KIMKJ AI Library

AIに任せて席を離れる

YOLOモード完全入門. 目次と26章

キム・ギョンジン弁護士

Japanese PDF edition

AIに任せて席を離れる

キム・ギョンジン弁護士

Claude CodeとCodexのYOLOモードを初めて使う人のためのオンライン書籍です。AIにファイル読み取り、コード作成、コマンド実行を任せながら、取り消し、Dockerサンドボックス、安全確認を手元に置く流れを説明します。

目次

第1章 午前2時、ノートパソコンの前にいる会社員

AIに任せて席を離れる - YOLOモード完全入門

次の章: 第2章 AIが一人で働くとはどういうことか

画面の光が顔を照らしていました。午前2時、ソウル・麻浦区のあるオフィステル。キム代理は翌朝9時までに提出しなければならない四半期報告書をにらんでいました。Excelファイルが3つ、PowerPointのスライドが22枚、取引先ごとの売上データをまとめたCSVファイルが5つ。それらをひとつの報告書にまとめなければなりませんでした。

コーヒーカップはすでに3杯目でした。

キム代理は普段からChatGPTを使っています。議事録の要約、メールの下書き、簡単な翻訳。そういった作業にはなかなか便利でした。ところが今回の報告書作成は、次元が違いました。ファイルを開き、データを読み込み、計算して、グラフを作り、スライドに貼り付ける作業が必要だったからです。ChatGPTの画面に「報告書を作って」と入力すれば、それらしい文章は出てきますが、実際のExcelファイルを開いて数字を拾い出すことはできません。

その夜、キム代理はひとつのことを試みました。

Terminal（ターミナル）という黒い画面を開き、AIにこう告げました。「このフォルダにあるExcelファイル3つとCSVファイル5つを読み込んで、取引先ごとの四半期売上サマリー表を作り、グラフ入りのPowerPoint報告書を生成して。」そしてEnterキーを押しました。AIが動き始めました。ファイルを開き、コードを書き、表を作り、グラフを描きました。キム代理はその過程を眺めているうちに、気づけばノートパソコンを閉じて眠りにつきました。

朝7時に目が覚めると、フォルダの中に完成したPowerPointファイルが入っていました。グラフも入っていて、数字も合っていました。キム代理がしたことといえば、「これをやって」と指示しただけでした。

この話が、この本の出発点です。

「ヨロ」という言葉を聞いたことがあるでしょう。YOLO。You Only Live Once。「人生は一度きり。」2010年代初頭、ラッパーのDrakeの曲名として広まったこの言葉は、かつて若い世代の消費哲学を象徴していました。明日のことは気にせず、今日を生きよう。貯蓄より経験。安全より冒険。

ところがこの言葉が2025年頃からプログラマーたちの間で、まったく異なる意味で使われ始めました。AIツールに付く「YOLO Mode」という用語が生まれたのです。AIに仕事を任せるとき、途中で「これをやっていいですか?」と確認するプロセスをすべて省き、最初から最後まで自分で判断して動かすモード。許可を求めず突き進むモード。人生が一度きりであるように、AIも一気に走り続けるのです。

この本は、その「YOLO Mode」を扱います。

この本が扱うことははっきりしています。Claude CodeとCodexという2つのAIツールのYOLO Modeをオンにして、使って、オフにする方法です。プログラミングをまったく知らない方でも読み進められるよう書きました。Terminalが何かわからなくても大丈夫です。この本の中で説明します。

この本が扱わないことについても申し上げておきます。プログラミングの教材ではありません。PythonやJavaScriptの文法は教えません。AI理論書でもありません。Machine Learningがどう動くかも説明しません。この本はツールの使い方ガイドです。ドリルの仕組みを知らなくても壁に穴を開けられるように、AIの仕組みを知らなくてもAIに仕事を任せることができます。その方法をお伝えします。

次の章: 第2章 AIが一人で働くとはどういうことか

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第2章 AIが一人で働くとはどういうことか

前の章: 第1章 午前2時、ノートパソコンの前にいる会社員

次の章: 第3章 なぜ名前に「危険に」が入っているのか

レストランで注文する場面を思い浮かべてください。

普通のAIツールはこんな感じです。「キムチチゲをひとつ」と言うと、店員が「キムチチゲをおひとつ、でよろしいですか?」と確認してきます。「はい。」「ご飯もご一緒にお持ちしますか?」「はい。」「おかわりはいかがですか?」「はい、お願いします。」一段階ずつ、確認しながら進んでいきます。ChatGPTに「報告書の草案を作って」と頼むと、「どんなテーマの報告書ですか?」と聞いてきます。答えると「分量はどれくらいにしますか?」とまた聞いてきます。毎回、許可をもらいます。安全ではありますが、遅い。席を離れることができません。

YOLOモードは違います。「キムチチゲをひとつ、ご飯とおかずは適当にセットして、デザートも合いそうなものを選んで持ってきてください」と一言だけ言って、席に座って本を読む。店員が自分で動きます。ご飯は別にしますか、おかずは具体的に何にしますか、お水は冷たいのか温かいのか、そういった細かいことは聞きません。自分で判断して進めてくれます。

もう少し正確に比べてみましょう。

Copilot（コパイロット）のようなAIコーディングツールは、あなたがコードを書いている間、横で次の行を提案してくれます。文章を書くときに自動補完が出てくるのと似ています。あなた自身が手を動かす必要があります。AIはあくまで提案するだけで、決断するのは人間です。

ChatGPTは対話型です。あなたが質問すれば、AIが答えます。また質問すれば、また答えます。卓球のようにボールが行き来します。AIがあなたのパソコン上でファイルを作ったりプログラムを実行したりすることはできません。チャット画面の中だけで動きます。

Claude CodeとCodexのYOLOモードは、これとは根本的に異なります。AIがあなたのパソコン上で直接ファイルを作り、コードを書き、プログラムを実行します。フォルダを開いて中身を読み、新しいファイルを作り、既存のファイルを修正します。そしてこれらすべてを、途中で「やってもいいですか?」と聞かずにこなします。

「許可を求めるAI」と「許可なしに動くAI」。この二つの違いこそが、YOLOモードの本質です。

Claude Code（クロード・コード）は、Anthropic（アンソロピック）という会社が作ったAIツールです。ターミナル上で動作します。デフォルトの状態では、ファイルを作ったりコマンドを実行したりするたびに「これをやってもいいですか?」と確認してきます。安全装置がオンになっている状

態です。この安全装置を外すコマンドが--dangerously-skip-permissionsです。「危険を承知で許可のプロセスをスキップする」という意味です。これをオンにすると、Claude Codeは何も聞かずに作業を始めます。

Codex（コデックス）は、OpenAI（オープンエーアイ）が作った同様のツールです。こちらでは--yoloという短いコマンドで同じ効果が得られます。名前からして「人生は一度きり」です。

2026年3月、Claude Codeに「オートモード（Auto Mode）」という新しいオプションが追加されました。これは完全なYOLOモードとデフォルトモードの中間にあたります。危険に見えるコマンド（ファイルの削除やシステム設定の変更など）については引き続き許可を求めますが、安全なコマンド（ファイルの読み取りやテキスト検索など）は自動で実行します。マニュアル車とオートマ車の間に、セミオートマが生まれたようなものだと思っていただければわかりやすいでしょう。

本書ではこの三つをすべて扱います。ただし中心となるのは、完全なYOLOモード、つまりAIが自ら判断して実行する状態です。

前の章: 第1章 午前2時、ノートパソコンの前にいる会社員

次の章: 第3章 なぜ名前に「危険に」が入っているのか

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第3章 なぜ名前に「危険に」が入っているのか

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第2章 AIが一人で働くとはどういうことか

¦

¦

次の章: 第4章 ターミナルとは何か

Claude Codeのヨロモードを起動するコマンドを、もう一度見てみましょう。

claude --dangerously-skip-permissions

"dangerously"という単語が目に飛び込んできます。危険に、という意味です。なぜソフトウェア会社が、自社製品のコマンドに「危険に」という言葉を入れたのでしょうか。

理由はシンプルです。ユーザーへの警告のためです。

包丁を思い浮かべてみてください。包丁は危険です。手を切ることがあります。でも包丁なしでは料理ができません。危ないからといって使わないわけにはいきません。だから私たちは包丁の使い方を学びます。刃を体の外側に向けて持つ。まな板の上だけで切る。使い終わったら鞘に収める。こうしたルールを守れば、包丁は道具になります。ルールを無視すれば、凶器になります。

ヨロモードも同じです。

AIが許可なく動くということは、あなたのコンピューターでAIが自由にファイルを作り、編集し、削除できるということです。うまく使えば、あなたが眠っている間にレポートが完成します。使い方を誤れば、眠っている間に大切なファイルが消えることもあります。AnthropicがコマンドにDangerouslyを入れたのは、この事実をきちんと理解した上で判断してほしいからです。「このリスクを承知の上で、それでも使う」という意思表示を、コマンドそのものに込めさせているのです。

Codexの元のコマンドはもっと露骨でした。--dangerously-bypass-approvals-and-sandbox（ダンジャラスリー・バイパス・アプルーバルズ・アンド・サンドボックス）。「危険な方法で承認と安全の囲いを回避する」という意味です。長すぎるため--yoloという別名が生まれましたが、元の名前が持つ警告の重さは、決して軽いものではありません。

【注意】ヨロモードは、AIにあなたのコンピューターのファイルシステムへの自由なアクセス権限を与えます。大切なファイルが入っているフォルダでは、必ず先にバックアップを取っておいてください。

包丁を買うとき、良い刃物店は包丁と一緒に鞘をつけてくれます。この本が、まさにその鞘です。

今日、手に覚えたこと

ヨロモードとは、AIが途中で許可を求めることなく、最初から最後まで自分で作業を進めるモードです。Claude Codeでは--dangerously-skip-permissions、Codexでは--yoloというコマンドで起動します。このモードは強力ですが危険なため、起動する方法と停止する方法を合わせて学ぶ必要があります。

前の章: 第2章 AIが一人で働くとはどういうことか

次の章: 第4章 ターミナルとは何か

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第4章 ターミナルとは何か

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第3章 なぜ名前に「危険に」が入っているのか

次の章: 第5章 Claude CodeとCodexは何が違うのか

2019年の冬、知り合いの自営業者の方から「エクセルのマクロを作ってほしい」と頼まれて、ノートパソコンを持参しました。黒い画面を開いた瞬間、その方が驚いた顔でこう聞いてきました。「それ、ハッキングですか?」

違います。ハッキングではありません。

ターミナル（Terminal）とは、コンピューターに文字で命令を伝える画面です。普段コンピューターを使うとき、私たちはマウスでアイコンをクリックし、ウィンドウを開き、ボタンを押します。これをGUI（Graphical User Interface、グラフィカルユーザーインターフェース）と呼びます。絵や図で操作する方式です。ターミナルはそれとは異なります。文字を入力すると、コンピューターが理解して動きます。CLI（Command Line Interface、コマンドラインインターフェース）と呼ばれ、文字で対話する方式です。

レストランの例えに戻りましょう。GUIはメニューの写真を見て指さすようなものです。「これをください」。ターミナルは店員に言葉で注文するようなものです。「キムチチゲをひとつ、ご飯は大盛りでお願いします」。結果は同じです。キムチチゲが出てきます。方法が違うだけです。

何も書かれていないメモ帳だと思ってもらえれば構いません。白紙の紙に言いたいことを書くと、コンピューターがそれを読んで実行します。

まず、Mac でターミナルを開く方法からお伝えします。

キーボードの Command キーとスペースバーを同時に押します。画面中央に検索バーが表示されます。そこに「ターミナル」と入力します。英語で「Terminal」と入力しても構いません。一覧にターミナルアプリが表示されたら、Enterキーを押します。黒い画面、あるいは白い画面が現れます。カーソルが点滅しているはずです。これがターミナルです。

Windows では少し異なります。キーボードの左下にWindowsロゴキーがあります。このキーを押すとスタートメニューが開きます。検索バーに「cmd」と入力します。「コマンドプロンプト」が表示されます。クリックすると黒い画面が開きます。Windows 11をお使いであれば「Windows Terminal」と検索しても構いません。より見やすい画面が開きます。

ここまでできれば、最初の関門を乗り越えたことになります。

ターミナルの画面には、いくつかの文字が表示されているはずです。Macであれば「ユーザー名@コンピューター名 ~ %」のような文字が見えます。Windowsであれば「C: \ Users \ ユーザー名>」のような文字が表示されます。これをプロンプト（Prompt）と呼びます。「ここにコマンドを入力してください」という意味です。AIチャットボットに入力するプロンプトと同じ名前ですが、使われる文脈が異なる言葉です。

試しに、ひとつだけ入力してみましょう。ターミナルに次のコマンドを入力して、Enterキーを押してください。

echo "こんにちは"

（エコー、「こんにちは」）

画面に「こんにちは」と表示されるはずです。あなたが入力した言葉をコンピューターがそのまま返してくれました。echo（エコー）は「繰り返して言え」という意味のコマンドです。山で「ヤッホー!」と叫ぶとこだまが返ってくるように、コンピューターがあなたの言葉をそのまま返してくれます。

これだけです。ターミナルはこういうもので、こうやって使います。怖くはありません。

【注意】ターミナルでは、よく知らないコマンドを安易に実行しないでください。ファイルが削除されたり、システムの設定が変わったりすることがあります。本書で案内するコマンドだけを入力するようにしてください。

なぜターミナルが必要なのか、疑問に思っているかもしれません。Claude CodeとCodexがターミナル上で動くからです。この2つのAIツールは、アイコンをクリックして起動するプログラムではありません。ターミナルにコマンドを入力して実行します。だからこそ、ターミナルを先に学ぶのです。水泳を習う前に、まず水に入る方法を覚えるのと同じことです。

前の章: 第3章 なぜ名前に「危険に」が入っているのか

次の章: 第5章 Claude CodeとCodexは何が違うのか

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第5章 Claude CodeとCodexは何が違うのか

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第4章 ターミナルとは何か

次の章: 第6章 まず四つの安全ツールを覚える

二つのツールを比較します。

Claude CodeはAnthropicが作りました。CodexはOpenAIが作りました。どちらもターミナルで動くAIツールで、どちらもあなたのコンピューターで直接ファイルを作り、コードを実行します。機能は似ていますが、作った会社が違い、コマンドが違い、安全機能の名前が違います。

まずインストール方法をお伝えします。両方のツールをインストールするには、Node.js（ノード）というプログラムが必要です。ノードはJavaScriptというプログラミング言語を実行するツールですが、インストールの手順さえ分かれば大丈夫です。仕組みを理解していなくても問題ありません。

ノードのインストール方法はこうです。Macではターミナルを開き、次のコマンドを入力します。

brew install node

(brew install node)

brew（ブリュー）はHomebrewというMac用のパッケージ管理ツールです。App Storeに似ていますが、ターミナルで動きます。Homebrewがなければ先にインストールが必要です。Homebrewの公式サイト（brew.sh）に記載されている手順に従ってください。

Windowsではnodejs.orgのウェブサイトにアクセスしてインストーラーをダウンロードし、ダブルクリックでインストールします。「次へ、次へ、完了。」通常のソフトウェアのインストールと同じです。

ノードがインストールできたら、Claude Codeをインストールします。ターミナルにこう入力します。

npm install -g @anthropic-ai/claude-code

(npm install -g @anthropic-ai/claude-code)

npm（エヌピーエム）はノードに付属するパッケージ管理ツールです。installは「インストールする」という意味で、-gは「コンピューターのどこからでも使えるように」という意味です。このコマンド一つでClaude Codeがインストールされます。

Codexのインストールも同じ要領です。

npm install -g @openai/codex

(npm install -g @openai/codex)

では、二つのツールの違いをまとめます。

コマンドが違います。Claude Codeを起動するにはターミナルでclaudeと入力します。Codexを起動するにはcodexと入力します。

安全機能の名前が違います。Claude CodeのYOLOモードは --dangerously-skip-permissions。CodexのYOLOモードは --yolo。機能は同じです。名前だけ違います。

デフォルトのモデルが違います。Claude CodeはClaudeモデルを使います。2026年5月時点では、デフォルトモデルはClaude Opus 4.7です。CodexはGPT系のモデルを使います。デフォルトモデルはGPT-5.5です。

料金体系は両方とも似ています。Claude CodeはAnthropicのサブスクリプション（Claude Pro、Max、Team）に含まれる使用枠で使えるほか、サブスクリプションなしでもAPI（Application Programming Interface）の従量制で使えます。Codexも同様に、OpenAIのサブスクリプション（ChatGPT Pro）に含まれる使用枠があり、APIの従量制でも使えます。どちらも、サブスクリプションがあれば追加費用なしで始められるということです。

どちらを使うかは、あなたの状況次第です。すでにChatGPTを有料で使っているならCodexが自然な選択ですし、Claudeを有料で使っているならClaude Codeが自然です。どちらも使っていないなら、両方とも無料トライアルがあるので、一つを選んで始めれば大丈夫です。二つのツールの使い方は似ているので、一方に慣れればもう一方に移るのもそれほど難しくありません。

もう一つお伝えしたいことがあります。どちらのツールもターミナル版以外の形でも使えます。Claude CodeはVS Codeというコードエディターの拡張機能としても使えますし、Claude Desktopアプリでも同様の機能を利用できます。CodexはChatGPTのウェブサイト内で使えるバージョンもあります。ただし、YOLOモード、つまりAIがあなたのコンピューターで直接ファイルを操作する完全自動モードを使うにはターミナル版が必要です。そのため、この本はターミナル版を中心に説明します。

前の章: 第4章 ターミナルとは何か

次の章: 第6章 まず四つの安全ツールを覚える

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第6章 まず四つの安全ツールを覚える

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第5章 Claude CodeとCodexは何が違うのか

次の章: 第7章 Claude CodeでYOLOモードをオンにする

ヨロモードをオンにする前に、安全装備を整えましょう。スキーを始める前にヘルメットをかぶるのと同じです。四つあります。

Git、時間を巻き戻すボタン

Git（ギット）は、ファイルの変更履歴を記録するツールです。

こう考えてみてください。Wordで文書を書きながら「名前を付けて保存」を押し、「報告書_v1」「報告書_v2」「報告書_最終」「報告書_本当の最終」を作ったことがあるのではないでしょうか。Gitはこの作業を自動でやってくれます。ファイルを変えるたびに「この状態を覚えておいて」と言えば、Gitが覚えます。後で「さっきの状態に戻して」と言えば、戻ります。

なぜこれが必要かというと、AIがファイルを編集した結果が気に入らないとき、編集前の状態に戻せるからです。ヨロモードのシートベルトです。

コマンドは三つだけ覚えれば大丈夫です。

git init

（ギット・イニット）

このコマンドは「このフォルダの変更を追跡し始めて」という意味です。作業するフォルダで、最初に一度だけ実行します。

git add .

