SuperClaude Command Reference 📋

Quick reference for win-claude-code with SuperClaude v3.0

🛠️ Development Commands

/analyze - Deep code analysis and investigation
Usage: `/analyze components/ --depth 3 --focus architecture`

/build - Compile and bundle projects with optimization
Usage: `/build --type prod --clean --optimize`

/dev-setup - Initialize development environment and dependencies
Usage: `/dev-setup --framework react --typescript`

/design - System architecture and component design planning
Usage: `/design "authentication system with OAuth"`

/improve - Code enhancement, refactoring, and optimization
Usage: `/improve components/Card.tsx --performance --shadcn-ui`

/troubleshoot - Debug issues and provide solutions
Usage: `/troubleshoot test-validation.js --verbose --suggest-fixes`

📋 Project Management Commands

/task - Multi-session project and feature management
Usage: `/task create "User Authentication" --epic --breakdown`

/estimate - Project complexity and time estimation
Usage: `/estimate "dashboard redesign" --detailed --resources`

/migrate - Data and code migration assistance
Usage: `/migrate --from "class components" --to "hooks"`

/deploy - Production deployment guidance and automation
Usage: `/deploy --platform vercel --environment production`

🔍 Quality & Testing Commands

/scan - Security vulnerabilities and code quality analysis
Usage: `/scan --security --performance --dependencies`

`/test` - Test execution, generation, and coverage analysis
Usage: `/test --all --coverage --e2e --playwright`

`/review` - Comprehensive code review and recommendations
Usage: `/review --comprehensive --security --performance`

📚 Documentation Commands

`/document` - Generate technical documentation and guides
Usage: `/document --api --examples --deployment-guide`

`/git` - Intelligent git operations and commit management
Usage: `/git commit "feature: add user authentication"`

🎭 Smart Personas (Auto-Activating)

`@architect` - Systems design and architecture specialist
Usage: `@architect "Design microservices architecture"`

`@frontend` - UI/UX specialist with shadcn-ui integration
Usage: `@frontend "Create modern dashboard with shadcn components"`

`@backend` - Server-side and API development expert
Usage: `@backend "Design REST API for user management"`

`@security` - Threat modeling and vulnerability assessment
Usage: `@security "Audit authentication implementation"`

`@analyzer` - Root cause analysis and investigation specialist
Usage: `@analyzer "Investigate performance bottlenecks"`

`@qa` - Quality assurance and testing strategy expert
Usage: `@qa "Generate test cases for payment processing"`

`@performance` - Speed optimization and resource management
Usage: `@performance "Optimize database queries and caching"`

`@refactorer` - Code quality and technical debt management
Usage: `@refactorer "Clean up legacy component architecture"`

`@devops` - Infrastructure and deployment automation
Usage: `@devops "Set up CI/CD pipeline with GitHub Actions"`

`@mentor` - Educational guidance and knowledge transfer
Usage: `@mentor "Explain React hooks best practices"`

`@scribe` - Professional documentation and technical writing
Usage: `@scribe "Create user onboarding documentation"`

🚀 MCP Server Integration

shadcn-ui - Professional React components and design blocks
Usage: `"Create card component with shadcn/ui"`

context7 - Official library documentation and patterns
Usage: `"Show me Next.js 14 app router patterns"`

sequential - Complex multi-step analysis and planning
Usage: `"Analyze entire codebase architecture systematically"`

playwright - Browser automation and E2E testing
Usage: `"Test user registration flow across browsers"`

magic - Custom UI component generation (fallback)
Usage: `"Create animated loading spinner"`

⚡ Power User Features

Command Chaining - Execute multiple commands in sequence
Usage: `/analyze && /improve --last-analysis && /test --changed`

Thinking Modes - Enhanced analysis depth

- `--think` (4K context): `/analyze --think components/`
- `--think-hard` (10K): `/design --think-hard "system architecture"`
- `--ultrathink` (32K): `/review --ultrathink --comprehensive`

Context Management - Optimize token usage

- `/load project-name --essential --exclude node_modules`
- `--compress --smart` (when hitting limits)

Multi-Session - Continue work across sessions
Usage: `/task continue "Feature Development" --restore-context`

🔧 Essential Flags

`--verbose` - Detailed output and explanations
`--clean` - Clean build or fresh start
`--optimize` - Enable performance optimizations
`--security` - Focus on security aspects
`--performance` - Focus on speed and efficiency
`--shadcn-ui` - Use shadcn/ui components
`--comprehensive` - Full analysis or review
`--e2e` - End-to-end testing
`--production` - Production environment settings

🧮 Quick Actions for Your Projects

Fix Test Suite

`/troubleshoot test-*.js --fix-suggestions && /improve tests/ --coverage`

Modern UI Upgrade

`@frontend "Replace components with shadcn/ui equivalents"`

Security Audit

`@security "Full security review" && /scan --vulnerabilities`

Performance Boost

`@performance "Optimize rendering" && /build --analyze`

New Feature

`@architect "Design feature" && /task create --breakdown`

🆘 Troubleshooting

Commands not working? → Check with `/help` or `@help`
Git issues? → Run git wrapper: `C:\claude\superclaude-tools\git-ultimate-wrapper.bat`
Context limits? → Use `--compress --smart`
Server errors? → Restart Claude Desktop

SuperClaude v3.0 | Windows Edition | Your AI Development Partner 🚀

Print this guide for quick reference during development sessions