

Claude Code の基礎と Obsidian 第二の脳

Claude Code × Obsidian 活用研修 | アーサー・ディ・リトル・ジャパン株式会社 御中

午前 Claude Code の操作

午後 Obsidian と自動化

目安 [1日]

START 今日やること

1日目は、Claude Code そのものの操作に午前中で慣れ、午後にObsidianの第二の脳とつながります。手を動かす演習と、講師のデモを見るパートが交互に来ます。

時間帯	やること	形式
午前	環境の起動と配布データの確認 → Claude Code の基本操作 (Agent view / StatusLine / パスの渡し方 / モデル・effort) → デスクトップに調査レポートを作る	手を動かす
午前	研修受講者による事例紹介 → 講師による活用デモ	見る・聞く
午後	Obsidian の基礎 (PMVault を開く / グラフビュー / メモ / リンク) → 第二の脳の考え方	手を動かす
午後	同じ保管庫をアプリとClaude Codeの両方で開く → 1日1回の自動調査システムを作る	手を動かす
午後	今日の設定を保存し、2日目の演習で使えるように持ち出す	手を動かす

進め方

各演習は「考える → やってみる → 確認する」で進みます。途中、**自分で指示を書く**と書かれたブロックが出てきます。そこでは指示文を渡さず、達成条件だけを示します。何をどこまでやってほしいかを自分の言葉で書いて、Claude Code に送ってください。書いて実行したあとに参考例を開くと、自分の指示に何が足りなかったかが分かります。確認できたらチェックを付けて次へ進みます。ページ下部のチェックリストが進み具合を覚えます。分からない操作は、その場で Claude Code に「〇〇のやり方を教えて」と聞いて構いません。

午前

Claude Code を使いこなす

まずAIの相棒そのものに慣れます。操作を一通り触ってから、実際に成果物を1つ作ります。

SETUP 環境の起動と配布データの確認

配布フォルダの中身を確認、Claude Code を起動して、AIが最初に何を読んで動くのかを見ます。

作業する場所	ターミナル (Claude Code) と Finder
対象	配布フォルダ (PMVault と 演習フォルダ)
目安時間	[20min]

1 確認

配布物の場所を確認める

配布ZIPを展開したフォルダを Finder で開き、**PMVault** (第二の脳の保管庫) があることを確認します。場所は分かるところ (例: 書類フォルダ) に置いておきます。

Tips

フォルダを深い階層に置くと、あとでパスが長くなります。~/Documents/PMVault のように浅い場所が扱いやすいです。

2 やってみる

Vault の場所で Claude Code を起動する

ターミナルを開き、PMVault のフォルダへ移動して Claude Code を起動します。

```
cd ~/Documents/PMVault  
claude
```

起動したら、最初にこう聞きます。

Claudeへの指示(コピーして送る)

このフォルダの構成と、あなたが起動時に最初に読むファイル、このVaultで使えるSkillを教えてください。

3 確認

AIが読む前提を押さえる

観察ポイント

Claude Code が `CLAUDE.md` と `40_運用/persona.md` ・ `40_運用/domain.md` を読むこと、いくつかのSkillが挙がることを確認します。これが毎回の起動時の前提になります。

Skill とは

このVaultには、よく使う手順をまとめた「Skill」が入っています。`/` に続けて名前を打つと呼び出せます(例 `/ask`)。何があるかは `/` だけ打つと一覧で出ます。2日目の演習でも使います。

注意

「ファイルが見つからない」と返るときは、起動フォルダが違います。`cd ~/Documents/PMVault` からやり直してください。

☐ 配布フォルダの PMVault の場所を確認し、扱いやすい場所に置いた

☐ `cd ~/Documents/PMVault` → `claude` で起動できた

☐ 起動時に CLAUDE.md と persona / domain を読むことを確認した

BASICS Claude Code の基本操作

これから毎回使う操作を、一通り自分の手で触ります。暗記はいりません。「こういう操作がある」と分かって、必要なときに呼び出せば十分です。

場所 Claude Code

目安 [70min]

