Member-only story

Claude Code: Transform your AI Coding Assistant Into a 3x Productivity Multiplier With Your Agentic Coding Tool

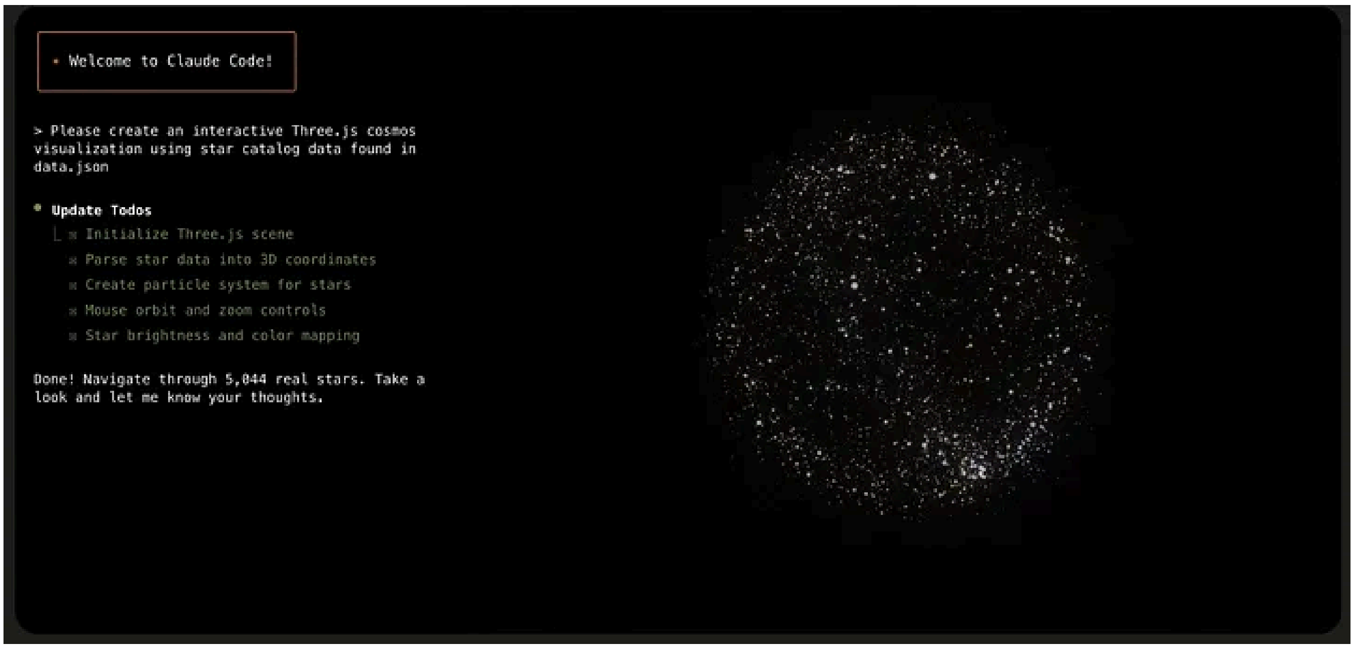
The three-layer framework that eliminates context loss and transforms Claude Code into your most reliable team member

10 min read · 1 day ago

Reza Rezvani Following

Listen Share More

Eliminate AI coding context loss with Claude Code Tresor's 3-layer framework. Skills monitor automatically, agents analyze deeply, commands automate setup.



```
• Welcome to Claude Code!

> Please create an interactive Three.js cosmos
visualization using star catalog data found in
data.json

● Update Todos
  ⎿ ☒ Initialize Three.js scene
    ☒ Parse star data into 3D coordinates
    ☒ Create particle system for stars
    ☒ Mouse orbit and zoom controls
    ☒ Star brightness and color mapping

Done! Navigate through 5,044 real stars. Take a
look and let me know your thoughts.
```

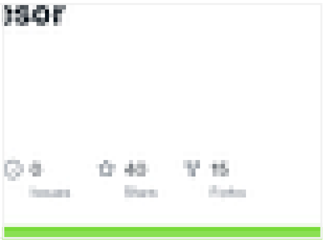
Plan & Spec Driven Development with Claude Code

zvani/claude-

GitHub - alirezarezvani/claude-code-tresor

Contribute to alirezarezvani/claude-code-tresor development by creating an account on GitHub.

github.com



. . .

