E4 Claude Code

Gitの導入と設定：変更履歴という「保存の金庫」

非開発者向け。ターミナルなしで進める：Gitの確認・設定、GitHub（非公開）、保存と戻し方、確認画面で見るべき言葉、守るルール

更新日：2026-10-03（2026年10月時点の情報）｜HUMAC 伊藤晴通

対象ツール：Claude Code

Git と GitHub は、**どのAIエージェントでも共通**です。「Claudeに頼む」「Codeタブ」などの操作は、**Claude Code**（Claudeのデスクトップアプリの「Code」タブ）の場合です。

対象の方：開発者ではない方（経営者・事務・現場の担当者）。プログラミングは不要です。**コードは書かず、AIエージェントに頼んで、自分の業務アプリをつくる**ところまでを、順に進めます（ターミナルの黒い画面は、使わない形で書いています）。

Gitとは、「セーブポイント」付きの保存機能

ゲームのセーブポイントのように、**仕事のフォルダの「ある時点の状態」を、何度でも保存**できる仕組み。あとから「昨日の状態に戻す」「いつ、何を変えたか見る」ができる。そして、保存したものを**インターネット上の「貸金庫」（GitHub）**にも預けておけば、パソコンが壊れても、別のパソコンでも、続きができる。

難しい操作は要らない。**ターミナル（黒い画面）は、開かなくてよい**。この資料は、Claudeに頼むか、画面のボタンだけで進められる形で書いてある。E1で作った仕事のフォルダを、ここで「保存できる状態」にする。

1. なぜ、AIエージェントにGitが要るのか

AIが壊しても、戻せる

AIは、ファイルを書き換えたり、消したりする。**保存した時点には、いつでも戻れる**。安心して任せられる。

「いつ・何を」が分かる

AIが何を変えたかを、履歴で確認できる。引き継ぎや、あとからの確認にも使える。

バックアップになる

貸金庫（GitHub）に預けておけば、パソコンの故障や紛失から守れる。別のPCにも、そのまま移せる。

並行して作業できる

Claudeのデスクトップアプリで、複数のセッションを、それぞれ独立した作業コピーで動かすには、Gitが必要（公式）。

AIの「巻き戻し」機能は、Gitの代わりにならない

Claude Code にも、AIの編集を取り消す機能（チェックポイント）がある。ただし**戻せるのは、AIが直接編集したファイルだけ**。コマンドで消したファイルや、外部への操作は戻せず、保存期間も限られる（約30日）。**長く残す履歴は、Gitで持つ**（公式の説明）。

2. 3つの段階：まずは①だけでもよい

段階	できること	用意するもの	目安	ターミナル
① PCの中に保存する	昨日の状態に戻せる。履歴が残る	Git	約15分	不要（Claudeに頼む）
② 貸金庫（GitHub）に預ける	バックアップ。別のPCで続きができる	GitHubのアカウント、GitHub Desktop	＋約20分	不要（ボタンで操作）
③ 「保存して、金庫に送って」と頼めるようにする	終わりの合言葉に、保存と送信まで含められる	GitHub CLI（gh）でのログイン	＋約10分	任意（Claudeに頼めば、操作は案内される）

おすすめは、**①と②**。③は、慣れてから。勉強会・サロンでは、①②を実際にやる。

3. 準備の手順

手順1：Gitが入っているか確認する（①）

ClaudeのデスクトップアプリのCodeタブで（まだなら、先に A4：契約と最初の設定）、仕事のフォルダを選び、次のように頼む。

コピーして使う

このパソコンで、Git が使える状態か確認してください。使えなければ、私のパソコン（MacかWindowsか）に合わせて、入れ方を1手順ずつ、やさしく教えてください。

パソコン	入れ方（Gitの公式の案内）
Mac	ほとんどのMacには、最初から入っている（Claudeの公式ドキュメントの説明）。入っていないときは、Macが「開発者ツールをインストールしますか」と案内を出すので、「インストール」を押して、終わるまで待つ（Gitの公式が、いちばん簡単な方法として案内している）
Windows	Gitの公式サイト（git-scm.com）から、「Git for Windows」のインストーラーをダウンロードして実行する。迷ったら、設定は初期のまま「Next」で進めてよい

手順2：最初の設定（名前とメールアドレス）

Gitは、保存のたびに「誰が保存したか」を記録する。最初に、名前とメールアドレスを1回だけ設定する。

コピーして使う

Git の最初の設定をしてください。名前は〔あなたの名前〕、メールアドレスは〔使っているメールアドレス〕です。設定したあと、設定が合っているか確認して、結果を見せてください。

参考：Claudeがやっていること

Gitの最初の設定は、`git config --global user.name "名前"` と `git config --global user.email メールアドレス` の2つ（Gitの公式の手順）。自分で打ちたい人は、この2行をターミナルに入れてもよい。名前とメールは、保存の履歴に載る。**だから、保存先は必ず非公開にする**（後述）。

手順3：仕事のフォルダを、Gitで保存できる状態にする（①）

先に、「入れてはいけないもの」を決める。鍵・パスワード・実データ・個人情報は、保存の対象から外す（D1）。この除外リストが「.gitignore」というファイル。

