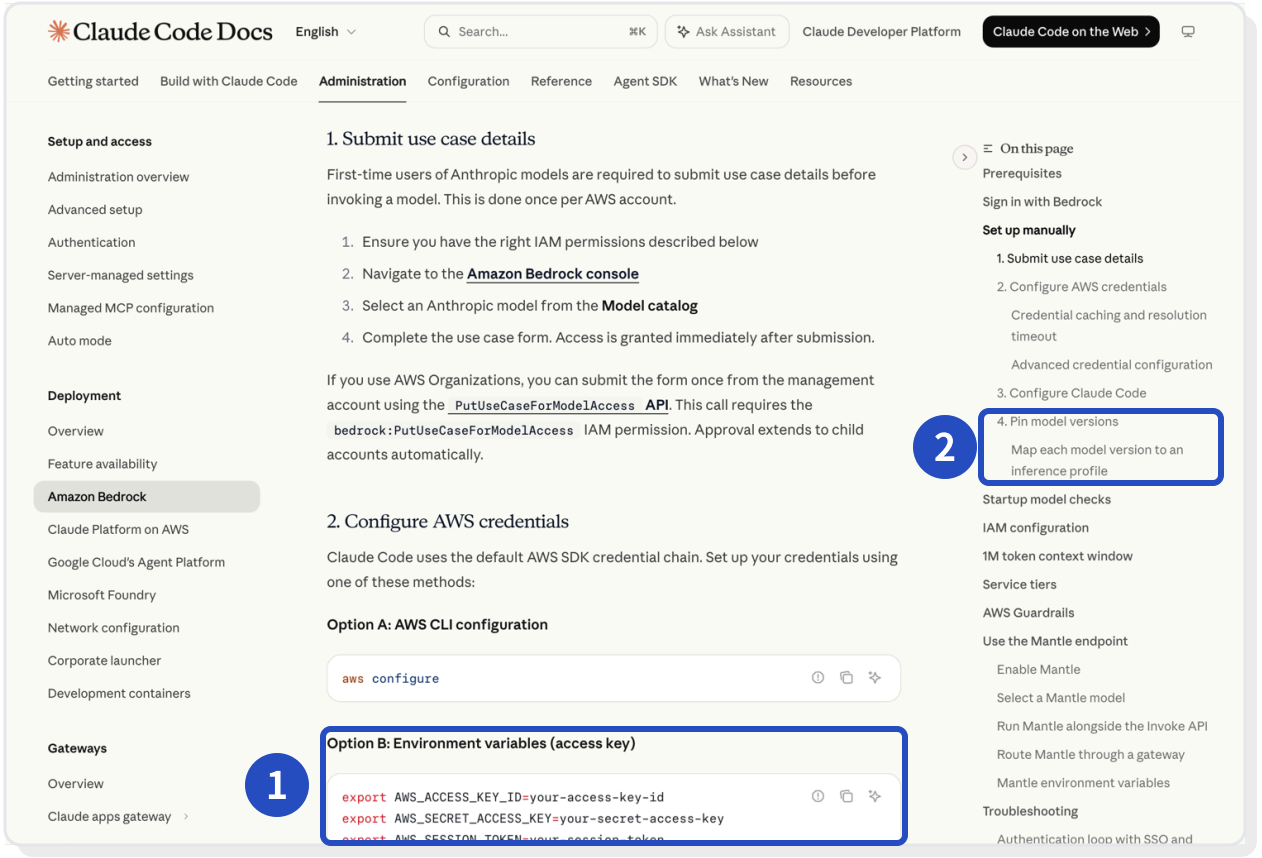
読む・計画・実装・実行・報告の5段が、毎回1周します。



最後の報告だけ読んで終わりにしがちですが、計画の段でいったん止めて
方針を返すほうが、あとの手戻りが減ります。

モデルと接続の指定箇所

モデル指定は設定ファイルと環境変数で決まります。



1 認証情報とリージョンは、環境変数で渡します。

2 使うモデルは、推論プロファイルを指定して固定します。

画面: Claude Code 公式ドキュメント

よく使う操作

今日使う操作は4つだけです。
残りは日本語で頼めば足ります。

ファイルを指す `@ファイル名`

開いていないファイルも読み込ませます

会話を切る `/clear`

直前までのやり取りを忘れさせます

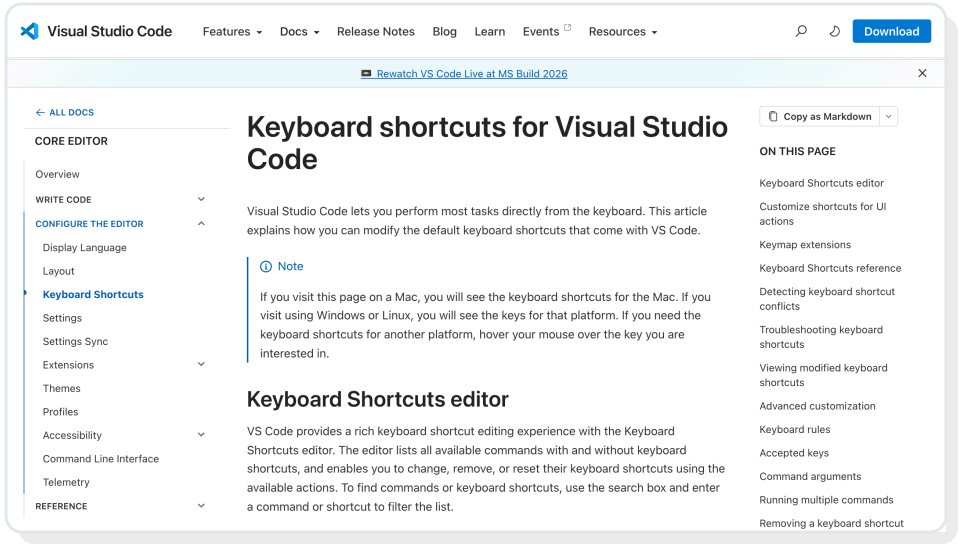
モードを変える `Shift+Tab`

Plan と Agent を切り替えます

差分を確かめる `提案の差分を読む`

受け入れる前に変更点を確認します

@ は入力欄で打つと候補が出ます。パスを覚えていなくても選べます。



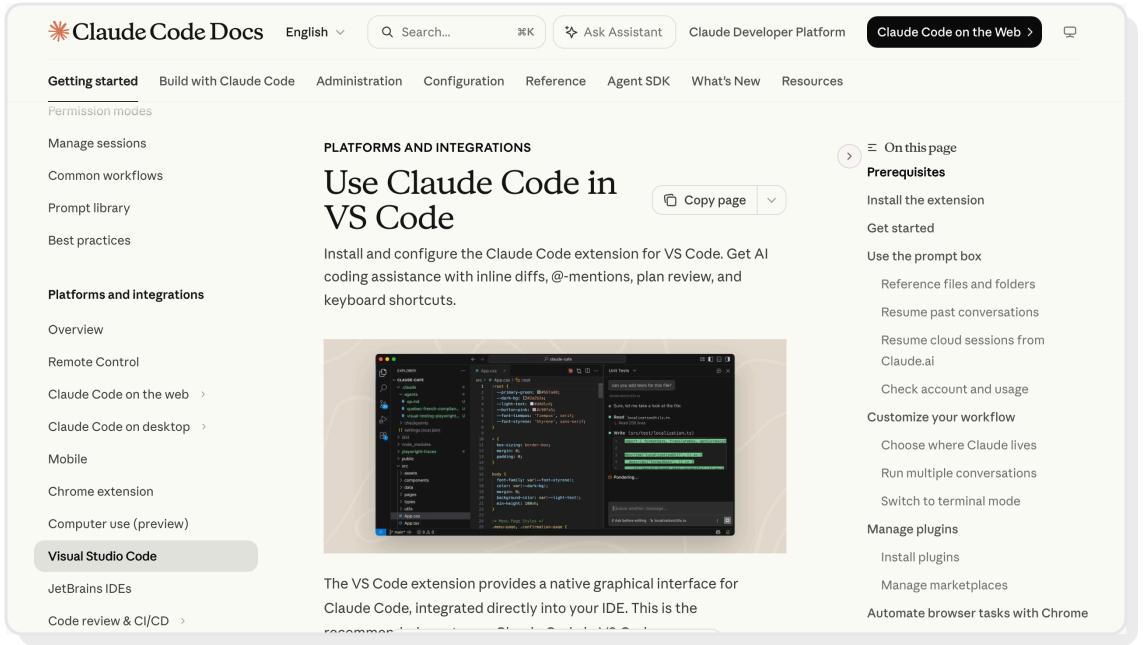
画面: VS Code 公式ドキュメント ショートカット

割り当ての一覧は VS Code 側で
確認できます。当日は変更しません。

解説

渡していない情報

AI に渡していない情報は、無いのと同じ扱いになります。



画面: Claude Code 公式ドキュメント VS Code 版

開いていないファイルの中身は、AI には見えていません。

口頭で説明したつもりでも、
パネルに書いていなければ
渡せていません。

渡すときの書き方

```
@src/main/java/com/example/service/ItemService.java
この中の countByCategory を読んでから答えてください。
```

クイズ

Q2 文脈のリセット

Q. /clear を送ったあと、次に送る文ではどうなりますか。

- **A** CLAUDE.md は読み直され、直前までの会話は引き継がれない
- **B** CLAUDE.md も直前までの会話も、どちらも引き継がれる
- **C** CLAUDE.md も直前までの会話も、どちらも引き継がれない
- **D** 直前までの会話は引き継がれ、CLAUDE.md だけ読み直されない

正解は次のページで確認します。まず自分で考えてみてください。

解説

正解はA

/clear は会話を切りますが、CLAUDE.md は読み直されます。

A **正解 CLAUDE.md は読み直され、直前までの会話は引き継がれない**
CLAUDE.md はパネルの起動時に読まれるので、会話を切っても前提は残ります。

