HANDS-ON • DAY 2

演習プロジェクトの調査と提案スライド

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

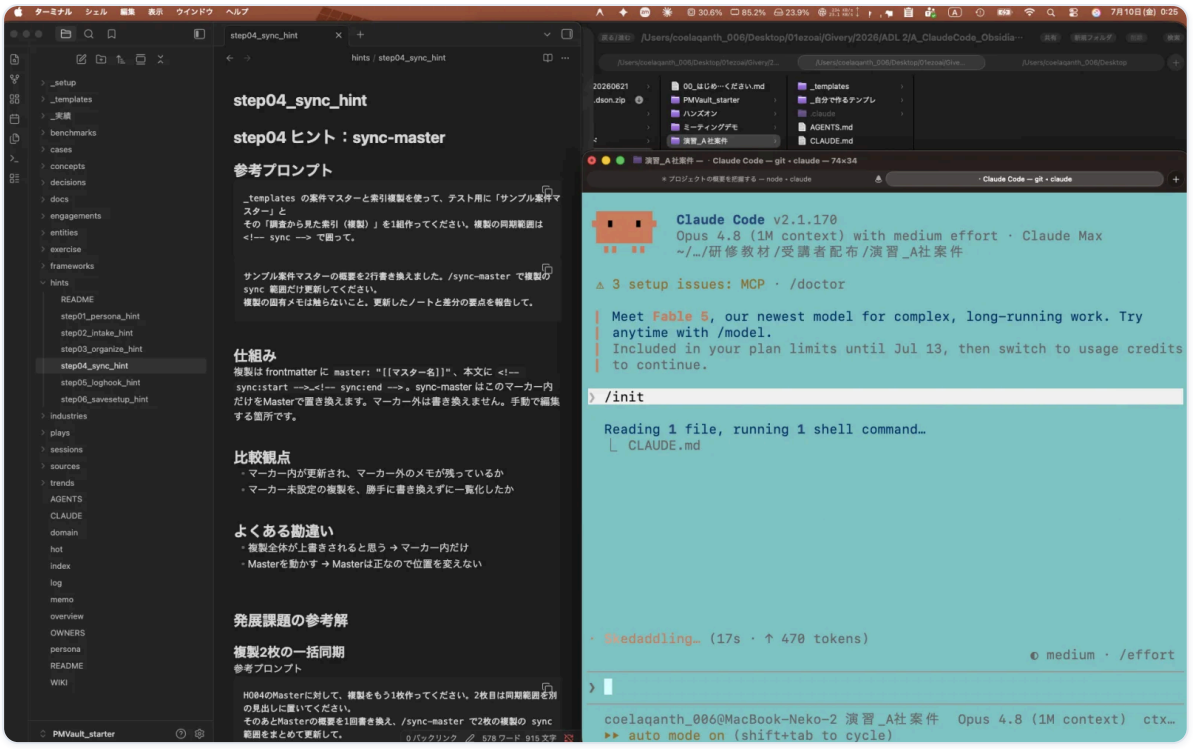
題材 **A社（架空）生成AI構想** | 進め方 **6ステップ** | 講師 **安田 光喜**

START 今日やること

Day2 は、別フォルダ **演習_A社案件** で Claude を起動し、公開情報の調査からレポート・提案スライド（pptx）までを作ります。仕上げに Vault 側へ戻り、「記録が勝手にたまり、たまったものが勝手に整う」仕組みを2つ作ります（EXERCISE 7・8）。Day1 で作った Vault からは `/load-setup` で設定を持ち込みます。`claude agents` を実行すると AgentView が開き、演習プロジェクトと Vault の2つのセッションを切り替えられます。

画面構成（2画面）

左：Obsidian / エディタ	右：ターミナル
案件の背景（Vaultの docs/）と成果物を見る	`cd ~/.../演習_A社案件` → `claude` で演習プロジェクトのセッションを起動。`claude agents` で AgentView を開くと、Vault のセッションと切り替えられる



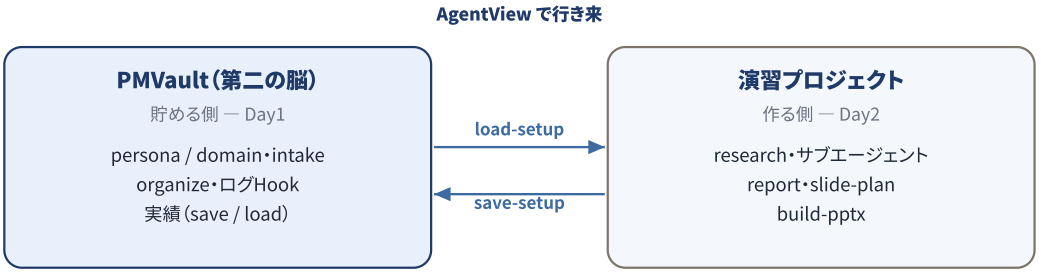
左で **演習_A社案件** の docs/ や成果物を開き、右のターミナルで同じフォルダに `cd` して `claude` を起動した状態です。