コピーして使う

この仕事のフォルダを、Git で保存できる状態にしてください。
先に .gitignore を作ってください。次のものは、保存の対象から外します：鍵・パスワード（.env など）、data フォルダ（実データ）、パソコンが勝手に作る不要なファイル（.DS_Store など）。
作ったら、保存の対象になっているファイルの一覧を見せてください。私が確認してから、最初の保存をします。

.gitignore に入れる例	理由
`.env  .env.*  *.key  *.pem`	鍵・パスワード。入れると、金庫から漏れる
`data/`	実データ（個人情報・決算書など）
`.DS_Store  Thumbs.db`	パソコンが自動で作る不要なファイル
動画・大きな画像のフォルダ	大きなファイルは、金庫に向かない

手順4：最初の保存（①）

コピーして使う

いまのフォルダの状態を、最初の保存として記録してください。保存の説明は「最初の保存：指示書と現在地ファイル」にしてください。保存したあと、保存の履歴を見せてください。

これで、「セーブポイント」が1つできた。**以後は、「保存して」と頼むだけ**。保存の説明（コミットメッセージ）は、日本語でよい。「何を変えたか」を一言で書く。

手順5：貸金庫（GitHub）を用意する（②）

1. **アカウントを作る**
   github.com で、無料のアカウントを作る。二段階認証を有効にしておくと安心
2. **GitHub Desktop を入れる**
   desktop.github.com から入れる（無料。ボタンで操作できるアプリ）。「Sign into GitHub.com」でログイン
3. **フォルダを追加する**
   メニュー「File」→「Add local repository」で、仕事のフォルダを選ぶ（ドラッグでも追加できる）
4. **金庫に預ける（公開）**
   画面の「Publish repository」から、GitHubに作る。**必ず「非公開（Private）」を選ぶ**

いちばん大事：必ず「非公開」にする

「公開」にすると、世界中の誰でも中身を見られる。**仕事のフォルダは、すべて非公開**。預ける前に、画面の「非公開（Private）」の選択を、毎回、目で確認する（ボタンの名前は、アプリの版で変わることがある）。

預けたあとは、GitHub Desktop の画面に、変更されたファイルの一覧が出る。**保存は「Commit」ボタン、金庫に送るのは「Push origin」ボタン**。まずは、この2つのボタンだけ覚えればよい。

手順6（任意）：Claudeに「保存して、金庫に送って」と頼めるようにする（③）

慣れたら、Claudeが金庫への送信までやれるようにできる。GitHubの公式コマンドラインツール「gh」を入れて、1回だけログインする。

コピーして使う

GitHub CLI（gh）が使える状態か確認してください。入っていなければ、私のパソコンに合う入れ方を1手順ずつ教えてください。入ったら、GitHub へのログインを、ブラウザで行う方法で案内してください。ログインが終わったら、ログイン状態を確認してください。

- ログインは、ブラウザが自動で開き、GitHubのアカウントで認証する形（GitHubの公式の説明）。認証の情報は、パソコンの安全な保管場所に残るので、以後は、毎回のログインは要らない。
- **パスワードやトークンを、チャットに書かない**。ログインは、必ずブラウザの画面で行う。

4. 毎日の使い方：終わりの合言葉に入れる

E1の「終わりの合言葉」に、保存を入れる。これで、毎回の作業の最後に、現在地・決めたこと・保存が、まとめて残る。

コピーして使う

今日の作業を締めます。STATUS.md を更新し、決めたことと理由を docs/決定ログ.md に追記してください。そのあと、変更を保存してください（保存の説明は、今日やったことを一言で）。保存した内容の一覧を見せてください。

（③まで設定した人は、最後に「金庫にも送ってください」を足す。②だけの人は、GitHub Desktop の「Push origin」を押す。）

5. 戻したいとき：「見る」→「戻す」の順で

いきなり戻さない。**先に、何が変わるかを見てから、戻す**。

やりたいこと	Claudeへの頼み方
AIがさっき直した所を、取り消したい	「さっきの変更を、すべて取り消してください。取り消す前に、どのファイルがどう変わるか教えてください」
昨日の状態を、見たい	「保存の履歴を、新しい順に10件、日付と説明で見せてください」→「〇日の保存のときの、〔ファイル名〕の中身を見せてください」（見るだけで、何も変えない）
昨日の状態に、戻したい	「いまの状態を、先に保存してください」→「〇日の保存の状態に戻したい。戻す前に、変わるファイルの一覧を見せてください。私が確認するまで、実行しないでください」
消してしまったファイルを、復活したい	「保存の履歴から、〔ファイル名〕を探して、元の場所に復元してください」

GitHub Desktop の「History」タブでも、保存の履歴を一覧で見られる。

6. Claudeの確認画面で、見るべき言葉

Claudeがコマンドを実行する前に、実行する内容が確認画面に出る（最初の設定の「確認あり」のとき）。**コマンドの意味は、読めなくてよい**。ただし、次の言葉があったら、承認せず、「これは何をするものですか？本当に必要ですか？」と聞く。