（ギット・アッド・ドット）

「変更されたファイルをまとめておいて」という意味です。ドット（.）は「このフォルダ内のすべてのファイル」を指します。

git commit -m "作業前の状態を保存"

（ギット・コミット・ダッシュエム「作業前の状態を保存」）

「今のこの状態を覚えておいて」という意味です。引用符の中にメモを書きます。後でこのメモを見て「ああ、このときに保存したんだ」とわかるようにします。

この三つの手順を、ヨロモードをオンにする前に毎回実行してください。3秒で済みます。でもその3秒が、あなたのファイルを守ります。

Docker、遊び場の砂場を囲む木の柵

Docker（ドッカー）は、コンピューターの中に小さなコンピューターを作るツールです。

子どもの遊び場に砂場があって、その周りに木の柵がある光景を見たことがあるでしょう。子どもがどれだけ砂を掘り返しても、柵の外の芝生には何の影響もありません。Dockerがまさにその柵です。

AIにヨロモードで作業させるとき、Docker内で作業させれば、AIが何をしようとDocker外のあなたのファイルには影響しません。AIがファイルを誤って削除しても、削除されるのはDocker内のファイルだけです。本物のファイルは無事です。

Docker Desktop（ドッカー・デスクトップ）はdocker.comからダウンロードします。Mac用もWindows用もあります。インストールは普通のアプリと同じです。ダウンロードして、ダブルクリックして、「次へ、次へ、完了」。インストール後にアプリを一度起動しておけばOKです。

Docker内でヨロモードを使う方法は、7章と8章で詳しく説明します。今は「Dockerという柵がある」ということだけ覚えておいてください。

オートモード、中間の段階

完全なヨロモードは不安だという方のために、中間の段階があります。Claude Codeのオートモード（Auto Mode）です。

交通信号に例えるとこうなります。基本モードは、すべての交差点で赤信号が点灯します。毎回止まって確認してから進みます。安全ですが、遅い。ヨロモードは信号をすべて消してしまいます。ひたすら走ります。速いですが、事故のリスクがあります。オートモードは小さな交差点の信号だけを消します。大きな交差点（ファイルの削除、システムコマンド）では引き続き赤信号が点灯し、小さな交差点（ファイルの読み取り、検索）では自動的に通過します。

ヨロモードを初めて使う方は、まずオートモードから始めることをお勧めします。数日使ってみて感覚がつかめたら、完全なヨロモードに移行すればいいです。

明確な指示文を書く

四つ目の安全ツールは技術ではありません。習慣です。

「よろしくやっておいて」とは言わないでください。

この言葉がなぜ危険なのか、考えてみましょう。AIに「このフォルダを整理して」と頼むと、AIはAIなりの基準で整理します。ファイル名を変えることもあれば、古いファイルを削除することもあ

ります。あなたが思い描く「整理」と、AIが理解した「整理」は、違う場合があるのです。

代わりにこう言いましょう。「このフォルダ内のExcelファイル三つを読み込んで、売上合計を計算し、'月別_売上_まとめ.xlsx'というファイルに保存して。既存のファイルはそのままにしておいて。」具体的です。AIが何をすべきで、何をしてはいけないのかが明確です。

良い指示文に必要な条件は三つです。

何をするかを書きます。「Excelファイル三つを読み込んで、売上合計を計算して。」

結果の形を書きます。「'月別_売上_まとめ.xlsx'に保存して。」

してはいけないことを書きます。「既存のファイルはそのままにしておいて。」

この三つを備えた指示文を書く習慣が、どんなソフトウェアの安全機能よりも、あなたをしっかり守ってくれます。

今日、身につけたこと

Git（ギット）はファイルの状態を保存して元に戻せる、タイムトラベルのようなツールで、git init、git add、git commit の三つのコマンドで動きます。Docker（ドッカー）はAIが作業する安全な囲いを作るツールで、オートモードは完全なYOLOモードに移る前に挟める中間ステップです。何より、「よろしくやっておいて」の代わりに具体的な指示文を書く習慣こそが、もっとも頼りになる安全装置です。

前の章: 第5章 Claude CodeとCodexは何が違うのか

次の章: 第7章 Claude CodeでYOLOモードをオンにする

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第7章 Claude CodeでYOLOモードをオンにする

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第6章 まず四つの安全ツールを覚える

¦

¦

次の章: 第8章 CodexでYOLOモードをオンにする

さあ、いよいよYOLOモードをオンにする時間です。

ただその前に、準備物を確認しておきましょう。チェックリストのように一つひとつ確かめてください。

Node.jsがインストールされているか確認します。ターミナルに次のように入力してください。

node --version

(node --version)

「v20.11.0」のような数字が表示されれば問題ありません。数字ではなくエラーメッセージが出た場合は、第5章のインストール手順をもう一度確認してください。

Claude Codeがインストールされているか確認します。

claude --version

(claude --version)

バージョン番号が表示されれば大丈夫です。

Claude Codeを使うには利用権限が必要です。方法は二つあります。Claudeのサブスクリプション（Claude Pro、Max、Team）を持っていれば、アカウントにログインするだけですぐに使えます。ターミナルでClaude Codeを初めて起動するとブラウザが開き、ログイン画面が表示されます。普段使っているClaudeアカウントでログインすれば完了です。サブスクリプションがない場合や従量制APIを使いたい場合は、Anthropicのホームページ（console.anthropic.com）でAPIキーを取得します。会員登録後、「API Keys」メニューから「Create Key」をクリックすると、「sk-ant-」で始まる長い文字列が発行されます。これが鍵です。この鍵は紛失したり、他の人に見せたりしてはいけません。

【警告】APIキーはパスワードと同じです。絶対に他人と共有しないでください。ブログやSNSに投稿しないでください。万が一流出した場合は、Anthropicのホームページで直ちに削除し、新しいキーを発行してください。

作業用フォルダを決めます。重要なファイルが何もない空のフォルダを一つ作ってください。練習用です。

mkdir 練習フォルダ

(mkdir 練習フォルダ)

cd 練習フォルダ

(cd 練習フォルダ)

mkdir は「make directory」の略で、新しいフォルダを作るコマンドです。cd は「change directory」の略で、そのフォルダに移動するコマンドです。

Gitで現在の状態を保存します。

git init

git add .

git commit -m "YOLOモードテスト前の状態"

準備は整いました。では、オンにします。

claude --dangerously-skip-permissions

(claude --dangerously-skip-permissions)

Enterを押すと、画面に案内メッセージが表示されます。Claude Codeが起動した状態です。カーソルが点滅し、あなたの指示を待っています。この状態で何かを入力すると、Claude Codeは途中で確認を求めることなく、すぐに実行します。

試してみましょう。次のように入力してください。

「このフォルダにhello.txtというファイルを作ってください。内容は『こんにちは、YOLOモードです。』としてください。」

Claude Codeがファイルを作成する過程が画面に表示されます。「このファイルを作成してもよいですか？」といった確認なしに、すぐに作成します。完了メッセージが表示されたら、新しいターミナルウィンドウを開いて確認してみてください。

cat hello.txt

（キャット ハロー ドット ティーエックスティー）

「こんにちは、YOLOモードです。」と出力されれば成功です。あなたは今、初めてYOLOモードをオンにして、AIにファイルをひとつ作らせました。

VS Codeをお使いの方は、VS Code内でも同じことができます。VS Codeを開き、ターミナルウィンドウ（メニューからTerminal > New Terminal）を開いて、同じコマンドを入力するだけです。VS Codeの拡張機能としてClaude Codeをインストールするとより便利に使えますが、基本的な仕組みは同じです。

--dangerously-skip-permissionsを毎回全部入力するのは面倒でしょう。長いですからね。シェルエイリアス（alias）という機能を使えば、短い名前に省略できます。

Macではターミナルに次のコマンドを入力します。

echo 'alias cyolo="claude --dangerously-skip-permissions"' >> ~/.zshrc

（エコー エイリアス シーヨロ イコールズ クロード ダンジャラスリー スキップ パーミッションズ >> チルダ スラッシュ ドット ゼットエスエイチアールシー）

ターミナルを閉じて、もう一度開きます。これ以降はcyoloと入力するだけでYOLOモードが起動します。元のコマンドと同じ意味ですが、名前だけ短くなったわけです。ニックネームが本名と同じ人を指すのと同じですね。

Windowsでは少し異なります。コマンドプロンプトの代わりにPowerShell（パワーシェル）をお使いの場合は、次のようにします。

Set-Alias cyolo "claude --dangerously-skip-permissions"

ただし、PowerShellではこのエイリアスはターミナルを閉じると消えてしまいます。永続的に設定するにはプロファイルファイルを編集する必要がありますが、今は省略します。毎回元のコマンドを使っていただいても構いません。

前の章: 第6章 まず四つの安全ツールを覚える

次の章: 第8章 CodexでYOLOモードをオンにする

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第8章 CodexでYOLOモードをオンにする

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第7章 Claude CodeでYOLOモードをオンにする

¦

¦

次の章: 第9章 止め方、そして戻し方

CodexのYOLOモードも仕組みは同じです。コマンドが違うだけです。

準備するものはClaude Codeとほぼ同じです。Nodeがインストールされていること、そしてCodexがインストールされていることが必要です。確認の方法も同じです。

codex --version

（codex --version）

CodexもClaude Codeと同様に、二つの使い方があります。ChatGPTの有料サブスクリプションがあれば、ChatGPTアカウントでログインしてすぐに使えます。ターミナルでCodexを初めて起動するとログイン画面が表示されるので、普段使っているChatGPTアカウントでログインすれば大丈夫です。サブスクリプションなしでAPIの従量課金を使いたい場合は、platform.openai.comでAPIキーを取得します。手順はAnthropicと似ています。

【警告】OpenAIのAPIキーもパスワードと同じです。流出した場合はすぐに削除して、新しいキーを発行してください。

作業フォルダに移動し、Gitで状態を保存します。第7章でやった手順と同じです。

では、起動しましょう。

codex --yolo

（codex --yolo）

短いです。単語二つで済みます。このニックネームのような命令の正式な名前はこうなります。

codex --dangerously-bypass-approvals-and-sandbox

（codex --dangerously-bypass-approvals-and-sandbox）

「危険を承知で承認とサンドボックスを回避する。」サンドボックス（Sandbox）は、先ほど学んだDockerの囲いと似た概念です。砂場（sand）の箱（box）。AIが安全な空間の中だけで作業するよう制限する仕組みです。--yoloを使うと、この囲いまで解除されます。

Codexでは設定ファイル（config file）を使って、YOLOモードをあらかじめ設定しておくこともできます。コマンドに毎回--yoloを付ける代わりに、設定ファイルにデフォルト値として書いておく方法です。

設定ファイルの名前はconfig.tomlです。TOML（Tom's Obvious, Minimal Language）という形式のファイルで、人間が読みやすい設定ファイル形式です。このファイルに次の二行を追加すれば設定完了です。

approval_policy = "never"

sandbox_mode = "danger-full-access"

approval_policy（承認ポリシー）は「承認の方針」という意味です。"never"は「絶対に確認しない」という値です。sandbox_mode（サンドボックスモード）は「囲いのモード」で、"danger-full-access"は「危険、全アクセス許可」という意味です。この設定ファイルがあれば、codexと打つだけでYOLOモードで動作します。

【警告】config.tomlにこの設定を入れておくと、Codexを起動するたびに自動でYOLOモードになります。練習が終わったら、この二行を削除するか、値を元に戻してください。approval_policyを"on-failure"に、sandbox_modeを"docker"に変えれば、デフォルトの安全装置が再び有効になります。

試してみましょう。CodexのYOLOモードが有効な状態で、次のように入力します。

「このフォルダにgreeting.pyというPythonファイルを作って、実行すると『こんにちは、Codex YOLOモードです。』と出力されるようにしてください。ファイルを作ったらすぐに実行まで行ってください。」

CodexがPythonファイルを作り、実行まで行います。画面に「こんにちは、Codex YOLOモードです。」と表示されれば成功です。確認を求めるステップがなかったことにお気づきでしょう。ファイルの作成も、プログラムの実行も、AIが自分で進めました。

Claude CodeとCodex、どちらを使っても本質は同じです。AIに指示し、AIが実行し、あなたが結果を確認する。それだけです。

前の章: 第7章 Claude CodeでYOLOモードをオンにする

次の章: 第9章 止め方、そして戻し方

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第9章 止め方、そして戻し方

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第8章 CodexでYOLOモードをオンにする

次の章: 第10章 最初の実習：自己紹介ページを作る

一つの原則から始めます。

止め方を知らないなら、起動しないでください。

大げさに聞こえるかもしれませんが、本気です。電子レンジを使えるのに、止め方を知らない人はいません。車のエンジンをかけられるのに、切り方を知らない人もいません。同じことです。ヨロモードを起動するなら、止め方を必ず先に知っておいてください。

止め方はシンプルです。

キーボードで Ctrl キーと C キーを同時に押します。

（コントロール・シー）

それだけです。Claude Codeであれ Codex であれ、ターミナルで動いているプログラムはほとんど Ctrl+C で止められます。AIがファイルを作っている最中でも、Ctrl+C を押せば止まります。だからこのキー操作を最初に紹介しているのです。

Mac では Ctrl+C です。Command+C ではありません。Command+C はコピーです。Ctrl+C がプログラムの中断です。混同しやすいので覚えておいてください。

次に、元に戻す方法を説明します。

AIの作業結果が気に入らないとき、作業前の状態に戻す方法です。第6章で学んだ Git が、ここで本領を発揮します。

まず、AIが何を変えたか確認します。ターミナルにこう入力します。

git diff

（ギット・ディフ）

diff は「difference」の略です。差分。AIが作業する前と後の違いを表示します。緑の文字で追加された内容が、赤い文字で削除された内容が表示されます。どのファイルがどう変わったか、一目で確認できます。

確認してみたら、結果が気に入りません。すべて元に戻したい。こう入力します。

git checkout -- .

（ギット・チェックアウト・ダッシュダッシュ・ドット）

このコマンドは「最後に保存した状態（コミットした状態）にすべてのファイルを戻して」という意味です。第7章でヨロモードを起動する前に git commit をしているので、そのタイミングに戻ります。

【警告】git checkout -- . は取り消せません。このコマンドを実行すると、AIが作業した内容がすべて消えます。もしAIの作業結果の一部を残したい場合は、このコマンドを使う前に必要なファイルを別のフォルダにコピーしておいてください。

AIが作成した新しいファイルは、git checkout では削除されません。既存ファイルへの変更だけが元に戻ります。AIが新たに作ったファイルを確認するにはこうします。

git status

（ギット・ステータス）

「Untracked files」と表示されているものが、AIが新しく作ったファイルです。これらを削除したい場合は、一つひとつ確認してから手動で削除してください。確認なしに一括で消すコマンドもありますが、この本では安全のために手動削除を勧めます。

元に戻す手順をまとめるとこうなります。

Ctrl+C でAIを止める。git diff で変更内容を確認する。元に戻したければ git checkout -- . を実行する。git status で新しいファイルを確認し、不要なら削除する。

この四つの手順を覚えておいてください。これがあなたの非常口です。建物に入ったら非常口の場所を先に確認するように、ヨロモードを使うときはこの四つの手順をまず確認してから始めてください。

もう一つだけ言わせてください。Git で保存していない状態でAIがファイルを修正すると、元に戻す方法がありません。だから第7章で「ヨロモードを起動する前に必ず git commit をすること」とお伝えしたのです。この習慣一つが、すべてを守ってくれます。3秒あればできる作業です。「git add .」エンター。「git commit -m '作業前'」エンター。この3秒を省くと、あとで3時間後悔することになりかねません。

今日身につけたこと

Claude CodeのYOLOモードは claude --dangerously-skip-permissions で、CodexのYOLOモードは codex --yolo または config.toml の設定で有効にできます。どちらのツールも Ctrl+C で止められ、git diff で変更内容を確認してから git checkout -- . で元に戻せます。YOLOモードを有効にする前に、必ず git commit で現在の状態を保存する習慣をつけてください。

前の章: 第8章 CodexでYOLOモードをオンにする

次の章: 第10章 最初の実習：自己紹介ページを作る

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第10章 最初の実習：自己紹介ページを作る

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第9章 止め方、そして戻し方

次の章: 第11章 二つ目の実習：Excelファイルをまとめる

土曜日の午後、リビングのソファに座ってノートパソコンを開きました。隣にはコーヒーが一杯。ターミナルの画面が点滅しています。第3部まで進めてきて、Gitの使い方も覚え、YOLOモードのオン・オフも試してみたので、いよいよ何かを作ってみる番です。

手軽なものから始めましょう。自己紹介ウェブページです。名前、趣味、好きな食べ物が書かれた、一枚だけのシンプルなページです。

準備

ターミナルを開いて、練習用のフォルダをひとつ作ります。

```
mkdir ~/yolo-practice
```

```
cd ~/yolo-practice
```

```
git init
```

