Claude Code クイックリファレンス

CLI コマンド、ワークフロー、MCP、フック、設定

はじめに

インストール & 起動

```
npm install -g @anthropic-ai/claude-code
claude           # start interactive session
claude --version  # check version
claude update     # update to latest
```

認証

```
claude login             # browser OAuth
export ANTHROPIC_API_KEY="sk-ant-..."
claude logout            # clear session
```

スラッシュコマンド

セッションコマンド

/help	利用可能なコマンドを表示
/clear	会話履歴をクリア
/compact	コンテキストを圧縮してトークンを節約
/cost	トークン使用量とコストを表示
/status	セッション情報とモデルを表示
/new	新しい会話を開始
/config	設定を開く / 表示
/doctor	設定の問題を診断
/init	プロジェクトの CLAUDE.md を作成
/login	Anthropic で認証
/logout	認証をクリア

ワークフローコマンド

/bug	バグレポートを送信
/review	コードレビューをリクエスト
/pr	プルリクエストを作成 / 更新
/commit	ステージ済みの変更をメッセージ付きでコミット

キーボードショートカット

Ctrl+C	現在の生成をキャンセル
Ctrl+D	Claude Code を終了（EOF）
Escape	現在の入力 / 編集をキャンセル
Tab	ファイルパスとコマンドをオートコンプリート
Up/Down	入力履歴をナビゲート

CLI フラグ

よく使うフラグ

-p, --print	非インタラクティブ（ヘッドレス）モード
--model	モデルを指定: **opus**、**sonnet**、**haiku**
--output-format	**text**、**json**、**stream-json** で出力
--allowedTools	ツールを制限: **Edit,Read,Bash**
--max-turns N	会話ターン数を制限
--debug	デバッグログを有効化
-n, --name	セッションに名前を付ける

ヘッドレス / スクリプト

```
claude -p "explain this error" < log.txt
claude -p "list todos" --output-format json
echo "fix typos" | claude -p
```

設定ファイル

CLAUDE.md の階層

./CLAUDE.md	プロジェクトレベルの指示（リポジトリにチェックイン）
./.claude/settings.json	プロジェクト設定（パーミッション、フック）
~/.claude/CLAUDE.md	ユーザーグローバルの指示（全プロジェクト共通）
~/.claude/settings.json	ユーザーグローバル設定
~/.claude/projects/*/CLAUDE.md	プロジェクト別ユーザー指示（リポジトリ外）

環境変数

ANTHROPIC_API_KEY	直接認証用の API キー
CLAUDE_MODEL	デフォルトモデルの上書き
CLAUDE_CONFIG_DIR	カスタム設定ディレクトリのパス

パーミッション

settings.json

```
{
  "permissions": {
    "allow": ["Read", "Glob", "Grep"],
    "deny": ["Bash(rm *)"]
  }
}
```

パーミッションモード

default	リスクのあるツールの前に確認
--dangerously-skip-permissions	全ツールを許可（CI / スクリプト専用）
--allowedTools	ツールセットを制限する CLI フラグ

MCP サーバー

MCP サーバーとは?

Model Context Protocol サーバーはカスタムツール（データベース、API、サービス）で Claude Code を拡張します。CLI または `.mcp.json` で管理します。

CLI による管理

```
claude mcp add server-name -- cmd arg1 arg2
claude mcp list
claude mcp remove server-name
```

.mcp.json の設定

```
{
  "mcpServers": {
    "my-db": {
      "command": "python",
      "args": ["-m", "mcp_server_db"],
      "env": { "DB_URL": "${DATABASE_URL}" }
    }
  }
}
```

スコープ

.mcp.json	プロジェクトレベルの MCP サーバー
~/.claude/mcp.json	ユーザーグローバルの MCP サーバー
claude mcp add -s user	ユーザースコープに追加
claude mcp add -s project	プロジェクトスコープに追加

フック

フックイベント

PreToolUse	ツール実行前に呼び出される
PostToolUse	ツール完了後に呼び出される
Notification	Claude が通知を送るときに呼び出される
Stop	Claude がレスポンスを完了したときに呼び出される

settings.json の例

```
{
  "hooks": {
    "PreToolUse": [{
      "matcher": "Bash",
      "hooks": [{
        "type": "command",
        "command": "check-command.sh $INPUT"
      }]
    }]
  }
}
```

フックの動作

exit 0	ツールの実行を許可
exit 2	ツールの実行をブロック
stdout	Claude への JSON フィードバック

SDK & 自動化

ヘッドレススクリプト

```
# Single prompt, get JSON output
claude -p "summarize main.py" \
  --output-format json

# Pipe input
git diff | claude -p "review this diff"
```

CI / GitHub Actions

```
- uses: anthropics/claude-code-action@v1
  with:
    claude_args: >
      --max-turns 5
      --model claude-sonnet-4-6
```

ヒント

自動化でコストを抑えるには `--max-turns` を使用。結果をプログラムで処理するには `--output-format json`。安全のために `--allowedTools` と組み合わせましょう。

モデル

モデルの選択

```
claude --model opus     # most capable
claude --model sonnet   # balanced (default)
claude --model haiku    # fastest, cheapest
```

モデルのショートカット

opus	Claude Opus — 最高性能
sonnet	Claude Sonnet — デフォルト、バランス型
haiku	Claude Haiku — 高速で経済的
--model full-name	例: **claude-sonnet-4-6**

ベストプラクティス

CLAUDE.md の書き方

プロジェクトの規約、技術スタック、ビルドコマンド、テスト手順を CLAUDE.md に記述しましょう。簡潔に — Claude は毎回セッションで読み込みます。`/init` でスキャフォールドできます。

コスト管理

/cost	現在のトークン使用量を確認
/compact	コンテキストが大きくなったら圧縮
--max-turns	自動化でターン数を制限
sonnet / haiku	シンプルなタスクには安いモデルを使用

効果的なプロンプト

具体的に	"auth.ts:42 の null チェックを修正" > "バグを修正"
コンテキストを与える	ファイル、関数、エラーメッセージを参照
@file を使う	直接ファイルを参照: **@src/main.ts**
画像を使う	スクリーンショットをドラッグ & ドロップで視覚的コンテキスト
繰り返し改善	フォローアップで精度を上げる。リセットには **/clear**