確認画面の言葉	意味	どうする
`rm`、`delete`、`clean`	ファイルを消す	消す対象を聞く。消さずに、退避フォルダへ（E1）
`--force`、`-f`	確認を飛ばして、強引に実行する	原則、承認しない。理由を聞く
`reset --hard`、`checkout .`、`restore .`	保存していない変更を、捨てて戻す	捨てられる内容を、先に確認する
`push` と `--force` の組み合わせ	金庫の履歴を、書き換える	承認しない
`public`（公開）	金庫を、世界に公開する	承認しない。必ず非公開

7. 守るルール（E2の「関所」）

1. **金庫は、必ず非公開**。
2. **鍵・パスワード・個人情報・実データは、保存しない**（.gitignore で除外。鍵の置き場所は E6）。間違えて入れてしまったら、履歴から消すだけでは足りない。**その鍵を、すぐ無効にして、新しいものに入れ替える**（講師の約6万8千円の事故の教訓。E2）。

3. **履歴を書き換える・消す操作は、AIに任せない**（上の表）。迷ったら、人が判断する。
4. **保存の前に、中身を見る**。「何が保存の対象になっているか」の一覧を、毎回、軽く眺める。
5. **大きなファイル（動画など）は、入れない**。

8. 用語のやさしい説明

用語	やさしく言うと
Git	変更履歴を残す仕組み（セーブ機能）。自分のパソコンの中で動く
GitHub	Gitで保存したものを預ける、インターネット上の貸金庫
リポジトリ	Gitで管理しているフォルダ（保存の履歴つきの仕事フォルダ）
コミット（保存）	セーブポイントを1つ作ること。説明を添える
プッシュ（金庫に送る）	保存したものを、GitHubに預けること
プル（金庫から取る）	GitHubにある最新の状態を、自分のパソコンに取り込むこと
ブランチ	履歴の枝分かれ。最初は、1本（main）だけで十分
ワークツリー	同じ仕事フォルダの、独立した作業コピー。複数のセッションを並行して動かすとき
.gitignore	「保存の対象にしないもの」の除外リスト

9. ターミナルを使う場面のまとめ

場面	ターミナル	代わりの方法
Gitの確認・設定・保存・履歴の確認・戻す	不要	Claudeに頼む
金庫に預ける、変更を送る	不要	GitHub Desktop のボタン
Claudeに、金庫への送信まで頼む（③）	任意	Claudeに、ghの導入とログインを案内してもらう（ログインはブラウザ）

ターミナルの操作に慣れた人向けの確認方法として、`git --version`（Gitが入っているか）、`git status`（いまの状態）、`git log --oneline`（保存の履歴）がある。

講師の実例

- 設定ファイル・手順書（スキル）・記憶のファイルを、非公開の金庫に、毎日自動で保存している。
- 保存の対象は「入れるものだけを許可する」形の除外リストにして、鍵や会話の記録が入らないようにしている。
- 作業の最後は「締めます」の一言で、記録・保存・バックアップまで、まとめて行う。
- Git はもともと開発者の道具だが、**「保存して」「金庫に送って」と頼むだけ**で、非開発者でも使える。

10. はじめての日のチェックリスト

□	確認すること
□	Gitが使える状態になった（Claudeに確認してもらった）
□	名前とメールアドレスを設定した
□	.gitignore を作り、鍵・実データが除外されている
□	最初の保存ができた。履歴に1件出ている
□	GitHubのアカウントを作った（二段階認証つき）
□	金庫に預けた。**非公開（Private）**になっている
□	終わりの合言葉に、保存を入れた
□	「見る→戻す」の順を、1回練習した（ファイルを1つ直して、取り消してもらう）

関連資料

- E1 はじめてのAIエージェント：フォルダ構成と管理 … 仕事のフォルダの作り方と、終わりの合言葉
- E2 ハーネスエンジニアリング入門 … 関所と事故から生まれたルール
- D1 信号機ルール … 入れてよい情報・いけない情報

出典（2026年10月確認）

- Git 公式：ダウンロード（Mac・Windows） (https://git-scm.com/downloads)

- Git 公式：最初の設定（First-Time Git Setup） (https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup)
- GitHub Docs：リポジトリの作成 (https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories)
- GitHub Docs：GitHub Desktop をはじめる (https://docs.github.com/en/desktop/overview/getting-started-with-github-desktop)
- GitHub Docs：自分のフォルダを GitHub Desktop に追加する (https://docs.github.com/en/desktop/adding-and-cloning-repositories/adding-a-repository-from-your-local-computer-to-github-desktop)
- GitHub CLI：gh auth login（ブラウザでのログイン） (https://cli.github.com/manual/gh_auth_login)
- Claude Code 公式：デスクトップアプリ（セッション、作業コピーにGitが必要） (https://code.claude.com/docs/en/desktop)
- Claude Code 公式：チェックポイント（Gitの代わりにならない） (https://code.claude.com/docs/en/checkpointing)