B CLAUDE.md も直前までの会話も、どちらも引き継がれる
会話まで残るなら、/clear を送る意味がなくなります。

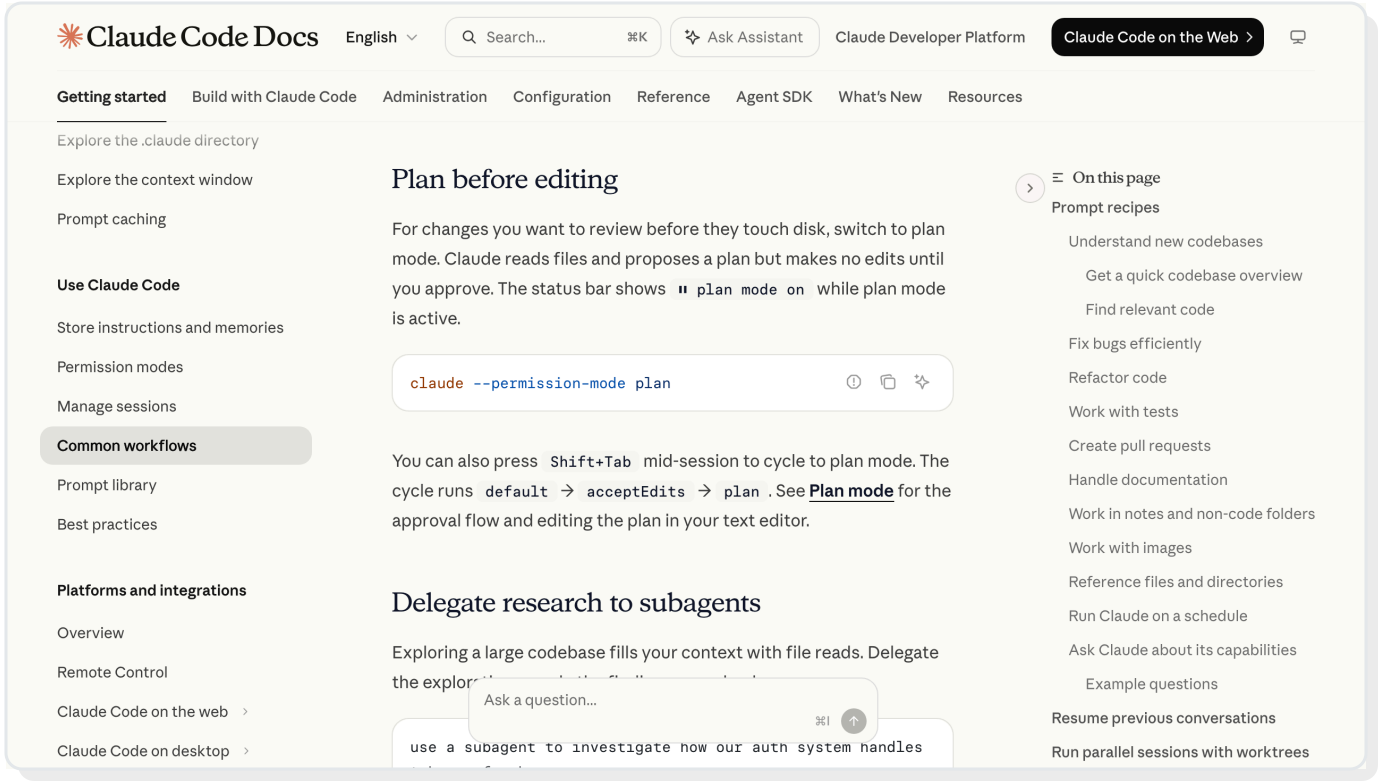
C CLAUDE.md も直前までの会話も、どちらも引き継がれない
設定ファイルは会話とは別に読まれるので、消えません。

D 直前までの会話は引き継がれ、CLAUDE.md だけ読み直されない
消えるのは会話のほうです。役割が入れ替わっています。

実務でのポイント
話が長くなって的が外れたら切ります。プロジェクトの前提は書き直さずに済みます。

PlanモードとAgentモード

Plan は調べて計画を返し、Agent はその場で書き換えます。



画面: Claude Code 公式ドキュメント Plan before editing

Planモード

調べて、計画だけを返す
ファイルは書き換えない
初見のコードを読むとき

Agentモード

その場でファイルを書く
差分を出して反映まで進む
直す場所が決まっているとき

切り替えは Shift+Tab

迷ったら Plan から入って、方針が固まってから Agent に切り替えます。

初見のコードの扱い

初見のコードと広い範囲の変更は、まず計画から入ります。

> **この2つに当てはまったら、書かせる前に Plan モードで計画を出させます。**

1. **初めて触るコード**
   どこに何があるか分からないまま直させると、関係のない場所まで書き換わります。
2. **複数のファイルにまたがる変更**
   先に対象ファイルとメソッド名を並べさせると、抜けと余計な変更が減ります。

1行の小さな直しは、計画を挟まずそのまま頼んで構いません。

演習Aの進め方

最初は設定ファイルが1つも無い状態で作ります。
設定は演習の途中で配ります。

handson/

- kadaiA/ — 作るアプリの置き場
- docs/ — 3つの規約
- hints/ — 詰まってから開く
- memo.md — 気づきを書く

CLAUDE.md **.claude/**

A-3 の号令でここへ置きます。それまでは空です。

進め方の約束

配布フォルダはそのまま使う
開けない場合はその場で手を挙げてください。

気づいたことは memo.md に書く
提出はありません。読み返すための記録です。

設定ファイルは号令で配置する
A-3 まで CLAUDE.md と .claude は触りません。

終わらなくても次へ進む
7ステップで70分です。止まったら手を挙げます。

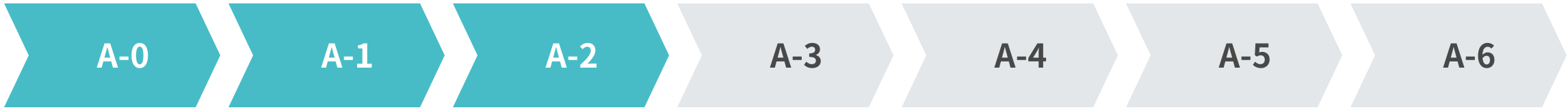
03

演習A 素の状態で作る

A-0 から A-2 まで、設定を1つも置かないままアプリを立ち上げます

演習Aの7ステップ

A-0 から A-6 まで、素の状態からハーネス投入、そして比較まで進みます。



A-0	A-1	A-2	A-3	A-4	A-5	A-6
素の確認 [5min]	アプリ作成 [15min]	懸念の洗い出し [8min]	ハーネス配置 [5min]	サマリー確認 [7min]	パイプライン [20min]	比較と考察 [10min]

章3で扱うのは A-2 までです

A-3 以降のハーネス投入と比較は、章4で扱います

各ステップの終わりに memo.md へ1行書き足しておくと、A-6 の比較がそのまま書けます。

手を動かす時間は7ステップ合計で70分です。

出発点

配布フォルダに CLAUDE.md と .claude は、まだ置かれていません。

```
handson/
├ kadaiA/
│  ├ docs/お題シート.md
│  └ dummy_data.csv
├ kadaiB/
├ exercises/
├ _harness_kit/
└ memo.md
```

CLAUDE.md と .claude
A-3 でここに置きます

設定一式は _harness_kit/ の中で
待機しています。
いま置くと、素の状態との差が
見えなくなります。
A-3 で、全員そろえて配置します。

配布ZIPを展開した直後の状態です。

ワーク

A-0 素の状態を確認する

[5min]

設定が1つも無いことを目で確かめ、差を測る基準点を作ります。

操作

1. VSCode で handson フォルダを開きます。
2. エクスプローラで直下を見て、CLAUDE.md と .claude が無いことを確認します。
3. Claude Code パネルに「いま参照している設定ファイルがあれば教えてください」と送ります。
4. kadaiA/docs/お題シート.md と kadaiA/dummy_data.csv を開いて中身を見ます。

OK基準

- handson 直下に CLAUDE.md と .claude が無いことを確認できた
- 「参照している設定ファイルはありません」に相当する回答が返った
- memo.md に、AI が知らないと思うことを3項目以上書けた

生成物　handson/memo.md の ## A-0 見出しの下（3行以上）

A-0で送る1文

参照している設定があるかを、そのまま聞きます。

Claude Code に送る文

いま参照している設定ファイルがあれば教えてください

返ってくる応答の例

参照している設定ファイルはありません。
CLAUDE.md や .claude ディレクトリは見つかりませんでした。

> 文面は毎回同じにはなりません。参照している設定は無い、という趣旨が
> 返っていれば A-0 は到達です。

課題Aで作るもの

作るのは一覧・検索・詳細の3画面です。

データは dummy_data.csv の50件、カテゴリは5種類です。

50件

dummy_data.csv の行数

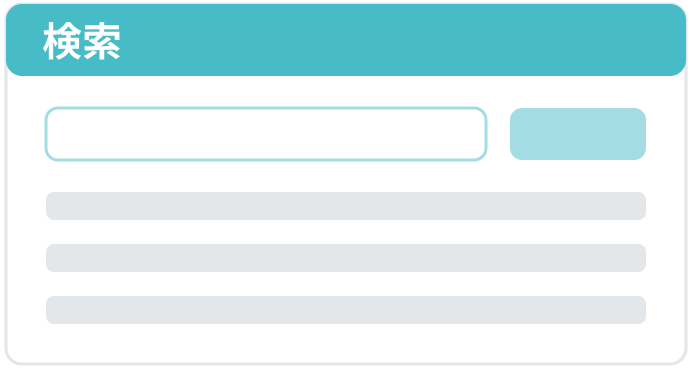
5種類

カテゴリの種類数

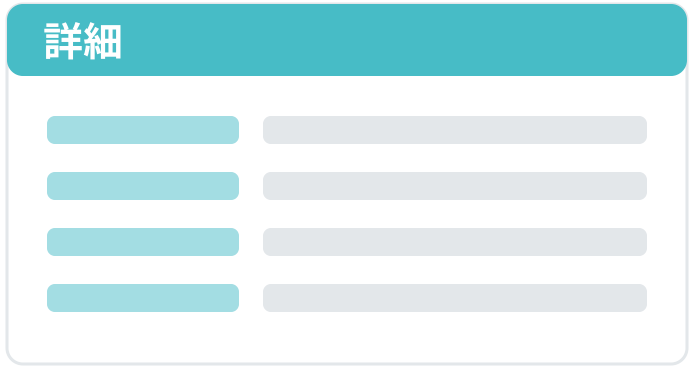
3画面

一覧・検索・詳細

50件がそのまま並ぶ



備品名で絞り込む



1件の全項目を見る

ワーク

A-1 CRUD業務アプリを作る

[15min]

指示を3回に分けて、動くアプリを1本立ち上げます。

操作

1. 1回目の指示で、備品の一覧画面を kadaiA/index.html として作らせます。
2. ブラウザで kadaiA/index.html を開き、一覧に50件が並ぶことを確認します。
3. 2回目の指示で、備品名で絞り込める検索ボックスを付けさせます。
4. 3回目の指示で、1件クリックの詳細表示と一覧への戻りを足させます。
5. 表示が崩れていたら「表を見やすく整えてください」と差分で伝えます。

OK基準

- 一覧に備品が50件表示される
- 備品名の一部を入力すると件数が絞り込まれる
- 1件クリックで詳細が表示され、一覧に戻れる
- index.html をダブルクリックするだけで動く

生成物　`handson/kadaiA/index.html`

3回に分けた指示

1回で全部を頼まず、途中で自分の意図を足していきます。

1回目

`@kadaiA/dummy_data.csv` を読み込んで、備品の一覧を表示する画面を
`kadaiA/index.html` として作ってください。ブラウザで開くだけで動く形にしてください。

