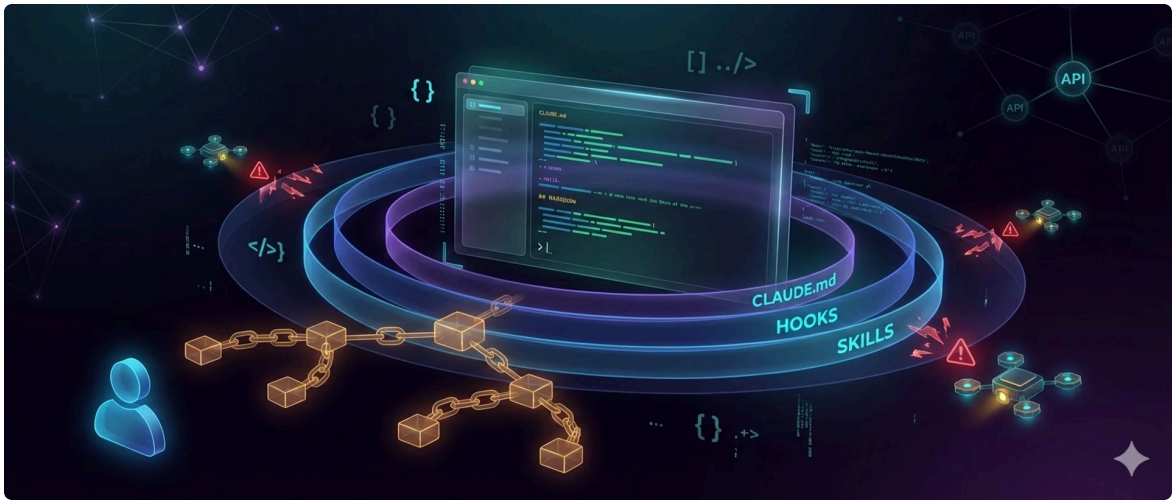

Claude Code V3 Guide

LSP, CLAUDE.md, MCP, Skills & Hooks

JANUARY 15, 2026

THEDECIPHERIST.COM

TABLE OF CONTENTS

🎉 V3: BUILT ON COMMUNITY FEEDBACK (AGAIN)

V2 hit #2 all-time on r/ClaudeAI. Your comments made V3 possible. Huge thanks to u/BlueVajra (commands/skills merge), u/stratofax (dotfiles sync), u/antoniocs (MCP tradeoffs), u/GeckoLogic (LSP), and everyone from V2: u/headset38, u/tulensrma, u/jcheroske.

What's new in V3:

- **Part 8: LSP** – Claude now has IDE-level code intelligence (900x faster navigation)
- **Commands & Skills merged** – Same schema, simpler mental model
- **Expanded MCP directory** – 25+ recommended servers by category
- **MCP tradeoffs** – When NOT to use MCP servers
- **Dotfiles sync** – Keep ~/.claude consistent across machines
- GitHub repo with templates, hooks, and skills

TL;DR: Your global `~/.claude/CLAUDE.md` is a security gatekeeper AND project blueprint. **LSP gives Claude semantic code understanding** – go-to-definition, find-references, diagnostics. MCP servers extend capabilities (but have tradeoffs). Commands and skills now share the same schema. **Hooks enforce rules deterministically** where CLAUDE.md can fail. And research shows mixing topics causes **39% performance degradation** – keep chats focused.

PART 1: THE GLOBAL CLAUDE.MD AS SECURITY GATEKEEPER

The Memory Hierarchy

Claude Code loads CLAUDE.md files in a specific order:

LEVEL	LOCATION	PURPOSE
Enterprise	`/etc/claude-code/CLAUDE.md`	Org-wide policies
Global User	`~/.claude/CLAUDE.md`	Your standards for ALL projects
Project	`./CLAUDE.md`	Team-shared project instructions
Project Local	`./CLAUDE.local.md`	Personal project overrides

Your global file applies to **every single project** you work on.

What Belongs in Global

1. Identity & Authentication

```
## GitHub Account
**ALWAYS** use **YourUsername** for all projects:
- SSH: `git@github.com:YourUsername/<repo>.git`

## Docker Hub
Already authenticated. Username in `~/.env` as `DOCKER_HUB_USER`
```

Why global? You use the same accounts everywhere. Define once, inherit everywhere.