操作	何をするものか	やり方
Agent view	裏で走らせている作業を、一画面で見張る監視画面。長い調査を投げて、別の作業をしながら様子を覗けます。	<code>/background</code> で裏に回し、ターミナルで <code>claude agents</code>
StatusLine	画面の下に出せる状態表示。自分で設定すると、いまのモデル・いる場所・コンテキスト(記憶)の使用率・料金が常に見えます。最初は出ていません。	<code>/statusline</code> に出したい内容を伝える
パスの渡し方	「このファイルを見て」とAIに指し示す方法。ファイルの中身を読ませたいときに使います。	入力中に <code>@</code> でファイル名を補完
モデルの変更	頭脳の種類を切り替えます。難しい調査は賢いモデル、軽い作業は速いモデル、と使い分けます。	<code>/model</code>
effort の変更	考える深さを変えます。深いほど丁寧ですが時間もかかります。	<code>/effort</code>

作業を裏に回して、Agent view で見張る

1
裏に回す

`/background` で作業を投げる

`/background` のあとに続けて指示を書くと、その作業を裏で走らせたまま、ターミナルが手元に戻ってきます。返事を待たずに次の作業へ移れます。

`/background` 生成AIを業務に取り入れている企業の事例を3つ調べて、それぞれ1行でまとめてください。

2 見張る

claude agents で一覧を開く

空いたターミナルで次を実行します。いま裏に回した作業が一覧に出ます。

```
claude agents
```

矢印キーで行を選び、`Space` で最新の様子を覗き見、`Enter` でその作業に入り、`←` で一覧に戻ります。`Ctrl + R` で名前を付けられます。`?` でキー一覧が出ます。

観察ポイント

裏に回しても作業は止まりません。時間のかかる調査を投げて別のことをし、終わったところに結果だけ受け取る、という使い方ができます。午後に作る自動調査も、この考え方の延長です。

StatusLine を出す

自分で指示を書く 画面の下に、いまの状態を出す

StatusLine は、最初はありません。`/statusline` のあとに「何を出したいか」を言葉で伝えると、Claude Code が表示用のスクリプトを作り、設定まで済ませてくれます。設定ファイルを自分で書く必要はありません。

達成条件

- 画面の下に帯が出ている
- いまいる場所(パス)が出ている
- いま使っているモデル名が出ている
- そのセッションでかかった料金が出ている
- コンテキストの使用率が%で出ている

書くときに使える情報

`/statusline` と打ったあと、そのまま続けて日本語で「～を表示してください」と書けます。途中でファイルの編集許可を聞かれたら許可してください。出す内容は、あとから何度でも変えられます。

確認

出てきた帯を読む

画面の下に帯が出たら、いまのモデル・いる場所・使用率が読めるか確かめます。

観察ポイント

長い作業を続けるとコンテキストの使用率が上がっていきます。上がりきる前に、区切りのよいところで新しいセッションに切り替えると安定します。料金も出しておく、どの作業にどれだけかかるかの感覚がつかめます。

パスの渡し方 (@メンション)

やってみる

入力欄で @ を打つと、フォルダ内のファイル候補が出ます。40_運用/persona.md を選んで、こう聞きます。

Claudeへの指示(コピーして送る)

@40_運用/persona.md これはどんなファイルですか。3行で説明してください。

Tips

ファイルを Finder からターミナルへドラッグしても、パスが貼り付きます。画像は `Cmd + V` で貼れます。「このファイルを見て」と口で言うより、@ で指し示すほうが確実です。

モデルと effort を切り替える

やってみる

`/model` と打つと、モデルの一覧が出て矢印キーで選べます。`/model opus` のように名前を添えると直接切り替わります (`sonnet` / `opus` / `haiku` などの呼び名で、その時点の最新版が選ばれます)。同じように `/effort` で考える深さを変えられます (`/effort high` など)。

```
/model opus
/effort high
```