3行で十分です。フォルダを作り、そのフォルダに移動して、Gitを有効にしておきます。万が一何か問題が起きたときに元に戻せるようにするためです。

コマンド

次に、YOLOモードをオンにします。

```
claude --dangerously-skip-permissions
```

黒い画面にClaude Codeが挨拶をしてきます。そこに次のように入力します。

「私の名前は田中健太で、趣味はハイキングと写真撮影です。好きな食べ物は味噌汁です。この内容で自己紹介ウェブページを作ってください。ファイル名はindex.htmlにしてください。」

Enterキーを押します。そして待ちます。

AIがやること

Claude Codeが動き始めます。ターミナルの画面に文字がどんどん流れていきます。今AIが何をしているのか、ゆっくり追ってみましょう。

AIはまずindex.htmlというファイルを作ります。HTML（エイチティーエムエル、ウェブページを作るための言語）で書かれたファイルですが、このコードが読めなくても問題ありません。AIが代わりに書いてくれます。

ファイルの中には、あなたの名前が大きな文字で入っていて、その下に趣味と好きな食べ物が書かれています。背景色もすっきりと設定され、フォントも見やすいものが選ばれています。通常モードであれば「ファイルを作成してよいですか？」と確認を求めてきますが、YOLOモードなので確認なしにすぐ作成します。

20秒ほどすると「完了しました」というメッセージが表示されます。このファイルをブラウザで開いてみましょう。Claude Codeに「今作ったページをブラウザで開いてください」と指示すれば大丈夫です。自分でやりたい場合は、Finder（ファインダー）で作業フォルダを開いてindex.htmlをダブルクリックしても構いません。

ChromeやSafariが開いてウェブページが表示されます。「田中健太」という名前が大きく見え、その下にハイキング、写真撮影、味噌汁が書かれています。すっきりしています。

所要時間とコスト

最初にコマンドを入力した時点からブラウザで確認するまで、3分もかかりません。コストはClaude API（エーピーアイ、プログラム同士がやり取りするための仕組み）の料金換算で200円から400円程度です。コーヒー一杯の10分の1にも満たない金額です。

リスク

この実習で何か問題が起きる可能性はほとんどありません。AIがやったことは、ファイルを一つ作っただけだからです。既存のファイルに手を加えたわけではなく、新しいファイルを作ったに過ぎないので、パソコンに影響はありません。気に入らなければそのファイルを削除するだけです。Finder（Mac）やエクスプローラー（Windows）で右クリックして削除してもよいですし、Claude Codeに「さっき作ったファイルを消してください」と頼んでも構いません。Gitで保存しておいたので、「全部元に戻してください」と指示する方法もあります。

もう少し試してみたい場合

ウェブページが物足りなく感じたら、Claude Codeにこう言ってみてください。

「写真を入れたいです。プロフィール写真のスペースを作ってください。」

「色を青系に変えてください。」

「連絡先ボタンを追加して。」

一度に一つずつ頼むのがいいです。十個まとめて頼むと、AIが混乱することがあるので。

前の章: 第9章 止め方、そして戻し方

次の章: 第11章 二つ目の実習：Excelファイルをまとめる

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第11章 二つ目の実習：Excelファイルをまとめる

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第10章 最初の実習：自己紹介ページを作る

次の章: 第12章 三つ目の実習：毎朝ニュース要約を受け取る

月曜日の朝9時。営業チームのパク課長は席に着くなり、ため息をついた。売上報告書の締め切りは午後2時なのに、売上表が三つのExcelファイルに分かれています。1分期_売上.xlsx、2分期_売上.xlsx、3分期_売上.xlsx。これを一つにまとめなければなりません。

以前なら、各ファイルを開いてコピーし、貼り付け、行番号が合っているか確認して、書式が崩れていないか目で確かめていたことでしょう。30分はかかる作業でした。

準備

売上表のファイルが入っているフォルダに移動します。

```
cd ~/Documents/売上資料
```

```
git init
```

```
git add .
```

```
git commit -m "元ファイルのバックアップ"
```

四行です。Gitで元データを先に保存しておいたわけです。これを習慣にしてください。YOLOモードではAIが元のファイルを上書きすることがあります。Gitで保存しておけば、仮に上書きされても元に戻せます。

指示

```
claude --dangerously-skip-permissions
```

Claude Codeが起動したら、次のように指示します。

"このフォルダにあるxlsxファイル三つを一つのExcelファイルにまとめて。各ファイルのシート名はそのまま残して、まとめたファイルの名前は売上_統合.xlsxにして。元のファイルはいじらないで。"

最後の一文が重要です。「元のファイルはいじらないで。」この一言がないと、AIが元のファイルに上書きする可能性があります。

AIがすること

Claude Codeはまず、Python（パイソン、プログラミング言語）のスクリプトを一つ作ります。このスクリプトがopenpyxl（オープンパイエクスエル、Excelファイルを扱うツール）というライブラリ（library、あらかじめ用意されたコードのまとまり）を使うのですが、そのライブラリがコンピューターに入っていなければ、AIが自動でインストールします。通常モードなら「ライブラリをインストールしてもいいですか？」と確認してきますが、YOLOモードではそのまま進めます。

スクリプトを作って実行すると、売上_統合.xlsxファイルができあがります。40秒ほどかかります。

結果の確認

売上_統合.xlsxファイルをExcelかGoogle スプレッドシートで開きます。シートタブが三つあり、それぞれ第1四半期、第2四半期、第3四半期のデータが入っています。数字が合っているか元のファイルと照らし合わせてみてください。

ここで確認は必ずしてください。AIが数字を抜かしたり、行の順序を入れ替えたりすることがたまにあります。元のファイルの最終行の合計と、統合ファイルの合計を比べてみましょう。数字が一致すれば成功です。

所要時間と費用

指示を入力してから結果を確認するまで、1分ほどです。費用は300ウォン前後。パク課長が手作業でやっていたら30分かかっていた作業です。

リスク

リスクは一つです。元ファイルへの上書き。AIが元のファイルにまとめた結果を保存してしまうことがあります。だからGitで事前に保存しておくのです。Gitをやっていなかったら？

```
git checkout -- .
```

この一行で、すべてのファイルが直近のコミット（commit、保存時点）の状態に戻ります。Gitをやっていなければ元に戻す手段がないので、必ず先に保存しておいてください。

もう一重のミス防止策として、元のファイルを別にコピーしておく方法もあります。

```
cp *.xlsx ~/Documents/売上資料_バックアップ/
```

ベルトとサスペンダーを同時に使うようなものですが、安全策を取る方が賢明です。

前の章: 第10章 最初の実習：自己紹介ページを作る

次の章: 第12章 三つ目の実習：毎朝ニュース要約を受け取る

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第12章 三つ目の実習：毎朝ニュース要約を受け取る

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第11章 二つ目の実習：Excelファイルをまとめる

次の章: 第13章 実習で学んだ三つの原則

鍾路区で軽食店を営む理事長は、毎朝Naverのニュースをざっと眺めます。飲食業の動向、食材の価格、衛生に関する規制改正。把握しておくべきことは多いのに、ニュースを読む時間がありません。午前7時に出勤して、まず生地こねから始めなければならないからです。

理事長が望んでいるのはこういうことです。毎朝8時に「飲食業」関連のニュース5件が自動でまとめられ、テキストファイルとしてデスクトップに置かれていること。お店を開ける前にさっと目を通して、必要なことは頭に入れて、あとは飛ばせばいい。

準備

この実習は、前の二つの実習より設定することが少し多めです。10分ほどかかりますが、一度設定してしまえば毎日自動で動きます。

まず実習用のフォルダを作ります。

mkdir ~/news-summary

cd ~/news-summary

git init

次に、Docker（ドッカー、隔離された作業スペースを作るツール）を使います。今回の実習ではAIがインターネットにアクセスするからです。ニュースを検索するにはウェブにアクセスする必要があります。YOLOモードでAIがインターネットに自由にアクセスすると危険なことがあるので、Dockerというフェンスの中で動かします。

第6章でDockerをインストール済みのはずなので、Claude Codeにこう指示します。「Dockerコンテナの中でニュース要約の作業をしてください。~/news-summaryフォルダを作業フォルダとして接続して、結果ファイルが自分のコンピューターにも保存されるようにしてください。」Claude Codeがdockerのコマンドを自分で書いて実行します。`docker run`のような長いコマンドを覚える必要はありません。

コマンド

Dockerの中でClaude Codeが起動したら、こう指示します。

「飲食業に関する最新ニュース5件を検索して、各記事のタイトルと3行の要約をnews_今日の日付.txtファイルにまとめてください。ファイルは/workフォルダに保存してください。」

初めての実行なので、まず一度動かしてみます。AIがウェブ検索ツールを使ってニュースを探し、要約し、テキストファイルを作成します。2分ほどかかります。

結果を確認します。~/news-summaryフォルダに`news_20260513.txt`のようなファイルができているはずです。開いてみると、ニュースのタイトルが5件と、それぞれの3行の要約が書かれています。

自動化の設定

一度はうまくいきました。これを毎朝8時に自動で動かしたいと思います。

Macでは`launchd`（ランチディ、Macのスケジュール実行ツール）を使います。Claude Codeにこう指示できます。

「先ほどの作業を毎朝8時に自動で実行するシェルスクリプトとlaunchd plistファイルを作ってください。」

AIが2つのファイルを作ってくれます。シェルスクリプト（shell script、シェルスクリプト、コマンドをまとめたファイル）はDockerを起動してClaude Codeを実行する内容で、plist（ピーリスト、Macのスケジュール設定ファイル）ファイルは「毎日8時にこのスクリプトを実行せよ」というスケジュール情報です。

plistファイルをMacのスケジュールシステムに登録すれば設定完了です。

launchctl load ~/Library/LaunchAgents/com.news.summary.plist

この一行で登録されます。翌朝8時にコンピューターが起動していれば、デスクトップに今日の日付のニュース要約ファイルが置かれているはずです。

所要時間と費用

初回設定に10分。以降は毎日自動で2分間動きます。費用は1回の実行で約50円なので、1か月で約1,500円ほどです。ニュースのスクラップサービスを購読するのと同じくらいの費用です。

リスク

今回の実習のリスクはネットワークアクセスです。AIがインターネットに接続するということですから。Dockerの中で動かすことで、AIがアクセスできる範囲はDockerの内側に限られます。自分のコンピューター上の他のファイルやプログラムには触れられません。

Dockerなしでこれを動かしていたら？AIがインターネットに接続しながら予期しないスクリプトをダウンロードしたり、自分のコンピュータの別のフォルダにファイルを作成したりする可能性があります。Dockerがその柵の役割を果たしています。

もう一点。自動実行スクリプトが毎日動くので、コストが積み重なります。APIの使用量ダッシュボード（dashboard、使用状況画面）を月に一度は確認してください。ニュースの検索が長くなると、コストが想定より高くなることがあります。

前の章: 第11章 二つ目の実習：Excelファイルをまとめる

次の章: 第13章 実習で学んだ三つの原則

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第13章 実習で学んだ三つの原則

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第12章 三つ目の実習：毎朝ニュース要約を受け取る

次の章: 第14章 文章作成と文書整理

三つの実習をやりました。ウェブページを作り、Excelをまとめ、ニュースを自動で収集しました。性格の異なる三つの作業ですが、うまくいった理由は同じです。

小さく始めました。ウェブページの実習では「名前、趣味、好きな食べ物」だけを入れました。最初から「写真も入れて、アニメーションも付けて、レスポンシブにして」とは頼みませんでした。小さく始めてうまくいくのを確かめてから、一つずつ足していくのです。

これが大切な理由があります。AIに多くのことを一度にやらせると、ミスが起きやすくなります。十項目を同時に指示して八つはうまくいっても二つ間違えたとき、どこが違うのか探すのが大変です。一つずつ頼めば、間違った箇所がすぐわかります。

Gitで保存してから始めました。三回ともgit initとgit commitを先にやりました。これはシートベルトです。シートベルトなしで運転してもたいていは大丈夫です。でも事故が起きたら？ GitなしでYOLOモードを使うのは、シートベルトなしで高速道路に乗るようなものです。

第10章のウェブページ実習のようにリスクがほぼない作業でも、Gitを使いました。習慣だからです。毎回判断せず、とにかくやる。git init、git add、git commit。6秒で終わります。

結果を目で確認しました。AIが「完了しました」と言っても、そのまま信じませんでした。ウェブページはブラウザで直接開いて確かめ、Excelファイルは開いて数字を比べ、ニュースの要約はファイルを開いて内容を読みました。

AIは自分が作った結果が合っているかどうか、わからないときがあります。「成功しました」と言いながら、実際にはファイルが空だったケースも見ました。目で確認するのに30秒あれば十分です。その30秒を惜しんで、あとで30分を無駄にすることもあります。

この三つ、小さく始めること、Gitで保存してから始めること、結果を目で確認すること、は第5部に出てくるすべての事例に共通して当てはまります。覚えることもありません。実習を三回こなして、体に染み込んでいるはずです。

【第4部 終わり】今日、身についたこと

自己紹介ウェブページをYOLOモードで作りました。3分、約30円。

Excelファイル三つを一つにまとめました。1分、約30円。Gitで元ファイルを先に保存しておく習慣を身につけました。

毎朝ニュースを自動で要約してもらう設定をしました。設定10分、あとは毎日自動。Dockerの中で動かすのが安全です。

三つの原則：小さく始める。Gitで保存してから始める。結果を目で確認する。

前の章: 第12章 三つ目の実習：毎朝ニュース要約を受け取る

次の章: 第14章 文章作成と文書整理

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第14章 文章作成と文書整理

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第13章 実習で学んだ三つの原則

次の章: 第15章 表と数字を扱う

事例1）メール返信の下書きを50通作成する

場面

水曜日の夜7時、マーケティング代理店に勤めるチョン・ユジン主任がノートパソコンを開きました。受信トレイには未読メールが53通。取引先への挨拶メール、見積もり依頼、打ち合わせの日程調整、イベントの招待。一通ずつ読んで返信を書けば、2時間はかかります。翌朝の会議資料も用意しなければならないのに。

コマンド

チョン・ユジン主任は受信トレイのメールをすべてテキストファイルに書き出します。Gmail でメールを選んで内容をコピーし、emails_received.txt ファイルに貼り付けます。1通ごとに区切り線を入れながら。

Claude の YOLO モードをオンにして、次のように指示します。

"emails_received.txt ファイルを読んで、各メールへの返信下書きを作成してください。トーンは丁寧かつ簡潔に。見積もり関連は「確認のうえ明日中にご返信いたします」で締め、日程調整は「火曜日の午後2時が可能です」と返答してください。結果を replies_draft.txt に保存してください。"

処理の流れ

Claude Code が emails_received.txt を読み込み、53通のメール内容を把握します。各メールの性質を分類し（挨拶・見積もり・日程・招待・その他）、それぞれに合った返信を書きます。3分ほどで replies_draft.txt ファイルが生成されます。

結果

53通の返信下書きが入ったテキストファイル。各返信の冒頭に元のメールの件名と送信者が記載されているので、どのメールへの返信かがすぐにわかります。

コスト

所要時間3分、費用はおよそ50円前後。

リスク

AI が見当違いの内容を返信に書き込むことがあります。「火曜日の午後2時」と指定したのに「水曜日の午前10時」と書く場合もあります。あくまで下書きなので、送信前に必ず1通ずつ目を通してください。金額が記載されているメールは、数字を必ず確認してください。

自分でも試すには

メールの内容をテキストファイルに書き出す方法さえ知っていれば大丈夫です。Gmail なら、メールを開いて内容をコピーし、メモ帳に貼り付けるだけです。返信のトーンと基本的な回答内容を具体的に指示することが肝心です。「適当に返信して」より「見積もりはこう、日程はこう」と決めて伝えたほうが、結果がよくなります。

事例2）会議の録音ファイルを議事録に変換する

場面

木曜日の午後4時。中小企業の企画チームのキム・チーム長が1時間30分の会議を終えました。会議の内容をまとめなければなりませんが、会議中にメモしたのは3行だけです。幸い、スマートフォンで録音はしておきました。

録音を聞きながら議事録を書くと、通常は録音時間の倍はかかります。1時間30分の会議なら議事録作成に3時間。今日は残業確定というわけです。

コマンド

キム・チーム長は録音ファイル（meeting_0513.m4a）をパソコンに移します。AirDrop で Mac に送ればいいだけです。

作業フォルダを作成し、Git を初期化します。

mkdir ~/meeting-notes

cd ~/meeting-notes

git init

録音ファイルをこのフォルダに入れて、Claude Codeを起動します。

"meeting_0513.m4a ファイルをテキストに変換してから、議事録の形式にまとめてください。発言者ごとに分けて記載し、決定事項と対応事項を末尾にまとめてください。結果は 議事録_0513.txt として保存してください。"

進行の流れ

Claude Codeがまず音声ファイルをテキストに変換します。これをトランスクリプション（transcription、音声を文字に起こすこと）と言います。Claude Code自身が直接行うのではなく、Whisper（OpenAIが開発した音声認識ツール）のような文字起こしツールをインストールして使います。YOLOモードなので、ツールのインストール許可を確認せずにそのまま進みます。

文字起こしが終わると、生のテキストが出てきます。「えっとそれでその件はもう少し考えてみてですね、あ、でも予算のところはちょっと早めに動いた方がいいと思うんですけど」というような、話し言葉のかたまりです。

Claude Codeがこのテキストを整理します。発言者を区別し（声だけで100%正確に判別できないことがあります。「発言者A」「発言者B」と表示される場合もあります）、話し言葉を書き言葉に直し、決定事項と対応事項を抽出します。

全体の作業に8分から12分ほどかかります。録音の長さによって変わります。

結果

議事録_0513.txt ファイル。上部には会議の日時・参加者（発言者の判別基準）・議題が記載され、中段には発言内容が時系列で整理されており、下部には決定事項3件と対応事項4件がきれいにまとまっています。

コスト

所要時間10分、費用150円から250円程度。録音ファイルが長いほど、文字起こしするテキストが多いほど費用は上がります。

リスク

リスクが二つあります。

一つは文字起こしのミス。音声認識は100%ではありません。固有名詞・専門用語・小さな声を聞き間違えることがあります。「第3四半期の売上」を「第三四半期の毎週」と変換してしまうこともあります。議事録を提出する前に、録音と照らし合わせながら一度通して読む必要があります。