進め方の型（4段）

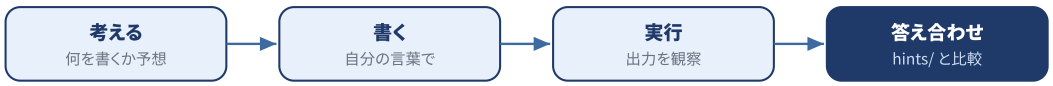
① 考える → ② 書く → ③ 実行 → ④ 答え合わせ。④では `hints/stepNN_xxx.md` を開いて比べます。**参考プロンプトは資料に載せていません。**

案件の背景

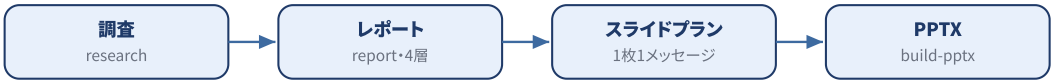
はじめに Vault の `_研修/docs/`（ハンズオンテーマ・案件ブリーフィング・キックオフ議事録）と、このプロジェクトの `data/与件メモ`・`data/リサーチ起点` を読む。A社は架空。公開情報のみ扱う。



各フォルダで claude を起動してセッションを持ち、claude agents で開く AgentView から2つを切り替える。Vaultに蓄えた設定を load-setup で演習へ持ち込み、成果は save-setup でVaultへ戻す。



ハンズオンの進め方。参考プロンプトは資料に載せず、自分で書いてから hints/ で答え合わせする。



Day2の流れ。公開情報の調査から提案スライドまでを通して作る。

EXERCISE 1 案件理解・load-setup・最初の調査

A社案件を掴み、Day1にVaultで整えた設定を演習プロジェクトへ持ち込みます。公開情報の調査を一手目まで進め、出典付きノートを1本残します。ここで作った出典付きノートが、この後のレポートとスライドの材料になります。

場所 **Claude Code** 対象 **data/与件メモ・リサーチ起点、Vault docs/**

成果物 **成果物/research/ に1本** 目安 **[20min]**

---

1 考える

設定の持ち込み方

Day1でVaultにためたpersonaや頼み方を、A社案件にどう読み替えるかを予想します。`memo.md`（このプロジェクトに作ってよい）に当たりを書きます。

2 書く

起動して load-setup を書く

達成条件

- Day1の設定のうち、そのまま使えるものと読み替えが要るものを分けている
- A社案件の固有名詞へ読み替える指示になっている
- 差分を確認してから反映する流れになっている

```
cd ~/Documents/演習_A社案件
claude
```

`data` の与件メモとリサーチ起点、Vaultの `_研修/docs/` を読ませます。続けて `/load-setup` のあとに、Day1設定を今回の固有名詞へ読み替える指示を自分の言葉で書きます。

Tips

起動は `cd ~/Documents/演習_A社案件` のあと `claude` 。別フォルダで起動していると設定やdataを見つけられません。Vault（ `~/Documents/PMVault` ）は演習フォルダの外にあるので、読み込みの許可を聞かれることがあります。聞かれたら許可してください。

3 実行

差分を確認して調査を1本

`/load-setup` を走らせ、差分を確認してから反映します。次にリサーチ起点を1つ選び、調査指示を書いて実行します。配布の `/research` はまだ使いません。

注意

`/load-setup` はDay1設定をそのまま上書きせず、差分を見て今回の案件に読み替えてから反映します。出典が薄いと感じたら「一次情報の出典を付けて」と追加で頼みます。

4 答え合わせ

hints と比べる

`hints/step01_research_hint.md` を開き、参考プロンプトと比較観点を見て、自分の調査ノートと突き合わせます。

自分で考えるポイント

- Day1で効いた頼み方やpersonaのうち、そのまま使えるものと読み替えが要るものはどれか
- 1つ目のリサーチ起点として、提案の優先順位づけに効く論点はどれか

達成チェック

- [ ] /load-setup の差分を確認し、A社案件向けに読み替えて反映できている
- [ ] リサーチ起点を1つ選び、調査指示を自分の言葉で書けている
- [ ] 成果物/research/ に調査ノートが1本あり、すべての主張に出典URLと日付が付いている

うまくいかないとき

`/load-setup` が見つからないと言われたときは、Day1の引き継ぎで Skill を持ち出せていません。「PMVault の .claude/skills にある load-setup と save-setup を ~/.claude/skills/ にコピーして」と頼めば、その場で使えるようになります。設定が反映されないときは、Day1の設定がVaultの `_研修/_実績` に保存されているか、INDEXに行があるかを確認します。そのうえで「_研修/_実績 に保存済みのDay1設定を探して、差分表示してから取り込んで」と再取り込みを頼みます。直らなければ「詰まったときの即応」と `hints/step01_research_hint.md` を見ます。