2回目

備品名で絞り込める検索ボックスを付けてください。

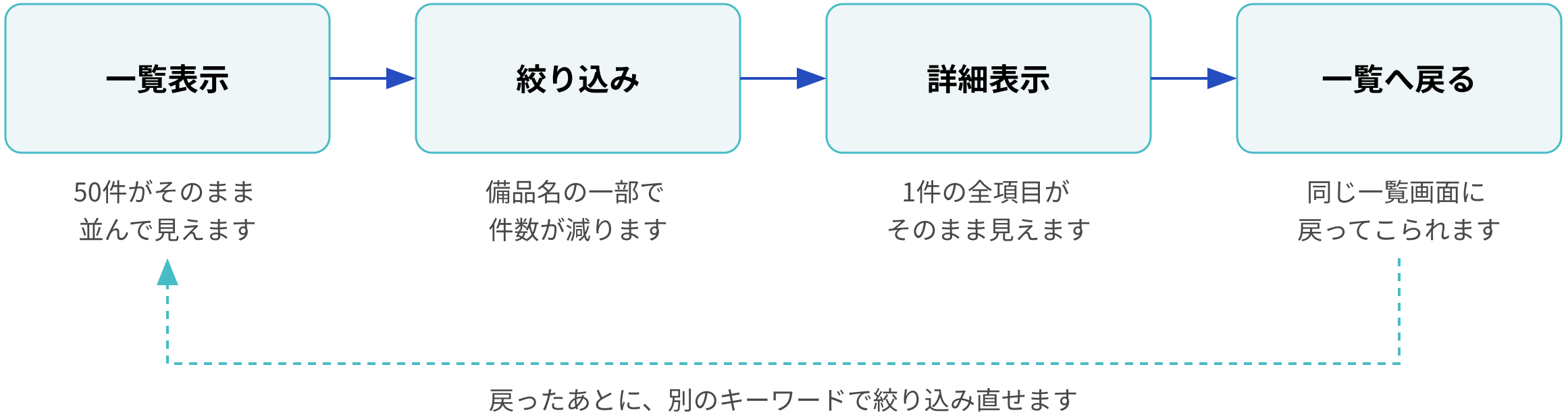
3回目

一覧の1件をクリックすると、その備品の全項目が見える詳細表示に
切り替わるようにしてください。詳細から一覧に戻れるようにもしてください。

1回目に添えた「ブラウザで開くだけで動く形」の一文が、実装のやり方を決めます。

課題Aの画面遷移

一覧から検索で絞り、1件を開いて
一覧へ戻れれば到達です。



この4つがつながって動けば、A-1 は到達です。

クイズ

Q3 指示に添えた一文

Q. 1回目の指示に添えた
「ブラウザで開くだけで動く形に」の役割はどれですか。

- **A** 生成するファイルを1つに限定する
- **B** CSSとJavaScriptを外部ファイルに分ける
- **C** 読み込むデータの件数を50件に固定する
- **D** サーバー起動が要らない実装を選ばせる

正解は次のページで確認します。まず自分で考えてみてください。

解説

正解はD

正解はD。サーバー起動が要らない実装を選ばせます。

D **サーバー起動が要らない実装を選ばせる**

ローカルのHTMLから fetch でCSVを読むと、ブラウザの制限で失敗します。

A 生成するファイルを1つに限定する

1ファイルになることはありますが、指示が求めているのは動く形です。

B CSSとJavaScriptを外部ファイルに分ける

分けても、ダブルクリックで開いたときに動くかどうかは変わりません。

C 読み込むデータの件数を50件に固定する

件数は dummy_data.csv 側で決まっていて、指示の一文では変わりません。

実務のポイント

動かす条件を一文添えるだけで、AI が選ぶ実装方式が変わります。

バイブコーディングの実感

速さはすぐ手に入りますが、抜けは画面には出てきません。

> **画面が動いていても、データの取りこぼしや**
> **入力の想定漏れは、そのまま残ります。**

- 入力に想定外の値が入ったときの動き
- 50件が本当に全部そろって出ているかどうか
- 半年後の自分が読んで直せるかどうか

この続きは A-2 で、AI に聞く前に自分の言葉で書き出します。

ワーク

A-2 懸念点を自分で洗い出す

[8min]

AI に聞く前に、自分の言葉で5個書き出します。

操作

1. memo.md の ## A-2 に、いま作ったアプリで不安な点を5分間で5個以上書き出します。
2. 書き終えてから、@kadaiA/index.html の危険な点と曖昧な点を深刻な順に指摘させます。
3. AI の指摘と自分の5個を突き合わせ、差分を1行書き足します。

手が止まったときに眺める観点

データの扱い　入力値　エラー時の挙動　他人が読めるか　明日の自分が直せるか

OK基準

- 自分の言葉で5個以上書けた
- AI の指摘が3件以上返ってきた
- 自分と AI の差分を1行以上書けた

生成物　handson/memo.md の ## A-2 見出しの下

発展

自分の指摘とAIの指摘

自分が気づいてAI が触れなかった項目が、文脈を知る人にしか出せない指摘です。

自分が書いた5個

- そのプロジェクトの事情から出る
- 運用の都合や過去の失敗が入る
- 数は少なく、粒度もそろわない
- 業務側の前提を織り込める

差分

AI が返した指摘

- コードから読み取れる範囲は速い
- 件数は多いが、毎回同じにならない
- 業務上の前提は入っていない
- 深刻な順に並べ替えられる

両方に出た項目より、片方にしかない項目を見てください。自分だけが挙げた行に印を付けて、memo.md に残します。

文脈のない指摘

同じ質問を送っても、返ってくる指摘は毎回ちがいます。

隣の受講者と内容が違っても、失敗ではありません。

粒度がばらつくのは、AIに文脈を渡していないからです。ここまでが素の状態です。
この章では、その文脈をファイルにして渡し、毎回同じ観点で見てもらう形に変えます。

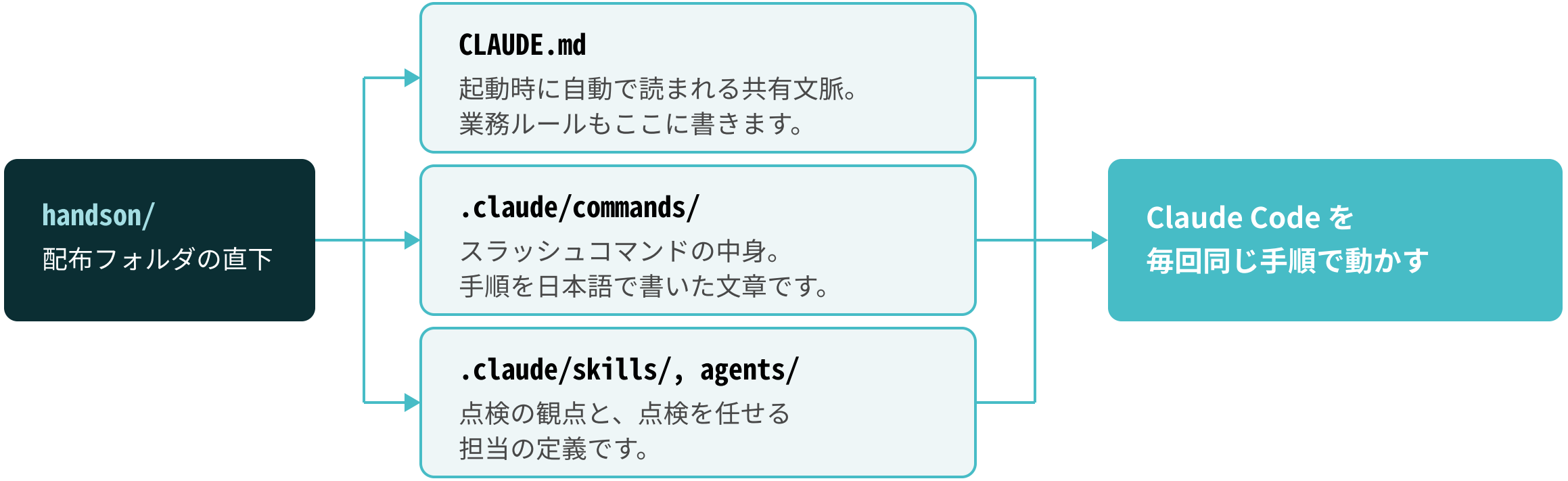
04

ハーネスを入れて回す

設定ファイルを置いて、コマンド1本で点検から報告まで回します

ハーネスとは

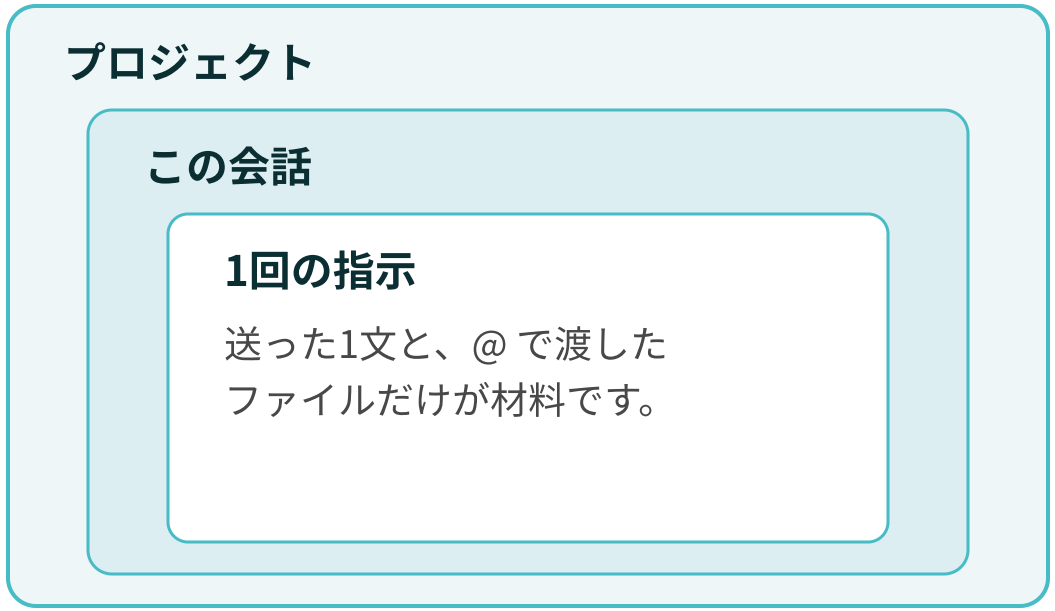
ハーネスは、AIを毎回同じ手順と同じ観点で動かすための設定一式です。



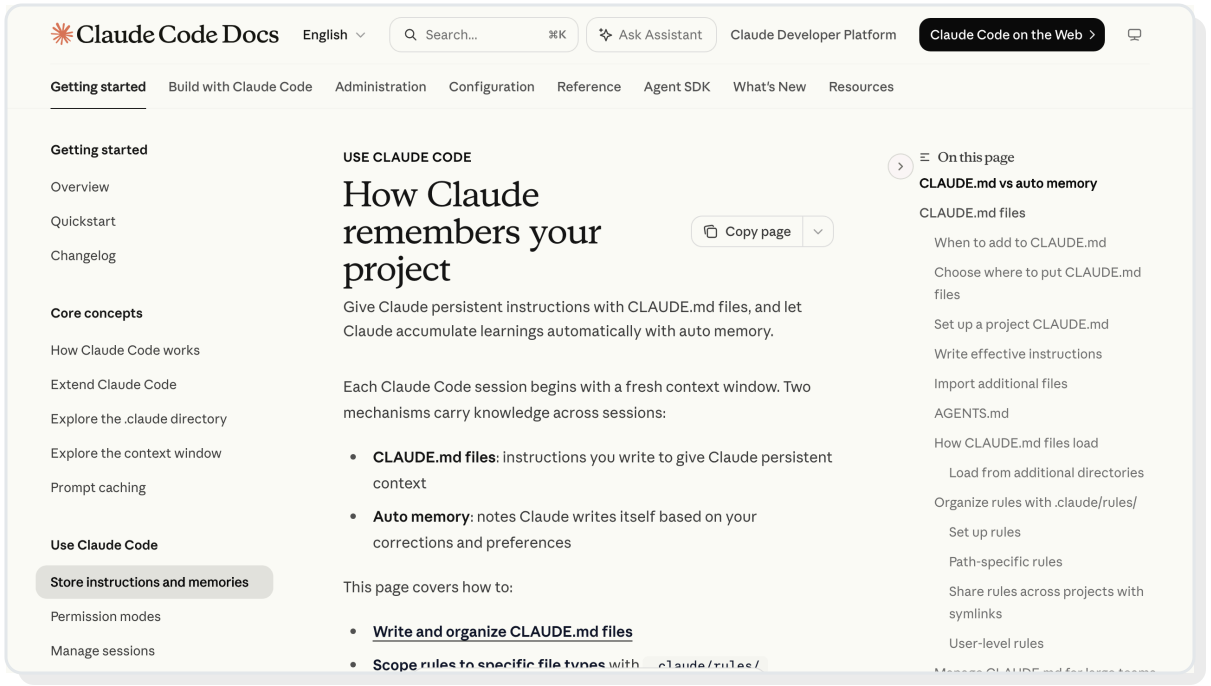
3つとも handson 直下に置きます。
場所が決まっているから、AIが自分で見つけます。

