

主要スラッシュコマンド	
コマンド	効果・使いどころ
/help	使えるコマンドとヘルプを表示。入力欄で / を打つと全コマンドが絞り込み表示される
/init	コードベースを解析して CLAUDE.md の雛形を生成。既存があれば改善提案。CLAUDE_CODE_NEW_INIT=1 で対話型
/clear (/reset /new)	文脈を空にして新規会話。前の会話は /resume に残る。名前を付けて区別できる
/compact [指示]	会話を要約して文脈を節約。長くなって動作が重い時に使う。焦点を引数で指定可
/context [all]	文脈の使用量をグリッドで可視化。どのツールやメモリが容量を食っているか分かる
/model [名前]	モデルを切替えて既定として保存。引数なしでピッカー。s キーで当セッション限定切替
/effort [low..max]	推論強度を変更。low/medium/high/xhigh/max、利用可否はモデル依存。即時反映
/plan [説明]	プランモードに入る。大きな変更の前に実行計画をレビューさせる
/agents	サブエージェントの作成・編集・管理を開く
/mcp	MCP サーバの接続状態確認と OAuth 認証。要認証サーバはここでブラウザログイン
/memory	CLAUDE.md と rules を編集、auto memory の表示と有効切替
/hooks	設定済みフックの一覧表示
/permissions (/allowed-tools)	allow/ask/deny ルールを対話編集。スコープ別表示と作業ディレクトリ管理
/config (/settings)	設定タブを開く。テーマ・モデル・出力スタイル等を変更
/review [PR]	PR をローカルで読みレビュー。深いレビューは /code-review ultra
/diff	未コミット差分とターン別差分を対話ビューアで確認
/rewind (/undo)	会話やコードを過去の地点へ巻き戻す（チェックポイント）
/resume (/continue)	ID か名前で会話を再開、引数なしでピッカー。背景セッションも bg 印付きで出る
/usage (/cost /stats)	セッションコスト・プラン使用量・skill/MCP 別内訳を表示
/skills	利用可能な skill 一覧。t でトークン量ソート、Space で非表示化
/reload-plugins /reload-skills	再起動せずプラグイン・skill をディスクから再読込
/background (/bg)	セッションを背景エージェント化して端末を解放

Skills / Subagents / Commands

種類	置き場所	役割	呼び出し
Skill	.claude/skills/<名>/SKILL.md (user は ~/.claude/skills/)	繰り返す手順や長い参照を必要時だけ文脈に載せる。本文は使う時のみロードされ普段はコスト最小	Claude が関連時に自動起動、または /名 で手動。frontmatter の disable-model-invocation で自動を抑制
Custom Command	.claude/commands/<名>.md	skill に統合済。従来の commands ファイルもそのまま動く	/名. skill と同じ仕組みで \$ARGUMENTS 等が使える
Subagent	.claude/agents/<名>.md (user は ~/.claude/agents/)	独立した文脈・専用 system prompt・ツール制限で側タスクを処理し要約だけ返す。Haiku 等への振り分けでコスト節約	description を見て Claude が委譲、または /agents で管理。--agent で指定
frontmatter 主フィールド	Skill: name/description/allowed-tools/model/disable-model-invocation / Subagent: name/description/tools/model	Skill は Agent Skills 標準準拠（Claude Code が起動制御・サブエージェント実行を拡張）	引数置換 \$ARGUMENTS / \$1,\$2 / \$名。プラグイン skill は /plugin:skill で名前空間化

Hooks / MCP / Permissions

Hooks とは ライフサイクルイベントで自動実行されるシェル等。CLAUDE.md と違い Claude の判断に関係なく必ず走る。settings.json の hooks に定義

Hooks 主なイベント 詳細は別シート。SessionStart/SessionEnd, UserPromptSubmit/UserPromptExpansion, PreToolUse/PostToolUse, PostToolUseFailure/PostToolBatch, PermissionRequest/PermissionDenied, PreCompact/PostCompact, Stop/StopFailure, SubagentStart/Stop, TaskCreated/TaskCompleted, InstructionsLoaded, ConfigChange, CwdChanged, FileChanged, WorktreeCreate/Remove, Elicitation/ElicitationResult, Notification, Setup