発展課題（早く終わった方向け）：2つ目のリサーチ起点で論点を広げる

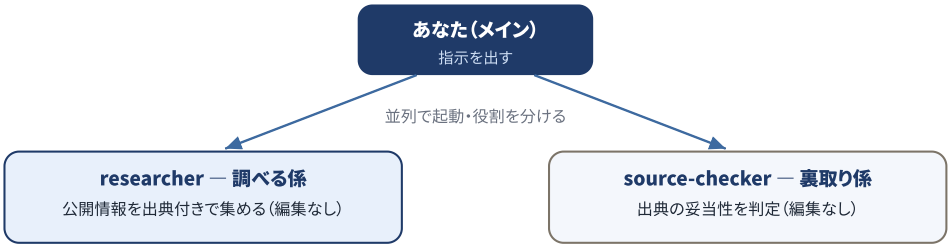
1本目のノートは書き換えず、1本目と別の起点を1つ選んで調査し、2本目のノートを別ファイルで作ります。対象は `data/リサーチ起点.md` と `成果物/research/`、公開情報のみを使います。達成条件は、出典付きノートが2本あり論点が重複していないこと。詳細は `hints/step01_research_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 2 researcher 並列・source-checker で裏取り

HO01で掴んだ論点を、researcherサブエージェントに複数まとめて調べさせ、重要な主張はsource-checkerで裏取りします。調査役と検証役を分けることで、出典の確かな主張だけを提案に残せる状態を目指します。

場所 **Claude Code**　対象 **data/リサーチ起点.md、成果物/research/**

成果物 **論点ごとの調査ノート・検証済み主張リスト**　目安 **[25min]**



調べる係と裏取り係を分け、検証役には編集権限を渡さない。並列に動かして質を上げる。

---

1 考える

なぜ役割を分けるか

調査役と検証役を分けると何が良いか、検証役に編集権限を渡すべきかを予想し、`memo.md` に書きます。

2 書く

並列調査の指示を書く

達成条件

- 3つの論点を researcher に渡し、並列で調べさせている
- 主張と、日付つきの出典URLが対になって返るよう求めている
- 軸になる主張は source-checker で裏取りする流れになっている

リサーチ起点から3論点を選び、`researcher` サブエージェントに並列で調べさせる指示を自分の言葉で書きます。各論点で「主張…出典URL（日付）」の対を返させます。

Tips

3論点を1つの指示にまとめて渡すと、researcherが並列で動きます。1論点ずつ順番に投げると逐次になり時間がかかります。

注意

researcherは調査専任、source-checkerは検証専任です。検証役には編集権限を渡さず、判定だけ返させます。

3 実行

裏取りする

集まった主張のうち、提案の軸になりそうな5つを `source-checker` に渡し、出典が一次情報か・日付が新しいかを検証させます。弱いものは差し替え候補を出させます。

Tips

全主張を検証すると時間が溶けます。軸になる主張を5つに絞ります。

4 答え合わせ

hints と比べる

`hints/step02_subagent_hint.md` を開き、並列の頼み方と権限分離の考え方を確認します。

自分で考えるポイント

- 3論点をどう選べば、後のレポートのWhat（分かったこと）が埋まるか
- source-checkerが「裏取れず」とした主張を、提案でどう扱うか

達成チェック

- [ ] 3論点をresearcherに渡し、並列で調査が走っている
- [ ] 各論点のノートで、主張と出典URL（日付）が対になっている
- [ ] 軸になる5つの主張をsource-checkerで検証し、弱い出典が差し替えまたは保留になっている

うまくいかないとき

サブエージェントが呼べないときは、`.claude/agents/` に該当ファイルがあるか、frontmatterの `name` がプロンプトの呼び名と一致するかを確認します。読めていなければファイル名とname欄を教えてもらいます。直らなければ「詰まったときの即応」と `hints/step02_subagent_hint.md` を見ます。

発展課題（早く終わった方向け）：「裏取れず」の主張を別ソースで再調査

source-checkerが弱いと判定した主張を1つ選び、researcherに別ソース（公式発表・統計・IR資料など一次情報）で再調査させ、再びsource-checkerにかけます。対象は `成果物/research/` の検証済み主張リスト。達成条件は、再調査した主張が「裏取れた」か「保留」に区分され根拠URLが残っていること。詳細は `hints/step02_subagent_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 3 reportで4層に束ねる

これまでの調査ノートを、Why / What / How / So what の4層に束ね、提案の素地を作ります。So whatをA社向けの優先順位の仮説まで踏み込ませることで、調べただけの資料から一歩進んだ状態を目指します。この4層レポートが、次のスライドプランの下敷きになります。

場所 **Claude Code**　対象 **成果物/research/、_templates/レポート_テンプレ.md**

成果物 **成果物/report/ に4層レポート（各主張に出典）**　目安 **[20min]**

1 考える

So what をどう出すか

調べただけの資料で終わらせないために、So whatに何を書くべきかを予想し、`memo.md` に書きます。

2 書く

4層の指示を書く

達成条件

- Why / What / How / So what の4層で束ねるよう指示している
- What の事実と数字に出典を残すよう求めている
- So what が「何から手を付けるか」の優先順位になるよう求めている

調査ノートを4層に束ねる指示を、自分の言葉で書きます。Whatは事実と数字に出典を残し、So whatは「A社が何から手を付けるべきか」の優先順位の仮説まで踏み込ませます。配布の `/report` はまだ使いません。

