Claude Code Starter Kit

23 Commands. 9 Hooks. 10 Rules. 12+ Profiles. One Template.

FEBRUARY 13, 2026

THEDECIPHERIST.COM

TABLE OF CONTENTS

23 slash commands. 9 hooks. 10 rules. 2 agents. 2 skills. 12+ project profiles. One template.

THE PROBLEM

You've read the guides. You know you need a `CLAUDE.md`, hooks for secret prevention, database connection pooling, quality gates, testing structure. But every new project starts the same way: 45 minutes of setup before you write a single line of business logic. And if you skip the setup, Claude creates `new MongoClient()` in six different files, commits your `.env`, and pushes untested Docker images to production.

WHAT THIS IS

The Claude Code Starter Kit is a ready-to-use project template based on Claude Code Mastery Guides V3-V5. Clone it, run `/setup`, and start building. Every best practice is already wired in.

It's not a runnable application — it's the scaffold that makes every application you build with Claude better from the first prompt.

THREE WAYS TO USE IT

1. **Scaffold** — `"/new-project my-app clean"` creates a new project with full infrastructure
2. **Convert** — `"/convert-project-to-starter-kit ~/existing-project"` merges the kit into an existing project without overwriting your code
3. **Customize** — Clone the repo, modify commands/hooks/skills, use as your own template source

---

WHAT'S INSIDE

23 Slash Commands

On-demand tools you invoke directly in Claude Code:

Setup & Scaffolding:

COMMAND	WHAT IT DOES
`/help`	Overview of all available commands and features
`/quickstart`	Fastest path from clone to building
`/install-global`	Install/merge global Claude config into `~/.claude/` (one-time)
`/setup`	Interactive .env configuration — GitHub, database, Docker, analytics
`/new-project`	Scaffold a new project from any of 12+ profiles
`/convert-project-to-starter-kit`	Merge the kit into an existing project

Code Quality:

COMMAND	WHAT IT DOES
`/review`	Code review with security, performance, and type safety checks
`/refactor`	Audit + refactor a file against all CLAUDE.md rules
`/security-check`	Scan for exposed secrets and security issues
`/test-plan`	Generate structured test plans

Development:

COMMAND	WHAT IT DOES
`/commit`	Smart commit with conventional commit format
`/progress`	Real-time project status from filesystem state
`/diagram`	Generate architecture, API, database, and infrastructure diagrams from actual code
`/architecture`	Display system architecture and data flow

Infrastructure:

COMMAND	WHAT IT DOES
`/optimize-docker`	Audit Dockerfile against 12 best practices
`/create-e2e`	Create Playwright E2E tests with explicit success criteria
`/create-api`	Scaffold a full API endpoint — route, handler, types, tests — wired into the server
`/worktree`	Create isolated branch + worktree for a task

Project Management:

COMMAND	WHAT IT DOES
`/set-project-profile-default`	Set the default profile for new projects
`/add-project-setup`	Add setup steps to the project registry
`/projects-created`	List all projects scaffolded from this kit
`/remove-project`	Remove a project from the registry

Monitoring:

COMMAND	WHAT IT DOES
`/what-is-my-ai-doing`	Live monitor — tokens, cost, violations, tool usage in real-time

10 Critical Rules (CLAUDE.md)

Battle-tested rules that prevent the most common Claude Code failures:

- **Never publish sensitive data** — No secrets in git, Docker, or anywhere online
- **TypeScript always** — Strict mode, no `any`. When editing `.js` files, convert to TypeScript first
- **API versioning** — Every endpoint uses `/api/v1/` prefix. No exceptions
- **Database wrapper only** — All database access through `src/core/db/index.ts`. No rogue `new MongoClient()` anywhere
- **Explicit test success criteria** — Every E2E test must have URL, element visibility, and content assertions. "Page loads" is not a test
- **Never hardcode credentials** — All secrets in `.env`, all configuration via environment variables
- **Quality gates** — No file > 300 lines. No function > 50 lines. TypeScript strict. All tests pass
- **Auto-branching on main** — Claude never touches `main`. Every code-modifying command auto-creates a feature branch
- **Parallelize independent awaits** — Claude must use `Promise.all` when multiple `await` calls don't depend on each other
- **Docker push gate** — When enabled, blocks `docker push` until the image is built, run locally, and health-checked

9 Deterministic Hooks

CLAUDE.md rules are suggestions. Hooks are guarantees — they run automatically, every time:

PreToolUse (blocks actions before they happen):

HOOK	WHAT IT DOES
`block-secrets.py`	Blocks Claude from reading `.env`, credentials, SSH keys
`check-rybbit.sh`	Blocks deploys without analytics setup
`check-branch.sh`	Prevents direct commits to main
`check-ports.sh`	Detects port conflicts before starting servers
`check-e2e.sh`	Blocks pushes without E2E test coverage

PostToolUse (validates after actions):

HOOK	WHAT IT DOES
`lint-on-save.sh`	Auto-validates TypeScript/ESLint after every file write

Stop (runs when Claude's session ends):

HOOK	WHAT IT DOES
`verify-no-secrets.sh`	Scans staged files for credentials
`check-rulecatch.sh`	Reports rule violations from the session
`check-env-sync.sh`	Warns on `.env` / `.env.example` misalignment

2 Skills (Automatic Template Loading)

Skills trigger automatically based on keywords in your prompts:

- **Code Review Skill** – Triggered by "review" or "audit". Loads a systematic 7-point checklist covering security, types, error handling, performance, testing, database patterns, and API versioning

- **Create Service Skill** – Triggered by "create service". Scaffolds microservices with server/handlers/adapters pattern