Hooks type command (stdin に JSON でシェル実行) / http / mcp_tool / prompt (yes/no 判定の単発LLM) / agent. command 系は exit 2 で Claude にブロック理由を返す

matcher で対象ツールを絞る

コマンド	効果・使いどころ
/doctor	インストールと設定を診断。f キーで Claude に自動修復させる
/login /logout /status	認証セッション管理。/status で現在の認証方式を確認
/add-dir <path>	作業ディレクトリを追加（ファイルアクセス付与、.claude 設定は読まない）
/mcp_server__p rrompt	MCP サーバが公開するプロンプトをコマンドとして実行（引数は空白区切り）

CLI起動フラグ

フラグ	効果
claude	対話セッションを開始。claude "query" で初期プロンプト付き
-p, --print "query"	1回答して終了（ヘッドレス/スクリプト用）。cat file claude -p も可
-c, --continue	カレントディレクトリの直近会話を継続
-r, --resume <id>[名>	ID か名前でセッション再開、引数なしでピッカー
--model <id>	セッションのモデル指定。設定と ANTHROPIC_MODEL を上書き
--fallback-model <m>	既定モデルが過負荷/利用不可の時に切替（-p と背景セッションで有効）
--effort low..max	推論強度をセッション/限定で上書き（永続しない）
--permission-mode <m>	default/acceptEdits/plan/auto/dontAsk/bypassPermissions で開始
--dangerously-skip-permissions	全許可プロンプトを飛ばす。bypassPermissions と同義。隔離環境専用
--allowedTools / --disallowedTools	実行前確認なしで許可/拒否するツール。Bash(log git log) 形式
--tools "Bash,Edit,Read"	使える組込ツールを制限。"" で全停止、"default" で全部
--add-dir <path>	追加の作業ディレクトリ（読み書き許可）
--settings <file>[json>	設定ファイル/インライン JSON を明示し当該キーを上書き
--mcp-config <file>	MCP サーバ定義を JSON から読込。--strict-mcp-config で他を無視
--agent <name> / --agents <json>	サブエージェント指定 / JSON で動的定義

フラグ	効果
--bare	hooks/skills/plugins/MCP/auto memory/CLAUDE.md を読まず高速起動（Bash/Read/Edit のみ）
--bg ["prompt"]	背景エージェントとして開始し即復帰。--exec でシェルジョブも可
--output-format text json stream-json	-p モードの出力形式
--append-system-prompt[-file]	既定 system prompt に追記。--system-prompt[-file] は丸ごと置換
--fork-session	resume 時に元を残し新 ID で分岐
--from-pr <PR>	PR に紐付くセッションを再開（番号/URL）
--worktree, -w [名] HPR]	隔離した git worktree で開始。--tmux で tmux ペイン化
--max-budget-usd / --max-turns	-p モードで上限金額/ターン数に達したら停止
--debug ["api,hooks"] / --debug-file	デバッグログをカテゴリ絞り込みで出力
--ide / --chrome	起動時に IDE 連携 / Chrome ブラウザ連携を有効化
--disable-slash-commands	当セッションの全 skill/コマンドを停止
--setting-sources user,project,local	読み込む設定ソースを限定

設定ファイル階層

優先順位（高→低） Managed > CLI|引数 > Local > Project > User。同じキーは上位が勝つ。ただし permissions ルールは各スコープでマージされる。多くのキーは再起動なしで反映（model と outputStyle は除く）

Managed（最優先・上書き不可） 組織が MDM 等で配布。個人設定で無効化できない macOS /Library/Application Support/ClaudeCode/managed-settings.json / Linux,WSL /etc/claude-code/managed-settings.json / Win C:\Program Files\ClaudeCode\managed-settings.json

Local（自分のみ・当リポジトリ限定） 個人の実験用。gitignore 対象。共有前のテストはここで .claude/settings.local.json

Project（チーム共有） リポジトリに commit してチームで共有 .claude/settings.json

User（全プロジェクト・自分のみ） 個人の既定設定 ~/.claude/settings.json

settings.json 主要キー model / defaultMode / permissions(allow,ask,deny,additionalDirectories) / env / hooks / autoMemoryEnabled / claudeMdExcludes / agent / includeCoAuthoredBy. \$schema を入れると IDE 補充が効く

```
{ "$schema": "https://json.schemastore.org/claude-code-settings.json", "model": "claude-opus-4-6", "defaultMode": "acceptEdits", "permissions": { "allow": [ "Bash(npm run *)"], "deny": [ "Read(./env)"] } }
```

