REFERÊNCIA RÁPIDA DE CLAUDE CODE

Comandos CLI, fluxos de trabalho, MCP, hooks, configuração

Início Rápido

Instalar & Iniciar

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

Autenticação

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

Slash Commands

Comandos de Sessão

`/help`	Mostrar comandos disponíveis
`/clear`	Limpar histórico da conversa
`/compact`	Condensar contexto para economizar tokens
`/cost`	Exibir uso de tokens e custo
`/status`	Exibir info da sessão e modelo
`/new`	Iniciar nova conversa
`/config`	Abrir ou ver configuração
`/doctor`	Diagnosticar problemas de configuração
`/init`	Criar CLAUDE.md para o projeto
`/login`	Autenticar com Anthropic
`/logout`	Limpar autenticação

Comandos de Workflow

`/bug`	Registrar um bug
`/review`	Solicitar revisão de código
`/pr`	Criar ou atualizar pull request
`/commit`	Fazer commit das mudanças staged com mensagem

Atalhos de Teclado

`Ctrl+C`	Cancelar geração atual
`Ctrl+D`	Sair do Claude Code (EOF)
`Escape`	Cancelar entrada / edição atual
`Tab`	Autocompletar caminhos e comandos
`Cima/Baixo`	Navegar no histórico de entrada

Flags CLI

Flags Comuns

`-p, --print`	Modo não-interativo (headless)
`--model`	Definir modelo: `opus`, `sonnet`, `haiku`
`--output-format`	Saída como `text`, `json`, `stream-json`
`--allowedTools`	Restringir ferramentas: `Edit,Read,Bash`
`--max-turns N`	Limitar turnos da conversa
`--debug`	Ativar log de debug
`-n, --name`	Nomear a sessão

Headless / Scripting

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

Arquivos de Configuração

Hierarquia CLAUDE.md

`./CLAUDE.md`	Instruções do projeto (versionado no repo)
`./.claude/settings.json`	Configurações do projeto (permissões, hooks)
`~/.claude/CLAUDE.md`	Instruções globais do usuário (todos os projetos)
`~/.claude/settings.json`	Configurações globais do usuário
`~/.claude/projects/*/CLAUDE.md`	Instruções por projeto do usuário (fora do repo)

Variáveis de Ambiente

`ANTHROPIC_API_KEY`	Chave API para autenticação direta
`CLAUDE_MODEL`	Override do modelo padrão
`CLAUDE_CONFIG_DIR`	Caminho de config personalizado

Permissões

settings.json

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

Modos de Permissão

`default`	Perguntar antes de ferramentas arriscadas
`--dangerously-skip-permissions`	Permitir todas as ferramentas (somente CI/scripts)
`--allowedTools`	Flag CLI para restringir conjunto de ferramentas

Servidores MCP

O que São Servidores MCP?

Servidores Model Context Protocol estendem o Claude Code com ferramentas customizadas — bancos de dados, APIs, serviços. Gerenciados via CLI ou `.mcp.json`.

Gerenciamento CLI

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

Config .mcp.json

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

Escopo

`.mcp.json`	Servidores MCP do projeto
`~/.claude/mcp.json`	Servidores MCP globais do usuário
`claude mcp add -s user`	Adicionar ao escopo do usuário
`claude mcp add -s project`	Adicionar ao escopo do projeto

Hooks

Eventos de Hook

`PreToolUse`	Executa antes de uma ferramenta ser usada
`PostToolUse`	Executa após uma ferramenta completar
`Notification`	Executa quando Claude envia uma notificação
`Stop`	Executa quando Claude termina uma resposta

Exemplo settings.json

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

Comportamento de Hook

`exit 0`	Permitir que a ferramenta prossiga
`exit 2`	Bloquear a ferramenta de executar
`stdout`	Feedback JSON para o Claude

SDK & Automação

Scripting Headless

```
# 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
```

Dicas

Use `--max-turns` para limitar custo em automação. Use `--output-format json` para parsear resultados programaticamente. Combine com `--allowedTools` para segurança.

Modelos

Seleção de Modelo

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

Atalhos de Modelo

`opus`	Claude Opus — máxima capacidade
`sonnet`	Claude Sonnet — padrão, balanceado
`haiku`	Claude Haiku — rápido e econômico
`--model full-name`	ex.: `claude-sonnet-4-6`

Boas Práticas

Escrevendo CLAUDE.md

Coloque convenções do projeto, stack, comandos de build e instruções de teste no CLAUDE.md. Seja conciso — Claude lê a cada sessão. Use `/init` para gerar um.

Gestão de Custo

`/cost`	Verificar uso de tokens em tempo real
`/compact`	Comprimir contexto quando crescer muito
`--max-turns`	Limitar turnos em automação
`sonnet / haiku`	Usar modelos mais baratos para tarefas simples

Prompts Eficazes

`Seja específico`	"Fix the null check in auth.ts:42" > "fix bug"
`Dê contexto`	Referencie arquivos, funções, mensagens de erro
`Use @arquivo`	Referencie arquivos diretamente: `@src/main.ts`
`Use imagens`	Arraste e solte screenshots para contexto visual
`Itere`	Continue refinando; use `/clear` para resetar