観察ポイント

難しい調査や設計は賢いモデル＋深いeffort、短い確認や整形は速いモデルで十分、と使い分けると、待ち時間とコストの無駄が減ります。次の調査レポートで、この使い分けを実際に試します。

- ☐ `/background` で作業を裏に回し、claude agents の一覧に出るのを確認した
- ☐ `/statusline` で状態表示を出し、モデル・場所・使用率を読めた
- ☐ `@` でファイルを指し示して質問できた
- ☐ `/model` と `/effort` で切り替えを試した

HANDS-ON デスクトップに調査レポートを作る

ここまでの操作を全部使って、成果物を1つ作ります。テーマは「生成AIの企業導入動向」。調べた結果を、そのままブラウザで開けるHTMLレポートとしてデスクトップに保存させます。

作業する場所	Claude Code
題材	生成AIの企業導入動向 (2026年時点)
成果物	デスクトップの調査レポート HTML (例: ~/Desktop/生成AI導入動向.html)
目安時間	[30min]

1 考える

どう頼めばファイルとして残るか

「調べて」とだけ頼むと、会話に文章が出て終わります。あとから開けるファイルとして残すには、何を伝える必要があるでしょうか。指示を書く前に予想します。少なくとも「調べる対象」「出力の形式」「保存する場所」の3つが要ります。

2 準備

モデルと effort を上げる

調査は深く考えさせたい作業です。指示を送る前に切り替えておきます。

```
/model opus  
/effort high
```

自分で指示を書く 調査してHTMLレポートに残すよう頼む

ここからは指示文を渡しません。下の達成条件を満たすように、自分の言葉で Claude Code に書いて送ってください。一度で全部そろわなくても、会話を続けて直せます。

達成条件

- 生成AIの企業導入動向について調べている
- 調べた結果が HTML ファイルとして保存されている
- 保存先が ~/Desktop/生成AI導入動向.html になっている
- 「導入が進んでいる業務」「代表的なツール」「つまずきやすい点」の3つが入っている
- 数字や事例に出典のリンクが付いている
- 最後に、根拠にしたURLの一覧が載っている

書くときに使える情報

保存場所は `~/Desktop/ファイル名.html` のように、置き場所とファイル名まで書くと確実です。`~` は自分のユーザーフォルダを指す省略記号で、`~/Desktop` はデスクトップのことです。入れてほしい内容は、箇条書きで並べて伝えても構いません。

ひらく + 書いて実行してから開く: 指示の例

3 確認

デスクトップで開く

できあがったら、デスクトップの `生成AI導入動向.html` をダブルクリックしてブラウザで開きます。読みやすいか、出典リンクが付いているかを見ます。

観察ポイント

物足りなければ、会話を続けて「事例をもう2つ増やして」「図を1つ入れて」と頼めば、同じファイルを直してくれます。作り直しでなく、対話で育てていけるのがポイントです。

できたら

調べた内容が、出典付きのHTMLレポートとしてデスクトップに残った。

- ☐ モデル・effort を上げ、自分で書いた指示で調査を頼んだ
- ☐ デスクトップに調査レポート HTML が保存された
- ☐ ブラウザで開いて、出典リンクを確認した

SHOWCASE 研修受講者による事例紹介

先に使い始めている受講者の方から、実際の業務での使い方を紹介していただきます。手を動かすパートではありません。自分の業務に置き換えながら聞いてください。

形式 見る・聞く

目安 [20min]

CASE 1

自動ニュース収集

毎朝、決めたテーマのニュースを自動で集めて、要約を残す仕組みです。午後の「1日1回の自動調査システム」で、これに近いものを自分の手で作ります。

CASE 2

作成済みのスライド

前回の研修で作ったスライドを見ながら、どんな指示でどこまで作れるのかの肌感をつかみます。2日目のスライド作成の予習にもなります。

聞くときのポイント

「便利そう」で終わらず、自分の担当業務で毎日・毎週繰り返している作業のうち、どれが同じやり方で自動化できそうかを1つメモしておきます。午後と2日目でそれを実際に組みます。

