

Claude Code Commands Reference

Quick Reference Card

Starting Claude Code

```
# Start interactive session in current directory
claude

# Start with a specific prompt
claude -p "Explain this codebase"

# Resume last conversation
claude --continue

# Start fresh (ignore history)
claude --no-history

# Enable MCP debug output
claude --mcp-debug
```

Slash Commands

Session Management

Command	Description
/help	Show all available commands
/quit or /exit	End the session

Command	Description
<code>/clear</code>	Clear conversation history
<code>/compact</code>	Summarize conversation to save tokens

Code Operations

Command	Description
<code>/commit</code>	Stage and commit changes with AI message
<code>/pr</code>	Create pull request with AI description
<code>/review</code>	Review recent code changes
<code>/diff</code>	Show uncommitted changes

Context & Planning

Command	Description
<code>/init</code>	Create CLAUDE.md for current project
<code>/agents</code>	List available subagents
<code>/permissions</code>	View/modify tool permissions
<code>/config</code>	View current configuration

Custom Commands

Command	Description
<code>/[your-command]</code>	Run custom command from <code>.claude/commands/</code>

@ Mentions (Context References)

Use `@` followed by Tab to autocomplete:

```
@filename.js      # Reference a specific file
@src/             # Reference a directory
@package.json     # Reference config files
@https://example.com # Reference a URL
```

Examples

```
> Explain @src/auth/middleware.js

> The bug is in @api/users.ts around line 45

> Follow the patterns in @src/services/
```

Keyboard Shortcuts

Shortcut	Action
Tab	Autocomplete file/folder names
Ctrl+C	Cancel current operation
Ctrl+D	Exit session
↑ / ↓	Navigate command history
Ctrl+L	Clear screen

Permission Modes

Default (Recommended)

Claude asks before: - Writing/editing files - Running shell commands - Making git commits

Allow Specific Tools

```
claude --allowedTools="Read,Write,Bash"
```

Dangerous Mode (Use with caution!)

```
claude --dangerously-skip-permissions
```

⚠️ Only use in isolated/containerized environments

Thinking Modes

Trigger extended reasoning with these phrases:

Phrase	Thinking Budget	Use Case
"think"	Low (~5s)	Simple questions
"think hard"	Medium (~15s)	Design decisions
"think harder"	High (~30s)	Complex architecture
"ultrathink"	Maximum (~60s)	Critical problems

Example

```
> Think harder about how we should structure  
the database schema for multi-tenancy.
```

Built-in Subagents

Agent	Model	Purpose
Explore	Haiku	Fast, read-only codebase search

Agent	Model	Purpose
Plan	Sonnet	Research and planning
General	Sonnet	Complex multi-step tasks

Invoke Subagent

```
> Use the explore agent to find all authentication code

> Have the plan agent outline a refactoring strategy
```

Configuration Files

CLAUDE.md Locations

Location	Scope	Priority
./CLAUDE.md	Project (checked in)	Highest
./.claude/CLAUDE.md	Project (local)	High
~/.claude/CLAUDE.md	User global	Low

Custom Commands Location

```
.claude/commands/
"my-command.md"
"code-review.md"
"test-gen.md"
```

MCP Server Configuration

```
./mcp.json           # Project (shared)
./claude/mcp_servers.json # Project (local)
~/.claude/mcp_servers.json # Global
```

Common Workflows

Explore → Plan → Code → Commit

```
1. > Give me an overview of this codebase
2. > Plan how to add user preferences feature
3. > Implement the plan, starting with the database schema
4. > /commit
5. > /pr
```

Test-Driven Development

```
1. > Write failing tests for email validation
2. > /commit -m "test: add email validation tests"
3. > Implement email validation to pass the tests
4. > /commit -m "feat: add email validation"
```

Bug Fix Workflow

```
1. > Find where the login timeout is configured
2. > Think hard about why timeouts might be occurring
3. > Fix the issue following existing patterns
4. > Write a regression test
5. > /commit
```

Git Integration

Commit with AI Message

```
> /commit
# Claude analyzes changes and suggests message
# You approve or edit
```

Create PR

```
> /pr
# Claude creates PR with description
# Opens in browser for final edits
```

Using gh CLI

```
> Use gh to list open issues assigned to me

> Create an issue for the bug we just found
```

Headless Mode (CI/Automation)

```
# Run single prompt, output to stdout
claude -p "List all TODO comments" --output-format json

# Pipe input
echo "Review this code" | claude

# Use in scripts
claude -p "Generate changelog for v2.0" > CHANGELOG.md
```

MCP Commands

```
# List available MCP connectors
claude mcp list

# Get info about a connector
claude mcp info github

# Add a connector
claude mcp add github

# Debug MCP issues
claude --mcp-debug
```

Troubleshooting

Issue	Solution
"command not found: claude"	Add npm global bin to PATH
Authentication fails	Run <code>claude login</code> again
Claude doesn't see files	Ensure you're in project root
Slow responses	Use <code>/compact</code> to reduce context
Changes not applying	Check if you approved the diff
MCP connector errors	Run with <code>--mcp-debug</code>

CLAUDE.md Template

```
# Project: [Name]

## Commands
```

- `npm run dev` - Start dev server
- `npm test` - Run tests
- `npm run build` - Production build

Tech Stack

- Node.js 20 + TypeScript
- PostgreSQL with Prisma
- Jest for testing

Conventions

- Use async/await, never callbacks
- Prefer named exports
- All functions need JSDoc comments

Do

- Run tests after code changes
- Use conventional commit messages
- Keep functions under 50 lines

Don't

- Use `any` type
- Commit console.log statements
- Modify database directly

Quick Tips

1. **Be specific** - "Add validation" â†’ "Add email regex validation to signup form"
 2. **Use @ mentions** - Reference files directly for accurate context
 3. **Approve carefully** - Always review diffs before accepting
 4. **Iterate** - Start broad, then refine with follow-up prompts
 5. **Use /compact** - Long sessions slow down; compact periodically
 6. **Commit often** - Smaller commits = easier to review and revert
-