2. The Gatekeeper Rules

```
## NEVER EVER DO

These rules are ABSOLUTE:

### NEVER Publish Sensitive Data
- NEVER publish passwords, API keys, tokens to git/npm/docker
- Before ANY commit: verify no secrets included

### NEVER Commit .env Files
- NEVER commit `.env` to git
- ALWAYS verify `.env` is in `.gitignore`
```

Why This Matters: Claude Reads Your .env

Security researchers discovered that Claude Code **automatically reads `.env` files** without explicit permission. Backslash Security warns:

> *"If not restricted, Claude can read `.env`, AWS credentials, or `secrets.json` and leak them through 'helpful suggestions.'"*

Your global CLAUDE.md creates a **behavioral gatekeeper** — even if Claude has access, it won't output secrets.

Syncing Global CLAUDE.md Across Machines

Thanks to u/stratofax for this tip.

If you work on multiple computers, sync your `~/.claude/` directory using a dotfiles manager:

```
# Using GNU Stow
cd ~/dotfiles
stow claude  # Symlinks ~/.claude to dotfiles/claude/.claude
```

This gives you:

- Version control on your settings
- Consistent configuration everywhere
- Easy recovery if something breaks

Defense in Depth

LAYER	WHAT	HOW
1	Behavioral rules	Global CLAUDE.md "NEVER" rules
2	Access control	Deny list in settings.json
3	Git safety	.gitignore

Team Workflows: Evolving CLAUDE.md

Boris Cherny shares how Anthropic's Claude Code team does it:

> *"Our team shares a single CLAUDE.md for the Claude Code repo. We check it into git, and the whole team contributes multiple times a week."*

The pattern: Mistakes become documentation.

```
Claude makes mistake → You fix it → You add rule to CLAUDE.md → Never happens a
```

Compounding Engineering

This embodies Compounding Engineering:

> *"Each unit of engineering work should make subsequent units easier."*

The 80/20 inversion: Spend 80% on planning and review, 20% on execution. Your CLAUDE.md becomes institutional knowledge that compounds over time.

PART 2: GLOBAL RULES FOR NEW PROJECT SCAFFOLDING

Your global CLAUDE.md becomes a **project factory**. Every new project automatically inherits your standards.

The Problem Without Scaffolding Rules

Research from project scaffolding experts:

> *"LLM-assisted development fails by silently expanding scope, degrading quality, and losing architectural intent."*

The Solution

```
## New Project Setup

When creating ANY new project:

### Required Files
- `.env` — Environment variables (NEVER commit)
- `.env.example` — Template with placeholders
- `.gitignore` — Must include: .env, node_modules/, dist/
- `CLAUDE.md` — Project overview

### Required Structure
project/
├── src/
├── tests/
├── docs/
├── .claude/skills/
└── scripts/

### Node.js Requirements
Add to entry point:
process.on('unhandledRejection', (reason, promise) => {
  console.error('Unhandled Rejection:', reason);
  process.exit(1);
});
```

When you say "create a new Node.js project," Claude reads this and **automatically** creates the correct structure. Zero manual setup.

PART 3: MCP SERVERS — CLAUDE'S INTEGRATIONS

MCP (Model Context Protocol) lets Claude interact with external tools.

Adding MCP Servers

```
claude mcp add <server-name> -- <command>
claude mcp list
claude mcp remove <server-name>
```

When NOT to Use MCP

Thanks to u/antoniocs for this perspective.

MCP servers consume tokens and context. For simple integrations, consider alternatives:

USE CASE	MCP OVERHEAD	ALTERNATIVE
Trello tasks	High	CLI tool (`trello-cli`)
Simple HTTP calls	Overkill	`curl` via Bash
One-off queries	Wasteful	Direct command

Rule of thumb: If you're calling an MCP tool once per session, a CLI is more efficient. MCP shines for *repeated* tool use within conversations.

Recommended MCP Servers for Developers

Core Development

SERVER	PURPOSE	INSTALL
Context7	Live docs for any library	`claude mcp add context7 -- npx -y @upstash/context7-mcp@latest`
GitHub	PRs, issues, CI/CD	`claude mcp add github -- npx -y @modelcontextprotocol/server-github`
Filesystem	Advanced file operations	`claude mcp add filesystem -- npx -y @modelcontextprotocol/server-filesystem`
Sequential Thinking	Structured problem-solving	`claude mcp add sequential-thinking -- npx -y @modelcontextprotocol/server-sequential-thinking`

Databases

SERVER	PURPOSE	INSTALL
MongoDB	Atlas/Community, Performance Advisor	`claude mcp add mongodb -- npx -y mongodb-mcp-server`
PostgreSQL	Query Postgres naturally	`claude mcp add postgres -- npx -y @modelcontextprotocol/server-postgres`
DBHub	Universal (MySQL, SQLite, etc.)	`claude mcp add db -- npx -y @bytebase/dbhub`