コンテキストの3層

プロジェクト・この会話・1回の指示の3層で、AIが見える範囲が決まります。



外側ほど長く効き、内側ほどその回だけに効きます。
渡していない情報は、無いのと同じです。



画面: Claude Code 公式ドキュメント CLAUDE.md の解説

CLAUDE.md
起動のたびに読まれます

直前までのやり取り
/clear を送ると切れます

今回のプロンプト
@ で名指しした分だけ渡ります

ワーク

A-3 ハーネスを配置する

[5min]

設定一式を handson 直下にコピーし、パネルを開き直します。

操作

1. _harness_kit/step1_kadaiA/ を開く
2. CLAUDE.md と .claude を handson 直下へ
3. 2つが並んだことを目で確認する
4. Claude Code パネルを開き直す
5. 参照している設定の要点を聞く

OK基準

- ✓ handson 直下に2つが置かれた
- ✓ CLAUDE.md の中身に触れた回答
- ✓ 末尾に実行サマリーが出ている

生成物

```
handson/CLAUDE.md
handson/.claude/
```

自分で考える

A-0 で同じ質問をしたときの回答と、
いまの回答は何が違いますか。

発展課題

.claude/settings.json を開き、どの設定が
どの動作を縛っているか読み取ります。

CLAUDE.mdの中身

技術スタック・業務ルール・してはいけないこと・実行サマリー
この4節とも日本語の文章で、プログラムは1行もありません。

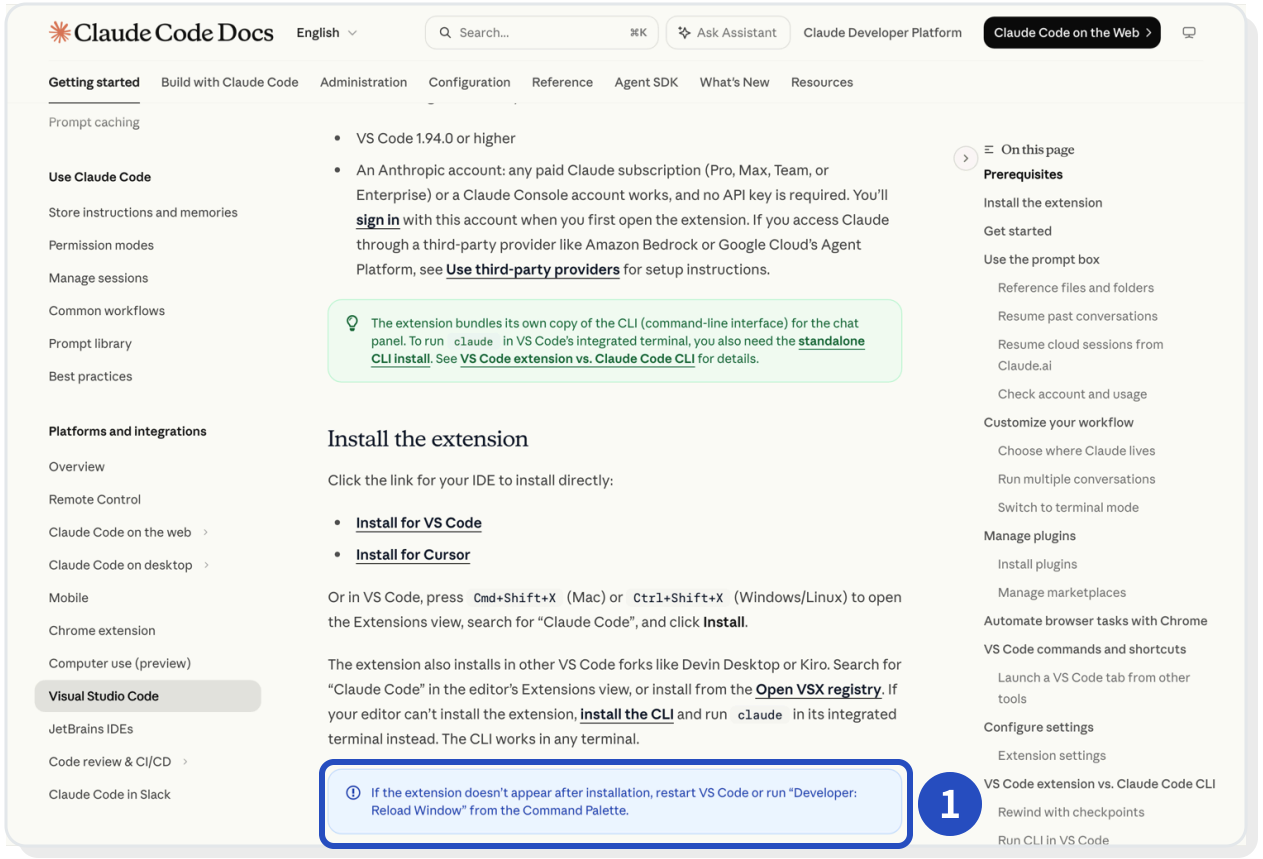
```
# 課題A 備品管理アプリ
## 技術スタック
HTML / CSS / JavaScript のみ。ブラウザで開くだけで動く形にします。
## 業務ルール（抜粋）
在庫が5個以下の備品は目立たせる。廃棄済みは一覧に出さない。
日付は YYYY-MM-DD で表示する。
## してはいけないこと
外部CDNの読み込み。実在しそうな個人情報のダミー生成。
## 応答の最後に必ず出すもの
動作、従った設定、対象ファイル、未実施・保留の4項目を付ける。
読み取りだけの依頼でも省略しない。
```

実物は handson/CLAUDE.md です。A-3 でコピーしたのは、このファイルです。

実演

パネルを開き直す

設定を読むのは起動のときだけです。
開き直すまで反映されません。



画面: Claude Code 公式ドキュメント Visual Studio Code のページ

開き直しの手順

1. パネル右上の × で閉じる
2. アイコンをもう一度押す
3. 起動時に CLAUDE.md が読まれる

1 拡張が出てこないときの案内

入れ直しではなく、VS Code か
ウィンドウの読み込み直しで直ります。

クイズ

Q4 CLAUDE.mdの読み込み

Q. handson 直下に CLAUDE.md を置きました。
中身をAIに反映させるには何が必要ですか。

- **A** 依頼のたびに「CLAUDE.md を読んで」と書き添える
- **B** パネルを開き直せば、起動時に自動で読み込まれる
- **C** /clear を送ると、置いたファイルが読み込まれる
- **D** settings.json に読み込むファイル名を登録する

正解は次のページで確認します。まず自分で考えてみてください。

解説

正解はB

正解はB。CLAUDE.md は起動時に自動で読まれます。

B **パネルを開き直せば、起動時に自動で読み込まれる**
読み込みは起動のときだけ。だから置いたあとに開き直します。

A 依頼のたびに「CLAUDE.md を読んで」と書き添える
書き添えなくても読まれます。毎回書く手間は要りません。

C /clear を送ると、置いたファイルが読み込まれる
/clear は会話を切る操作で、読み込みの引き金ではありません。

D settings.json に読み込むファイル名を登録する
登録は要りません。置き場所が決まっているので自動で見つかります。

現場での効き目

同じ CLAUDE.md を配れば、チーム全員が同じ前提でAIを動かせます。

ワーク

A-4 実行サマリーを確認する

[7min]

応答の最後に出る4項目を読み、実ファイルと突き合わせます。

操作

1. @kadaiA/index.html を指定して依頼
2. いちばん下の実行サマリーを読む
3. 書かれたパスを開いて照合する
4. ブラウザを再読込して表示を見る

OK基準

- ✓ 末尾に4項目が出ている
- ✓ 対象ファイルと実際の変更が一致
- ✓ 在庫5個以下の行が目立つ表示に

生成物

handson/kadaiA/index.html
memo.md の A-4 にサマリーを貼る

自分で考える

未実施・保留の欄に何か書かれていましたか。
それは省いてよかったものでしたか。

発展課題

実行サマリーに、自分の仕事で欲しい項目を
1つ足して、実際に出るか試します。

実行サマリーの書式

動作・従った設定・対象ファイル・未実施の4項目が、応答のいちばん下に毎回出ます。

```
--- 実行サマリー ---
動作: 読み取り N件 / 編集 N件 / コマンド実行 N件
従った設定: 業務ルール（課題A） / kadaia-review
対象ファイル:
- kadaiA/index.html : 在庫5個以下の行に色を付けた
未実施・保留: 廃棄済みの非表示は未対応
------------------
```

- **動作**
  何回読んで、何回書いたか
- **従った設定**
  どの見出しに従ったか
- **対象ファイル**
  どのパスに何をしたか
- **未実施・保留**
  頼まれたのにやらなかったこと

> **未実施・保留の欄を読む**
> ここが空でない応答は、人が判断すべき残件があるという合図です。

設定1節の効き目

書き足したのは、CLAUDE.md の 1つの見出しだけです。

A-1 のとき

何をしたかは、本文を読んで
自分で拾う必要がありました。

A-4 のとき

動作・従った設定・対象ファイル・
未実施が、毎回そのまま出ます。

変えたのは設定ファイル1つ。AIそのものは同じままです。

ワーク

A-5 パイプラインを回す

[20min]

コマンドを1本叩くと、点検から報告まで自動で回ります。

操作

1. パネルで /selfcheck kadaiA と送る
2. .work/ を開いたまま眺める
3. 完了したら REPORT.md を読む
4. 中間データを番号順に開いて見る
5. 04_loop_log.md で周回数を見る

自分で考える

02_findings.json の指摘は、A-2 で自分が
書いた5個と比べてどうですか。

OK基準

- ✓ REPORT.md に件数が数字で出る
- ✓ .work/ に中間ファイルが4つ
- ✓ 04_loop_log.md に周回の記録
- ✓ 修正後もブラウザでアプリが動く