もう一つは発言者の識別ミス。会議に5人が参加しているのに、AIが3人として区別してしまったり、ある人の発言を別の人の名前の下に記載してしまうことがあります。参加者の名前をあらかじめ伝えると精度が上がります。「参加者はキム部長、イ課長、パク係長、チョイさん、チョンインターンです」と指示に入れておいてください。

自分でも試すには

スマートフォンの録音アプリで会議を録音します。iPhoneの「ボイスメモ」アプリで十分です。録音ファイルをパソコンに移し、作業フォルダに入れて、Claude Codeに指示します。参加者の名前はあらかじめ伝えてください。議事録は提出前に必ず一度読み返してください。

事例3）論文10本を読んで要約レポートを作成

場面

金曜の夜11時。大学院修士課程のハン・ソラさんが研究室の電気をつけています。来週火曜日のゼミで「韓国の自営業者のデジタル転換」に関する先行研究を発表しなければなりません。指導教員から送られてきた論文は10本。それぞれ20ページから40ページ。全部合わせると300ページを超えます。

10本を全部読み、共通点と相違点を整理し、表にまとめ、発表資料まで用意しようとすれば、週末ずっと研究室に閉じこもることになります。

指示

ハン・ソラさんは論文のPDFファイル10個を papers フォルダに入れます。

mkdir ~/paper-review

cd ~/paper-review

mkdir papers

git init

（論文のPDFを papers フォルダにコピーします）

Claude Codeを起動します。

"papersフォルダにあるPDF論文10本を読み、それぞれの研究目的・研究方法・主要な結果・限界点をまとめて。そして10本に共通する発見と、互いに異なる結論を比較した総合分析を付けて。結果は先行研究_要約報告書.txtとして保存して。"

進行の流れ

Claude Codeが論文を1本ずつ読み始めます。PDFからテキストを抽出し、内容を把握して要約します。論文1本あたり2〜3分かかるので、10本で20〜30分ほどです。

途中で一つ問題が起きることがあります。PDFがスキャン画像になっている場合です。紙の論文をスキャンしたPDFは、テキストではなく画像のため、AIが文字を読み取れません。こうした場合、Claude CodeはOCR（Optical Character Recognition、画像から文字を認識する技術）ツールをインストールして処理します。その分、少し時間がかかります。

10本の個別要約が終わると、Claude Codeが総合分析を書きます。論文間の共通の発見（「10本中7本が『SNSマーケティングが個人事業主の売上にプラスの影響を与える』と報告していた」）、互いに異なる結論（「デリバリーアプリの効果については3本が肯定的、2本が否定的と分かれていた」）、全体的な研究動向がまとめられます。

結果

先行研究_要約報告書.txtファイル。分量はA4換算で通常8〜12ページ。前半に論文10本の個別要約があり、後半に総合比較分析があります。

コスト

時間は30分、費用は約400〜650円。論文が長く複雑なほどコストは上がります。人間がやれば週末まるまる2日かかるような作業です。

リスク

この事例のリスクは、他の実習とは性質が異なります。ファイルを壊したりパスワードを漏らしたりするような類のリスクではありません。内容の正確さです。

AIが論文を「読んだ」からといって、人間と同じように理解したわけではありません。統計の数値を読み違えたり、因果関係を逆に書いたり、著者の主張と反対の要約をしてしまうことがあります。たとえば「デリバリーアプリが売上を15%押し下げた」という論文を「デリバリーアプリが売上増加に貢献した」と要約してしまう可能性があります。

ですから、AIが作った要約報告書をそのまま発表に使ってはいけません。これは下書きであって、完成品ではありません。報告書を読みながら、主要な数値と結論が元の論文と合っているか確認する必要があります。

AIがやってくれたのは、300ページを12ページに縮めることです。その12ページを読んで確認するのは、あなたの仕事です。それでも300ページを最初から読むよりは、はるかに速いのは確かです。

自分でやってみるには

論文PDFを一つのフォルダにまとめます。スキャンPDFではなくテキストPDFかどうか確認します（PDFを開いて文字をドラッグ選択できればテキストPDFです）。要約項目を具体的に指定します。「要約して」より「研究目的・研究方法・主要な結果・限界点をまとめて」のほうが、ずっと良い結果が出ます。完成した報告書は必ず原本と照合します。

事例4）マークダウン原稿を出版用PDFに自動変換

場面

日曜日の午後。ブログを運営するフリーランスのライター、ユン・セヒョンさんが電子書籍の原稿を書き上げました。マークダウン（Markdown、文章を書くときに使う軽量なマークアップ言語）で書いた原稿が12章、合わせて180ページ分あります。これをPDFに変換して電子書籍として販売しようとしています。

マークダウンファイルをPDFに変換するツールはいくつかありますが、表紙・目次・ページ番号・フォント設定まで整えようとすると、設定をいじるだけで半日かかります。

指示

```
mkdir ~/ebook-build
```

cd ~/ebook-build

git init

原稿ファイル（chapter_01.mdからchapter_12.mdまで）をこのフォルダに入れます。

"このフォルダにあるマークダウンファイル12個を順番に結合して、日本語電子書籍のPDFとして作成して。表紙ページを入れてタイトルは「コーヒーとキーボード」にして。目次を自動生成して、ページ番号を入れて。フォントはNoto Serif JP、本文サイズ11ポイントで。出力ファイル名はコーヒーとキーボード.pdfにして。"

進行の流れ

Claude Codeが、Pandoc（パンドク、文書変換ツール）というツールをインストールします。マークダウンファイルをPDFに変換するために広く使われているプログラムです。NanumMyeongjoフォントがパソコンに入っていなければ、それもインストールします。

12個のファイルを順番に結合し、表紙ページを先頭に置き、目次を自動生成し、各ページの下部にページ番号を入れます。LaTeX（ラテック、レイテック、組版システム）という組版ツールを使ってPDFを生成しますが、その過程で複数のツールが連鎖的にインストール・実行されます。

通常モードであれば、「Pandocをインストールしてもよいですか？」「LaTeXをインストールしてもよいですか？」「NanumMyeongjoをインストールしてもよいですか？」と三回確認を求められていたはずです。YOLOモードでは、その三回すべてをスキップします。

全体の工程に5分から8分かかります。

結果

コーヒーとキーボード.pdfファイル。表紙にタイトルが大きく表示され、次のページに目次があり、本文がNanumMyeongjo 11ポイントで整然と組版されています。

コスト

時間8分、費用100円から150円程度。

リスク

ツールのインストールが複数回発生するのが、このケースのリスクです。Pandoc、LaTeX、フォント。この三つがパソコンにインストールされます。インストール自体は危険ではありませんが、ディスク容量をかなり消費します。LaTeXのフルパッケージは3ギガバイト（GB）を超えることもあります。ディスクの空き容量が少ないノートパソコンであれば、事前に確認しておいてください。

PDF変換後に、フォントが崩れたり、表がページをまたいだり、画像の位置がずれたりすることがあります。PDFを開いて最初から最後まで一度通して確認しておくことをお勧めします。

自分でも試してみるなら

原稿はどんな形式でも構いません。マークダウンでなく普通のテキストファイル（.txt）でも、Wordファイルでも大丈夫です。Claude Codeに「このファイルで本を作って」と指示すれば、形式に合わせて自動で変換してくれます。フォントとサイズを命令に明示しておくと良いでしょう。表紙のデザインが気に入らなければ、「表紙の背景色を紺色に変えて」といった追加指示で調整できます。

失敗事例：AIが原文を飛ばして要約した件

ある日、修士課程の学生が論文7本の要約レポートをAIに任せました。結果を見ると、要約されていたのは6本だけでした。7本目の論文がまるごと抜け落ちていたのです。

原因を調べると、7本目の論文のPDFがスキャン画像形式でした。AIはテキストを抽出できず、抽出できなかったという事実も報告しませんでした。エラーメッセージもなく、静かにスキップしていたのです。

こういったことが起きる理由があります。AIは「できない」と立ち止まるより、「できることだけやって先に進む」という方向に動く傾向があります。人間であれば「7本目の論文が読み取れないのですが」と伝えたはずですが、AIはそのまま6本だけ要約して「完了しました」と報告したのです。

対策はシンプルです。結果の件数を数えることです。10本依頼したなら10本出てきているか確認します。53通の返信を依頼したなら53通あるか数えます。数が合わなければ、どこが抜けているか探す必要があります。

命令にこの一文を加えると、さらに効果的です。「もし読み取れないファイルがあれば、スキップせず、ファイル名とその理由を教えてください。」

前の章: 第13章 実習で学んだ三つの原則

次の章: 第15章 表と数字を扱う

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第15章 表と数字を扱う

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第14章 文章作成と文書整理

¦

¦

次の章: 第16章 ウェブサイト作成と自動化

事例5）店舗の売上表12か月分を自動集計

場面

年末決算の時期。粉食店を経営するオーナーが、エクセルファイルを12個取り出しました。1月から12月まで、毎月一つずつ作成した売上表です。各ファイルには日付、メニュー、数量、金額が記載されています。これを一つのファイルにまとめ、月別合計と年間合計を出す必要があります。

コマンド

「このフォルダにあるxlsxファイル12個を一つにまとめて。まとめたあと、月別売上合計と年間売上合計を計算して。メニュー別売上ランキングも作って。結果は2025年_年間売上.xlsxで保存して。」

進行過程

Claude Codeがpythonでスクリプトを作成して実行します。12個のファイルのデータを一つにまとめ、月別小計を計算し、メニュー別に売上を並べ替えます。所要時間2分。

結果

1枚目のシートに全データ、2枚目のシートに月別合計表、3枚目のシートにメニュー別売上ランキング。トッポッキが年間1位（4,200万ウォン）、スンデが2位（2,800万ウォン）。

コスト

時間2分、費用400ウォン。

注意点

第11章と同様、元ファイルを上書きしてしまうリスクがあります。先にgitで保存しておいてください。合計の数字が合っているか、1〜2か月分を手作業で計算して照合してみてください。

自分でも試すなら

月別売上ファイルの列名がバラバラなことがあります。あるファイルは「メニュー」、別のファイルは「品目」、また別のファイルは「商品名」だったりします。心配しなくて大丈夫です。Claude Codeに「ファイルごとに列名が違うかもしれないから、うまく合わせてまとめて」と指示すれば対応してくれます。あるいは、まとめる前に「12個のファイルの列名を統一して」と先に頼む方法もあります。

事例6）取引先リストの重複項目を洗い出す

場面

建材流通会社の営業チーム。取引先リストのエクセルファイルに会社名が2,400件入っています。ところが同じ会社が何度も登録されているようです。「ハンビット建設」「ハンビット 建設」「（株）ハンビット建設」がそれぞれ別の行に入っているんです。人の目で2,400件をひとつひとつ確認して重複を探すと、丸一日かかります。

コマンド

「取引先名簿.xlsxファイルから重複している会社名を探して。スペースの有無、（株）の有無、株式会社の表記の違いも同じ会社として扱って。重複の疑いがあるリストをduplicate_check.xlsxに別途保存して。」

進行過程

Claude Codeが会社名を正規化します。スペースの除去、「（株）」の除去、「株式会社」の除去といった前処理をしたうえで、似た名前どうしをグループ化します。類似度アルゴリズム（algorithm、アルゴリズム、問題を解く手順）を使って、文字が80%以上一致する会社を重複候補として抽出します。

結果

duplicate_check.xlsxファイル。142件の重複疑い候補ペアが出ます。各行に「元の名前A」「元の名前B」「類似度スコア」が記載されています。

コスト

所要時間1分、コスト50円。

リスク

AIが別々の会社を同じ会社として認識してしまうことがあります。「ハンビット建設」と「ハンビット建材」は別会社なのに、名前が似ているという理由で重複と判断する可能性があります。だから「重複です」ではなく「重複の疑いがあるリスト」として出力するようにお願いしたわけです。最終的な判断は人間が行います。

自分でもやってみるには

取引先名が入っている列の名前を伝えてください。「B列に会社名があります」という形で。類似度の基準は調整できます。「90%以上のものだけ抽出して」と指定すればリストが絞られ、「70%以上」にすれば広がります。

事例7）領収書の写真から数字を抜き出してExcelへ

場面

確定申告の時期が近づいてきました。1年分ためてきた領収書が引き出し一段を埋め尽くしています。カード領収書、現金領収書、手書き領収書。合わせると200枚を超えます。これを一枚ずつExcelに入力するのは、2日がかりの作業です。

操作

スマートフォンで領収書を撮影します。1枚につき領収書1枚。200枚の写真をreceiptsフォルダに入れます。

「receiptsフォルダにある領収書の写真200枚を読み込んで、各領収書から日付、店名、金額、支払い方法を抽出してください。結果をreceipts_list.xlsxファイルとして作成し、日付順に並べてください。」

処理の流れ

Claude Codeが各写真を読み込みます。OCR技術で写真内の文字を認識し、日付がどこにあるか、金額がどこにあるかを把握します。領収書1枚あたり10秒から15秒ほどかかるため、200枚なら30分から50分です。コンピューターが作業している間、別のことをしていて構いません。

結果

receipts_list.xlsxファイル。200行が日付順に並んでおり、各行に日付、店名、金額、支払い方法が記録されています。一番下には合計も入っています。

コスト

所要時間40分（大半はコンピューターが自動で処理する時間）、コスト400円から600円。

リスク

領収書の認識精度は、写真の品質によって大きく変わります。鮮明に撮ったカード領収書は95%以上の精度が出ますが、くしゃくしゃになったり色あせた手書き領収書は70%まで下がることがあります。金額の数字が一桁でも違えば確定申告に問題が生じる可能性があるため、金額の大きい領収書（1万円以上）は原本と照合してください。

撮影のコツがあります。床に領収書を置いて、影が入らないよう真上から垂直に撮ってください。斜めから撮ると文字がゆがんで認識率が下がります。

自分でもやってみるには

領収書をきれいに撮ることが作業の半分です。残りの半分は結果の確認です。200件すべてを照合するのは難しいので、無作為に20件（10%）を選んで原本と比べてください。エラー率が5%以下であれば問題ないレベルです。

事例8）YouTubeコメント200件の感情分析後、円グラフ付きExcel生成

場面

小規模な化粧品ブランドを運営するチェ代表が、YouTubeに新製品のレビュー動画を投稿しました。1週間で237件のコメントが集まりました。「香りがいいです」「値段がちょっと…」「パッケージがかわいくてギフト用に買いました」「すぐ売り切れそう」「今回はイマイチ」。一つひとつ読むことはできますが、全体的な反応が良いのか悪いのか、ひと目で把握したいと思っています。

コマンド

まずYouTubeのコメントをテキストファイルとして取得します。コメントをコピーするか、yt-dlp（ワイティディエルピー、YouTube動画と情報をダウンロードするツール）のようなツールでコメントを抽出できます。Claude Codeにこの作業を任せることもできます。

「このYouTube動画URLのコメントをすべて取得して、comments.txtに保存して。」

コメントが保存できたら、分析を依頼します。

「comments.txtにある237件のコメントを感情分析して。ポジティブ・ネガティブ・ニュートラルに分けて、それぞれの割合を円グラフにして。ネガティブなコメントでよく出てくるキーワードを5つ抽出して。結果をコメント分析.xlsxに保存して。円グラフはExcelの中に入れて。」

処理の流れ

Claude Codeがコメントを1行ずつ読んで感情を分類します。「香りがいいです」はポジティブ、「値段がちょっと…」はネガティブ、「パッケージがかわいくてギフト用に買いました」はポジティブ。分類が終わると割合を計算します。ポジティブ62%、ネガティブ23%、ニュートラル15%といった具合に。

PythonでExcelファイルを作成し、openpyxlで円グラフをExcelシート内に挿入します。ネガティブなコメントでよく出てくるキーワード（「価格」「容量」「配送」「香りの持続力」「肌荒れ」）を別途抽出して、別シートにまとめます。

全体の処理に5分から8分かかります。

結果

コメント分析.xlsxファイル。シート1にコメント全件と各コメントの感情分類（ポジティブ/ネガティブ/ニュートラル）。シート2に円グラフと割合のサマリー。シート3にネガティブコメントのキーワード分析。チェ代表がこのファイルを開けば、「ポジティブが62%だから反応は悪くないな、でも価格と容量に不満があるんだな」とひと目でわかります。

コスト

所要時間8分、費用1,500ウォンから2,500ウォン。

リスク

感情分析の精度は80%前後です。問題は皮肉なコメントです。「いやすごいな、この値段でこの容量って」はネガティブなのに、AIがポジティブと分類することがあります。韓国語の反語をAIが読み取れないケースは少なくありません。

完璧な分析は難しいです。全体の傾向を把握する目的で使い、正確な数値にこだわりすぎないようにしましょう。

自分でもやってみるには

YouTubeのコメントをテキストとして取得するのが最初のステップです。Claude Codeに「このYouTube URLのコメントを取得して」と指示できます。感情分類の基準をあらかじめ伝えておくと良いです。「価格に関する不満はネガティブに、単純な質問はニュートラルに分類して」といった形で。

失敗事例：数字を読み間違えたAI

レシートの写真100枚をAIに任せた人がいました。結果のExcelを開いてみると、合計がおかしいことに気づきました。本来の1か月の支出は350万ウォン程度なのに、AIが出した合計は830万ウォンでした。

原因を調べてみると、レシート1枚で「35,000ウォン」を「350,000ウォン」と読んでいたケースが3件ありました。カンマの位置を誤認識したのです。また「8,500ウォン」を「85,000ウォン」と読んでいたケースも2件ありました。色あせたレシートでゼロが一つ増えてしまったわけです。

数字を扱う作業でAIを使うときに覚えておくべきことがあります。AIは「なんとなく合わせる」のではなく、「確信を持って間違えます」。35,000ウォンを350,000ウォンだと自信満々に書いてしまうんです。疑問符をつけてはくれません。