Documents & RAG

SERVER	PURPOSE	INSTALL
Docling	PDF/DOCX parsing, 97.9% table accuracy	`claude mcp add docling -- uvx docling-mcp-server`
Qdrant	Vector search, semantic memory	`claude mcp add qdrant -- npx -y @qdrant/mcp-server`
Chroma	Embeddings, vector DB	`claude mcp add chroma -- npx -y @chroma/mcp-server`

Browser & Testing

SERVER	PURPOSE	INSTALL
Playwright	E2E testing, scraping	`claude mcp add playwright -- npx -y @anthropic-ai/playwright-mcp`
Browser MCP	Use your logged-in Chrome	browsermcp.io
Brave Search	Privacy-first web search	`claude mcp add brave -- npx -y @anthropic-ai/brave-search-mcp`

Cloud & Hosting

SERVER	PURPOSE	INSTALL
AWS	Full AWS service access	`claude mcp add aws -- uvx awslabs.aws-api-mcp-server@latest`
Cloudflare	Workers, KV, R2	`claude mcp add cloudflare -- npx -y @cloudflare/mcp-server`
Hostinger	Domains, DNS, VMs, billing	`npm i -g hostinger-api-mcp` then configure
Kubectl	Kubernetes natural language	`claude mcp add kubectl -- npx -y @modelcontextprotocol/server-kubernetes`

Workflow & Communication

SERVER	PURPOSE	INSTALL
Slack	Messages, channel summaries	`claude mcp add slack -- npx -y @anthropic-ai/slack-mcp`
Linear	Issue tracking	`claude mcp add linear -- npx -y @linear/mcp-server`
Figma	Design specs, components	`claude mcp add figma -- npx -y @anthropic-ai/figma-mcp`

Discovery

Find more servers:

- awesome-mcp-servers – 76k+ stars, hundreds of servers
- mcpservers.org – Searchable directory
- Claude Market – Curated marketplace

PART 4: CONTEXT7 — LIVE DOCUMENTATION

Context7 gives Claude access to **up-to-date documentation**.

The Problem

Claude's training has a cutoff. Ask about a library released after training → outdated answers.

The Solution

```
You: "Using context7, show me the Next.js 15 cache API"
Claude: *fetches current docs* → accurate, up-to-date code
```

Installation

```
claude mcp add context7 -- npx -y @upstash/context7-mcp@latest
```

PART 5: SKILLS (COMMANDS ARE NOW SKILLS)

Thanks to u/BlueVajra for the correction.

Update: As of late 2025, **commands and skills have been merged**. They now share the same schema.

> *"Merged slash commands and skills, simplifying the mental model with no change in behavior." — Claude Code Changelog*

The New Structure

OLD LOCATION	NEW LOCATION
`~/.claude/commands/review.md`	`~/.claude/skills/review/SKILL.md`

Key Difference

- **Slash commands** (`/review`) – You explicitly invoke them
- **Skills** – Claude can trigger automatically based on context

Both use the same SKILL.md format:

```
---
name: review
description: Review code for bugs and security issues
---

# Code Review Skill

When reviewing code:
1. Check for security vulnerabilities
2. Look for performance issues
3. Verify error handling
```

Progressive Disclosure

Skills use **progressive disclosure** for token efficiency:

1. **Startup**: Only name/description loaded
2. **Triggered**: Full SKILL.md content loaded
3. **As needed**: Additional resources loaded

Rule of thumb: If instructions apply to <20% of conversations, make it a skill instead of putting it in CLAUDE.md.

PART 6: WHY SINGLE-PURPOSE CHATS ARE CRITICAL

Research consistently shows mixing topics destroys accuracy.

Studies on multi-turn conversations:

> *"An average **39% performance drop** when instructions are delivered across multiple turns."*

Chroma Research on context rot:

> *"As tokens in the context window increase, the model's ability to accurately recall information decreases."*

The Golden Rule

> *"One Task, One Chat"*

SCENARIO	ACTION
New feature	New chat
Bug fix (unrelated)	`/clear` then new task
Research vs implementation	Separate chats
20+ turns elapsed	Start fresh

Use `/clear` Liberally

```
/clear
```

Anthropic recommends:

> *"Use `/clear` frequently between tasks to reset the context window."*

PART 7: HOOKS — DETERMINISTIC ENFORCEMENT

This section added based on V2 feedback from u/headset38 and u/tulensrma.

