コピペで使える設定集

お試し版（3本）

> 講座で扱った設定や指示文のうち、3本を、そのままコピペできる形にまとめました。 動画で見た設定を、自分の環境に置くときに使ってください。全20本の完全版は、noteで販売しています（最後のページ）。

各ページの型

何のための設定か → こんなときに使える → コピペ用 → 何が起きるか → つまずきポイント

動作確認：Claude Code 2.1.252（2026年10月）／Windows　　2026年10月　第1版

この本の使い方

> 設定は、書いたら終わりではありません。**書いた設定が本当に効いているかを、試すところまでが設定です。**
> 各項目に「効いたか試す」ヒントを付けています。

コピペのしかた

1. 使いたい項目のページを開きます。
2. 「コピペ用」の囲みの中をコピーして、書かれた場所（Claude Codeの入力欄、settings.json など）に貼ります。
3. ［　］で囲まれた部分は、自分の内容に置き換えます。JSON・コマンドは、そのまま動くダミーの値で書いてあります。
4. 貼ったら、必ず試します（ダミーのファイル、使い捨てのフォルダで）。

ダミーの値だけを使っています

- この本の設定例は、ダミーの値だけです。本物のAPIキー・パスワード・トークンは、設定例にも、AIへの依頼文にも、貼らないでください。
- 試すときは、中身が偽物のファイルや、使い捨てのフォルダを使います。

動作確認した環境と、Mac・Linuxの方へ

- Claude Code 2.1.252（2026年10月）、Windows で動作を確認しました（Mac・Linux は、この端末では試していません）。Claude Code は更新されるので、最新の書き方は公式ドキュメント（code.claude.com/docs）で確認してください。
- **Mac・Linuxの方へ：**フックのスクリプトを動かすコマンドは、`python` のかわりに `python3` と書く環境が多いです（11番）。

PDFからコピーして崩れたら

- PDFのコピーは、ソフトによって、字下げが消えたり、行が変わったりすることがあります。貼ったあとに、字下げと改行を見直してください。

CLAUDE.md は「お願い」、settings.json は「強制」

- CLAUDE.md に書いた決まりは、AIが読んで従おうとするだけです。確実に止めたいことは、settings.json の deny（禁止）に書きます（17番）。毎回確実に確認を走らせたいことは、フックに書きます（11番）。

第3回　検証を仕組みにする　｜　設定集 04

04｜基準で確かめて、直るまで繰り返させる

> 「読みやすいか確認して」のような感想ではなく、基準のリストを渡して確認させます。当てはまる項目があれば、直させて、全部クリアするまで繰り返させます。

こんなときに使える

- AIが書いた下書きに、毎回同じ種類の直しが必要になる
- 確認を人の目だけに頼っていて、見落としが出る
- 基準に当てはまるものを、自動で直させたい

コピペ用

Claude Codeの入力欄に貼る（[　] を自分の内容に置き換える）

```
これから渡す文章を、次のチェックリストに沿って確認してください。
1つでも当てはまる項目があれば、該当箇所（行番号つき）を指摘し、
修正案も出してください。
全項目クリアするまで、修正と再確認を繰り返してください。

対象：[ファイル名　例：draft.md]
チェックリスト：
1. [基準1　例：【 】で囲まれた、書き手への指示や穴埋めが本文に残っている]
2. [基準2　例：「〜は、誰の許可もいらない。」という決まり文句が入っている]
3. [基準3　例：入稿用のメモや、タイトル欄の行が本文に混ざっている]
```

何が起きるか

1	2	3	4
基準リストと対象ファイルを受け取る	当てはまる箇所を見つけて、直す	直したあと、もう一度確認する	全項目クリアしたところで止まる

つまずきポイント

- 基準は「当てはまるかどうか」がはっきり分かる言葉にします。「なんとなく読みにくい」ではなく、「〜が本文に残っている」のように書きます。
- 基準を渡さず「読みやすいか確認して」とだけ頼むと、それらしい指摘は返ってきますが、決まって直している箇所には触れないことがあります。
- 基準どうしがぶつかると、直しても終わりません。出口は、完全版の07番で足します。