生成物

`kadaiA/REPORT.md`
.work/ に 01 から 04 の4ファイル

発展課題

観点を1つ足して、もう一度回します。

パイプラインの全体像

入力3種から中間データ4点を残し、ループで報告書を出します。

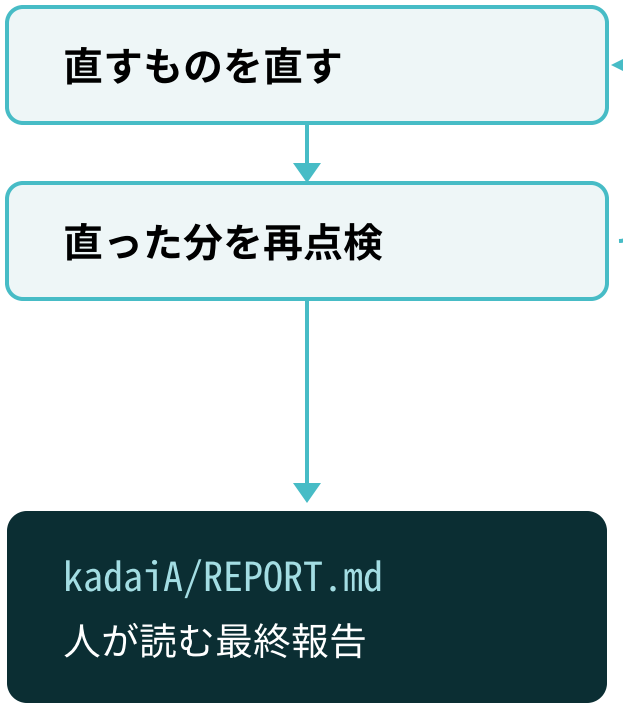
入力

- kadaiA/ の生成物
- docs/ の3つの規約
- memo.md の A-2 の懸念

中間データは .work/ に残ります

- 01_inventory.md 点検対象
- 02_findings.json 指摘と深刻度
- 03_fix_plan.md 直す・直さない
- 04_loop_log.md 残件の推移

ループと出力



自分が A-2 で書いた懸念も入力に混ざります。
だから A-6 の比較が成り立ちます。

深刻度の高い指摘が0件で停止。
止まらなければ3周で打ち切り。

.work/ に残るもの

中間データが残るので、
AIが何を見て何を直したかを後から追えます。

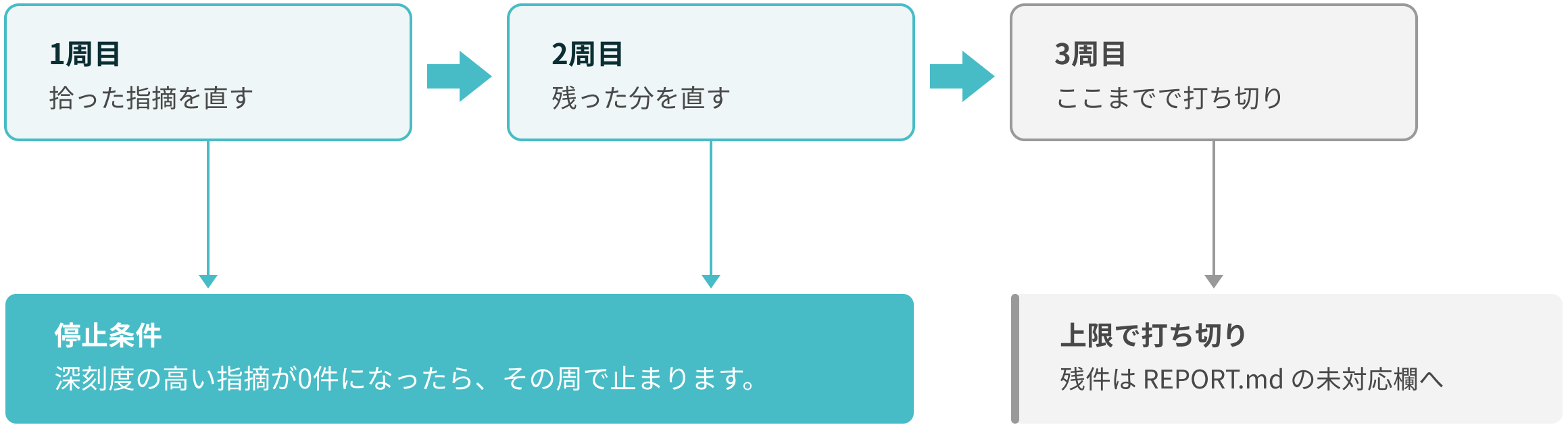
ファイル	中身	読むポイント
01_inventory.md	拾ったファイルの一覧と行数	AIが何を見たか
02_findings.json	観点ごとの指摘。深刻度と行番号	指摘に根拠が付いている
03_fix_plan.md	直すもの、直さないものと理由	全部は直していない
04_loop_log.md	周回ごとの残件数と打ち切り理由	ループが回った実体

消さずに残す

.work/ は実行後も残ります。あとから開けることが、この演習の中身です。

ループの止まり方

深刻度の高い指摘が0件になるか、3周で打ち切ります。



何周回って各周で何件残ったかは、04_loop_log.md にそのまま書かれています。

ワーク

A-6 ハーネスの有無を比較する

[10min]

素の状態との差を、持ち帰れる1ファイルとして残します。

操作

1. 下のプロンプトをそのまま送る
2. 3列目に自分の手で1行足す
3. 末尾に持ち帰る見出しを足す

```
@memo.md の A-2 と @kadaiA/REPORT.md を
突き合わせて、kadaiA/COMPARE.md を作って
ください。素の状態で気づけたこと、ハーネス
で拾えたこと、まだ誰も拾えていないこと、の
3列の表にしてください。
```

OK基準

- ✓ COMPARE.md に3列の表がある
- ✓ 3列目に自分の手で1行足した
- ✓ 持ち帰る見出しに1行書いた

生成物

handson/kadaiA/COMPARE.md

自分で考える

増えたのは指摘の数だけですか。それとも
毎回同じ観点で見る再現性のほうですか。

素の状態とハーネス投入後

増えたのは指摘の数だけでなく、毎回同じ観点で見る再現性です。

素の状態 A-1 から A-2

速く動くものが出てくる

指摘は聞いたときだけ返る

粒度は毎回変わる

見る観点は人によって違う

何を見たかは残らない

ハーネス投入後 A-3 から A-5

出てくる速さは変わらない

コマンド1本で点検が回る

観点は SKILL.md に固定される

誰が回しても同じ観点で見る

.work/ に見た跡が残る

1人で使うときより、5人のチームで使うときのほうが効きます。
COMPARE.md に書いた差が、そのときの説明材料になります。

05

演習B Skillとバグ修正

Spring Boot の不具合を、Skill を足しながら直しきります。

演習Bの到達基準

バグを2件直し、ブラウザで直ったことを確認し、./mvnw test を BUILD SUCCESS にします。

2件

修正するバグ

Issue を根拠に直します

1回

ブラウザで確認

直ったことを目で見ます

緑

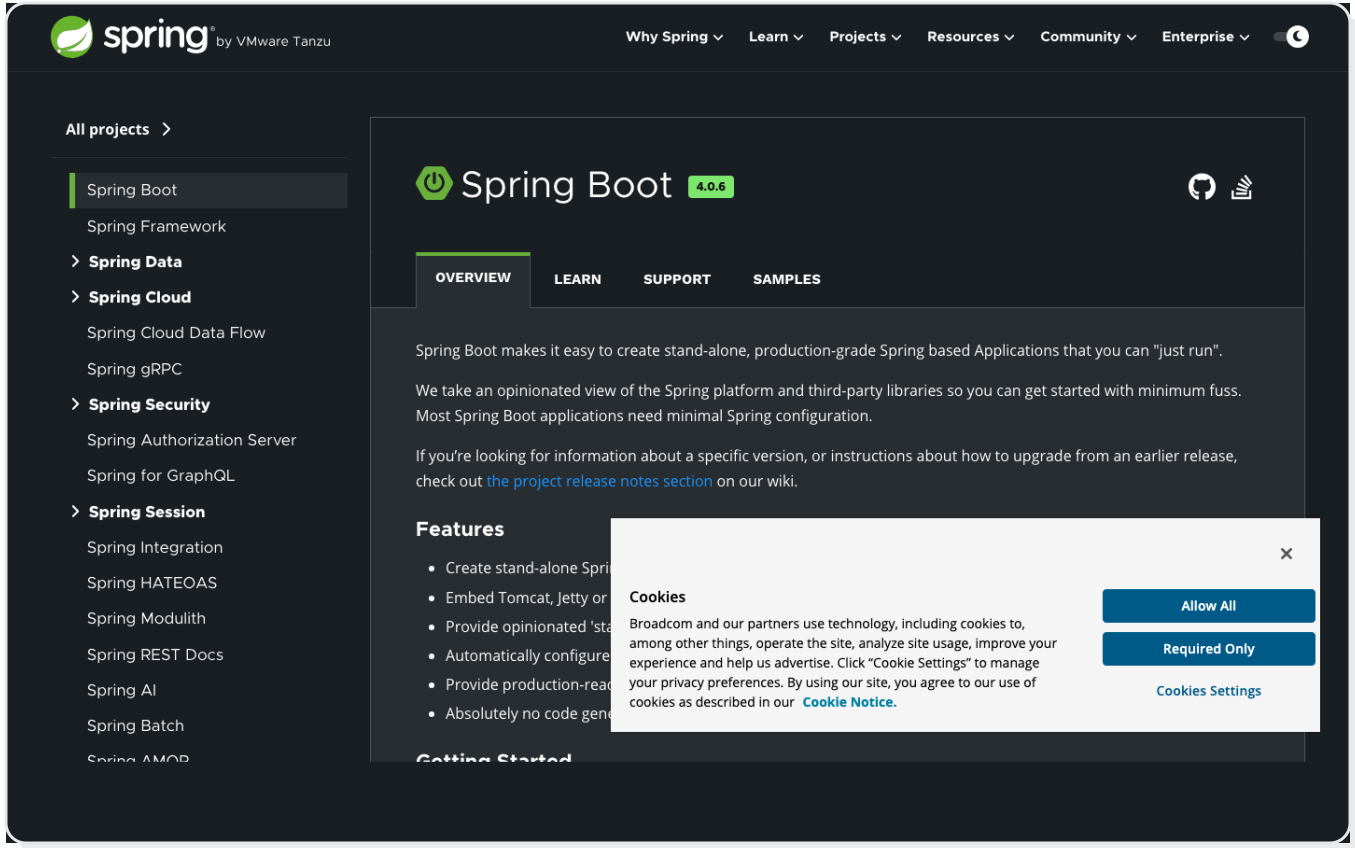
テストの結果

./mvnw test が BUILD SUCCESS

> テストは2件以上作ります。3件目のバグを自分で探すところからは発展課題です。
> 時間が余った方だけ進めてください。

題材はSpring Boot

課題Bは Spring Boot と Thymeleaf、H2 で動く備品管理システムです。



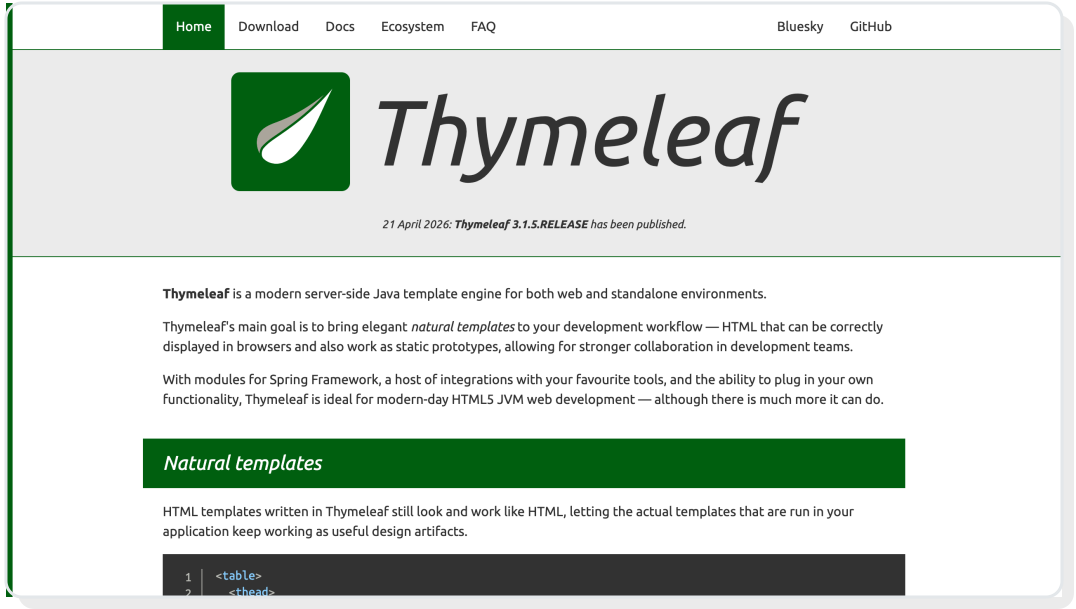
画面: spring.io の Spring Boot プロジェクトページ

- **Spring Boot**
  Java で Web アプリを動かす枠組み。
  起動は ./mvnw の1本で足ります。
- **Thymeleaf**
  Java 側のデータを HTML に流し込んで画面を組み立てます。
- **H2 インメモリDB**
  メモリ上だけのデータベース。
  アプリを止めると中身は消えます。