Tips

`/report` は同じ意図のひな型です。まず自分で指示を書き、答え合わせのときに見比べます。

注意

Whatを盛りすぎると論点がぼやけます。提案に効く事実に絞ります。

3 実行

レポートを出す

指示を実行し、`成果物/report/` にレポートを出します。結論から始まり、段落が短いかを確認します。

Tips

So whatが「重要だ」で止まっていたら、「何から着手するかの順番まで書いて」と追加で頼みます。

4 答え合わせ

hints と比べる

`hints/step03_report_hint.md` を開き、4層の型と比較観点を確認します。配布の `/report` が同じ意図のひな型だと分かります。

自分で考えるポイント

- Why（なぜ今）を、A社の相談内容とどう結び付けるか
- So whatを一般論でなくA社向けの優先順位にするには、何を根拠にするか

達成チェック

- [ ] 成果物/report/ にレポートが作られ、Why / What / How / So what の4層がすべて埋まっている
- [ ] Whatの事実と数字に出典が残っている
- [ ] So whatが「何から手を付けるか」の優先順位の仮説になっている

うまくいかないとき

So whatが一般論のままのときは、与件メモの制約（半年で意思決定、来期から本格投資）を根拠に着手順を出させます。「半年で意思決定・来期から本格投資という制約を踏まえ、最初に手を付ける2〜3手を理由と順番付きで示して」と頼みます。直らなければ「詰まったときの即応」と `hints/step03_report_hint.md` を見ます。

発展課題（早く終わった方向け）：読み手をCIOに変えてレポートを組み替え

調査ノートは変えず、読み手を経営からCIOに変え、技術的な実現性と情報管理の論点を厚くしたレポートを別ファイルで作ります。対象は `成果物/report/` の4層レポート。達成条件は、経営向けとCIO向けの2版があり、強調点の違いが見出しとSo whatに表れていること。詳細は `hints/step03_report_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 4 slide-plan で 1枚1メッセージ

レポートを、そのままスライドの設計図に割ります。pptxを生成する前に構成を文章で固めておくと、各ページのメッセージが1つに絞られた状態を目指せます。調査で束ねた中身を、提案の形に組み替える最初

の一歩です。

場所 **Claude Code** 対象 **成果物/report/ のレポート** 成果物 **成果物/slides/plan.md** 目安 **[25min]**

---

1 考える

1枚1メッセージとは

1ページに何を1つだけ載せるか、表紙からまとめまでの並びを頭で描き、`memo.md` に予想を書きます。

---

2 書く

構成案を書く

達成条件

- 提案の幹（What）が1文で言える構成になっている
- 各ページの見出しを体言止めにするよう指示している
- 1ページに載せるメッセージを1つに絞るよう求めている

レポートをスライドプランに割る指示を、自分の言葉でターミナルに書きます。提案の幹（What）を1文にし、各ページに見出し（体言止め）・要点3つ以内・根拠の出典を含める、と条件を渡します。

注意

配布の `/slide-plan` はまだ使いません。まず自分でプロンプトを組みます。

---

3 実行

plan.md を出す

指示を送り、`成果物/slides/plan.md` が作られるのを待ちます。

Tips

1ページに2つ以上のメッセージが混ざっていないか、見出しが説明文になっていないかを声に出して確認します。

---

4 答え合わせ

hints と比べる

`hints/step04_slideplan_hint.md` を開き、割り方と比較観点を照らします。

自分で考えるポイント

- 提案の幹（What）を1文で言うと何になるか
- 表紙・背景・現状と論点・分かったこと・打ち手・優先順位・進め方・まとめ のどこに、どの主張を割り当てるか
- 要点3つに絞ると、落とす情報はどれか

達成チェック

- [ ] 成果物/slides/plan.md が作られ、What（提案の幹）が1文で言える
- [ ] 各ページの見出しが体言止めになっている
- [ ] 1ページに載るメッセージが1つに絞られ、hintsと割り方を比べた

うまくいかないとき

1ページに要点が詰まりすぎるときは、plan.mdを正として、そのページを論点ごとに2枚へ割る指示を出します（各ページ1メッセージに直す）。見出しが長い説明文になるときは「見出しを体言止めの短い名詞句に直して」と頼みます。直らなければ「詰まったときの即応」と `hints/step04_slideplan_hint.md` を見ます。

発展課題（早く終わった方向け）：1ページ2枚割りの読み比べ

plan.mdの中身（主張・出典）は変えず、要点が多いページを2枚に割った版を `plan_split.md` として作り、元の1枚版と読み比べます。割る前に、そのページの要点を「同格に並ぶもの」か「順番があるもの」かで見分けると割り方が決まります。達成条件は、`plan_split.md` があり1枚版と2枚版の違いを2〜3行で言えること。詳細は `hints/step04_slideplan_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 5 ・ 本日の山場　DESIGN.md × build-pptx で生成