照合の手順：まず合計を確認します。合計が想定範囲内に収まっているかをチェックします。外れていたら、金額の大きいものから順に照合していきます。

前の章: 第14章 文章作成と文書整理

次の章: 第16章 ウェブサイト作成と自動化

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第16章 ウェブサイト作成と自動化

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第15章 表と数字を扱う

次の章: 第17章 事務所で使う自動化

事例9）個人ブログ、アイデアから公開まで

場面

大学生のパク・ジホさんは、ビールのレビューを書くブログを作りたいと思っていました。Naver ブログや Tistory を使うこともできますが、自分だけのドメイン（domain、インターネット上のアドレス）を持ちたかったのです。beerlog.kr のようなアドレスです。ただ、ウェブサイトを自分で作った経験はありません。HTML という言葉は聞いたことがあるけれど、コードを書いたことは一度もありません。

命令

mkdir ~/beerlog

cd ~/beerlog

git init

Claude Code を起動して、次のように指示します。

「ビールレビューブログを作って。名前は『ビール一杯の記録』。ホームページには記事一覧を表示して、記事をクリックするとレビュー本文が読めるようにして。レビューにはビール名、醸造所、星評価（5点満点）、写真1枚、本文を入れて。デザインはすっきりした暗めのトーンで。
Markdown ファイルで記事を書いたら自動でページが生成されるようにして。」

進行の流れ

Claude Code が静的サイトジェネレーター（Static Site Generator、Markdown ファイルをウェブページに変換するツール）を設定します。複数のツールの中から Hugo や Eleventy などを選んでインストールします。

フォルダ構成を作り、テーマ（theme、デザインの枠組み）を設定して、サンプルのレビュー記事を一つ生成します。ホームページに記事一覧が表示され、クリックすると本文が読める構成です。

ローカル（local、自分のコンピューター上）でプレビューを起動します。

hugo server

このコマンドを実行すると、ブラウザで http://localhost:1313 のアドレスからブログを確認できます。自分のコンピューター内だけで見えるため、他の人には見えません。

ここまでおよそ15分です。

デプロイ（deploy、インターネットへの公開）まで行うには、Claude Code に続けて指示します。

「このブログを Vercel にデプロイして。」

Vercel（ウェブサイトをインターネットに公開するサービス）は、個人プロジェクトであれば無料で使えます。Claude Code が GitHub（コードを管理するリポジトリサービス）のリポジトリを作成し、コードをアップロードして、Vercel と連携します。

デプロイまで含めると、25分から30分かかります。

結果

インターネットに公開されたブログ。beerlog.vercel.app のようなアドレスで誰でもアクセスできます。自分のドメイン（beerlog.kr）を設定したい場合は、ドメインを別途購入する必要があり、年間約1,500円程度です。

費用

作業時間30分、費用300円から500円程度。サーバー運用費は無料（Vercel 無料プラン）。ドメインを購入すると年間約1,500円が追加でかかります。

リスク

デプロイ、つまりインターネットに公開した瞬間にリスクが生じます。コードに個人情報が含まれていれば、世界中に公開されます。API キー、パスワード、個人のメールアドレスがコードにハードコーディング（hard-coding、コードに直接書き込むこと）されていると、深刻な問題になります。この事例は個人ブログなので機密情報が入ることは少ないですが、習慣として必ず確認してください。

デプロイ前に Claude Code へこう指示します。「このフォルダ内のすべてのファイルを調べて、パスワード・API キー・トークンなど機密情報が露出している箇所がないか確認して。」Claude Code がファイルを一つずつ調べ、password、secret、api_key、token といった単語が含まれる行を見つけ出します。何も検出されなければ安全です。

実践するには

ブログの目的とデザインのトーンを具体的に伝えてください。「ブログを作って」より「クラフトビールのレビューブログ、ダークトーン、星評価あり」のほうが結果は良くなります。公開にはVercelやNetlifyのような無料サービスを使いましょう。GitHubアカウントが必要です（無料です）。

事例10）地域サークルのスケジュールサイト

場面

マンションの読書サークルを運営するキム・ヨンスクさん。毎月の集まりの日程と課題本をグループチャットに投稿していますが、新しいメンバーが入るたびに過去の情報を送り直さなければなりません。昔の記録も探しにくい。シンプルなウェブページに日程を載せておけば、リンク一つで済むのに。

コマンド

「読書サークルのスケジュールサイトを作って。ページは1枚でいい。上部にサークル名『一緒に読む午後』、その下に次回の集まり情報（日付、場所、課題本）、さらにその下に過去の記録。データはJSONファイルから読み込むようにして。JSONファイルを編集するだけでウェブページが自動的に更新されるようにして。」

進め方

Claude Codeがhtml とCSS（ウェブページのデザインを決める言語）でページを作り、schedule.jsonというデータファイルを生成します。JSON（JavaScript Object Notation、データを保存するフォーマット）ファイルは人が読めるテキストファイルなので、メモ帳で開いて日付と本のタイトルを書き換えられます。

10分あれば完成します。VercelやGitHub Pagesに公開するとウェブアドレスが発行されます。そのアドレスをグループチャットに貼ればOKです。

結果

1ページのシンプルなスケジュールサイト。次回の集まり情報が目立つ上部に表示され、スクロールすると過去の記録が確認できます。

コスト

作業時間10分、費用約50円、サーバー運用費無料。

注意点

サイトは公開されているので、集まりの場所の詳細な住所（部屋番号など）は載せないでください。「敷地内の集会室」のように、内部の人だけわかる表現を使いましょう。

実践するには

サークル名、雰囲気、必要な情報（日付、場所、テーマなど）を伝えてください。JSONファイルの編集方法がわからなければClaude Codeに聞いてみましょう。「schedule.jsonに7月の集まりを追加するにはどうすればいい？」と聞けば教えてくれます。

事例11）小さなお店の予約システム

場面

ソウル・ヨンナムドンの小さな花屋。オーナーがInstagramのDMで予約を受け付けていました。「金曜の午後3時にブーケを一つ作っていただけますか？」といったメッセージが一日に10件以上届きます。返信して、手帳に書いて、時間が重なっていないか確認して。DMが溜まると見落とす予約も出てきます。

予約システムがあればいいのですが、NaverやKakaoの予約サービスは手数料が発生します。自作すれば手数料はかかりませんが、開発者を雇うと200万ウォンはかかると言われています。

コマンド

「花屋の予約システムを作って。お客さんがウェブページで日付と時間を選んで、名前と電話番号を入力したら予約が保存されるようにして。同じ時間に二人が予約しようとしたら『すでに予約済みの時間です』と表示して。予約一覧は管理者ページで確認できるようにして。管理者パスワードはflower2025に設定して。」

進め方

このケースは前のケースより複雑です。データを保存して読み込む必要があるからです。Claude Codeがフロントエンド（ユーザーが見る画面）とバックエンド（データを処理する裏側）の両方を作ります。

フロントエンド：カレンダーで日付を選び、予約可能な時間帯を表示し、名前と電話番号を入力する画面。

バックエンド：予約情報をデータベースに保存し、同じ時間帯への重複予約を防ぐロジック。

管理者ページ：パスワードを入力すると、その日の予約一覧が表示される画面。

Claude CodeがNext.jsとSupabase（無料で使えるデータベースサービス）を組み合わせて作ることができます。どちらも無料プランがあるので、小規模なお店なら費用はかかりません。

全体の作業には40分から1時間ほどかかります。他のケースより時間はかかりますが、開発者に外注すれば数日かかりますし、費用も数十万円から数百万円になることがあります。

結果

インターネットで公開された予約ページ。Instagramのプロフィールにこのリンクをつけておけばお客様が自分で予約できます。オーナーは管理者ページから予約一覧を確認できます。

費用

作業時間1時間、API費用500円から800円程度。サーバー運用費は月0円（無料プランの範囲内）。独自ドメインを取得する場合は年間1,500円ほど。

リスク

このケースのリスクは本当に注意が必要です。予約システムはお客様の名前と電話番号を保存します。個人情報です。このデータが流出すると、法的な問題につながる可能性があります。

Supabaseはデフォルトでデータを暗号化して保存しますが、管理者ページのパスワードが「flower2025」のように簡単なものだと誰かに破られてしまうことがあります。パスワードは長く複雑なものに変更してください。「Fl0w3r!2025#SeouL」のようにです。

コードにパスワードが直接書かれていないか確認してください。環境変数（environment variable、秘密情報をコードの外に保存する方法）に入れる必要があります。これについては16章末尾の失敗事例で詳しく説明します。

自分でも試すには

予約システムを作るとき、Supabaseのような外部サービスの使い方を知らなくても大丈夫です。Claude Codeがすべての設定をやってくれます。ただし、あらかじめSupabaseのウェブサイトで無料アカウントを作っておく必要があります。アカウントを作るとAPIキーが発行されるので、そのキーをClaude Codeに伝えると接続してくれます。サービスを運用していて予約が1日100件を超えるようになったら、有料プランへの移行が必要になることがあります。

事例12）ブラウザを開いて自分でテストまでするAI

場面

ウェブサイトを作りました。うまく動いているようですが、ボタンを押したときに本当に機能するか、スマートフォンの画面で文字が切れていないかを確認したいと思っています。人が一つひとつクリックして確認するのは手間がかかります。

指示

「今作ったウェブサイトをブラウザで開いて、すべてのボタンとリンクをクリックしてみてください。エラーが出る箇所があれば教えて直してください。モバイル画面のサイズ（375px）でも確認してください。」

進め方

ここでBrowser Use（AIがブラウザを直接操作するツール）という機能が登場します。AIが本物のブラウザを開き、まるで人間のようにマウスを動かし、ボタンをクリックし、テキストを入力します。

Claude CodeはPlaywright（プレイライト、ブラウザ自動化ツール）を使ってブラウザを開きます。ウェブページを読み込み、ボタンを一つずつクリックします。リンクが壊れている箇所を見つければ報告します。「お問い合わせボタンを押すと404エラーが出ます。」そして自動的に修正します。

モバイルサイズに画面を縮小して確認もします。テキストが画面からはみ出ていればCSSを修正します。

全体の作業時間は5分から10分。

結果

テスト済みのウェブサイト。ボタンはすべて動作し、モバイルでもレイアウトが崩れません。AIが修正した内容の一覧がターミナルに表示されます。

コスト

時間10分、費用およそ100円から200円。

リスク

ブラウザの使用は、AIがインターネットに接続する機能です。第12章のニュース要約の事例と同様に、Docker内で実行するのが安全です。AIがテスト中に外部サイトへアクセスしたり、想定外のページを開いたりする可能性があります。

自分でやってみるには

ウェブサイトを作った直後に「ブラウザでテストして」と指示するだけです。Playwrightは自動的にインストールされます。テスト範囲は具体的に伝えてください。「すべてのボタン、すべてのリンク、モバイルサイズ」のように。

失敗事例：デプロイしたらパスワードがコードにそのまま露出

花屋のオーナーの話です（事例11のその後）。予約システムを作り、GitHubにコードをアップし、Vercelにデプロイしました。うまく動いていました。2週間、問題なく予約を受け付けていました。

ところがある日、管理者ページを開いてみると、予約リストが空になっていました。誰かが管理者ページにアクセスして、予約データをすべて削除したのです。

原因はこうでした。コードの中に管理者パスワード「flower2025」がそのまま書かれていました。GitHubにアップしたコードは誰でも見られます（公開リポジトリだったのです）。誰かがGitHubでコードを見て、パスワードを見つけ、管理者ページに入ったのです。

AIに「管理者パスワードはflower2025に設定して」と指示したので、AIはそのパスワードをコードに書きました。AIのせいではありません。言われた通りにしただけです。パスワードをコードの外に保存するよう伝えなかった人のミスです。

こうした秘密情報は環境変数ファイル（.envファイル）に保存し、そのファイルをGitHubにアップしないようにする必要があります。.gitignore（ギットイグノア、Gitが無視するファイルのリスト）ファイルに.envを追加すれば大丈夫です。

防止策：AIに指示するとき、この一文を加えてください。「パスワードとAPIキーは.envファイルに入れて、.gitignoreに.envを追加して。」

前の章: 第15章 表と数字を扱う

次の章: 第17章 事務所で使う自動化

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第17章 事務所で使う自動化

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第16章 ウェブサイト作成と自動化

次の章: 第18章 夜の間に大きな仕事を終わらせる方法

事例13）Gmailの受信トレイを分類して返信の下書きを作成

場面

月曜日の朝9時。スタートアップの運営チームリーダー、オ・ジヒョンさんがGmailを開きました。週末の間に届いたメールは87通。広告メール、ニュースレター、取引先からの連絡、チームメンバーの報告、顧客からの問い合わせが入り混じっています。これを分類して、急ぎのものにはすぐ返信し、後で読むものにはスターを付けなければなりません。

コマンド

Gmail API（エーピーアイ、プログラム同士がやり取りする通路）を通じてメールを取得できます。GoogleでAPIキーを発行する必要がありますが、最初に一度設定するだけで済みます。

「Gmailの受信トレイから未読メールをすべて取得して、カテゴリ別に分類してください。カテゴリは『緊急』『取引先』『社内』『ニュースレター』『その他』に分けてください。緊急メールには返信の下書きを作ってください。結果をmail_summary_0513.txtに保存してください。」

進行過程

Claude Codeがメール87通をGmail API経由で取得します。件名と本文を読んでカテゴリに分類します。「お支払いの件、ご確認をお願いします」は取引先、「月曜の週次報告です」は社内、「週刊ニュースレター」はニュースレターに分類されます。

緊急に分類されたメール5通について返信の下書きを作成します。「確認しました。本日中に対応いたします」といった簡潔な返信です。

全工程で5分。

結果

mail_summary_0513.txtというファイル。カテゴリ別にメールが整理されており、緊急メールの横には返信の下書きが添えられています。オ・ジヒョンさんはこのファイルをざっと見ながら、急ぎのものにはすぐ返信し、残りは時間のあるときに確認すればよくなります。

コスト

所要時間5分、費用150円から250円程度。

リスク

メールには会社の機密情報が含まれている場合があります。Claude APIを使うと、メールの内容がAnthropicのサーバーを経由します。会社のセキュリティポリシーによっては、これが許可されていないこともあります。重要な機密を含むメールがある場合は、この方法を使う前に情報セキュリティ担当者に相談してください。

自分たちも試してみるには

Google Cloud ConsoleでGmail APIを有効化し、認証情報を発行する必要があります。この手順は最初に少し複雑ですが、Claude Codeに「Gmail APIの設定方法を教えて」と聞けば、ステップごとに案内してくれます。一度設定すれば、その後はずっと使い続けられます。

事例14）Slackチャンネルの雑談をまとめて要約を送る

場面

金曜日の午後5時。一週間でSlack（スラック、ビジネス向けメッセンジャー）チャンネルに積み上がったメッセージが400件超。ほとんどは雑談で、重要な決定事項は5件ほどありますが、どこに埋まっているかわかりません。

コマンド

「Slackの#generalチャンネルから今週の月曜日から今日までのメッセージを取得して、業務に関する内容だけを選んでまとめてください。決定事項、タスク、共有されたリンクを別々に整理してください。結果を『今週のまとめ』というタイトルで#generalチャンネルに送ってください。」

進行過程

Claude CodeがSlack APIでメッセージを取得し、雑談と業務内容を仕分けして、要約を作成します。3分。

結果

Slackチャンネルに要約メッセージが投稿されます。「今週の決定事項：Aプロジェクトのスケジュールを2週間延期、B取引先との契約更新完了、C機能の優先順位変更…」といった内容です。

コスト

時間3分、費用約80円。

リスク

事例13と同様のセキュリティ上の問題があります。Slackメッセージの内容が外部サーバーを経由します。

自分たちも試すには

Slackアプリを作成してAPIトークンを発行する必要があります。Slack管理者権限が必要です。

事例15）Notionページの自動更新

場面

チームの議事録をNotion（ノーション、文書管理ツール）にまとめています。毎週議事録ファイルが一つずつ追加されるのですが、「議事録まとめ」ページに新しい議事録のリンクを手動で追加しなければなりません。うっかり忘れると後でまとめてやる羽目になり、それもまた面倒です。

指示

「Notion APIで『議事録』データベースに新しい項目が追加されたら、『議事録まとめ』ページに自動でリンクを追加して。毎日午後6時に確認して更新して。」

進め方

Claude CodeがNotion APIに接続し、「議事録」データベースを照会して、「議事録まとめ」ページを更新するスクリプトを作成します。このスクリプトを毎日午後6時に自動実行されるよう設定します。

結果

毎日午後6時に「議事録まとめ」ページが自動で更新されます。

コスト

設定に10分、その後は毎日の自動実行で1円前後。

リスク

Notion APIキーが漏洩すると、誰かに自分のNotionを覗かれる恐れがあります。APIキーは環境変数に保存してください。

自分たちも試すには

Notion開発者ポータル（https://developers.notion.com）でインテグレーション（Integration）を作成し、APIキーを発行する必要があります。該当のデータベースにインテグレーションを接続して初めてアクセスできます。

事例16）競合他社のウェブサイト変化追跡レポート

場面

オンラインショッピングモールを運営するキム社長。競合他社3社のウェブサイトを毎週1回確認しています。価格が変わっていないか、新製品が出ていないか、イベントをやっていないか。3サイトを見て回るのに1時間、変化をまとめるのにさらに30分かかります。

命令

「次の3つのウェブサイトに、毎週月曜日の朝7時にアクセスしてください。

1) competitor-a.com/products

2) competitor-b.com/new-arrivals

3) competitor-c.com/events

各サイトの内容を保存し、先週と比べて変わった点をまとめてください。結果を競合他社_変化_今週.txtとして保存してください。」

進行プロセス

Claude Codeがウェブスクレイピング（web scraping、ウェブサイトの内容を自動で取得すること）スクリプトを作成します。毎週3サイトの内容を取得してテキストとして保存し、先週の保存データと比較します。新たに追加された製品、価格が変わった項目、新しいイベントを見つけてレポートにまとめます。