DEMO 講師による活用デモ

講師が実際に Claude Code を動かして、いくつかの成果物をその場で作ります。指示の出し方と、AIの動き方を見てください。

形式 見る

目安 [30min]

DEMO 1

調査レポートの作成

午前の演習と同じ流れを、より踏み込んだ指示で。どこまで深く調べさせられるかを見ます。

DEMO 2

解説動画の作成

調べた内容を、動画のかたちに組み立てる例です。テキスト以外の成果物も作れることを見ます。

DEMO 3

競馬・競輪の予想

公開データを集めて予想を組み立てる例です。数字を扱う作業の進め方の参考にします。

見るときのポイント

成果物そのものより、講師がどう指示を分割しているか、途中でどう軌道修正しているかに注目します。うまくいく指示の出し方は、まねるのが一番早いです。

お昼休憩 [60min]

午後

Obsidian と第二の脳

午前のAIを、知識をためる保管庫とつなぎます。ためて、つないで、最後は自動でためる仕組みまで作ります。

OBSIDIAN Obsidian の基礎

午前中に Claude Code から触っていた PMVault を、今度は Obsidian (人が読み書きするアプリ) で開きます。中身の見方と、メモの残し方を覚えます。

場所 Obsidian

目安 [60min]

1 開く

PMVault を保管庫として開く

Obsidian を起動し、「フォルダを保管庫として開く」から **PMVault** のフォルダを選びます。左にファイルの一覧(フォルダツリー)が出ます。

Tips

Obsidianの「保管庫 (Vault)」は、ただのフォルダです。特別な形式ではないので、中の `.md` ファイルはテキストとしてそのまま読めます。

2 見る

フォルダ構成をつかむ

左のツリーで、番号付きのフォルダの役割を確認します。

フォルダ	入れるもの
00_ナレッジ	概念・手法・型 (自分の理解)
10_業界と事例	業界プロファイル・事例・動向
20_案件	案件のメモ・会話ログ・決めたこと
30_インプット	取り込んだ元資料と、そこから作ったメモ
40_運用	AIの設定 (persona/domain)・操作ログ・直近メモ (hot)

3 見る

グラフビューを開く

左端のリボンにあるグラフのアイコン(つながりの図)を開きます。ノート同士のつながりが点と線で見えます。いまはまだ点が少ないですが、書いてリンクするほど増えます。

4 書く

persona と domain を自分の言葉に直す

40_運用/persona.md (AIの口調) と 40_運用/domain.md (あなたの役割) を Obsidian で直接開いて、自分に合わせて書き換えます。マークダウンは、行頭に # で見出し、- で箇条書きです。

Tips

ここでの書き換えは、午前の Claude Code にそのまま効きます。人が書いても AI が書いても、同じファイルを見ているからです。

5 作る

思いつきメモの置き場を作る

日々の思いつきをためる場所を用意します。40_運用 の中に journal というフォルダを新しく作り、その中に今日の日付のノート YYYYMMDD.md (例 20260725.md) を作って、頭に浮かんだことを2〜3行書きます。

Tips

日付をファイル名にすると、あとで時系列に並びます。走り書きはここ、まとまった内容は 20_案件 や 00_ナレッジ へ、と使い分けます。

6 つなぐ

別のファイルへリンクする

いま作った日付ノートの中で [[と打つと、他のノートの候補が出ます。[[persona]] のように選ぶと、そのノートへのリンクになります。試しに、さっき直した persona か domain へリンクを1本張ります。

観察ポイント

リンクを張ってからグラフビューに戻ると、日付ノートと persona が線でつながって見えます。更新した場所が図に現れることを確認します。これが「つながる知識」の最小単位です。