見た目の方針を DESIGN.md に決めてから、plan.md の構成を提案スライド（PPTX）に起こします。中身の正は plan.md、見た目の正は DESIGN.md と役割を分けることで、生成結果が方針どおりに収まる状態を目指します。

場所 **Claude Code＋PowerPoint等で確認**

対象 **成果物/slides/plan.md、_templates/DESIGN.md**

成果物 **成果物/slides/output.pptx**　目安 **[35min]**

---

1 考える

DESIGNに何を書くか

配色・フォント・1枚1メッセージを崩さないために、DESIGN.md に何を決めておくべきかを `memo.md` に書きます。

---

2 書く

DESIGNを案件向けに調整

達成条件

- plan.md の構成どおりの枚数・順番になるよう指示している
- 配色を2系統に収め、信号機カラーと絵文字を使わないよう指定している
- 文字がはみ出さないよう、1ページあたりの分量の上限を伝えている

`_templates/DESIGN.md` を案件向けに調整します。配色は2系統、フォントはNoto系、絵文字なし、が条件です。

注意

赤と緑の信号機カラー、3色以上の原色は避けます。同系色の濃淡で差を出します。

---

3 実行

build-pptx で生成して開く

`/build-pptx` で plan.md から `成果物/slides/output.pptx` を作ります。生成後、ファイルを開いて中身を見ます。

Tips

生成は python-pptx を uv 経由（ `uv run --with python-pptx python ...` ）で動かします。uvが無いとエラーになります。

---

4 答え合わせ

hints と比べる

`hints/step05_pptx_hint.md` を開き、生成の頼み方とつまずきの対処を照らします。

自分で考えるポイント

- DESIGN.md に「配色2系統」以外で書いておくべき方針は何か
- plan.md の順番と、生成された pptx の順番はそろっているか
- 生成結果が plan.md から盛られていたら、どう戻すか

達成チェック

- [ ] _templates/DESIGN.md を案件向けに調整した
- [ ] 成果物/slides/output.pptx が生成され、開けた
- [ ] plan.md の構成どおりの枚数・順番になっている
- [ ] 配色が2系統に収まり、信号機カラー・絵文字が無く、文字のはみ出しが無い
- [ ] hints/step05_pptx_hint.md と比べた

うまくいかないとき

pptxが生成できないときは、uv が入っているか確認し、事前セットアップのDay2依存（uv/Python）を見直します。「エラーメッセージをそのまま見て、uv や python-pptx が原因なら uv run --with python-pptx python の形で実行し直す手順を教えて」と頼みます。plan.mdを無視してAIが内容を盛るときは「plan.md を正として、planから外れた点を戻して」と指示します。直らなければ「詰まったときの即応」と `hints/step05_pptx_hint.md` を見ます。

発展課題（早く終わった方向け）：配色違い版の印象比較

plan.mdの中身は変えず、DESIGN.mdの配色だけ変えた別版を `output_alt.pptx` として生成し、元版と並べて見比べます。配色は2系統を守り、信号機カラーと原色レインボーは使いません。色相を変えるより、同系色の濃淡や彩度を動かすと2系統に収めやすいです。達成条件は、`output_alt.pptx` があり2版の印象差を2〜3行で言えること。詳細は `hints/step05_pptx_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 6 critic で反論・Vaultへ実績還元

提案の穴を critic サブエージェントに突かせ、経営が聞いてくる反論に先回りして直します。直したあと、今回の研究から提案までで効いた設定をVaultへ戻し、次の案件で `/load-setup` から再現できる状態を目指します。Day2の締めで、演習プロジェクト側の成果をVault（貯める側）へ還元します。

場所 **Claude Code＋Vaultへ保存**

対象 **レポート・プラン・pptx、templates_master、Vault _研修/_実績**

成果物 **反論リストと修正、Vault _研修/_実績 の設定**

目安 **[30min]**

---

1 考える

経営は何を聞くか

経営が必ず突いてくる反論（コスト・リスク・実行可能性）を、`memo.md` に予想で挙げます。

---

2 書く

criticで反論させる

達成条件

- 提案の弱点を突く立場で読ませる指示になっている
- 反論が重要度の高い順に出るよう求めている
- 各反論が、直し方とセットで返るよう求めている

配布フォルダの `templates_master/agents/critic.md` を、演習プロジェクトの `.claude/agents/` にコピーします（zipで受け取った場合は解凍してから取り出します）。置いたら、レポートとプランに反論させ、重要度順に、直し方とセットで出させます。

注意

criticは賛成せず穴だけを突く役です。褒めさせず、弱点だけ出させます。

3 実行

直して実績に戻す

重要な反論を反映して直します。そのあとVaultに `/save-setup` で今回の設定を残します。

> **Tips**
> 保存日は自分で指定します。日付を渡さないと当日になります。

4 答え合わせ

hints と比べる

`hints/step06_critic_hint.md` を開き、反論の引き出し方と実績還元を照らします。

自分で考えるポイント

- criticが出した反論のうち、経営会議で最も刺さるものはどれか
- 反論に「直し方」が付いているか。付いていなければどう促すか
- Vaultに残す設定は、次回どの粒度で再現できると使えるか