> 公式サイトの表示は最新版に上がります。
> 課題Bは Spring Boot 3.4 です。

課題Bの構成

画面からデータまで4層あり、
不具合は**サービス層**に集まっています。



画面: Thymeleaf 公式サイト thymeleaf.org

画面: H2 Database 公式サイト h2database.com

Controller ▶ **Service** ▶ **Repository** ▶ **H2 インメモリDB**

画面は Thymeleaf が組み立て、データは H2 に入ります。
仕込まれた不具合5件は Service 層に集まっています。

ワーク

B-0 起動して現状を掴む

[8min]

Skill 3種を置き、アプリを起動して壊れた画面を1件見つけます。

操作

1. _harness_kit の Skill 3種と _lib をコピーする
2. コピー先は handson/.claude/skills/
3. CLAUDE_追記.md の中身を CLAUDE.md の末尾へ
4. Claude Code パネルを開き直す
5. ターミナルで cd kadaiB のあと ./mvnw spring-boot:run
6. localhost:8080/equipments を開き、詳細も見る

OK基準

- .claude/skills/ にフォルダが4つある
- ターミナルに Started が出ている
- 一覧画面に備品が並んでいる
- エラーになる備品を1件見つけた

生成物

handson/.claude/skills/（4フォルダ）
handson/CLAUDE.md（追記済み）
起動中のアプリ localhost:8080

起動コマンド

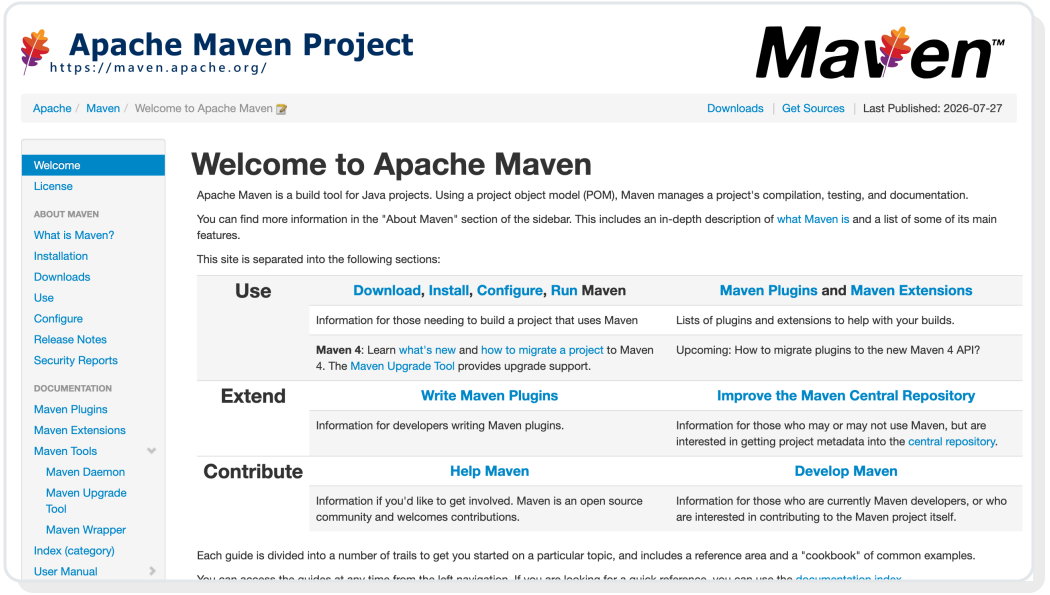
Mac と Windows でコマンドが変わります。

```
# Mac / Linux
$ cd kadaiB && ./mvnw spring-boot:run

# Windows PowerShell
> cd kadaiB
> .\mvnw.cmd spring-boot:run
```

起動できたときの目印

ターミナルに Started EquipmentApplication
が出れば起動できています。ブラウザで
http://localhost:8080/equipments を開きます。



画面: Apache Maven 公式サイト maven.apache.org

mvnw は Maven Wrapper です。
Maven を入れていなくても、
同じ版が使えます。

> 初回起動はライブラリの取得で数分かかります。3分待っても進まないときは挙手してください。

公式Skill 3種

Anthropic が公開している3種を、
提案・作成・点検で使い分けます。

Skill	本研修での役割	使うステップ	ライセンス
claude-automation-recommender	初見のプロジェクトに合う自動化を提案する。書き換えない	B-1	Apache-2.0
skill-creator（軽量版）	自分の観点を Skill として書き出す	B-5	Apache-2.0
threat-model	コードを実行せずに脅威モデルを作る	B-6	Apache-2.0

> 3種とも配布ZIPの _harness_kit に同梱しています。当日その場で配置します。

出所: anthropics の claude-plugins-official、skills、defending-code-reference-harness

ワーク

B-1 自動化を提案させる

[8min]

初見のプロジェクトに何を仕込むかを、5カテゴリで提案させます。

操作

1. パネルで「@kadaiB に合った自動化を提案してください。」と送る
2. 応答に claude-automation-recommender の名前が出るかを見る
3. MCP・Skill・フック・サブエージェント・プラグインの提案を読む
4. 1件だけ選び、「.claude/ の下に実際に作ってください」と依頼する
5. 作られたファイルを開いて中身を読む

OK基準

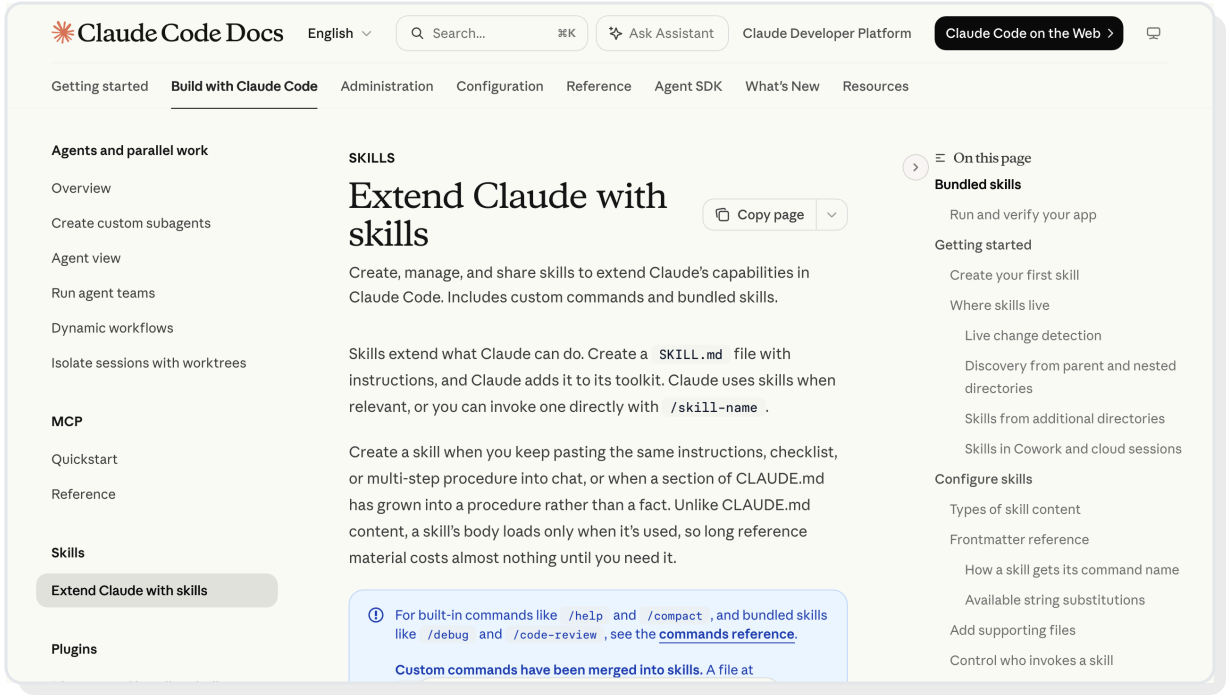
- 5カテゴリの提案が返ってきた
- 1件を選んで依頼し、ファイルが実際に作られた
- memo.md の ## B-1 に選んだ理由を1行書いた

生成物

handson/.claude/ 配下のファイル1本
（settings.json への追記、または SKILL.md）

読むAIと書くAI

提案する側と書き換える側を分けておくと、チームでも安全に回せます。



画面: Claude Code 公式ドキュメント Skills の解説

読むだけ（read-only）

コードと設定を読んで提案する。
ファイルは書き換えない。

B-1 claude-automation-recommender

書き換える（write）

差分を出してファイルを直す。
受け入れる前に人が読む。

B-4 Issue を根拠にした修正依頼

> この分け方は、SKILL.md に read-only と書くだけで作れます。
> B-1 で「実際に作ってください」と別に頼んだのは、そのためです。

ワーク

B-2 Planモードで壁打ちする

[12min]

直させる前に調べさせ、ファイル名とメソッド名まで特定します。

操作

1. Shift+Tab で Plan モードに切り替える
2. 1件目の症状を素直な言葉で聞く
3. 計画に出たファイル名と行を読む
4. その場所を自分で開いて確かめる
5. 業務ルールとの食い違いを聞く
6. 2件目の候補も自分で開いて確認する