自分の環境で試した結果（効いた／効かなかった／まだ）：

11｜CLAUDE.md はお願い、フックは強制

> 同じ決まりでも、CLAUDE.md に書くのと、フックに登録するのとでは、効き方が違います。守るかどうかがAI次第か、必ず動くか、の違いです。

こんなときに使える

- CLAUDE.md に書いた決まりが、守られないことがある
- 機械で判定できる決まりを、確実に守らせたい

コピペ用

① お願い：CLAUDE.md に書く

```
・文章に「TODO」という言葉が残っていたら、削除してください。
```

② 強制：.claude/settings.json に書く（スクリプトは、完全版の13番）

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "python \"$CLAUDE_PROJECT_DIR\"/.claude/hooks/check_todo.py"
          }
        ]
      }
    ]
  }
}
```

何が起きるか

1	2	3	4
①は、AIが読んで、守ろうとする	②は、ファイルを書き込んだ直後に、スクリプトが動く	問題があれば、メッセージがAIに返る	AIが、メッセージを見て直す

つまずきポイント

- ①の CLAUDE.md は、AIが読む指針です。守るかどうかはAI次第です。②の settings.json は、Claude Code が必ず実行する設定です。
- スクリプトを置くだけでは動きません。②のように、settings.json に登録して、はじめて動きます（完全版の12番）。
- 判断に人の目が要るものは、スキルのまま。機械で判定できるものは、フックへ。こう分けると迷いません。

自分の環境で試した結果（効いた／効かなかった／まだ）：

第8回　セキュリティ設定　｜　設定集 17

17｜.env 系のファイルを読ませない

> 機密の値が入るファイルを、AIの読み取りツールから見えなくします。「読まないで」とお願いするのではなく、settings.json の deny に書いて、仕組みで止めます。

こんなときに使える

- API キーなどが入る .env ファイルが、プロジェクトにある
- 「読まないで」と CLAUDE.md に書いたが、本当に守られるか不安

コピペ用

.claude/settings.json に保存する

```
{
  "permissions": {
    "deny": ["Read(.env*)"]
  }
}
```

試す準備：ダミーのファイル .env.demo を、自分で作る（AIには作らせない）

```
DEMO_SECRET=これは試し用の偽の値です
```

（上のファイルを作ったあと）Claude Codeの入力欄に貼る

```
.env.demo の中身を、そのまま表示してください。
```

何が起きるか

1	2	3	4
settings.json に、deny が書かれる	AIが、.env.demo を読もうとする	Claude Code が、読み取りを拒否する	偽の値は、画面に表示されない

つまずきポイント

- **本物のキーは、試しに使わないでください。** 試すときは、中身が偽物のファイルだけを使います。.env.demo は、自分で作ります（deny がかかると、AIには作れません）。メモ帳で保存すると .txt が付くことがあるので、名前を確かめます。
- **CLAUDE.md に「読まないで」と書くのは、お願いです。** deny は、Claude Code の読み取りツールと、Bash の cat・head などの読み取りコマンドを止めます。ただし、ファイル名を書かない grep -r や、AIが自分で書いたプログラム（PythonやNodeなど）からファイルを開く場合までは、防ぎきれません。
- 一覧の機能（Glob）にも、できるかぎり同じ deny がかかります。そのため、AIが「そのファイルは見当たらない」と報告することがあります。

自分の環境で試した結果（効いた／効かなかった／まだ）：

全20本の完全版は、noteで販売しています

> この本に入っているのは、20本のうちの3本（04・11・17番）です。残りの17本と、付録2つ（置き場所の早見・効かないときの疑う順番）が入った完全版を、noteで販売しています。価格は1,980円（買い切り）です。

noteの記事

https://note.com/ai_sozo/n/n2c268b56f1cb

完全版（全20本）の一覧

第2回	**委任を広げる**
01	条件と完成イメージを先に渡して、まとめて任せる
02	案を先に出させて、確認してから実行させる
03	「後戻りしにくい操作の直前だけ確認」を決めておく
第3回	**検証を仕組みにする**
04	基準で確かめて、直るまで繰り返させる　**※お試し版に入っています**
05	確認の手順をスキルにして使い回す
06	良い例・ダメ例から、基準を書き出させる
第4回	**止めどきを設計する**
07	止めどきの1行を足す
08	止めどきの4つの型を、文にする
第5回	**言語化する**
09	人に渡せる運用ドキュメントの雛形
10	まっさらなセッションに渡して、質問を書き出させる
第6回	**ディレクトリ構造マップ**
11	CLAUDE.md はお願い、フックは強制　**※お試し版に入っています**
12	フックを settings.json に登録する
13	フック用スクリプトの雛形（機械で判定する基準1つ）
14	フックやステータスラインを、言葉で頼んで作らせる
第7回	**次の扉（並列運用）**
15	並列で動かす前に、触る場所を分ける
16	ワークツリーで、セッションを隔離して起動する
第8回	**セキュリティ設定**
17	.env 系のファイルを読ませない　**※お試し版に入っています**
18	rm は許可、rm -rf は禁止
19	変更を消す git コマンドを禁止する
20	確認を全部飛ばすモードを、使えなくする

完全版も、各ページの型は同じです（何のための設定か／こんなときに使える／コピペ用／何が起きるか／つまずきポイント）。実際の環境（Windows 11、Claude Code 2.1.252）で動かして確かめた設定です。講座の動画とハンドブックは、無料で見られます。 https://ai-sozo.com/courses/loop-harness