☐ PMVault を Obsidian で開き、フォルダ構成を確認した

☐ グラフビューを開いた

☐ persona / domain を Obsidian で書き換えた

☐ 40_運用/journal/ を作り、YYYYMMDD.md にメモを書いた

☐ [[]] で他のノートへリンクを張り、グラフビューで確認した

CONCEPT AIエージェントと Obsidian の関係

なぜ、クラウドのチャットではなく、手元のフォルダに知識をためるのか。第二の脳の考え方を整理します。
手は止めて聞くパートです。

形式 座学

目安 [20min]

POINT 1

手元にためる(ローカル構築)

知識は自分のパソコンの中のフォルダにたまります。中身はただのテキストなので、どのアプリでも読めて、消えても復元でき、どこかのサービスに縛られません。

POINT 2

AIが第一の読み手

ためたメモは、人だけでなくAIが読んで使えるように整えます。だから4つの手がかかり(種類・時期・主題・状態)を付けます。付けておくと「あの案件の進行中の判断だけ」を後から引き出せます。

POINT 3

ためると引き出すを1つで

資料を取り込む・会話を保存する(ためる)と、自然な言葉で過去を呼び出す(引き出す)を、同じ保管庫で回します。使うほど賢くなる、自分専用の第二の脳になります。

LINK 同じ場所をアプリと Claude Code で開く

Obsidian で開いたままの PMVault を、同時に Claude Code でも開きます。人が見る画面とAIが動く画面が、同じ保管庫を指している状態を作ります。

場所 Obsidian + Claude Code

目安 [30min]

1
並べる

左に Obsidian、右にターミナル

Obsidian は開いたまま、ターミナルで同じ PMVault に入って Claude Code を起動します (午前と同じ)。画面を左右に並べます。

```
cd ~/Documents/PMVault
claude
```

2
名前を付ける

セッションを保持する

この保管庫用のセッションだと分かるよう、Agent view (`claude agents`) で `Ctrl + R` を押し、「PMVault」など分かる名前を付けておきます。次からこの作業に戻りやすくなります。

自分で指示を書く この保管庫の使い方をAIに解説させる

いま開いている PMVault について、Claude Code に説明させます。あとで自分が読み返せる説明を引き出すのが目的です。

達成条件

- はじめて使う人に分かる言葉で説明されている
- 「ためる方法」と「あとから引き出す方法」の両方が入っている
- それぞれに、実際の頼み方の例が付いている

書くときに使える情報

誰に向けた説明かと、入れてほしい項目を指示に書くと、返ってくる説明の細かさがそろいます。長すぎたら「短く」、例が少なければ「例を増やして」と続けて調整できます。

ひらく + 書いて実行してから開く: 指示の例

4
確認

人の操作とAIの操作がつながるのを見る

観察ポイント

Claude Code に「40_運用/journal/(今日の日付)に『〇〇を調べたい』とメモして」と頼み、Obsidian 側でそのファイルが増えるのを見ます。さらにグラフビューを開き直すと、更新した場所が図に反映されます。人の操作とAIの操作が、同じ保管庫でつながっていることを確認します。

- ☐ Obsidian と Claude Code で同じ PMVault を開き、左右に並べた
- ☐ Agent view でセッションに名前を付けて保持した
- ☐ 自分で書いた指示で使い方を解説させ、AIの書き込みが Obsidian に反映されるのを確認した

AUTOMATION 1日1回の自動調査システムを作る

1日目の仕上げです。決めたテーマを毎日1回、自動で調べてレポートにする仕組みを、Claude Code への指示だけで組みます。作り方は「指示 → 確認」のくり返しです。

場所 Claude Code

題材 生成AIの最新動向

目安 [60min]



重い成果物は Obsidian の外に置き、Vault には要約と索引だけを残してリンクでつなぐ。

なぜ本体を外に置くのか

調査レポートは重く、毎日増えます。全部を保管庫に入れると、第二の脳がすぐ散らかります。だから本体は外 (~/Documents) に置き、保管庫には「いつ・何を調べ・要点は何か」の短い要約と、本体への道しるべ (リンク) だけを残します。保管庫は索引に徹する、という考え方です。