CLAUDE.md rules are **suggestions** Claude can ignore under context pressure. Hooks are **deterministic** — they always run.

The Critical Difference

MECHANISM	TYPE	RELIABILITY
CLAUDE.md rules	Suggestion	Can be overridden
Hooks	**Enforcement**	Always executes

Hook Events

EVENT	WHEN	USE CASE
`PreToolUse`	Before tool executes	Block dangerous ops
`PostToolUse`	After tool completes	Run linters
`Stop`	Claude finishes turn	Quality gates

Example: Block Secrets Access

Add to `~/.claude/settings.json`:

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Read|Edit|Write",
        "hooks": [{
          "type": "command",
          "command": "python3 ~/.claude/hooks/block-secrets.py"
        }]
      }
    ]
  }
}
```

The hook script:

```
#!/usr/bin/env python3
import json, sys
from pathlib import Path

SENSITIVE = {'.env', '.env.local', 'secrets.json', 'id_rsa'}

data = json.load(sys.stdin)
file_path = data.get('tool_input', {}).get('file_path', '')

if Path(file_path).name in SENSITIVE:
    print(f"BLOCKED: Access to {file_path} denied.", file=sys.stderr)
    sys.exit(2)  # Exit 2 = block and feed stderr to Claude

sys.exit(0)
```

Hook Exit Codes

CODE	MEANING
0	Allow operation
1	Error (shown to user)
2	**Block operation, tell Claude why**

PART 8: LSP — IDE-LEVEL CODE INTELLIGENCE

Thanks to u/GeckoLogic for highlighting this.

New in December 2025 (v2.0.74), Claude Code gained native Language Server Protocol support. This is a game-changer.

What LSP Enables

LSP gives Claude the same code understanding your IDE has:

CAPABILITY	WHAT IT DOES
Go to Definition	Jump to where any symbol is defined
Find References	See everywhere a function is used
Hover	Get type signatures and docs
Diagnostics	Real-time error detection
Document Symbols	List all symbols in a file

Why This Matters

Before LSP, Claude used **text-based search** (grep, ripgrep) to understand code. Slow and imprecise.

With LSP, Claude has **semantic understanding** – it knows that `getUserById` in file A calls the function defined in file B, not just that the text matches.

Performance: 900x faster (50ms vs 45 seconds for cross-codebase navigation)

Supported Languages

Python, TypeScript, Go, Rust, Java, C/C++, C#, PHP, Kotlin, Ruby, HTML/CSS

Setup

LSP is built-in as of v2.0.74. For older versions:

```
export ENABLE_LSP_TOOL=1
```

What This Means for You

Claude can now:

- Navigate massive codebases instantly
- Find all usages before refactoring
- Catch type errors in real-time
- Understand code structure semantically

This shifts AI coding from **text manipulation** to **semantic understanding**.

QUICK REFERENCE

TOOL	PURPOSE	LOCATION
Global CLAUDE.md	Security + Scaffolding	`~/.claude/CLAUDE.md`
Project CLAUDE.md	Architecture + Team rules	`./CLAUDE.md`
MCP Servers	External integrations	`claude mcp add`
Context7	Live documentation	MCP server
Skills	**Reusable expertise**	`.claude/skills/*/SKILL.md`
Hooks	**Deterministic enforcement**	`~/.claude/settings.json`
LSP	**Semantic code intelligence**	Built-in (v2.0.74+)
`/clear`	Reset context	Type in chat

GITHUB REPO

All templates, hooks, and skills:

github.com/TheDecipherist/claude-code-mastery

- CLAUDE.md templates (global + project)
- Ready-to-use hooks (block-secrets.py, etc.)
- Example skills
- settings.json pre-configured

SOURCES

- Claude Code Best Practices – Anthropic
- Effective Context Engineering – Anthropic
- Model Context Protocol – Anthropic
- Agent Skills – Anthropic
- Claude Code LSP Setup – AI Free API
- Claude Code December 2025 Update – Geeky Gadgets
- MongoDB MCP Server – MongoDB
- Hostinger MCP Server – Hostinger
- Docling MCP – IBM Research
- awesome-mcp-servers – GitHub
- Context Rot Research – Chroma
- LLMs Get Lost In Multi-Turn – arXiv
- Claude Code Hooks Guardrails – Paddo.dev
- Claude loads secrets without permission – Knostic
- Compound Engineering – Every

What's in your setup? Drop your hooks, skills, and MCP configs below.