達成チェック

- [ ] critic が .claude/agents に置かれ、反論が出た
- [ ] 反論が重要度順で、直し方とセットになっている
- [ ] 重要な反論を反映してレポート・プランを直した
- [ ] /save-setup でVaultの _研修/_実績 に今回の設定を残した
- [ ] hints/step06_critic_hint.md と比べた

うまくいかないとき

criticが褒める・賛成してしまうときは、弱点だけ出す役だと明示し直します。「賛成やよい点は書かず、弱点だけを経営が突く順（コスト・リスク・実行可能性）で、各指摘に直し方を付けて出して」と頼みます。サブエージェントが呼べないときは `.claude/agents/critic.md` のパスとfrontmatterの name を確認します。直らなければ「詰まったときの即応」と `hints/step06_critic_hint.md` を見ます。

発展課題（早く終わった方向け）：効いた指摘の persona 反映

persona.mdの既存の役割は消さず、criticの指摘で最も効いたものを次回の頼み方として persona.md に1〜2行で追記します。対象はVaultの `40_運用/persona.md` 。具体の指摘そのものより「コスト・実行可能性を先に自己点検する」といった手順に一般化すると次回に効きます。達成条件は、persona.md に観点が追記され次回 `/load-setup` で読み込める場所にあること。詳細は `hints/step06_critic_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 7　どこで使っても記録が残る仕組み

ここからは、Vault そのものを自動で回す仕組みを2つ作ります。1つ目は記録です。どのフォルダで Claude Code を使っても、送った命令が Vault に残るようにします。記録が1か所にたまると、「自分がどんな作業をくり返しているか」が後から見えます。それを見て頼み方を型にすれば、たまったログが次の案件で使える資産に変わります。

場所 **Claude Code（Vaultを開いたもの）**　対象 **~/.claude/settings.json、~/.claude/hooks/**

成果物 **40_運用/log.md に自動でたまる記録**　目安 **[25min]**

1 考える

どこに登録すれば「どこでも」効くか

Claude Code の設定には、そのフォルダを開いたときだけ効くものと、どのフォルダでも効くものの2種類があります。今回のゴールは「演習フォルダで打った命令も、Vault で打った命令も、同じ場所に残す」ことです。どちらに登録すべきかを、書く前に決めます。

2 書く

記録の仕掛けを登録する指示を書く

達成条件

- 命令を送るたびに動く仕掛け（UserPromptSubmit）が登録されている
- 登録先が、どのフォルダでも効く設定（`~/.claude/settings.json`）になっている
- 記録用のスクリプトが `~/.claude/hooks/` に置かれ、絶対パスで呼ばれている
- Vault の絶対パスが、環境変数 `OBSIDIAN_VAULT` で渡されている
- もとから入っていた設定は壊れておらず、JSON として妥当なままである

書くときに使える情報

Vault の `.claude/hooks/log-prompt.mjs` に、命令を `40_運用/log.md` へ書き足すスクリプトが入っています。これを共通の置き場へ持っていきます。Vault の絶対パス（`/Users/…/PMVault` のような正式な住所）は `pwd` で分かるので、先に確認しておくと確実です。既存の設定を壊さないために、「マージして」「変更前と変更後を差分で見せて」と伝えてください。

注意

設定を書き換えても、いま開いている Claude Code には反映されません。次の「実行」では**新しいターミナル**を開いて確かめます。いまのセッションは、あとで確認に使うので閉じないでください。

3 実行

Vault と関係のない場所から命令を送る

新しいターミナルを開き、デスクトップへ移動して Claude Code を起動します。

```
cd ~/Desktop
claude
```

起動したら、自分で考えた命令を1つ送ります。中身は何でも構いません。「いま何時か教えて」のような、すぐ終わるもので十分です。ここで見たいのは命令の結果ではなく、その命令が Vault 側に残るかどうかです。

4 答え合わせ

ログを見て、hints と比べる

最初に開いていた Vault 側の Claude Code に戻り、`40_運用/log.md` の末尾を見せてもらいます。さきほど Desktop から送った命令が `[Desktop]` の印つきで並んでいれば成功です。どこで打った命令かまで残ります。そのうえで `_研修/hints/step05_loghook_hint.md` を開き、仕組みと自分の指示を照らします。

自分で考えるポイント

- そのフォルダだけに効く設定に入れると、何が記録されなくなるか
- たまったログから何を読み取れば、次の案件の頼み方に活かせるか

達成チェック

- [ ] どのフォルダでも効く設定に、命令を記録する仕掛けを登録できている
- [ ] Desktop など Vault 以外の場所から送った命令が 40_運用/log.md に残っている
- [ ] 記録にフォルダ名の印が付き、どこで打った命令か分かる

うまくいかないとき

記録されないときは、まず設定を書き換えたあとに開いたターミナルかどうかを確認します。古いセッションのままだと新しい設定が効きません。次に `OBSIDIAN_VAULT` のパスが、いま使っている Vault と一致しているかを見ます。ずれていると別の場所の log.md に書かれます。「~/.claude/settings.json の UserPromptSubmit を見せて。OBSIDIAN_VAULT の値がこの Vault のパスと一致しているか確認して」と頼めば分かります。JSON が壊れているとフック自体が読み込まれないので、「JSON として妥当か検証して」も試してください。