自分で指示を書く 調査スクリプトを作らせる

毎日動かす調査の本体を、Obsidian の外 (~/Documents/daily-research/) に作ります。この段階ではまだ自動で動かしません。手で1回動かして、正しく動くところまでを確認します。

達成条件

- ~/Documents/daily-research/run.sh ができている
- 動かすと、生成AIの最新動向を調べて reports/(今日の日付).html にレポートが保存される
- あわせて summary.md に、日付・調べたテーマ・要点3行の短い要約が追記される
- 実行できる状態になっていて、手で1回動かして結果を確認できている
- 人が見ていなくても、途中で止まらずに最後まで動く

書くときに使える情報

Claude Code は `claude -p "指示"` と書くと、画面を開かずに指示を1回だけ実行できます。スクリプトの中から呼ぶときはこの形を使います。また、自動で動かすものは、途中で「実行してよいですか」と確認待ちになると止まってしまいます。その点も指示に含めてください。`chmod +x` は「このファイルを実行してよい」という印を付けるコマンドです。

ひらく + 書いて実行してから開く: 指示の例

できたら

~/Documents/daily-research/ に run.sh ができ、手で動かすと reports にレポート、summary.md に要約が出る。

自分で指示を書く 要約を保管庫に記録してつなぐ

いまできた要約を PMVault に取り込み、そこからレポート本体へたどれるようにします。重い本体は外、保管庫には索引だけ、という形を作ります。

達成条件

- PMVault の `30_インプット/sources/` に `(今日の日付)-調査-生成AI最新動向.md` ができている
- 中身が、いま `summary.md` にある一番新しい要約になっている
- ファイルの先頭に `type` ・ `date` ・ `topic` ・ `status` の4つが付いている
- 本文の末尾に、レポート本体 (`~/Documents/daily-research/reports/(今日の日付).html`) への道しるべが1行ある

書くときに使える情報

ファイルの先頭に置く `type` などの情報を frontmatter と呼びます。AI が後から「調査ノートだけ」「この主題だけ」と絞り込むための見出しです。この保管庫では、種類・時期・主題・状態の4つをそろえる決まりにしています。

ひらく + 書いて実行してから開く: 指示の例

観察ポイント

Obsidian 側で `30_インプット/sources` に要約ノートが増え、そこから本体レポートへたどれることを確認します。重い本体は外、索引は保管庫、という形ができています。

仕上げ

1日1回、自動で動くようにする

Claudeへの指示(コピーして送る)

この `run.sh` を毎日1回、朝9時に自動で動かすように、macOSのlaunchdへ登録してください。plistは
~/Library/LaunchAgents/com.dailyresearch.plist に置き、標準出力と標準エラーは ~/Library/Logs/daily-research.log に出して、登録(load)まで実行してください。終わったら `launchctl list` に出るか確認してください。

Tips

1日待たずに試すには「いま `launchctl start com.dailyresearch` で手動実行して、ログとレポートを見せて」と頼みます。動きを確認できるまでは、テーマを軽いものにしておくと安心です。

できたら

毎朝9時に自動で調査が走り、要約が保管庫にたまる仕組みができた。

- ☐ 達成条件を見て自分で指示を書き、`run.sh` を作って手で動かせた
- ☐ 要約を `30_インプット/sources` に記録し、本体レポートへの道しるべを付けた
- ☐ 1日1回、自動で動くように登録した

ひらく + もっと安全に運用するには

HANDOVER 2日目への引き継ぎ

今日Vaultで整えた設定を保存し、明日の演習プロジェクトでも同じ設定とSkillが使える状態にします。ここをやっておかないと、2日目の最初でつまづきます。

場所 Claude Code (PMVaultで起動中)

目安 [15min]