OK基準

- 計画だけが返り、コードは変わっていない
- バグ候補2件をメソッド名まで特定できた
- 特定した箇所を自分で開いて確認した

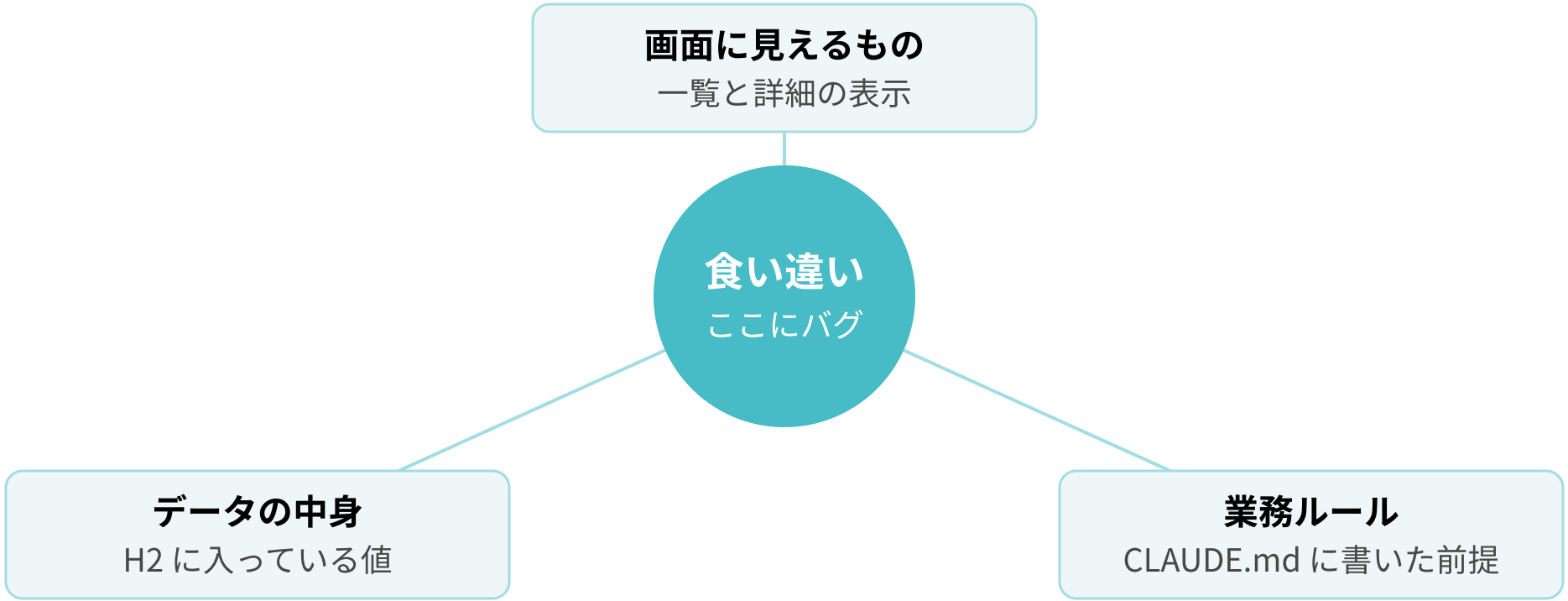
```
# 1件目 エラーが出るバグ
詳細画面を開くとエラーになる備品が
あります。原因を調べて、直すならどこ
をどう直すかの計画を出してください。
まだコードは変更しないでください。
```

```
# 2件目 エラーが出ないバグ
CLAUDE.md の業務ルールと実装が食い
違っている箇所はありませんか。画面・
データ・ルールを突き合わせて。
```

生成物: handson/memo.md の ## B-2
（2件のファイル名・メソッド名・症状）

バグの見つけ方

画面・データ・業務ルールの3つを突き合わせると、
エラーの出ない不具合が見えます。



3つのうち2つしか見ていないと、エラーの出ないバグは残ります。

クイズ

Q5 Planモードの動き

Q. Plan モードで「原因を調べて計画を出して」と頼んだとき、Claude Code は何をしますか。

- **A** 調べた結果と修正案を返し、ファイルは書き換えません
- **B** 調べたうえで、その場でファイルを書き換えます
- **C** 調べずに、計画の書き方だけを説明します
- **D** テストを実行してから、必要な箇所を書き換えます

正解は次のページで確認します。まず自分で考えてみてください。

解説

正解はA

正解はA。Plan モードは調べて計画を返し、書き換えません。

A **調べた結果と修正案を返し、ファイルは書き換えません**
計画を読んでから承認できるので、初見のコードでも安全に進められます。

B 調べたうえで、その場でファイルを書き換えます
それは Agent モードの動きです。Shift+Tab で切り替わります。

C 調べずに、計画の書き方だけを説明します
Plan モードはコードを読んだうえで計画を出します。

D テストを実行してから、必要な箇所を書き換えます
Plan モードでは、書き換えも実行も起きません。

初見のコードと、広い範囲を触る変更は Plan から入ります。
本研修では B-2 が Plan、B-4 が Agent です。

ワーク

B-3 Issueを起票する

[7min]

直す対象を、他人が再現できる文章にしてから修正に入ります。

操作

1. kadaiB/issues/ISSUE_TEMPLATE.md を開く
2. ISSUE_001 を読み、書き方の見本にする
3. 2件目を ISSUE_002_〈名前〉.md で新規作成
4. 5見出しを埋める。迷う欄は AI に相談する
5. 再現手順を自分でなぞり、症状が出るか見る

OK基準

- ファイル名が ISSUE_002_ で始まっている
- 5つの見出しがすべて埋まっている
- 手順どおりに操作して症状が再現した

ISSUE_TEMPLATE.md の5見出し

- 現象
- 期待する動作
- 発生箇所
- 再現手順
- 修正方針

生成物: kadaiB/issues/ISSUE_002_〈名前〉.md

Git と GitHub は使いません。
Markdown ファイル1本で回します。

Issue駆動の流れ

起票から修正結果の追記までが、1本の流れになります。

> 起票が B-3、計画が B-2、修正から追記までが B-4 にあたります。

Issue に修正結果を書き足して、はじめて1周が閉じます。次の不具合も同じ流れで回します。

ワーク

B-4 修正してテストを通す

[20min]

Issue を根拠に2件直し、ブラウザで確認しテストを緑にします。

操作

1. Shift+Tab で Agent モードに戻す
2. ISSUE_001 を指して修正を依頼する
3. 変更行を目で読んでから受け入れる
4. Ctrl+C で止めて ./mvnw spring-boot:run
5. ブラウザで直ったことを確認する
6. 2件目も同じ手順で直す
7. 2件を守るテストを src/test/java に作らせる
8. ./mvnw test を実行する（Windows は .\mvnw.cmd）
9. 落ちたら出力をそのまま貼って直させる

OK基準

- バグを2件修正した
- 修正前にエラーだった画面が正常に表示される
- ./mvnw test が BUILD SUCCESS で終わる
- テストが2件以上あり、対応するバグがわかる

生成物

EquipmentServiceImpl.java（修正）
EquipmentServiceImplTest.java（新規）
Issue の末尾に ## 修正結果 を追記

補足

テスト結果の読み方

BUILD SUCCESS が出れば到達です。
落ちたら、出力をそのまま貼って聞きます。

通ったとき

```
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO] BUILD SUCCESS
```

落ちたとき

```
[ERROR] Tests run: 2, Failures: 1, Errors: 0, Skipped: 0
[ERROR] EquipmentServiceImplTest.searchByKeyword:31
[ERROR] BUILD FAILURE
```

> 落ちた出力は要約せずそのまま貼ります。クラス名と行番号が、直す場所の手がかりです。

ワーク

B-5 自分のSkillを作る

[8min]

レビュー観点を Skill に書き出し、/clear 後に選ばれるか見ます。

操作

1. 「バグの探し方を Skill にしたい」と頼む
2. skill-creator の質問に答える
3. できた SKILL.md の description を読む
4. /clear で文脈をリセットする
5. Skill 名を出さずに別の言い方で質問する
6. 「従った設定」欄に Skill 名が出るか見る

OK基準

- SKILL.md に name と description がある
- /clear 後に使われたか、理由を説明できる

作る
SKILL.md の description に条件を書く

↓

/clear
会話の文脈だけを切る。設定は残る

↓

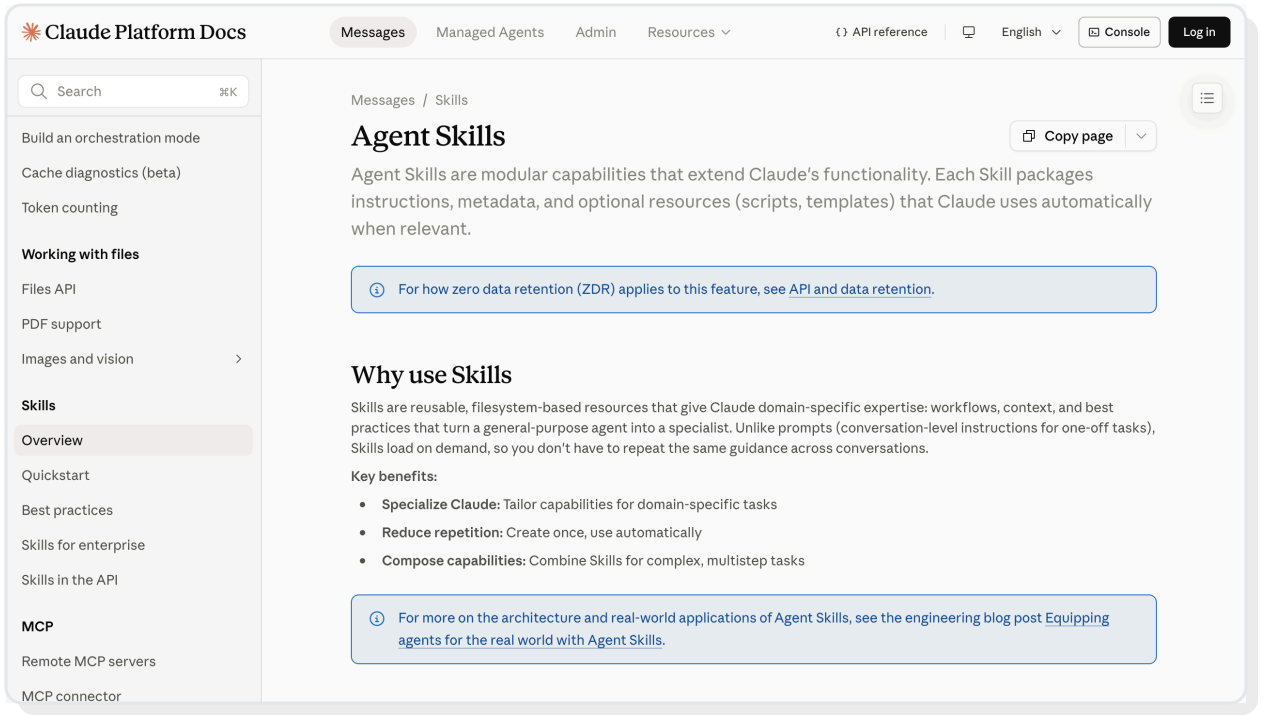
発火を確認
「従った設定」に Skill 名が出るか

生成物:
handson/.claude/skills/〈名前〉/SKILL.md

解説

Skillが選ばれる条件

Skill が発火する条件は、description に書いた文章です。



画面: Anthropic 公式ドキュメント Agent Skills

あなたが打った質問

業務ルールと実装が食い違って
いそうな箇所を探してください。

言葉が重なると発火

SKILL.md の description

業務ルールと実装の食い違いを
探すときに使います。

> Skill の名前を呼ばなくても選ばれます。description が広すぎると、関係のない質問でも発火します。
> 発火しなかったときに直すのは、質問ではなく description です。

