

Riferimento Rapido Claude Code

Comandi CLI, workflow, MCP, hook, configurazione

Per Iniziare

Installazione e Avvio

```
npm install -g @anthropic-ai/claude-code
claude # avvia sessione interattiva
claude --version # controlla versione
claude update # aggiorna all'ultima versione
```

Autenticazione

```
claude login # OAuth via browser
export ANTHROPIC_API_KEY="sk-ant-..."
claude logout # cancella sessione
```

Comandi Slash

Comandi di Sessione

```
/help Mostra i comandi disponibili
/clear Cancella la cronologia della conversazione
/compact Comprime il contesto per risparmiare token
/cost Mostra utilizzo token e costo
/status Mostra info sessione e modello
/new Avvia una nuova conversazione
/config Apre o visualizza la configurazione
/doctor Diagnostica problemi di configurazione
/init Crea un CLAUDE.md per il progetto
/login Autentica con Anthropic
/logout Cancella l'autenticazione
```

Comandi Workflow

```
/bug Segnala un bug
/review Richiede una revisione del codice
/pr Crea o aggiorna una pull request
/commit Esegue commit delle modifiche con messaggio
```

Scorciatoie da Tastiera

```
Ctrl+C Annulla la generazione corrente
Ctrl+D Esci da Claude Code (EOF)
Escape Annulla input/modifica corrente
Tab Completamento automatico percorsi file e comandi
Su/Giù Naviga nella cronologia degli input
```

Flag CLI

Flag Comuni

```
-p, --print Modalità non interattiva (headless)
--model Imposta modello: opus, sonnet, haiku
--output-format Output come text, json, stream-json
--allowedTools Limita gli strumenti: Edit, Read, Bash
--max-turns N Limita i turni della conversazione
--debug Abilita logging di debug
-n, --name Assegna un nome alla sessione
```

Headless / Scripting

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

File di Configurazione

Gerarchia CLAUDE.md

./CLAUDE.md Istruzioni a livello progetto (versionato nel repo)

./.claude/settings.json Impostazioni progetto (permessi, hook)

~/ .claude/CLAUDE.md Istruzioni globali utente (tutti i progetti)

~/ .claude/settings.json Impostazioni globali utente

~/ .claude/projects/*/CLAUDE.md Istruzioni utente per progetto (non nel repo)

Variabili d'Ambiente

ANTHROPIC_API_KEY Chiave API per autenticazione diretta

CLAUDE_MODEL Override del modello predefinito

CLAUDE_CONFIG_DIR Percorso directory di configurazione personalizzata

Permessi

settings.json

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

Modalità Permessi

default Chiede conferma prima di strumenti rischiosi

--dangerously-skip-permissions Consente tutti gli strumenti (solo CI/script)

--allowedTools Flag CLI per limitare il set di strumenti

Server MCP

Cosa Sono i Server MCP?

I server Model Context Protocol estendono Claude Code con strumenti personalizzati — database, API, servizi. Gestiti via CLI o `~.mcp.json`.

Gestione via CLI

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

Configurazione .mcp.json

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

Scope

.mcp.json Server MCP a livello progetto

~/ .claude/mcp.json Server MCP globali utente

claude mcp add -s user Aggiunge allo scope utente

claude mcp add -s project Aggiunge allo scope progetto

Hook

Eventi Hook

PreToolUse Eseguito prima che uno strumento venga avviato

PostToolUse Eseguito dopo che uno strumento è completato

Notification Eseguito quando Claude invia una notifica

Stop Eseguito quando Claude termina una risposta

Esempio settings.json

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

Comportamento degli Hook

exit 0 Consente allo strumento di procedere

exit 2 Blocca l'esecuzione dello strumento

stdout Feedback JSON a Claude

SDK e Automazione

Scripting Headless

```
# Prompt singolo, output JSON
claude -p "summarize main.py" \
  --output-format json
```

```
# Input da pipe
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
```

Consigli

Usa `--max-turns` per limitare i costi nell'automazione. Usa `--output-format json` per elaborare i risultati a livello di codice. Combina con `--allowedTools` per sicurezza.

Modelli

Selezione Modello

claude --model opus # più capace

claude --model sonnet # bilanciato (predefinito)

claude --model haiku # più veloce, meno costoso

Scorciatoie Modello

opus Claude Opus — massima capacità

sonnet Claude Sonnet — predefinito, bilanciato

haiku Claude Haiku — veloce ed economico

--model full-name es. **claude-sonnet-4-6**

Best Practice

Scrivere CLAUDE.md

Inserisci in CLAUDE.md le convenzioni del progetto, lo stack tecnologico, i comandi di build e le istruzioni di test. Mantienilo conciso — Claude lo legge ad ogni sessione. Usa `/init` per generarne uno.

Gestione dei Costi

/cost Controlla l'utilizzo corrente dei token

/compact Comprimi il contesto quando cresce

--max-turns Limita i turni nell'automazione

sonnet / haiku Usa modelli più economici per task semplici

Prompting Efficace

Sii specifico "Correggi il null check in auth.ts:42" > "correggi bug"

Fornisci contesto Cita file, funzioni, messaggi di errore

Usa @file Riferisci file direttamente: **@src/main.ts**

Usa immagini Trascina screenshot per contesto visuale

Itera Fai follow-up per raffinare; usa `/clear` per azzerare