

Claude Code Quick Reference

Shortcuts, slash commands, memory, MCP, hooks, agents, automation

Slash Commands

Daily Commands

<code>/compact [focus]</code>	Compress context; keep key memory
<code>/resume</code>	Resume or switch sessions
<code>/context</code>	Visualize context usage
<code>/memory</code>	Open or edit memory files
<code>/permissions</code>	View or change approval mode
<code>/model [model]</code>	Switch model
<code>/effort [level]</code>	Set reasoning effort

Project & Workflow

<code>/init</code>	Create a project CLAUDE.md
<code>/mcp</code>	Manage MCP servers
<code>/hooks</code>	Manage hooks
<code>/agents</code>	Manage agents
<code>/plan [desc]</code>	Enter plan mode
<code>/branch [name]</code>	Fork conversation / branch context
<code>/config</code>	Open settings

Memory & Project Files

Where Context Lives

CHOICE	USE WHEN	WATCH FOR
<code>./CLAUDE.md</code>	You want shared repo and instructions, build a command and project conventions	Keep it durable and concise, dump session chatter
<code>~/claude/CLAUDE.md</code>	You want personal defaults across many projects	Avoid specific rules that teammates should see
<code>/etc/claude-code/CLAUDE.md</code>	Your org needs managed instructions, users should not bypass	Too heavy for personal workflow notes, machine-remembers and corrects local facts over time
<code>auto memory</code>	Claude should remember and correct local facts over time	Machine-remembers and corrects local facts over time

Rules & Imports

<code>.claude/rules/*.md</code>	Project rules file set for more focused guidance
<code>@path/to/file</code>	Import files inside CLAUDE.md
<code>paths:</code>	Path-scoped rules in frontmatter

CLAUDE.md vs Auto Memory

CHOICE	USE WHEN	WATCH FOR
<code>CLAUDE.md</code>	You want durable instructions you author deliberately	Best for commands, repo structure, coding rules, and safety policy
<code>auto memory</code>	Claude should retain corrections and learned facts across sessions	Useful, but less predictable and not as portable

Permissions, Hooks & MCP

Permission Modes

CHOICE	USE WHEN	WATCH FOR
<code>plan</code>	You want research, planning, or architectural thinking without edits	Safest mode, but intentionally slower for execution
<code>default</code>	You want the normal interactive baseline	Expect prompts before riskier actions
<code>acceptEdits</code>	You want faster editing while still gating risky tools	Good speed trade-off, but still not full automation
<code>bypassPermissions</code>	You are in tightly trusted automation or fully controlled contexts	High risk; not a normal day-to-day setting

settings.json & Hooks

```
{
  "permissions": { "allow": ["Read", "Glob"],
    "deny": ["Bash(rm *)"] },
  "hooks": {
    "PreToolUse": [{
      "matcher": "Bash",
      "hooks": [{ "type": "command", "command":
        "check-command.sh $INPUT" }]
    }]
  }
}
```

Hook Events & Results

<code>PreToolUse</code>	Before tool execution
<code>PostToolUse</code>	After tool execution
<code>Stop</code>	At response completion
<code>exit 0</code>	Allow tool to proceed
<code>exit 2</code>	Block tool execution
<code>stdout JSON</code>	Structured feedback back to Claude

MCP Commands

```
claude mcp add my-server -- npx my-mcp-server
claude mcp add --transport http sentry https://
mcp.sentry.dev/mcp
claude mcp list
claude mcp get my-server
claude mcp remove my-server
claude mcp serve
```

MCP Scope

CHOICE	USE WHEN	WATCH FOR
<code>project</code>	The repo should share the same MCP server config	Put it in <code>~/.mcp.json</code> so the setup travels with the project
<code>user</code>	You want personal MCP servers across many repos	Lives in <code>~/.claude.json</code> ; teammates will not inherit it
<code>studio</code>	The server runs as a local process on your machine	Best for local tooling; install/runtime is your responsibility
<code>http / sse</code>	You connect to remote MCP servers over the network	Credentials, latency, and remote availability matter more

CLI & Automation

Core Commands

```
claude # interactive
claude "query" # start with prompt
claude -p "query" # headless
claude -c # continue last
claude -r "name" # resume named session
```

Key Flags

<code>-p</code>	Headless / print mode
<code>--output-format</code>	text / json / stream / stream-json
<code>--json-schema</code>	Constrain structured output
<code>--model</code>	Set model ID or shortcut
<code>--effort</code>	low / medium / high / max / auto
<code>--allowedTools</code>	Pre-approve restricted tool set
<code>--max-turns</code>	Cap turns in automation

Automation Examples

```
claude -p "summarize main.py" --output-format json
claude -p "list TODOs" --json-schema
todo.schema.json
git diff | claude -p "review this diff"
cat log.txt | claude -p "explain the failure"
```

Agents, Skills & Worktrees

Choose The Execution Path

CHOICE	USE WHEN	WATCH FOR
<code>/btw</code>	You want a side question using the current conversation context	No tools; it cannot inspect the repo or gather new facts
<code>Explore agent</code>	You want fast read-only investigation	Good for research, not normal editing
<code>General agent</code>	You want a delegated worker with normal tool access	Fresh context means you must specify the task clearly
<code>Skill</code>	You want a reusable workflow or parameterized prompt bundle	Better for repeated tasks than ad hoc questions
<code>/batch</code>	You want parallel worktree execution across subtasks	Higher overhead; best when the task splits cleanly

Worktrees & Skill Fields

<code>--worktree name</code>	Run session in dedicated git worktree
<code>isolation: worktree</code>	Agent frontmatter for worktree isolation
<code>background: true</code>	Run agent in background
<code>maxTurns</code>	Limit agentic turn count
<code>allowed-tools</code>	Pre-approve tools inside a skill
<code>context: fork</code>	Run the skill in a subagent
<code>\$ARGUMENTS</code>	User input placeholder in a skill

Keyboard & Input

High-Value Keys

<code>Ctrl+C</code>	Cancel current input or generation
<code>Ctrl+G</code>	Open the current prompt in your editor
<code>Ctrl+T</code>	Toggle the task list
<code>Shift+Tab</code>	Cycle permission modes
<code>\ + Enter</code>	Insert a quick newline
<code>/ / ! / @</code>	Command, shell, and file-path prefixes

Claude Code Quick Reference

Config & Defaults

Config Files

CHOICE	USE WHEN	WATCH FOR
<code>.claude/settings.json</code>	The project should share the same defaults	Commit only when the team should inherit the behavior
<code>.claude/settings.local.json</code>	You want local-only project overrides	Keep personal tweaks and secrets out of shared config

Context Moves

- `/compact` Compress context near capacity
- `/context` See token or context usage