CLAUDE CODE QUICK REFERENCE

Getting Started

Install & Launch

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

Authentication

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

Slash Commands

Session Commands

`/help`	Show available commands
`/clear`	Clear conversation history
`/compact`	Condense context to save tokens
`/cost`	Show token usage and cost
`/status`	Show session info and model
`/new`	Start a new conversation
`/config`	Open or view configuration
`/doctor`	Diagnose configuration issues
`/init`	Create a CLAUDE.md for the project
`/login`	Authenticate with Anthropic
`/logout`	Clear authentication

Workflow Commands

`/bug`	File a bug report
`/review`	Request a code review
`/pr`	Create or update a pull request
`/commit`	Commit staged changes with a message

Keyboard Shortcuts

`Ctrl+C`	Cancel current generation
`Ctrl+D`	Exit Claude Code (EOF)
`Escape`	Cancel current input / edit
`Tab`	Autocomplete file paths and commands
`Up/Down`	Navigate input history

CLI Flags

Common Flags

`-p, --print`	Non-interactive (headless) mode
`--model`	Set model: `opus`, `sonnet`, `haiku`
`--output-format`	Output as `text`, `json`, `stream-json`
`--allowedTools`	Restrict tools: `Edit`,`Read`,`Bash`
`--max-turns N`	Limit conversation turns
`--debug`	Enable debug logging
`-n, --name`	Name the session

Headless / Scripting

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

Configuration Files

CLAUDE.md Hierarchy

`./CLAUDE.md`	Project-level instructions (checked into repo)
`./.claude/settings.json`	Project settings (permissions, hooks)
`~/.claude/CLAUDE.md`	User-global instructions (all projects)
`~/.claude/settings.json`	User-global settings
`~/.claude/projects/*/CLAUDE.md`	Per-project user instructions (not in repo)

Environment Variables

`ANTHROPIC_API_KEY`	API key for direct authentication
`CLAUDE_MODEL`	Default model override
`CLAUDE_CONFIG_DIR`	Custom config directory path

Permissions

settings.json

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

Permission Modes

`default`	Ask before risky tools
`--dangerously-skip-permissions`	Allow all tools (CI/scripts only)
`--allowedTools`	CLI flag to restrict tool set

MCP Servers

What Are MCP Servers?

Model Context Protocol servers extend Claude Code with custom tools — databases, APIs, services. Managed via CLI or `.mcp.json`.

CLI Management

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

.mcp.json Config

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

Scope

`.mcp.json`	Project-level MCP servers
`~/.claude/mcp.json`	User-global MCP servers
`claude mcp add -s user`	Add to user scope
`claude mcp add -s project`	Add to project scope

Hooks

Hook Events

`PreToolUse`	Runs before a tool is executed
`PostToolUse`	Runs after a tool completes
`Notification`	Runs when Claude sends a notification
`Stop`	Runs when Claude finishes a response

settings.json Example

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

Hook Behavior

`exit 0`	Allow the tool to proceed
`exit 2`	Block the tool from running
`stdout`	JSON feedback to Claude

SDK & Automation

Headless Scripting

```
# 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
```

Tips

Use `--max-turns` to cap cost in automation. Use `--output-format json` to parse results programmatically. Combine with `--allowedTools` for safety.

Models

Model Selection

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

Model Shortcuts

`opus`	Claude Opus — highest capability
`sonnet`	Claude Sonnet — default, balanced
`haiku`	Claude Haiku — fast and economical
`--model full-name`	e.g. `claude-sonnet-4-6`

Best Practices

Writing CLAUDE.md

Put project conventions, tech stack, build commands, and test instructions in CLAUDE.md. Keep it concise — Claude reads it every session. Use `/init` to scaffold one.

Cost Management

`/cost`	Check running token usage
`/compact`	Compress context when it grows large
`--max-turns`	Cap turns in automation
`sonnet / haiku`	Use cheaper models for simple tasks

Effective Prompting

`Be specific`	"Fix the null check in auth.ts:42" > "fix bug"
`Give context`	Reference files, functions, error messages
`Use @file`	Reference files directly: `@src/main.ts`
`Use images`	Drag & drop screenshots for visual context
`Iterate`	Follow up to refine; use `/clear` to reset