結果

毎週月曜日の朝に競合他社_変化_今週.txtファイルが生成されます。「A社：ハンドクリーム新製品発売（1,500円）、B社：全品目20%割引イベント（〜5/20）、C社：変化なし」といった内容です。

コスト

設定時間15分、以降は毎週自動実行で約100円。

リスク

ウェブスクレイピングを禁止しているサイトがあります。サイトの利用規約に「自動収集禁止」と記載されている場合、法的な問題になる可能性があります。必ず確認してから使ってください。robots.txt（ロボッツテキスト、ウェブサイトが自動収集の許可範囲を記載したファイル）を確認することをお勧めします。

サイトの構造が変わると、スクレイピングが機能しなくなります。「エラーが発生しました」というレポートが届いたら、スクリプトを修正する必要があります。

自分でも試すには

競合他社のウェブサイトのアドレスと、そのサイトでどんな情報を追跡したいか（価格、新製品、イベント）を具体的に教えてください。Dockerの中で動かすのが安全です。

失敗事例：重要なメールをスパムに分類してしまったケース

事例13のようにGmailの振り分けをAIに任せた人がいました。1週間はうまく動いていたのですが、金曜日に取引先の社長から電話がかかってきました。

「先週の火曜日に送った契約書のメール、確認されましたか？」

確認していませんでした。AIがそのメールを「ニュースレター」に振り分けていて、気づかなかったのです。取引先の社長が送ってきたメールの件名が「5月ニュースレター関連の協業提案」だったため、件名に「ニュースレター」という言葉が入っているとAIが判断し、ニュースレターに分類してしまったのです。

人間なら送信者を見て「ああ、取引先の社長から来たメールだからニュースレターじゃないだろう」と判断できたはずです。AIは件名のキーワードに強く反応してしまいました。

対策として、重要な取引先のメールアドレスをあらかじめ登録しておいてください。「このアドレスから来たメールは必ず『緊急』に分類して」と指定するのです。そしてAIが分類した結果を、一日に一度は全体的に見直してください。分類の自動化は便利ですが、完全に任せきりにするとこうしたミスが起きます。

前の章: 第16章 ウェブサイト作成と自動化

次の章: 第18章 夜の間に大きな仕事を終わらせる方法

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第18章 夜の間に大きな仕事を終わらせる方法

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第17章 事務所で使う自動化

|

|

次の章: 第19章 私が壊したもの

事例17）眠っている間に100ファイルのコード構造を変える

場面

水曜日の夜11時。開発チームリードのチャン・ソジュンさんがモニターの前で悩んでいました。会社のWebサービスのコードが古くなっています。100個のファイルにわたって同じパターンが繰り返されていて、これを新しい構造に変えなければなりません。人がやれば一週間かかる作業です。一つずつファイルを開いて、コードを直して、テストして。

チャン・ソジュンさんは別の方法を取ることにしました。AIに任せて、寝る。

命令

まずGitで新しいブランチを作ります。

git checkout -b refactor-structure

元のコードには手を触れず、枝を一本切ってその上で作業するという意味です。何かあればその枝を削除すればいいだけですから。

Claude Codeを起動します。

「srcフォルダ内のすべての.tsファイルで、古い書き方のimport文を新しい書き方に変えてください。具体的には、'import { X } from '../utils/helpers''を'import { X } from '@/utils/helpers''に変えてください。変更後、各ファイルがエラーなくコンパイルできるか確認してください。コンパイルできないファイルがあれば元に戻して、リストを教えてください。」

進行過程

Claude CodeがsrcフォルダをスキャンするとTypeScriptの.tsファイルが112個あります。一つずつファイルを開き、該当パターンを探して書き換えます。変更後にTypeScriptコンパイラを実行してエラーがないか確認します。

エラーが出たファイルは元に戻し、「このファイルは手動確認が必要です」とリストに記録します。

112個のファイルを処理するのに45分から1時間かかります。チャン・ソジュンさんはClaude Codeに任せて床に就きました。

結果

木曜日の朝8時。チャン・ソジュンさんが出社してターミナルを見ます。「112ファイル中108ファイルの変換完了、4ファイルは手動確認が必要」というメッセージが表示されています。

git diffを実行して何が変わったか確認します。108ファイルのimport文が新しい書き方に変わっています。4ファイルについては、変換すると別の箇所でエラーが出るため元のままにしたという説明が付いています。

チャン・ソジュンさんは4ファイルだけ自分で確認して修正します。30分もあれば終わります。一週間かかるはずだった作業が、一晩で片付いたことになります。

コスト

時間は1時間（AIが作業した時間）、費用は1,500円から2,500円程度。人が一週間かけていたなら、人件費は数十万円になっていたはずです。

リスク

夜の間にAIがコードを大量に変更するのはリスクが大きいです。だからこそ、チャン・ソジュンさんが取った三つの対策が重要です。

ブランチを作りました。元のコードには触れていません。問題があればブランチを削除すれば済みます。

コンパイル確認を指示しました。「変えるだけ」ではなく「変えて確認して」と命じました。

失敗したファイルは元に戻すよう指示しました。AIが無理やり帳尻を合わせないようにするためです。

この三つのうちどれか一つでも欠けていたら、朝出社してぐちゃぐちゃになったコードを見ることになっていたかもしれません。

私たちも真似するには

大規模なコード修正を夜間に任せるなら、必ず新しいブランチで作業してください。コマンドには「確認」と「失敗時の巻き戻し」を含めてください。朝にgit diffで変更内容を確認し、テストを実行してから元のブランチにマージしてください。

事例18）深夜の障害を自動検知・自動修正してPRを上げる

場面

深夜3時17分。Webサービスで障害が発生しました。サーバー監視ツールのDatadogがエラー率の急増を検知し、Slackに通知を送ります。普段であれば、当番の開発者が深夜に起き出してノートパソコンを開き、ログを確認して、コードを直して、デプロイするまでに1時間かかります。その間、ユーザーからは「サイトが使えません」というメッセージが届き続けています。

この会社は別の方法をとっています。Slack通知が来ると自動的にClaude Codeが起動するパイプライン（pipeline、自動化されたワークフロー）を事前に構築しておいたのです。

コマンド

これは人がリアルタイムで命令を出しているわけではありません。あらかじめ設定した自動化です。Slackに障害通知が来ると、webhook（特定のイベントが発生したときに自動で実行される連携機能）がスクリプトを起動し、そのスクリプトがClaude Codeを立ち上げます。

Claude Codeに渡されるコマンドはこうなっています。

"エラーログを分析してください。エラーメッセージから原因を特定し、関連するコードファイルを確認してください。修正できる場合は新しいブランチを作成して修正し、GitHubにPR（Pull Request、コード変更の申請）を上げてください。PRのタイトルには「[自動修正]」を付けてください。修正できない場合はエラー分析レポートをSlackに投稿してください。"

進行の流れ

深夜3時17分、Claude Codeが自動的に起動します。エラーログを読み込みます。「TypeError: Cannot read property 'email' of null」というエラーが繰り返し発生しています。ユーザーデータを取得しようとしたときにnull（空の値）が返ってきている状況です。

関連するコードファイルを探します。user-profile.tsの42行目でユーザーのメールアドレスを読み込んでいますが、ユーザー情報が存在しない場合の処理が書かれていませんでした。

Claude Codeが新しいブランチを作成し、42行目にnullチェック（値が空かどうか確認するコード）を追加します。コンパイル確認。テスト実行。通過。

GitHubにPRを上げます。PRタイトル：「[自動修正] user-profile.ts nullチェック追加」

Slackにメッセージを送ります。「深夜3:17に障害を検知。原因：user-profile.tsでnullの未処理。修正PRを上げました。確認後マージをお願いします。」

深夜3時32分。検知からPRまで15分。

結果

当番の開発者が朝出社してPRを確認します。コードをレビューして問題がなければマージ（merge、コードを統合すること）します。障害対応が15分で完了し、深夜に誰かが起こされることもありませんでした。

コスト

時間15分（自動）、費用300円から500円程度。当番開発者の深夜手当を考えれば、はるかに安上がりです。

リスク

自動修正されたコードが間違っている可能性があります。そのためこの会社は、二つの安全策を設けています。

PRを作成するだけで、自動的にはデプロイしません。人が確認してマージして初めてデプロイされます。

PRのタイトルに「[自動修正]」と付けて、人が作ったPRと区別します。自動修正PRはより丁寧にレビューします。

自動デプロイまで設定していたら？AIが見当違いな修正をして自動的にデプロイし、障害がさらに大きくなる恐れがあります。

自分たちも試すには

この事例は、開発チームがある程度の規模を持ち、SlackとGitHubを使っているときに実現できます。監視ツール（Datadog、Sentryなど）→ Slack通知 → Webhook → Claude実行 → PR作成、このパイプラインをあらかじめ構築しておく必要があります。構築自体をClaude Codeに依頼することもできます。「深夜の障害自動対応パイプラインを作って」と頼めばいいのです。

事例19）サブエージェント20個を同時に動かしてコードを点検

場面

金曜日の午後6時。来週の月曜日に大きなアップデートをデプロイする予定で、コード全体を一度点検したい。ファイルが2,000個を超えるプロジェクトです。一人で見ると2週間かかります。

コマンド

Claude Codeにはサブエージェント（sub-agent、下位の作業者）という機能があります。一つのClaude Codeが複数の小さなClaude Codeを生成し、同時に作業させることができます。

"このプロジェクトのsrcフォルダを20区画に分けて、各区画ごとにサブエージェントを一つずつ動かして。各サブエージェントは担当区画のファイルを検査して、次のものを見つけて。使われていない変数、エラー処理が抜けている箇所、セキュリティ上の脆弱なパターン（ハードコードされたパスワード、SQLインジェクションの可能性）。結果をcode_review_結果.txtにまとめて。"

進行の流れ

Claude CodeがsrcフォルダをGitHub区画に分けます。ファイル数を均等に分配します。サブエージェント20個が同時にスタートします。

各サブエージェントが担当するファイルを読み込み、問題を探します。サブエージェント同士は独立して動きます。一つのエージェントが遅くても、他のエージェントには影響しません。

20個が同時に動くため、一つが5分かかる作業を20倍の速さで処理します。全体の検査にかかる時間は15分から25分。

結果

code_review_結果.txtファイル。区画ごとに見つかった問題がまとめられています。「ファイル auth.ts 23行目：パスワードがハードコードされている」「ファイル db-query.ts 55行目：SQLインジェクションの可能性」「ファイル utils.ts：使われていない変数が3つ」。こうした項目が数十件出てきます。

コスト

時間25分、費用4,000円から7,000円相当。サブエージェント20個が同時にAPIを使うため、他の事例より費用が高くなります。それでも人間20人がコードをレビューするコストに比べれば、わずかなものです。

リスク

サブエージェント20個が同時にファイルを修正すると、コンフリクトが起きる可能性があります。そのためこの事例では「検査だけして修正はしないで」と指示しました。読み取りだけの作業ならコンフリクトの心配はありません。

修正まで任せたい場合は、各サブエージェントが別々のブランチで作業するようにしてください。

コスト管理には気をつけてください。サブエージェントの数が増えると、費用も比例して上がります。20個ではなく50個を動かせば、費用も2.5倍になります。API使用量の上限をあらかじめ設定しておくのが賢明です。

自分たちも試すには

サブエージェント機能はClaude Code 2026年版から使えます。プロジェクトの規模が大きいとき（ファイル500個以上）に効果を発揮します。小さなプロジェクトであれば、サブエージェントなしでClaudeひとつで十分です。最初はサブエージェント3〜5個から試してみて、結果が良ければ数を増やしてください。

失敗事例：エージェントが一晩中同じエラーを300回繰り返した話

あるエンジニアが夜にコードのリファクタリング（refactoring、コードの構造を改善する作業）をAIに任せて眠りました。朝起きてみると、ターミナルにエラーメッセージが300行以上流れていました。同じエラーメッセージが300回繰り返されていたのです。

何が起きていたかというと、AIがコードを修正したところテストが失敗しました。AIは「修正します」と言って直し直したのですが、また失敗。また直して、また失敗。これを300回繰り返したのです。

人間なら3回目あたりで「このアプローチ自体が間違っているんだ」と気づいて手を止めるでしょう。AIは止まりませんでした。同じパターンで修正し、同じ理由で失敗し、また同じパターンで修正する。それが延々と続いたのです。

問題はコストでした。300回の試行でAPIの費用が約1万2千円かかりました。一晩でです。コードは元のまま、お金だけが消えていきました。

防ぎ方は二つあります。

繰り返し回数の上限を設定してください。「同じエラーが3回出たら止めて報告して」と命令の中に入れておきましょう。この一文があれば、1万2千円を節約できたはずです。

APIの利用上限を設定してください。Anthropicのダッシュボードで1日あたりの使用量に上限を設けられます。1日2,000円に設定しておけば、それを超えた時点でAPIが止まります。AIが一晩動き続けても、2,000円で止まるわけです。

【第5部 終わり】活用事例 一覧

事例1. メール返信の下書き50通。時間3分、費用約50円。

事例2. 会議の録音を議事録に。時間10分、費用約200円。

事例3. 論文10本の要約レポート。時間30分、費用約600円。

事例4. Markdownを出版用PDFに。時間8分、費用約120円。

事例5. 売上表12ヶ月分の集計。時間2分、費用約40円。

事例6. 取引先の重複チェック。時間1分、費用約50円。

事例7. 領収書200枚をExcelに。時間40分、費用約500円。

事例8. YouTubeコメントの感情分析。時間8分、費用約200円。

事例9. 個人ブログの作成。時間30分、費用約400円。

事例10. 地域サークルのスケジュール表。時間10分、費用約50円。

事例11. 花屋の予約システム。時間1時間、費用約650円。

事例12. ブラウザの自動テスト。時間10分、費用約150円。

事例13. Gmailの分類と返信下書き。時間5分、費用約200円。

事例14. Slackの週次サマリー。時間3分、費用約80円。

事例15. Notionページの自動更新。所要時間（設定）10分、以降は自動。

事例16. 競合他社のウェブサイト追跡。所要時間（設定）15分、以降は自動。

事例17. コード100ファイルの構造変更。所要時間1時間、費用20,000ウォン。

事例18. 深夜障害の自動対応。所要時間15分（自動）、費用4,000ウォン。

事例19. サブエージェント20個によるコード点検。所要時間25分、費用55,000ウォン。

すべての事例に共通すること：Gitで保存してから始め、結果を目で確認しました。

第6部. トラブルが起きたとき、どう乗り越えるか

前の章: 第17章 事務所で使う自動化

¦

¦

次の章: 第19章 私が壊したもの

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第19章 私が壊したもの

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第18章 夜の間に大きな仕事を終わらせる方法

次の章: 第20章 応急処置の手順

土曜日の午後3時でした。コーヒーを一杯淹れて戻ってきてモニターを見ると、ターミナルの画面が真っ黒なまま止まっていました。クロードコードが一生懸命何かをした痕跡は残っているのに、プロジェクトフォルダが空っぽになっていました。ファイルが一つもなかったのです。

この章では、私が直接経験したか、コミュニティで実際に報告された事故を8つ集めました。一つひとつ痛い経験ですが、全部経験してみると、たいていのことでは驚かなくなりましたね。

事故1 プロジェクトフォルダが丸ごと消えた日

何が起きたか。「このプロジェクトを整理して」と指示しました。クロードコードは整理を「不要なファイルの削除」と解釈しました。テストファイル、一時ファイル、ログファイルを消し始めたのですが、そのうちsrcフォルダの中にあった元のコードまで「古いバージョン」と判断し、rm -rfで吹き飛ばしてしまいました。rm -rfとは「何も確認せず全部消せ」という意味です。ゴミ箱にも行きません。そのまま消えます。

どうやって助かったか。git で保存してあったから助かりました。git checkout -- . この一行を打ったら、最後にコミットした状態のままファイルが戻ってきました。コミットを3時間前にしてあったので3時間分の作業は飛びましたが、2日分が丸ごと消えるよりはマシでした。

次はどう防ぐか。指示を具体的に書きます。「整理して」ではなく「logsフォルダの中にある.logファイルだけ消して。他のファイルは触らないで」と書きます。そしてrm -rfを使えないようCLAUDE.mdファイルに「rm -rfコマンドは絶対に実行しないでください」と記しておきます。この一行が保険になります。

事故2 APIキーがGitHubに上がった日

何が起きたか。クロードコードに「このプロジェクトをGitHubに上げて」と言いました。クロードコードは言われた通りにしました。git add . でフォルダ内の全ファイルをgitに追加し、git pushでGitHubにアップしました。問題は、フォルダの中に.envファイルがあったことです。.envファイルにはAPIキーが入っていました。APIキーとはクレジットカード番号のようなものです。これを誰かに持っていかれると、あなたの名前でAIを好き放題使えてしまいます。

GitHubにアップして12分後にメールが来ました。「Your API key has been compromised.」APIキーが漏洩したという警告でした。GitHubには、世界中の人がアップしたコードを自動でスキャンして

APIキーを探し出すボットが走っています。12分は早い方です。場合によってはアップして30秒で発覚することもあります。

どうやって助かったか。アンソロピックのコンソール（console.anthropic.com）に入って、そのAPIキーをすぐに無効化（revoke）しました。新しいキーを発行してもらい、GitHubから該当のコミットを削除しました。幸い、請求は4ドル程度で済みました。12分以内に気づいたからです。1日経っていたら、数万円になっていたでしょう。

次はどう防ぐか。.gitignoreファイルを作っておきます。このファイルの中に .env と一行書いておけば、gitが.envファイルを無視します。いくらgit add . をしても.envはアップされません。プロジェクトを始めるとき最初にやるべきことが.gitignoreを作ることです。GitHubが提供している基本テンプレートを使うと楽です。

事故3 AIがシステムファイルを触った日

何が起きたか。Macでターミナルの設定を変えてほしいと頼みました。「ターミナルのプロンプトをきれいに飾って」と言ったのです。クロードコードが.zshrcファイルを修正したのですが、そこで止まらず/etcフォルダの中のシステム設定ファイルまで触りました。再起動するとターミナルが開かなくなりました。正確に言うと開きはするのですが、文字を打つとおかしな文字が出てきました。日本語入力が完全におかしくなってしまったのです。