2 Custom Agents

Specialists that Claude delegates to automatically:

- **Code Reviewer** – Read-only agent for security audits, type safety, and quality checks. No write permissions – can't accidentally "fix" things during review
- **Test Writer** – Writes tests with explicit assertions and proper structure. Not just "passes" – real success criteria

Database Wrapper (`src/core/db/index.ts`)

A production-grade MongoDB wrapper that enforces every database best practice:

```
import { queryOne, queryMany, insertOne, updateOne, bulkOps } from '@/core/db/i

// Singleton pool — one connection per URI, prevents connection exhaustion
// NoSQL injection sanitization — automatic on ALL inputs
// $limit BEFORE $lookup — enforced automatically
// Graceful shutdown — idempotent, wired to SIGTERM/SIGINT/uncaught errors
```

- **3 pool presets**: `high` (20 connections), `standard` (10), `low` (5)
- **Automatic sanitization**: Strips `$` operators and `.` path traversal from user input
- **Aggregation-only reads, BulkWrite-only writes**: Consistent patterns, no mixed approaches
- **E11000 concurrent upsert auto-retry**: Handles race conditions automatically
- **Transaction and change stream support**: Built in when you need them
- **Next.js hot-reload safe**: Connection pool persists via `globalThis`

12+ Project Profiles

Pre-configured stacks via `/new-project`:

PROFILE	STACK
`clean`	Claude infrastructure only — zero coding opinions
`default`	Next.js + TypeScript + MongoDB + Docker
`go`	Go with Gin/Chi/Echo, standard layout, multi-stage Docker
`vue`	Vue.js frontend
`nuxt`	Nuxt full-stack
`svelte`	Svelte frontend
`sveltekit`	SvelteKit full-stack
`angular`	Angular frontend
`python-api`	Python REST API
`django`	Django full-stack
`flask`	Flask API
`api`	REST API focused

```
/new-project my-app go             # Go API with standard layout
/new-project my-app clean          # Just the Claude infrastructure
/new-project my-app sveltekit      # SvelteKit full-stack app
```

Test Infrastructure

- **`CHECKLIST.md`** — Master test status tracker with visual pass/fail indicators
- **`ISSUES_FOUND.md`** — User-guided testing log for manual QA
- **Playwright E2E Config** — Auto-spawns test servers on fixed ports (4000, 4010, 4020)
- **Test Query Master** (`scripts/db-query.ts`) — Centralized dev/test database queries, prevents script scatter

Content Builder

Markdown-to-HTML article builder (`scripts/build-content.ts`) with:

- Config-driven article registry
- SEO optimization (Open Graph, Twitter Cards, Schema.org)
- Syntax highlighting and table of contents generation
- Parent/child article relationships

Documentation Templates

Pre-structured docs that Claude actually follows:

- **`ARCHITECTURE.md`** – System overview with authority boundaries and "STOP" scope-creep preventers
- **`INFRASTRUCTURE.md`** – Deployment topology, environments, prerequisites
- **`DECISIONS.md`** – Architectural Decision Records with rationale

MCP Servers

Pre-configured integrations:

- **Context7** – Live documentation lookup for any library
- **GitHub** – PR management and issue tracking from Claude
- **Playwright** – E2E test debugging and browser automation
- **ClassMCP** – Semantic CSS patterns (auto-included in CSS-enabled profiles)

Featured npm Packages

Three open-source packages integrated into the kit:

- ClassMCP – MCP server providing semantic CSS patterns
- Classpresso – Post-build CSS consolidation
- TerseJSON – Memory-efficient JSON with ~70% size reduction

QUICK START

```
# 1. Clone
git clone https://github.com/TheDecipherist/claude-code-mastery-project-starter
cd my-project && rm -rf .git && git init

# 2. Install global config (one-time, never overwrites existing)
claude /install-global

# 3. Configure your project
claude /setup

# 4. Start building
claude
```

Already have a project? Convert it without losing anything:

```
claude "/convert-project-to-starter-kit ~/my-existing-project"
```

LIVE AI MONITOR

Run `/what-is-my-ai-doing` in a separate terminal and watch everything Claude does in real-time:

- **Every tool call** – what files it reads, writes, and executes
- **Token usage** – input, output, cache hits, running total
- **Cost tracking** – per-call and cumulative session cost

Zero token overhead. Runs completely outside Claude's context window – it doesn't know it's being watched, and it doesn't cost you anything extra.

The command runs the RuleCatch AI pooler in free monitor mode (`--no-api-key`) – no account, no subscription, no setup. For the full RuleCatch platform (rule engine, dashboards, violation alerts, MCP server), there's a 7-day free trial with no credit card required.

RULECATCH INTEGRATION

The starter kit includes RuleCatch.AI integration because it's part of TheDecipherist's own development workflow. **RuleCatch is not required to use the starter kit** — everything else works without it. But if you want 200+ pre-built rules, session analytics, violation alerts, and an MCP server so you can ask Claude directly *"RuleCatch, what was violated today?"* — it's there and ready to go.

The AI pooler's monitor mode is free — no API key, no account required:

```
npx @rulecatch/ai-pooler monitor --no-api-key
```

Real-time monitoring of tokens, costs, and tool usage right in your terminal. For the full platform (rule engine, dashboards, MCP server), RuleCatch offers a 7-day free trial with no credit card required.

The `check-rulecatch.sh` stop hook automatically reports violations at the end of every session.

LINKS

- GitHub Repository
- Interactive Guide (GitHub Pages)
- Claude Code Mastery Guides V3-V5
- RuleCatch.AI
- TheDecipherist