発展課題（早く終わった方向け）：たまったログを頼み方に還元する

`40_運用/log.md` の今日の分だけを読ませ、結果がよかった命令を3つ選ばせます。その3つに共通する頼み方を1〜2行にまとめ、`40_運用/persona.md` の応答方針へ反映します。書き換える前に変更前後を見せてもらってください。達成条件は、命令文そのままでなく、ほかのテーマでも効く形に一般化されていること。詳細は `_研修/hints/step05_loghook_hint.md` の「発展課題の参考解」を参照します。

EXERCISE 8

1日1回、Vault が自動で整う仕組み

2つ目は整理です。放っておくと、Vault の直下にはメモが散らかっていきます。それを1日1回、自動で決まったフォルダへ移す仕組みを作ります。ここまでできると、記録は勝手にたまり、たまったものは勝手に整う状態になります。

場所 **Claude Code（Vaultを開いたもの）** 対象 **~/.claude/hooks/organize-vault-daily.sh**

成果物 **1日1回動く整理の仕組みと 40_運用/organize.log** 目安 **[25min]**

---

1 考える

何を動かし、何を触らせないか

自動でファイルを動かす仕組みなので、範囲を決めずに任せると事故になります。「何を移してよいか」「何は絶対に触らせないか」を、指示を書く前に決めます。整理の考え方そのものは、Vault に入っている organize-vault の型（決まったフォルダへ移す、消さない）をそのまま使います。

2
書く

自動で整理するスクリプトを作らせる

達成条件

- `~/.claude/hooks/organize-vault-daily.sh` ができている
- 動かすと、Vault の直下に散らかった新しいノートだけが、決まったフォルダへ移される
- もとからあるフォルダの構造は触らず、ファイルの削除はしない
- 人が見ていなくても、確認を求めずに最後まで動く
- 実行した日時と結果が `40_運用/organize.log` に追記される
- 実行できる状態になっていて、中身が何をしているかの説明も受けている

書くときに使える情報

スクリプトの中から Claude Code を呼ぶときは `claude -p "指示"` と書きます（画面を開かずに1回だけ実行する形）。Vault の場所は絶対パスで伝えてください。`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。「削除はしない」「確認を求めずに最後まで動く」の2つは、必ず言葉にして伝えてください。書かないと、途中で止まるか、意図しない動きになります。

安全のために

いきなり全部お任せにせず、最初は「移動せずに提案だけ」から始めるのが安全です。数日ログを眺めて動きに納得してから、実際に移動させる形へ切り替えてください。切り替えも Claude Code に頼めば直してくれます。

3 実行

1日1回動くように登録し、その場で試す

作ったスクリプトを、決まった時刻に自動で動かす登録をします。macOS には launchd という、決めた時刻にプログラムを動かすしくみがあります（目覚まし時計のようなものです）。設定ファイルの書き方は覚えなくてよいので、置き場所とやってほしいことを伝えて任せます。

Claudeへの指示（コピーして送る）

```
いま作った ~/.claude/hooks/organize-vault-daily.sh を、1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。要件は次のとおりです。

・plistの置き場所は ~/Library/LaunchAgents/com.pmvault.organize-daily.plist。
・毎日09:00に起動する（StartCalendarInterval）。
・環境変数 OBSIDIAN_VAULT に、このVaultの絶対パスを設定する。
・標準出力と標準エラーを ~/Library/Logs/pmvault-organize.log に出す。
・登録（load）まで実行する。

終わったら、launchctl list に com.pmvault.organize-daily が出ているか確認して教えてください。
```

登録できたら、朝9時を待たずにその場で1回動かして確かめます。手で動かすコマンドは `launchctl start com.pmvault.organize-daily` です。自分で打たず、Claude Code に頼んで構いません。動き終わるまで少し時間がかかるので、待ってからログを見るように伝えてください。

4 答え合わせ

2つのログを見て、hints と比べる

`40_運用/organize.log` に実行日時の行が増えているか、`~/Library/Logs/pmvault-organize.log` に目立ったエラーが出ていないかを確認します。直下に散らかったノートがあれば、適切なフォルダへ移り、削除は起きていないはずです。そのうえで `_研修/hints/step07_autoorganize_hint.md` を開き、自分の指示と比べます。

自分で考えるポイント

- 自動整理に任せてよい範囲と、人が決めるべき範囲の線引きはどこか
- 毎日動かすとして、何を見れば「うまく回っている」と判断できるか

達成チェック

- [ ] organize-vault-daily.sh を作り、実行できる状態にした
- [ ] 1日1回、自動で動くように登録した
- [ ] 手で1回動かし、organize.log に記録が残ってエラーが出ていない

