Clineの基本的な利用方法

1. ActモードとPlanモードの使い分け

Actモード（実行モード）

- すぐにコードの変更・実行を開始する
- 明確なタスクで即座に作業を進めたい場合に使用
- 実装方針が決まっており、短時間で進めたい時に最適

Planモード（計画モード）

- 実行前にClineが計画（Plan）を立て、内容を確認できる
- 複雑なタスクや不明点のある依頼時に使用
- ステップバイステップで提案を確認・修正できる
- 大規模変更や安全性を重視する際に推奨

2. 変更の承認フロー

Auto-approve（自動承認）設定

- Actモード中に、**変更を手動承認するか自動で適用するか**を切り替え可能
- 自動適用をオンにすると、変更確認なしで即反映
- 信頼できるタスクや単純な修正時に便利
- 重要な変更時は、確認を挟む設定（手動承認）を推奨

承認の流れ

- **Saveボタン**：変更を承認・適用
- **Rejectボタン**：変更を拒否・再提案を依頼
- 差分表示で内容を確認しながら判断

差分表示の色分け

- **緑（Green）**：追加された行
- **赤（Red）**：削除された行
- 差分で変更内容を視覚的に把握可能

確認のポイント

- コードの意図が正しいか
- 副作用やセキュリティ影響がないか
- 既存機能や依存関係への影響を確認

3. 変更を元に戻す方法

1

Clineの履歴機能を使う

- 「元に戻して」「Undo」などと指示することで、Clineがセッション内の変更を復元
- セッション内であれば、Gitを使わずに元に戻せる

2

Gitで復元（推奨）

- git diff でClineの変更を確認
- git checkout ‹ファイル› や git reset --hard で元に戻す
- VSCodeのGitタブからもGUIで確認可能
- セッションをまたいだ変更や複雑な修正はGitでの管理が安全

安全な運用のポイント

- 作業前に git commit でスナップショットを残す
- 実験的な作業は git checkout -b ‹ブランチ名› でブランチを作成
- Cline履歴＋Git履歴の二重管理で安心

4. Clineが実行できる主な操作

ターミナル操作

- npm、pip、git などのコマンドを実行
- ビルド・テスト・サーバー起動なども可能
- 出力結果をコンテキストとしてClineに渡せる

プログラム実行

- **Python / Node.js** スクリプトの実行に対応
- プロジェクト環境に応じて他言語も利用可能

ブラウザ操作

- WebドキュメントやAPI仕様の閲覧
- スクリーンショット取得・Web動作確認
- コード修正とブラウザ確認を往復可能

ファイル操作

- ファイルの読み込み・編集・作成・削除
- ディレクトリ検索（grep相当）
- 複数ファイルの同時編集

5. コンテキストの追加方法

ファイルやフォルダを追加

- **Shift + ドラッグ＆ドロップ**でファイル／フォルダをコンテキストに追加
- 複数選択にも対応

@記号で追加

- チャット欄で @ を入力すると候補一覧が表示
- **@file**：特定ファイルを追加
- **@folder**：フォルダ全体を追加
- **@terminal**：ターミナル出力を追加

6. ワークフロー機能の活用

概要

- 定型タスクをショートカットで実行できる機能
- 「/」を入力すると利用可能なワークフロー一覧が表示される（バージョン依存）

配置場所

- .clinerules/workflows/ フォルダに配置
- YAML または Markdown で定義可能

活用例

- コードレビューリクエスト
- テストコード生成
- ドキュメント作成
- デプロイチェックリスト

7. .clinerules設定

.clinerulesとは

プロジェクトごとのルールやガイドラインを定義する設定ファイル

プロジェクトルートに配置し、Clineの動作基準を統一

設定内容の例

コーディング規約

言語スタイル・命名ルール

禁止事項

使用禁止ライブラリ・関数

推奨事項

デザインパターンや構成指針

テスト要件

カバレッジやフレームワーク指定

ドキュメント方針

READMEやコメント整備ルール

メリット

- 一貫したAI提案・コード生成が可能
- チーム開発での統一ルール維持
- 新メンバーのオンボーディングが容易

まとめ

効果的な**Cline**活用のポイント

01

タスクに応じて **Act / Plan** モードを使い分け

02

自動承認（Auto-approve）は状況に応じて設定、**差分確認 → Save / Reject** の流れを習慣化

03

Gitで変更を追跡し、いつでも復元可能に

04

ターミナル・ブラウザ・ファイル操作などを活用

05

@記号・ドラッグ&ドロップ で適切にコンテキスト追加

06

ワークフローで定型タスクを効率化

07

.clinerules でプロジェクトの品質と統一性を維持