1 保存

今日の設定をVaultに残す

PMVaultで起動しているClaude Codeで、次を実行します。今日書き換えた persona や domain、うまくいった頼み方が、Vaultの `_研修/_実績` に記録されます。

```
/save-setup
```

できたら

`_研修/_実績` に今日の設定が保存され、INDEX に行が増えた。

2 持ち出す

Skill をどのフォルダでも使えるようにする

Skill は、置いた場所によって使える範囲が変わります。いま PMVault に入っている Skill は、PMVault のフォルダで起動したときだけ使えます。明日は別のフォルダ(演習プロジェクト)で作業するので、持ち出しておきます。

置き場所	使える範囲
<code>~/.claude/skills/</code>	どのフォルダで起動しても使える
そのフォルダの <code>.claude/skills/</code>	そのフォルダの中だけ

Claudeへの指示(コピーして送る)

このVaultの `.claude/skills` にある `load-setup` と `save-setup` を、`~/.claude/skills/` にコピーして、どのフォルダで起動しても使えるようにしてください。コピーしたら、両方が使える状態になっているか確認して教えてください。

観察ポイント

明日の演習では、PMVault ではなく別のフォルダで Claude Code を起動します。そこで `/load-setup` を使うので、この持ち出しが要ります。同じやり方で、気に入った Skill はいくつでも共通化できます。

☐ `/save-setup` で今日の設定を Vault に保存した

☐ load-setup と save-setup を ~/.claude/skills/ にコピーした

CHECK 理解度チェック

Q1. 長い調査を投げたまま別の作業をして、あとで様子を覗きたいときに使うのは？

StatusLine

Agent view (claude agents)

@メンション

グラフビュー

Q2. 自動調査システムで、重いレポート本体を Obsidian の外 (~ / Documents) に置くのはなぜ？

Obsidian は HTML を保存できないから

外に置かないと launchd が動かないから

Claude Code は保管庫の中に書き込めないから

保管庫を索引に保ち、毎日増える重い成果物で散らからないようにするため

Q3. Obsidian で、別のノートへのリンクを作る書き方は？

[[ノート名]]

@ノート名

#ノート名

/link ノート名

WRAP UP 1日目のまとめと2日目への接続

午前でClaude Codeの操作に慣れ、午後で保管庫とつなぎ、最後に自動でためる仕組みまで作りました。手元には、今日つくった資産が残っています。

- 自分好みに直した `persona.md` / `domain.md`
- 画面の下に出るようにした StatusLine
- 思いつきをためる `40_運用/journal/`
- デスクトップの調査レポート、1日1回動く自動調査システムと、その要約が入り始めた保管庫
- `_研修/_実績` に保存した今日の設定と、どこでも使えるようにした Skill

2日目の予告

2日目は、この保管庫を使った案件の調査から提案スライドまでを通します。あわせて、保管庫を育て・整える機構(取り込み・引き出し・整理・点検・実績化)を演習で扱います。今日ためた journal や要約が、そこで効いてきます。

発展 早く終わった人への追加課題

ひらく + StatusLine に情報を足す

HELP 詰まったときの即応

症状	まず試すこと
「ファイルが見つからない」と返る	起動フォルダ違いです。 <code>cd ~/Documents/PMVault</code> からやり直します。
@ を打っても候補が出ない	そのフォルダで起動しているか確認します。別の場所で起動していると候補が出ません。
グラフビューにノートが出ない	リンク ([[]]) を張ると線でつながります。1本張ってから開き直します。
自動調査が動かない	ログ(<code>~/Library/Logs/daily-research.log</code>)のエラー文を、そのまま Claude Code に貼って原因を聞きます。

いちばん確実な手

迷ったら、いま起きていることをそのまま Claude Code に伝えて「どうすればいい?」と聞くのが最短です。エラー文があれば、それをそのまま貼ってください。



Claude Code × Obsidian 活用研修 Day1 ハンズオン v2.0 (2026年7月)
制作: Givery 株式会社 / アーサー・ディ・リトル・ジャパン株式会社 御中