うまくいかないとき

登録されているかは「launchctl list に com.pmvault.organize-daily が出ているか確認して」と頼めば分かります。動かない場合は、設定ファイルの中に書かれたスクリプトのパスが実在するかを見ます。`~/Library/Logs/pmvault-organize.log` にエラーが出ていたら、その文をそのまま Claude Code に貼って「これは何が原因？」と聞くのが最短です。時刻は9時でなくても構いません。自分の生活に合う時刻を伝えれば直してくれます。

発展課題（早く終わった方向け）：整理の結果を要約させる

1日1回の整理が走ったあと、その日に何がどこへ移ったかを3行で要約し、`40_運用/hot.md` に追記する処理を足します。対象は `40_運用/organize.log` と `40_運用/hot.md` 。達成条件は、翌朝 hot.md を見るだけで前日の整理内容が分かること。詳細は `_研修/hints/step07_autoorganize_hint.md` の「発展課題の参考解」を参照します。

CHECK 理解度チェック

Q1. researcher / source-checker に共通する設計は？

- 何でもできるよう全権限を渡す
- 必ず1体ずつ順番に動かす
- 役割を絞り、検証役には編集権限を渡さない
- 調べる役と検証する役を1体にまとめる

Q2. pptxを作る前に slide-plan を挟む理由は？

- pptxは作らなくてよいから
- 1枚1メッセージの構成を先に固め、詰め込みすぎを防ぐため
- 配色を決めるためだけ
- スライドの枚数を自動で増やすため

Q3. Day2の最後に /save-setup でVaultに戻すものは？

- 今回効いた設定・頼み方（次案件で load-setup できる）
- クライアントの機密データ
- pptxの実ファイルだけ
- 調査に使った元データを丸ごと全部

Q4. 演習フォルダで打った命令も Vault に記録したい。どこに登録する？

- Vault の .claude/settings.json
- 演習プロジェクトの .claude/settings.json
- Obsidian の設定画面
- ~/.claude/settings.json（どのフォルダでも効くユーザー設定）

Q5. 1日1回の自動整理を作らせるとき、指示に必ず入れるべきものは？

- 実行する時刻を秒まで指定すること
- ファイルは削除しないこと、確認を求めずに最後まで動くこと
- 整理したファイルを圧縮すること
- 古いノートを自動で消すこと

WRAP UP 2日間のまとめ

到達点

Vault（第二の脳）に知識を蓄え、別フォルダの演習で公開情報の調査から提案スライドまでを通しました。効いた設定は /save-setup でVaultに残し、次の案件で再現できます。さらに、どこで使っても記録が残り、1日1回ひとりでに整う仕組みまで組み込みました。特別な操作をしなくても知識はたまり、たまったものは整っていきます。

持ち帰り

- Vault と演習プロジェクトの2フォルダを、自分の業務テーマで1セット作る
- ログを見直し、効いた頼み方を persona.md に反映する
- たまったログをときどき見返し、くり返している作業を仕組みに変える
- 機密は入れない。公開情報と架空題材で仕組みを回す

発展 コンサルの引き出し（研修後）

提案を速くするため、繰り返し参照する層を Vault に蓄積します。いずれも4手がかり（type/when/topic/status）にそのまま乗り、研修で使ったリサーチ・提案の流れと同じ作法です。

毎日引く層

答え合わせ ー

置き場	何を貯めるか
`10_業界と事例/industries/`	1業界1枚。論点・KPI・規制・代表的取り組み（一次情報URL＋日付）
`10_業界と事例/cases/`	1事例1カード。成熟度（PoC／本番／全社）と一次情報URL
`00_ナレッジ/benchmarks/`	再利用する数値。取得日と確からしさ付き
`20_案件/engagements/`	1案件1フォルダ。初日に開く案件プロファイル1枚
`00_ナレッジ/frameworks/`	3C・5F など、何を分析するかの型

提案の型と例え話

- 提案ストーリーの型（SCQA・ピラミッド・空雨傘）は `00_ナレッジ/concepts/提案ストーリーの型.md` 。見出しは主張の完全文にして横読みで筋を確認します
- 見聞きした提案の構成は `_templates/参考ストーリーメモ_テンプレ.md` で型として貯めます（公開情報のみ）。AI に型を渡すと提案ドラフトの初稿を作れます
- 経営層への例え話は `00_ナレッジ/concepts/アナロジー設計の型.md` と `00_ナレッジ/concepts/例え話見本_技術とAIを経営層へ.md` 。必ず崩れる点を併記し、陳腐な比喩は使いません

置き場の地図

全体の入口は Vault の `index.md` のサブ索引。何がどこにあるかはそこから辿れます。

HELP 詰まったときの即応

答え合わせ ー

症状	初動
load-setup が効かない	Vaultの `_研修/_実績` に保存済みか、INDEXに行があるか確認
サブエージェントが呼べない	`.claude/agents/名前.md` のパスと frontmatter の name
出典が薄い	「一次情報の出典を付けて」と明示／source-checkerで検証
pptxが生成できない	uv の有無を確認（事前セットアップDay2依存）。`uv run --with python-pptx python スクリプト名.py` の形で実行し直す
planから外れて盛られる	plan.md を正として「planから外れた点を戻して」と指示