CLAUDE CODE QUICK REFERENCE

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

Slash Commands

Daily Commands

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

Project & Workflow

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

Memory & Project Files

Where Context Lives

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

Rules & Imports

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

CLAUDE.md vs Auto Memory

CHOICE	USE WHEN	WATCH FOR
`CLAUDE.md`	You want durable instructions you author deliberately	Best for commands, repo structure, coding rules, and safety policy
`auto memory`	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
`plan`	You want research, planning, or architectural thinking without edits	Safest mode, but intentionally slower for execution
`default`	You want the normal interactive baseline	Expect prompts before riskier actions
`acceptEdits`	You want faster editing while still gating risky tools	Good speed trade-off, but still not full automation
`bypassPermissions`	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

`PreToolUse`	Before tool execution
`PostToolUse`	After tool execution
`Stop`	At response completion
`exit 0`	Allow tool to proceed
`exit 2`	Block tool execution
`stdout JSON`	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
`project`	The repo should share the same MCP server config	Put it in `.mcp.json` so the setup travels with the project
`user`	You want personal MCP servers across many repos	Lives in `~/.claude.json`; teammates will not inherit it
`stdio`	The server runs as a local process on your machine	Best for local tooling; install/runtime is your responsibility
`http / sse`	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

`-p`	Headless / print mode
`--output-format`	text / json / stream / stream-json
`--json-schema`	Constrain structured output
`--model`	Set model ID or shortcut
`--effort`	low / medium / high / max / auto
`--allowedTools`	Pre-approve restricted tool set
`--max-turns`	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
`/btw`	You want a side question using the current conversation context	No tools; it cannot inspect the repo or gather new facts
`Explore agent`	You want fast read-only investigation	Good for research, not normal editing
`General agent`	You want a delegated worker with normal tool access	Fresh context means you must specify the task clearly
`Skill`	You want a reusable workflow or parameterized prompt bundle	Better for repeated tasks than ad hoc questions
`/batch`	You want parallel worktree execution across subtasks	Higher overhead; best when the task splits cleanly

Worktrees & Skill Fields

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

Keyboard & Input

High-Value Keys

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

Config & Defaults

Config Files

CHOICE	USE WHEN	WATCH FOR
`.claude/settings.json`	The project should share the same defaults	Commit only when the team should inherit the behavior
`.claude/settings.local.json`	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