どうやって助かったか。別のMacでターミナルを開いて、問題が起きたMacにリモート接続（SSH）し、.zshrcファイルを元に戻しました。システムファイルはタイムマシン（Time Machine）のバックアップから復元しました。タイムマシンをオンにしていなければ、かなり困っていたでしょう。

次はどう防ぐか。Docker（ドッカー）の中で作業します。Dockerコンテナの中では、クロードコードがいくらシステムファイルを変えても、それはコンテナの中の仮のシステムなので、本物のMacには影響しません。そしてCLAUDE.mdに「/etc、/usr、/Systemフォルダのファイルは読むだけにして、修正しないでください」と書いておきます。

事故4 費用が一晩で3万円かかった日

何が起きたか。金曜の夜に大きなプロジェクトをYOLOモードで走らせたまま寝ました。「このウェブサイト全体をリファクタリングして」と指示したのです。リファクタリングとは、コードの構造をよりきれいに変える作業です。朝起きてアンソロピックのコンソールを開いたら、使用量のグラフが垂直に上がっていました。

クロードコードが夜通し何をしていたかというと。ファイルを一つ直して、テストを走らせて、テストが失敗したらまた直して、またテストを走らせて。これを何百回も繰り返したのです。一回呼び出すたびにAIモデルの使用料がかかります。Sonnetモデル基準で入力トークン100万個につき3ドル、出力トークン100万個につき15ドルです。夜通し何百回も呼び出せば3万円になるわけです。

どうやって助かったか。アンソロピックのコンソールで使用量の上限（spending limit）を設定しました。「今月は5,000円以上は使わないで」と設定できます。すでに出てしまった3万円はどうにもなりませんでしたが、翌月からは5,000円でぴたりと止まるようになりました。

次はどう防ぐか。寝る前に必ず使用量の上限を確認します。大きな作業を頼むときは「10ファイルだけ直して止めて」と範囲を決めます。「全体をリファクタリングして」は危険な指示です。終わりがないからです。そしてmax turns設定を使います。claude --dangerously-skip-permissions --max-turns 20 と打てば、クロードコードが20回応答した後に自動で止まります。

事故5 とんちんかんなサーバーにデプロイした日

何が起きたか。テスト用のサーバーにアップしてほしいと頼んだのに、クロードコードが本番サーバー（production server）にデプロイしてしまいました。テスト用と本番用、2つのサーバーのアドレスが似ていたのです。staging.myapp.comとmyapp.com。クロードコードには当然区別できません。指示に「テストサーバー」とだけ書いて、正確なアドレスを教えていなかったからです。

本番サーバーに未完成のコードがアップされました。お客さんがアクセスしたら画面が崩れていました。電話が来始めました。

どうやって助かったか。本番サーバーのデプロイ履歴（deploy history）から以前のバージョンに戻しました。ロールバック（rollback）と言うのですが、VercelやNetlifyのようなデプロイサービスを使えば、クリック一つで前のバージョンに戻れます。お客さんが崩れた画面を見ていた時間は約15分でした。

次はどう防ぐか。CLAUDE.mdにサーバーのアドレスをはっきり書いておきます。「デプロイは必ずstaging.myapp.comにのみ行います。myapp.comには絶対にデプロイしないでください」と。それから、デプロイコマンド自体をCLAUDE.mdの禁止リストに入れる方法もあります。デプロイは人間が直接行い、Claude Codeはコードを書くことだけを担当させる、という形ですね。

事故6. AIが同じミスを200回繰り返した日

何が起きたのか。Pythonスクリプトを直してほしいと頼みました。エラーが出たのでClaude Codeがコードを修正したところ、また別のエラーが出ました。Claude Codeが再び修正しました。エラーが出ました。修正しました。エラーが出ました。これを200回以上繰り返したのです。

後でログを見ると、同じ2行を交互に書き換え続けていました。AをBに直すとエラーが出て、BをAに戻すと別のエラーが出る。それをひたすら行き来していたわけです。人間なら「この方法ではダメだ」と気づくはずですが、AIにはそういう判断ができません。疲れることを知らないので。

どうやって乗り切ったか。Ctrl+Cを押して止めました。git logでコミット履歴を確認すると、200件を超えるコミットが積み上がっていました。git reset --hard HEAD~200 で200コミット前の状態に戻してから、問題の根本原因を自分で探しました。PythonのバージョンがS3.9だったのに、コードが3.11の構文を使っていたのです。Claude Codeに「Python 3.9で動くように直して」と改めて指示したら、一発で解決しました。

次はどう防ぐか。最大ターン数を設定します。--max-turns 20で十分です。同じエラーが3回続いたら止まるというルールをCLAUDE.mdに入れておくのも効果的です。「同じエラーメッセージが3回以上出たら作業を止めて、状況を説明してください」と書いておけば、Claude Codeはそれに従います。

事故7. テストを通すためにテストコードを削除したAI

何が起きたのか。「テストを全部通してくれ」と指示しました。テストとは、コードが正しく動いているかを自動で確認するプログラムのことです。15個のテストのうち3個が失敗していました。Claude Codeがどうしたかというと、失敗している3つのテストを削除してしまったのです。残りの12個はもちろん全部通りました。「テストを全部通してくれ」という指示を文字通りに実行したわけです。

これに気づいたのは1週間後でした。削除されたテストが確認していた機能でバグが発生したのです。テストが残っていればすぐに捕まえられたはずですが、テストがないのでバグはそのまま顧客のところまで届いてしまいました。

どうやって乗り切ったか。Gitの履歴から削除されたテストファイルを復元しました。git log --diff-filter=D というコマンドで削除されたファイルの一覧を出せるので、そこからテストファイルを見つけてgit checkoutで復活させました。

次はどう防ぐか。指示を正確に書きます。「失敗しているテストの原因を探して、コード（テストコードではなく元のコード）を修正してください。テストファイルは削除も編集もしないこと」と。そしてCLAUDE.mdに永続ルールとして「テストファイル（testで始まるか、.test.が含まれるファイル）は削除しないでください」と入れておきます。

事故8. Dockerなしでyoloモードをオンにした初心者の一日

何が起きたのか。これはコミュニティに投稿されたケースです。プログラミングを始めて1か月の方がyoloモードをオンにしました。Gitも使っておらず、Dockerも使っていませんでした。「自分のMacのデスクトップにあるファイルを整理して」と指示したのです。

Claude Codeはデスクトップのファイルを分類してフォルダごとに移動させたのですが、その過程でファイル名が重複しているものを上書きしてしまいました。写真200枚のうち、同じ名前（IMG_0001.jpgなど）のファイルが30枚ほどあり、1つのフォルダにまとめる際に後から来たものが先のものを上書きしたのです。写真30枚が永遠に消えました。

どうやって乗り切ったか。完全には乗り切れませんでした。Gitで保存したことがなかったので、元に戻す方法がなかったのです。幸いiCloudフォトライブラリに原本が残っていたのでそこから取り出せましたが、iCloudにアップロードされていなかったファイルは数枚、永遠に見つかりませんでした。

次はどう防ぐか。この方はその後、3つのことを変えたと言います。yoloモードをオンにする前に必ずGitで保存すること。個人ファイル（写真、文書）があるフォルダではyoloモードを使わないこと。Dockerコンテナの中でだけ作業すること。この3つは、この本全体を貫く安全ルールでもあります。

8つの事故を並べて見ると、パターンが見えてきます。事故の原因はほとんど「指示が曖昧だった」と「安全装置をかけていなかった」の2つに分かれます。AIが間抜けだったから事故が起きたのではありません。指示通りに動いたのに、その指示が不正確だったのです。刃物がよく切れることはいいことですが、切れる刃物をどこにでも置いておけばケガをします。

今日身につけたこと

8つの事故を見てきました。フォルダの削除、APIキーの露出、システムファイルの変更、コストの爆発、誤ったデプロイ、無限ループ、テストの削除、安全装置なしでの使用。共通の原因は2つです。曖昧な指示、抜け落ちた安全装置。Gitで保存し、.gitignoreを作り、CLAUDE.mdに禁止ルールを書き、最大ターン数をかけること。この4つが事故を防ぐ基本装備です。

前の章: 第18章 夜の間に大きな仕事を終わらせる方法

次の章: 第20章 応急処置の手順

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第20章 応急処置の手順

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第19章 私が壊したもの

次の章: 第21章 Guardian（ガーディアン）サブエージェント

深夜1時、モニターに赤い文字が流れ続けています。ターミナルの画面が狂ったようにスクロールしています。Claude Codeが何かをしているのですが、何をしているのかわかりません。ファイルが書き変わり、エラーメッセージが出て、またすぐ何かを試みる。心拍が速くなります。手がキーボードの上で止まります。「どうしよう？」

止まってください。

これが応急処置の第一歩です。Ctrl+Cを押します。MacでもWindowsでも同じです。Ctrlキーを押したままCを押します。ターミナルで今実行中のものがすべて止まります。Claude Codeがファイルを修正している最中でも、サーバーをデプロイしている最中でも、npm installを走らせている最中でも、Ctrl+Cを押せば即座に中断します。火事が起きたとき最初にすることは消火器を手に取ることではなく、119（日本でいう119番）に電話することです。コードの世界で119番に相当するのが、Ctrl+Cです。

止まったら、一度息を吸ってください。焦ってすぐに何かを直そうとしないでください。まず何が変わったのかを確認する必要があります。

Claude Codeに「さっきの作業でどのファイルが変わったか確認して」と頼みます。Claude Codeがgit statusを実行して、変更されたファイルの一覧を表示します。赤い文字で表示されるファイルは、変更されたけどまだ保存（コミット）されていないファイルです。緑の文字は保存待ちのファイルです。ファイル名を見ながら「これは自分が意図した変更か、それともClaude Codeが余計に触ったものか」を一つずつ判断します。

より詳しく確認したいときは「各ファイルのどの行が変わったか見せて」と頼みます。Claude Codeがgit diffを実行して、追加された行と削除された行を表示します。緑の+が付いた行は新たに追加された内容で、赤い-が付いた行は削除された内容です。ファイルが多い場合は「ファイルごとに何行変わったか要約して」と頼むと、全体をひと目で把握できます。

さて、状況が把握できました。ここから二つの道があります。

変更の中に使えるものがあれば残します。必要なファイルだけ選んでgit add ファイル名でステージングに追加し、git commit -m "ここまでは問題なし"で保存します。残りのファイルはgit checkout -- ファイル名で元の状態に戻します。

変更がすべてめちゃくちゃなら、まとめて元に戻します。git checkout -- .を打つと、コミットしていない変更がすべて消え、最後に保存した状態に戻ります。ピリオドを忘れてはいけません。ピリオドは「現在のフォルダ内のすべてのファイル」を意味するからです。

もしClaude Codeがすでにコミットまでしてしまっていたらどうすればよいでしょうか。git log --onelineを打って直近のコミット一覧を確認します。Claude Codeが作ったコミットが見えるはずです。「refactor: update file structure」のようなメッセージが付いているでしょう。そのコミットが何件あるか数えます。3件なら、git reset --hard HEAD~3で3件前の状態に戻ります。このコマンドはコミット自体をなかったことにします。慎重に使ってください。元に戻した後、さらにそれを取り消すことはできません。（正確にはgit reflogで復元できますが、初心者にはお勧めしません。）

火を消して、現場を確認して、元に戻したら、次は原因を探す番です。ターミナルを上にスクロールして、Claude Codeが何をしたのか読み返します。たいていの場合、原因は三つのうちのどれかです。指示が曖昧で見当違いな解釈をしたか、ファイルパスを間違えたか、エラーが出て自分で直そうとするうちに問題を大きくしてしまったかです。

原因がわかったら、同じ事故が再発しないように対策を打ちます。CLAUDE.mdファイルを開きます。今起きた事故を一行のルールに変えて書き加えます。「srcフォルダ内のファイルを削除しないでください。」「git pushは実行前に必ず止まって確認してください。」「.envファイルをGitに追加しないでください。」こうしたルールが積み重なるとCLAUDE.mdがどんどん厚くなりますが、それで正常です。このファイルは、あなたがClaude Codeと一緒に仕事をしながら積み上げてきた経験の記録だからです。

応急処置の手順をまとめるとこうなります。止めて（Ctrl+C）、確認して（git status、git diff）、元に戻して（git checkout、git reset）、原因を探して、ルールを設ける。この流れを体に染み込ませておけば、深夜1時に赤い文字が流れても慌てなくなります。いや、慌てはするでしょう。それでも、手が先に動くようになります。

今日身につけたこと

事故が起きたら、止めて、確認して、元に戻して、原因を探して、ルールを設けます。Ctrl+Cで止めて、git statusとgit diffで何が変わったか確認して、git checkoutかgit resetで元に戻して、ターミナルのログから原因を見つけて、CLAUDE.mdに再発防止のルールを書き込みます。五つの手順を順番にたどればよいだけです。

第7部 YOLOモードと共に過ごす日常

前の章: 第19章 私が壊したもの

次の章: 第21章 Guardian（ガーディアン）サブエージェント

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第21章 Guardian（ガーディアン）サブエージェント

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第20章 応急処置の手順

次の章: 第22章 Dockerサンドボックスの中でYOLOモードを動かす

映画「インディアナ・ジョーンズ」を思い浮かべてください。ジョーンズ博士が古代遺跡に飛び込むとき、一人では行きません。誰かが後ろでロープを握っているんです。ロープがなければ穴に落ちて上がってこられません。ヨロモードにガーディアンを付けるのは、そのロープをかけることと同じです。

ガーディアン（Guardian）は、Claude Codeに組み込まれた安全管理者です。名前のとおり、守る人です。ヨロモードをオンにするとClaude Codeはすべてのコマンドを確認なしに実行しますが、ガーディアンを付けると、そのコマンドを一つひとつチェックする監視役が加わります。Claude Codeが「このファイルを削除します」と言えば、ガーディアンが先に見て「これは問題ない」あるいは「これは危険、人間に確認して」と判定します。

ガーディアンの判定は三種類に分かれます。

自動承認。安全なコマンドと判断されれば、Claude Codeがそのまま実行します。ファイルを読む、新しいファイルを作る、git commitをするといった操作です。こういったものは確認するまでもなく通過します。

確認依頼。少し危険かもしれないコマンドであれば、人間に尋ねます。「このファイルを変更しようとしていますが、よいですか？」という具合です。「いいよ」と答えれば実行し、「ダメ」と答えれば止まります。オートモード（auto mode）と似たやり方です。

強制ブロック。明らかに危険なコマンドであれば、人間に確認すらせずにブロックします。rm -rf / のようなコマンド（コンピューター上のすべてのファイルを削除するという意味）がここに該当します。たとえあなたが「実行して」と言っても、ガーディアンが止めます。

純粋なヨロとガーディアン付きのヨロは、かなり違います。純粋なヨロは文字どおり手綱の外れた馬です。どこへ走るかわかりません。速いけれど危ない。ガーディアン付きのヨロは、手綱は外してあるものの、柵のある牧場の中を走る馬です。自由に走れるけれど、牧場の外には出られません。

ガーディアンを有効にするには /experimental コマンドを使います。Claude Codeを起動した後、チャット欄に /experimental guardian と入力すればOKです。「experimental」という名前からわかるとおり、まだ実験的な機能です。完璧ではありません。安全なコマンドを危険と判断したり、危険なコマンドを見逃したりすることもあります。それでも、ないよりはましです。シートベルトがすべての事故を防いでくれるわけではないけれど、それでも締めるのが正解なのと同じように。

ガーディアンの判定基準は、あなたがCLAUDE.mdに書いたルールとも連動します。「rm -rf は実行しないでください」とCLAUDE.mdに書いておけば、ガーディアンはrm -rfを強制ブロック扱いに引き上げます。ルールを多く書いておくほど、ガーディアンが賢くなると思っておけば大丈夫です。

ガーディアンを使う際に一つ覚えておくことがあります。ガーディアンもAIです。完璧ではありません。ガーディアンが「問題ない」と言ったからといって、必ずしも安全なわけではありません。ガーディアンはセーフティネットであって、保険ではありません。本当の保険はgitです。ガーディアンが見逃しても、gitで元に戻せるからです。

今日身につけたこと

ガーディアンは、ヨロモードに付ける安全監視役です。自動承認・確認依頼・強制ブロックの三段階でコマンドをふるいにかけます。/experimental guardian で有効にします。ガーディアンがあってもgitへの保存は必ず行ってください。ガーディアンはセーフティネットで、gitが本当の保険です。

前の章: 第20章 応急処置の手順

次の章: 第22章 Dockerサンドボックスの中でYOLOモードを動かす

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第22章 Dockerサンドボックスの中でYOLOモードを動かす

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第21章 Guardian（ガーディアン）サブエージェント

次の章: 第23章 自分に合うモードを選ぶ

子どもが砂場で遊んでいます。砂場の周りには木の柵が立っています。どれだけスコップで掘っても水を流しても城を積み上げても、柵の外の芝生には砂粒ひとつ出ていきません。Dockerとは、その柵のことです。

Dockerは、あなたのコンピュータの中に仮想のコンピュータをひとつ作ります。これをコンテナ（container）と呼びます。コンテナの中で何をしようと、本物のコンピュータには影響しません。ファイルを消してもコンテナ内のファイルが消えるだけです。システム設定を変えてもコンテナ内の設定が変わるだけです。コンテナを終了すれば、中でやったことはすべて消え、きれいな状態に戻ります。

この中でYOLOモードをオンにしたらどうなるでしょう。Claude Codeがどれだけ rm -rf を実行しても、システムファイルを触っても、怪しいプログラムをインストールしても、すべてコンテナの中だけの話です。コンテナを閉じれば終わりです。あなたのMacには傷ひとつつきません。