ワーク

B-6 脅威モデリングを回す

[7min]

コードを実行せずに、守るものと入口を洗い出させます。

操作

1. 「kadaiB の脅威モデルを作って」と送る
2. 「対象コードは実行しません」の宣言を読む
3. 生成を待つ（3〜5分かかります）
4. kadaiB/THREAT_MODEL.md を開いて読む
5. 一番上の脅威の該当箇所を聞く

> 実行を伴う部分は配っていません。読み取りだけの構成です。

THREAT_MODEL.md の読みどころ

- 守るべきもの
- 入口（外から触れる場所）
- 信頼の境目
- 脅威の並び順と根拠

OK基準

- THREAT_MODEL.md が生成されている
- 守るべきもの・入口・脅威の並びが読める
- 一番上の脅威の該当箇所を挙げさせた

生成物

handson/kadaiB/THREAT_MODEL.md
（B-4 で直した2件が載っているかを見る）

発展

脅威と脆弱性

脆弱性は1行直せば消えますが、脅威は設計を変えないと消えません。

脆弱性

見つけたら直せる欠陥

入力チェックの抜けや、権限を見ていない分岐のように、直す場所が特定できる欠陥です。Issue に書き出して順番に消せます。

対処

Issue に書いて該当する行を直せば、その1件は消えます。

脅威

誰が何を狙うかの想定

守るものと入口を並べ、どこから何を取りに来るかを先に想定したものです。1行直しても、想定そのものは残ります。

対処

入口を減らす、権限を分けるなど、設計のほうを変えます。

B-6 で作った THREAT_MODEL.md は、右側を文章にしたものです。

06

コンテキストとハーネス

今日置いた設定が、明日の仕事でどう効くのかを整理します

バイブコーディングの強みと弱み

立ち上がりは速い一方で、抜けは画面に出てきません。

強み

- **動くものが最初に出る**
  仕様を詰める前に画面で確かめられます
- **試して捨てられる**
  気に入らなければ作り直す判断が早まります
- **書き出しの手間が小さい**
  空のファイルから書き始めずに済みます

弱み

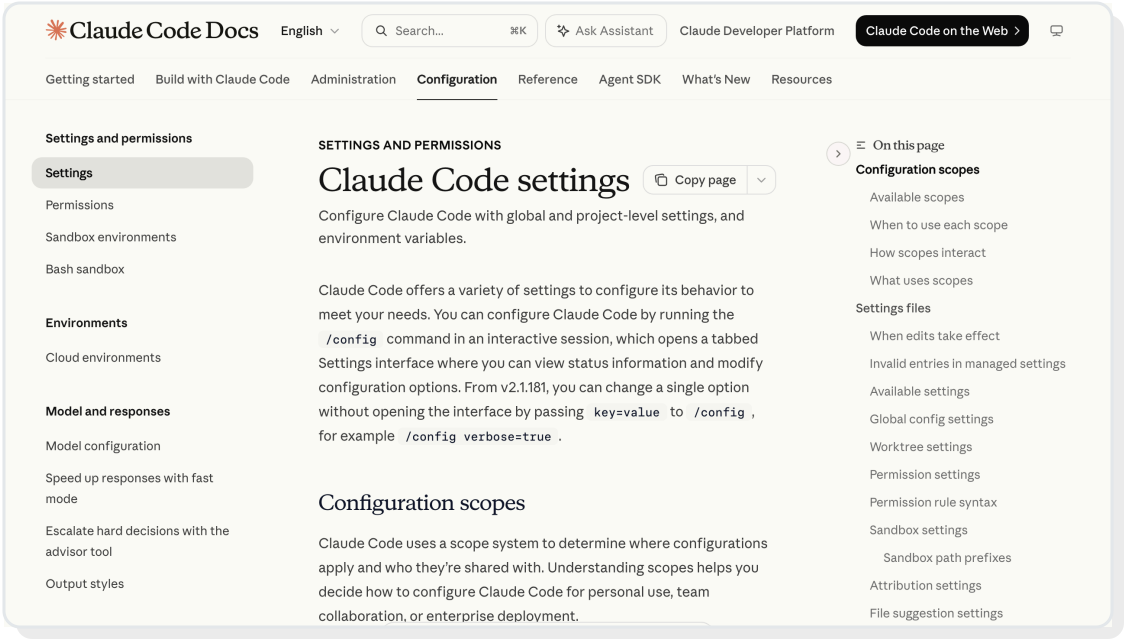
- **抜けが画面に出ない**
  動いて見えるので、見落としに気づけません
- **毎回やり方が変わる**
  同じ依頼でも、返ってくる作りが揃いません

> **A-2 で書き出した懸念**
> 入力チェック、エラー処理、件数が増えたときの動き。memo.md を読み返します。

コンテキストエンジニアリング

会社のナレッジと個人の暗黙知を、AIが読める形に整える作業です。

> **頭の中にあるうちは、AI には渡せません。**
> **ファイルに書いた分だけ、AI は前提を踏まえて動きます。**



画面: Claude Code 公式ドキュメント 設定の解説

前提を置く場所

CLAUDE.md	技術スタックと業務ルール、してはいけないことを書いた、プロジェクト共通の前提です。
docs/	用語集や業務ルールを置く場所。@ で名指しすると、その中身がそのまま渡ります。

コンテキストを置き、ハーネスで手順を固定し、ループで直しきります。

3つには順番があります。前を飛ばすと、後ろが効きません。

01
コンテキスト
前提をファイルに置く

02
ハーネス
手順と観点を固定する

03
ループ
直るまで回し直す

A-3 で配置しました
CLAUDE.md と .claude/

A-5 で回しました
/selfcheck を1本

B-2〜B-4 で使いました
Issue から修正まで

クイズ

Q6 コンテキストエンジニアリング

**Q. コンテキストエンジニアリングの説明として、
最も近いものはどれですか。**

A　AIに送る文章を、できるだけ長く書くこと

B　使うモデルを、より新しいものに入れ替えること

C　会社のナレッジと個人の暗黙知を、AIが読める形に整えること

D　会話が長くなったら履歴を消して、やり直すこと

正解は次のページで確認します。まず自分で考えてみてください。

解説

正解はC

暗黙知をファイルに書けば、全員同じ前提でAIを使えます。

C **正解 会社のナレッジと個人の暗黙知を、AIが読める形に整えること**

頭の中と口伝えのままでは、AIには渡りません。ファイルにすれば毎回同じ前提で動きます。

A AIに送る文章を、できるだけ長く書くこと

長さは関係ありません。要るものを選んで渡すほうが効きます。

B 使うモデルを、より新しいものに入れ替えること

モデルを替えても、渡していない前提は渡りません。

D 会話が長くなったら履歴を消して、やり直すこと

履歴を切るだけの操作で、前提を整える作業ではありません。

実務のポイント　前提が書いてあれば、後から参加した人も同じところから始められます。

実演

ハーネス動作デモ

/selfcheck を1回叩くところを、講師端末で見せます。

1. **パネルを開く**
   handson を開いた状態から
2. **/selfcheck と打つ**
   引数なしで1本だけ送る
3. **点検が回る**
   指摘を集めて直します（3周まで）
4. **REPORT.md が出る**
   .work/ に途中経過も残ります

[要挿入：図版]

/selfcheck 実行中の VS Code 画面
講師端末で撮影して差し替えます

同じことを手元で試す時間は取りません。流れだけ見てください。

質問

振り返りの4問

1問ずつmemo.mdに書きます。発表はありません。 [6min]

1

今日いちばん手応えがあった操作はどれですか

A-0からB-6のどこか、ステップ名で書いてください

2

ハーネスを入れる前と後で、応答の何が変わりましたか

A-1の応答とA-4の応答を思い出して、1行で書いてください

3

自分が気づいて、AIが触れなかった懸念はどれでしたか

A-2で書いた5個のうち、AIの指摘に無かったものを挙げます

4

明日の仕事で最初に試すのは、どの手順ですか

手順の名前と、試す対象の業務を1つずつ書いてください

答え合わせはしません。自分の言葉で残すことが目的です。

今日の到達点

今日つくったものは、すべて手元のファイルに残ります。

1本	1本	2本	2件
動く業務アプリ	比較メモ	起票したIssue	緑になったテスト
kadaiA/index.html	kadaiA/COMPARE.md	ISSUE_002_*.md	./mvnw test

ほかに残るファイル
memo.md、REPORT.md、THREAT_MODEL.md、自分で書いた SKILL.md

次の一歩

明日の仕事で最初に試すことを、1つだけ決めて帰ってください。

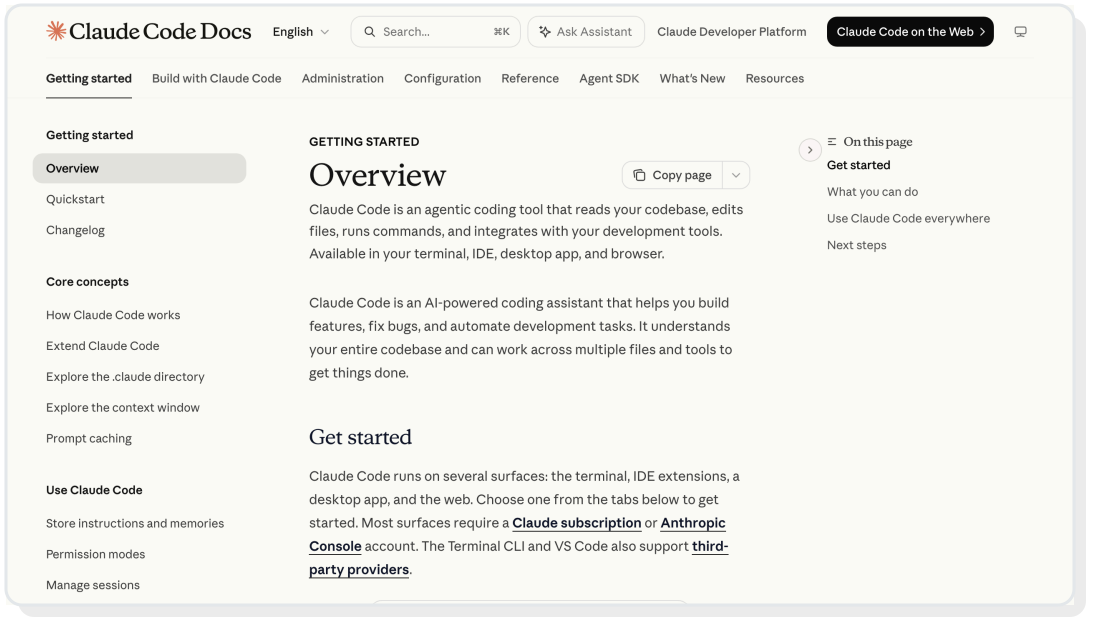
今日
動くアプリとテストを手元に持ち帰ります。

明日
最初に試す手順を1つ決めて、自分の手で回します。

1か月後
自分の CLAUDE.md をチームのルールに育てます。

アンケート

研修の最後にURLをご案内します。
その場で回答してください。3分ほどです。



画面: Claude Code 公式ドキュメント トップ

続きは公式ドキュメントで追えます。
docs.claude.com/en/docs/claude-code

決めた1つは、memo.md の最後に書き足してから閉じてください。

株式会社ギブリー (Givery, Inc.)
〒150-0036 東京都渋谷区南平台町15-13 帝都渋谷ビル8F