OAuth/MCP/キャッシュ 認証トークン、local/user スコープの MCP 定義、プロジェクト状態を保持。手書きしない ~/.claude.json

MCP（プロジェクト共有） project スコープの MCP 定義。git 管理。\${VAR:-default} 展開対応、初回は承認ダイアログ .mcp.json

CLAUDE.md（メモリ）

役割 毎セッション冒頭に丸ごと読み込まれる恒久指示。ビルド手順・規約・アーキ・常時ルールを書く。system prompt でなくユーザーメッセージとして渡るので強制力はない。厳格に止めたい処理は hook を使う

読込順（広→狭） Managed → User → 親ディレクトリ → Project → Local の順に連結。狭いスコープほど後に読まれる。サブディレクトリの CLAUDE.md は該当配下を触る時に遅延読込

Managed policy 組織全体。個人で除外不可。settings の claudeMd キーで直接埋め込みも可

OS別 managed パス配下の CLAUDE.md

User 全プロジェクトに効く個人指示

~/claude/CLAUDE.md

Project / Local チーム共有はどちらかに。個人用 Local は gitignore ./.CLAUDE.md / ./claude/CLAUDE.md / ./CLAUDE_local.md

Rules（パス限定） frontmatter の paths でファイルパターンに紐付け。マッチしたファイルに触る時だけ読込。paths なしは常時読込 .claude/rules/*.md (---|npaths:[\n - "src/api/**/*.ts" \n ---))

@import 構文 @path/to/file で他ファイルを取り込み。相対パスは当該ファイル基、最大4ホップ。起動時に展開されるので文脈は節約されない。1ファイル200行以内が目安

@AGENTS.md / @-/claude/my-prefs.md

AGENTS.md 連携 Claude は CLAUDE.md しか読まない。既存の AGENTS.md は @AGENTS.md で取り込むか symlink で共用

~/claude/settings.json

主要環境変数

認証・モデル ANTHROPIC_API_KEY 直接API / ANTHROPIC_AUTH_TOKEN Bearerプロキシ / ANTHROPIC_MODEL 既定モデル上書き / ANTHROPIC_BASE_URL エンドポイント ["type": "http", "url": "..."]' claude mcp list | get <名> | remove <名>

プロバイダ切替 CLAUDE_CODE_USE_BEDROCK / CLAUDE_CODE_USE_VERTEX / CLAUDE_CONFIG_DIR で設定ディレクトリ変更

メモリ・挙動 CLAUDE_CODE_DISABLE_AUTO_MEMORY=1 auto memory 無効 / CLAUDE_CODE_ADDITIONAL_DIRECTORIES_CLAUDE_MD=1 追加ディレクトリの CLAUDE.md も読む / CLAUDE_CODE_NEW_INIT=1 対話型 init / CLAUDE_PROJECT_DIR (hook/MCP に渡るプロジェクト root)

テレメトリ・更新 CLAUDE_CODE_ENABLE_TELEMETRY / DISABLE_AUTOUPDATER 自動更新停止 / CLAUDE_CODE_DISABLE_THINKING

MCP・ツール検索 MCP_TIMEOUT 起動ms / MAX_MCP_OUTPUT_TOKENS 出力上限(既定25000、超過警告は10000) / ENABLE_TOOL_SEARCH true/auto/auto:N/false / ENABLE_CLAUDEAI_MCP_SERVERS=false で claude.ai connector 無効

Sources（公式・一次情報 / クリックで開く）

1. Commands（全スラッシュコマンド一覧） — [code.claude.com/docs/en/commands](#)
2. CLI reference（コマンドとフラグ） — [code.claude.com/docs/en/cli-reference](#)

3. Settings（設定階層・権限・環境変数） — [code.claude.com/docs/en/settings](#)
4. Memory（CLAUDE.md / rules / auto memory） — [code.claude.com/docs/en/memory](#)

5. Skills（skill と custom command） — [code.claude.com/docs/en/skills](#)
6. Sub-agents（サブエージェント） — [code.claude.com/docs/en/sub-agents](#)

7. Hooks（フックイベント・type） — [code.claude.com/docs/en/hooks](#)
8. MCP（サーバ追加・スコープ・OAuth） — [code.claude.com/docs/en/mcp](#)