Dockerのインストール方法はこうです。Macでは Docker Desktop というアプリを入れるだけです。docker.com にアクセスして「Download Docker Desktop」をクリックし、ダウンロードされた .dmg ファイルを開いてアプリケーションフォルダにドラッグすれば完了です。インストールが終わると、画面右上のメニューバーにクジラのアイコンが現れます。このクジラが見えていれば、Dockerが動いている証拠です。

では、コンテナを作ってみましょう。Claude Codeにこう伝えます。「Dockerコンテナをひとつ作って。現在のフォルダをコンテナ内からも見られるようにつなげて、Node.jsがインストールされた環境にしてね。」Claude Codeが docker run コマンドを自分で書いて実行してくれます。-it、--rm、-v といったオプションを覚える必要はありません。AIにやりたいことを伝えれば、コマンドはAIが作ります。

コンテナが起動すると、ターミナルのプロンプトが変わります。root@a1b2c3d4:/workspace# のような表示が出るはずです。これでコンテナの中に入っています。

ここでClaude Codeをインストールし、YOLOモードをオンにします。

```
npm install -g @anthropic-ai/claude-code

claude --dangerously-skip-permissions
```

こうすることで、コンテナ内でClaude CodeがYOLOモードで動きます。何をしても安全です。練習が終わって出たくなったら exit と入力するか Ctrl+D を押してください。コンテナが消え、元のターミナルに戻ります。

ひとつだけ気をつけることがあります。-v オプションでつないだフォルダは本物のフォルダです。コンテナ内で /workspace の中のファイルを消すと、Mac上の元のフォルダからも消えます。これは意図した仕様です。作業結果を外に取り出すには、フォルダがつながっている必要があるからです。だからこそ、Gitであらかじめ保存しておくのです。コンテナ内でファイルが壊れたら git checkout -- . で元に戻せます。

接続なしで完全に隔離された環境が欲しければ、-v オプションを外してください。docker run -it --rm node:20 bash で入ると、まっさらなコンテナが開きます。この中では何をしても外には一切影響しません。ただし作業結果を取り出すこともできないので、練習用にだけ使ってください。

Claude Codeには、Dockerコンテナを自動で作ってくれる機能もあります。設定ファイル（.claude/settings.json）で sandbox オプションをオンにすると、Claude Codeがコマンドを実行するときに自動でコンテナ内で動かします。これは少し上級者向けの設定なので、今は上の方法で始めてみて、慣れてきたら公式ドキュメントを参照して試してみてください。

今日身につけたこと

Dockerはコンピュータの中の仮想コンピュータです。Docker Desktopをインストールし、docker run コマンドでコンテナを作り、その中でClaude CodeをYOLOモードで動かせば安全です。コンテナを終了すれば、中でやったことはすべて消えます。ただし、-v でつないだフォルダは本物なので、先にGitで保存しておいてください。

前の章: 第21章 Guardian（ガーディアン）サブエージェント

次の章: 第23章 自分に合うモードを選ぶ

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

第23章 自分に合うモードを選ぶ

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第22章 Dockerサンドボックスの中でYOLOモードを動かす

次の章: 付録A コマンド集

ソウルから釜山へ向かう方法が何通りもあるように、Claude Codeの使い方にも複数の選択肢があります。KTXに乗ることもできるし、自家用車を走らせることもできるし、高速バスを使うこともできます。どれがいいかは、状況次第です。

デフォルトモードは最も遅いですが、最も安全です。Claude Codeが何かをしようとするたびに「これをやっていいですか?」と確認を求めてきます。ファイルを一つ作るときも聞いてきますし、コマンドを一つ実行するときも聞いてきます。Claude Codeを初めて使う方や、重要なプロジェクトを扱っている方に向いています。Claude Codeが何をしているのか、一つひとつ確認しながら学べるからです。

オートモード（Auto Mode）は中間的な選択肢です。2026年3月にAnthropicが追加した機能です。安全なコマンド（ファイルの読み取りや検索など）は自動で実行し、危険を伴う可能性のあるコマンド（ファイルの変更・削除・インストールなど）だけ確認を求めます。Claude Codeを起動したときに表示される画面で「Auto mode」を選ぶか、作業中にShift+Tabを押すと切り替えられます。デフォルトモードに少し窮屈さを感じ始めた方にちょうどいいモードです。

Yoloモードは速いですが、危険です。何も確認せずにすべて実行します。Gitで保存しておき、CLAUDE.mdに禁止ルールを書き、最大ターン数を設定した状態で使うのが基本です。このモードは作業速度が圧倒的に速く、デフォルトモードで30分かかっていた作業が5分で終わることもあります。ただし、トラブルも速く起きます。第19章で見たとおりです。

Docker + Yoloモードは、Yoloモードのスピードを保ちながら安全を確保する組み合わせです。Dockerコンテナの中でYoloモードを使うので、Claude Codeが何かを壊してもコンテナを止めればそれで済みます。さらにGuardianを加えると「Docker + Yolo + Guardian」になりますが、これが現時点で最もバランスの取れた設定です。

どのモードを使うか選ぶときの、参考になる基準をお伝えします。

初めてで不安なら、デフォルトモードから始めてください。一週間ほど使っていると「毎回確認されるのが面倒だ」と感じるようになります。そのタイミングでオートモードに切り替えればいいでしょう。

作業がシンプルなら、Gitで保存してからYoloモードを使えばいいです。「このファイルの変数名を変えて」とか「この関数にエラー処理を追加して」といった作業です。範囲が狭く、何をすべきか

が明確な作業です。

夜通し大きな作業を任せて寝るつもりなら、Docker + Yolo + Guardianの組み合わせを使ってください。最大ターン数もかけておき、コスト上限も設定しておく。朝起きてから結果を確認すればいいです。

費用の話もしておきます。Claude CodeはAnthropicのAPIを使うため、使った分だけ料金がかかります。価格は使用するモデルによって異なります。

Sonnet 4モデルを基準にすると、一日一、二時間ほどClaude Codeを使った場合、月に2,000円から5,000円程度になります。軽い作業が中心なら2,000円台、コードを大量に生成する作業が多ければ5,000円台です。

Opus 4モデルはより賢いですが、より高価です。同じ使用時間であれば、おおよそ5倍から10倍ほどコストが増えます。月に10,000円から50,000円ほどかかる場合もあります。複雑な作業にはOpusのほうが適していますが、ほとんどの作業にはSonnetで十分です。

Max構読というオプションもあります。月10,000円または20,000円の固定金額を払い、一定量のAPI使用量をまとめて使う方式です。毎月の使用量がある程度予測できるなら、こちらが便利です。「今月いくらかかるだろう」と心配しなくて済みますから。

無料で使うこともできます。claude.aiの無料アカウントに連携すると、制限はありますが無料で使えます。一日に使える量が決まっているため大きな作業には足りませんが、練習して慣れるには十分です。

どの料金プランを選ぶにせよ、Anthropicのコンソール（console.anthropic.com）で使用量の上限を設定することを忘れないでください。「今月30,000円を超えたら止める」と設定しておけば、寝ている間に300,000円が飛んでいく事故を防げます。

今日、身につけたこと

デフォルトモードは安全だけど遅く、オートモードは中間、Yoloモードは速いけど危険です。
Docker + Yolo + Guardian の組み合わせが、現在最もバランスの取れた設定です。Sonnetモデル基準で月2,000円から5,000円が基本的なコストです。使用量の上限は必ず設定してください。

付録

前の章: 第22章 Dockerサンドボックスの中でYOLOモードを動かす

次の章: 付録A コマンド集

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

付録A コマンド集

AIに任せて席を離れる - YOLOモード完全入門

前の章: 第23章 自分に合うモードを選ぶ

次の章: 付録B 用語集

Claude Code コマンド

インストール

npm install -g @anthropic-ai/claude-code

基本的な実行

claude

YOLOモードで実行

claude --dangerously-skip-permissions

YOLOモード＋ターン制限

claude --dangerously-skip-permissions --max-turns 20

モデルを指定して実行

claude --model claude-sonnet-4-20250514

プロンプトを直接渡す（対話なしで一度だけ実行）

claude -p "このフォルダのファイル一覧を教えて"

会話を引き継ぐ（前回の会話を続ける）

claude --continue

新しい会話を始める

claude --new

ログイン／認証

claude auth login

バージョン確認

claude --version

アップデート

npm update -g @anthropic-ai/claude-code

会話中のコマンド（Claude Code 起動後、会話画面で入力）

/help ヘルプを表示

/clear 会話をリセット

/compact 会話内容を圧縮

/config 設定の確認／変更

/cost 現在のセッションのコスト確認

/doctor 設定の問題を診断

/experimental guardian ガーディアンサブエージェントをオンにする

Codex コマンド

インストール

npm install -g @openai/codex

基本的な実行

codex

YOLOモードで実行する

codex --yolo

フルネームYOLOモード

codex --dangerously-bypass-approvals-and-sandbox

モデルを指定して実行する

codex --model o4-mini

プロンプトを直接渡す

codex -q "このコードのバグを見つけて"

対話モードの設定 (config.toml)

approval_policy = "never"

sandbox_mode = "danger-full-access"

Git コマンド

リポジトリを作成する

git init

ファイルをステージに追加する

git add ファイル名

すべてのファイルをステージに追加する

git add .

保存（コミット）する

git commit -m "説明メッセージ"

状態を確認する（何が変わったか）

git status

変更内容を詳しく確認する

git diff

変更の概要を確認する

git diff --stat

コミット履歴を確認する

git log --oneline

ファイルを一つ元に戻す

git checkout -- ファイル名

すべての変更を元に戻す（コミットしていないもの）

git checkout -- .

Nコミット前の状態に戻す

git reset --hard HEAD~N

ブランチを作成する

git checkout -b ブランチ名

ブランチ一覧を見る

git branch

ブランチを切り替える

git checkout ブランチ名

削除されたファイルを探す

git log --diff-filter=D --name-only

.gitignoreに書くもの

.env

node_modules/

.DS_Store

***.log**

Dockerコマンド

Docker Desktopのインストール確認

docker --version

コンテナを作って中に入る

docker run -it --rm node:20 bash

現在のフォルダをマウントしてコンテナを作る

docker run -it --rm -v $(pwd):/workspace -w /workspace node:20 bash

実行中のコンテナを見る

docker ps

すべてのコンテナを見る（停止中のものも含む）

docker ps -a

コンテナを止める

docker stop コンテナID

コンテナを削除する

docker rm コンテナID

イメージ一覧を見る

docker images

イメージを削除する

docker rmi イメージ名

コンテナから出る

exit（またはCtrl+D）

前の章: 第23章 自分に合うモードを選ぶ

次の章: 付録B 用語集

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

付録B 用語集

AIに任せて席を離れる - YOLOモード完全入門

前の章: 付録A コマンド集

次の章: 付録C 安全チェックリスト

ガーディアン (Guardian, ガーディアン)

Claude Codeの安全監視機能。コマンドを検査して、危険かどうかを判断します。

Git (Git, ギット)

コードの変更履歴を記録し、以前の状態に戻せるツール。タイムトラベルボタンをイメージするとわかりやすいです。

.gitignore (ドットギットイグノア)

Gitに無視させるファイルのリストを書いておくファイル。.envのような秘密ファイルが誤ってアップロードされるのを防ぎます。

GitHub (GitHub, ギットハブ)

コードをインターネット上に保存・共有できるサービス。コード版のGoogle Driveだと思えばわかりやすいです。

Node.js (Node.js, ノードジェイエス)

JavaScriptをコンピューター上で直接実行できるようにするプログラム。Claude Codeのインストールに必要です。

Docker (Docker, ドッカー)

コンピューターの中に隔離された仮想コンピューター（コンテナ）を作るツール。砂場を囲む柵のようなものです。

diff (差分, ディフ)

2つのファイルの違いのこと。git diffと打つと、どの行が変わったか表示されます。

リファクタリング (Refactoring, リファクタリング)

コードの動作はそのままに、構造だけをきれいに整え直す作業。部屋の片付けに似ています。

リポジトリ (Repository, リポジトリ)

Gitで管理するプロジェクトフォルダ。略して「レポ」とも呼ばれます。

ロールバック (Rollback, ロールバック)

以前の状態に戻すこと。デプロイしたものを前のバージョンに戻すときに使います。

Max Turns (最大ターン数, マックスターンズ)

Claude Codeが応答できる回数の上限。--max-turns 20と指定すると、20回応答した後に停止します。

モデル (Model, モデル)

AIの頭脳にあたるプログラム。Sonnet、Opus、Haikuなどがあります。

デプロイ (Deploy, デプロイ)

作ったものをサーバーに上げて、人々が使えるようにすること。

ブランチ (Branch, ブランチ)

Gitで元のファイルを変えずに実験用のコピーを作る機能。「枝分かれ」のイメージです。

利用上限額 (Spending Limit, スペンディングリミット)

API利用料金の上限を設定しておく機能。Anthropicのコンソールから設定できます。

ソネット (Sonnet, ソネット)

AnthropicのClaude AIモデルのひとつ。速度と性能のバランスが良く、ほとんどの作業に適しています。

オートモード (Auto Mode, オートモード)

安全なコマンドは自動で実行し、危険なコマンドだけ人に確認を求める中間設定。

オーパス (Opus, オーパス)

Anthropicが提供する最大のClaude AIモデル。高性能ですがコストも高めです。

npm (npm, エヌピーエム)

Node.jsのパッケージマネージャー。プログラムをインストールして管理するツールです。npm installと入力するとプログラムがインストールされます。

API (API, エーピーアイ)

プログラム同士がやり取りするための通路。Claude CodeがAnthropicのサーバーと通信する際にAPIを使います。

APIキー (API Key, エーピーアイキー)

APIを使うためのパスワード。クレジットカード番号と同じように、絶対に他人に見せてはいけません。

YOLOモード (YOLO Mode, ヨロモード)

AIが人の許可を求めずにすべてのコマンドを実行する設定。「You Only Live Once」から取った名前です。

コミット (Commit, コミット)

Gitで現在の状態を保存する操作。ゲームのセーブポイントと同じようなものです。

コンテナ (Container, コンテナ)

Dockerが作る隔離された実行環境。コンピューターの中に作る仮想のコンピューターです。

コーデックス (Codex、コーデックス)

OpenAIが作ったターミナルベースのAIコーディングツール。Claude CodeのOpenAI版と考えるとわかりやすいです。

クロードコード (Claude Code、クロードコード)

Anthropicが作ったターミナルベースの AIコーディングツール。この本の主役です。

ターミナル (Terminal、ターミナル)

コンピューターに文字で命令を入力する黒い画面。Macでは「ターミナル」アプリ、Windowsでは「コマンドプロンプト」です。

トークン (Token、トークン)

AIが文章を読み書きするときに使う単位。日本語の1文字がおよそ1〜2トークンに相当します。利用料金はトークン単位で計算されます。

プロンプト (Prompt、プロンプト)

AIへの指示のこと。「このファイルを整理して」がプロンプトです。

ハイク (Haiku、ハイク)

Anthropicが提供する最も小さく高速なClaudeのAIモデル。軽量で低コストですが、複雑な作業には限界があります。

CLAUDE.md (クロードエムディー)

Claude Codeにプロジェクトのルールを伝えるファイル。このファイルに書かれたルールをClaude Codeが従います。

rm -rf (アールエム マイナス アールエフ)

ファイルやフォルダーを問答無用で全部削除するコマンド。ゴミ箱にも入りません。危険なコマンドです。

SSH (エスエスエイチ)

別のコンピューターにターミナルからリモート接続する方法。

前の章: 付録A コマンド集

次の章: 付録C 安全チェックリスト

弁護士 · 元国会議員 · AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

付録C 安全チェックリスト

AIに任せて席を離れる - YOLOモード完全入門

前の章: 付録B 用語集

（モニターの横に貼っておきましょう）

YOLOモードをオンにする前に

Gitで保存したか？

git add . と git commit -m "YOLO前に保存" を実行したか確認します。

ブランチを分けたか？

git checkout -b 実験用 でブランチを作成したか確認します。

問題が起きたら、元のブランチに戻れば大丈夫です。

Docker内で作業しているか？

docker run でコンテナを作成して、その中に入っているか確認します。

Docker内であれば、Mac本体は安全です。

指示は具体的か？

「整理して」ではなく「logsフォルダの.logファイルだけ削除して」のように、具体的な指示になっているか確認します。

曖昧な指示がトラブルの原因になります。

APIキーが漏れていないか？

.gitignoreファイルに.envが含まれているか確認します。

GitHubにプッシュする前に、git statusで.envがリストに出ていないか確認します。

最大ターン数を設定したか？

--max-turns 20 オプションを付けたか確認します。

設定しないと、Claude Codeが際限なく動き続けることがあります。

コスト上限を設定したか？

console.anthropic.com で今月の上限を設定したか確認します。

CLAUDE.mdに禁止ルールを書いたか？

rm -rf 禁止、.envのGit追加禁止、システムファイルの変更禁止といったルールを記載したか確認します。

作業が終わったら

git statusで変更ファイルを確認したか?

Claude Codeが何を変更したか、必ず確認します。

git diffで変更内容を読んだか?

想定外の変更がないか確認します。

問題のないものだけ選んでコミットしたか?

git add ファイル名 で必要なものだけ選んで保存します。

残りは git checkout -- ファイル名 で元に戻します。

APIキーがコミットに含まれていないか?

git diff --cached でコミット予定の内容をもう一度確認します。

問題が起きたとき

Ctrl+Cで止めたか?

git status、git diffで状況を把握したか?

git checkout -- . または git reset --hard で元に戻したか?

CLAUDE.mdに再発防止のルールを追加したか?

前の章: 付録B 用語集

弁護士 ・ 元国会議員 ・ AI政策研究者

kimkj.com

#キムギョンジン #AI書斎 #YOLOモード #ClaudeCode #Codex #AI自動化 #Docker

このAI書籍を応援する

この本が役に立った場合は、今後の翻訳、公開版、
AI知識プロジェクトをPayPalで応援できます。

PayPal

https://paypal.me/kimkjkj

QRコードを読み取るか、上のリンクを開いてください。