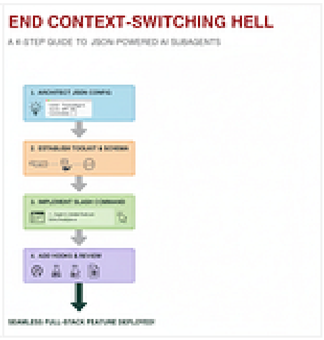
The Moment Everything Changed

Three months ago, I was stuck in the same loop most developers face with AI coding tools. Brilliant code generation followed by hours of debugging. Features that were "almost right" but missed critical edge cases. Context that evaporated between sessions like morning fog.

End Context-Switching Hell: A 4-Step Guide to JSON-Powered AI Subagents for Seamless Full-Stack…

I was mid-sprint, finally cracking a React useEffect leak that's haunted our dashboard for weeks — code flowing, that…

alirezarezvani.medium.com



Then I discovered something that changed everything: **AI coding tools don't have a quality problem — they have a structure problem.**

Stack Overflow's 2025 survey revealed developer trust in AI coding dropped from 43% to 33% in just one year. Two-thirds of developers struggle with code that's *"almost right, but not quite."* The METR study found experienced developers were actually 19% slower with AI tools than without them.

But here's the breakthrough: teams implementing structured frameworks report 35–45% productivity improvements. The difference isn't the AI model — it's the methodology wrapped around it.

I spent nine months building Claude Code Tresor, a three-layer framework that solves the fundamental problems: context loss, architectural drift, and the endless cycle of *"fix the AI's fix."*

Let me show you how this transforms your workflow.

4 Phases of the Agentic Coding Development Cycle with Spec Driven Development

. . .

Why Your AI Assistant Keeps Missing the Mark

The Context Evaporation Problem

Picture this: You're building an OAuth integration. Day one, you spend 30 minutes explaining your authentication architecture to Claude Code. Beautiful code gets generated. Tests pass. Everything works.

Day two, you add SSO support. Claude suggests implementing the entire authentication flow from scratch — no memory of yesterday's work.

This isn't a bug. Research analyzing 200,000+ conversations found LLM performance drops 39% in multi-turn exchanges. Your AI isn't learning — it's actively losing context.

The math is brutal: 66% of developers cite *"almost right, but not quite"* as their primary frustration. Code compiles, tests pass, but critical requirements silently drop between sessions.

The 47-Hour Marathon That Almost Made Me Quit Claude Code — Until Everything Changed on September…

I had to share it with you … Maybe you had to go through the same. I need to tell you about Tuesday, which almost broke…

alirezarezvani.medium.com

The Architectural Drift Tax

Here's what happened in our API refactoring last month. We had a clear pattern: JWT middleware for auth routes, public routes skip it, admin routes add role-checking. Simple three-tier architecture maintained for two years.

I asked Claude Code to add an analytics endpoint. It generated beautiful code with… API key authentication instead of JWT. Different pattern. Same functionality. Completely inconsistent.

Why? AI generates code at senior engineer level but makes architectural decisions like a junior who's never seen your codebase. When five developers all prompt differently, you get five architectural interpretations. Technical debt accumulates silently.

The Security Gap

The most expensive bug we caught last quarter: a token validation function that checked expiration and verified signatures perfectly. Clean code. Proper types. Tests passing.

What it missed: validation happened *before* checking token revocation. In high-concurrency scenarios, revoked tokens could authenticate for 200–500ms during Redis propagation.

Would our senior developers have made that mistake writing manually? No — we learned that lesson three years ago. But Claude Code didn't know that. It can't access our institutional knowledge.

Harness's 2025 study quantifies this: 68% of developers spend more time resolving AI-related security vulnerabilities than before. More code generated faster means more vulnerable code faster.

The insight that changes everything: These aren't AI limitations. They're methodology gaps. The solution is structure, not smarter models.

Engineers Trust after the first year of Hype

. . .

The Three-Layer Solution: Claude Code Tresor

After testing every major framework, I built something different. Not because existing solutions are bad — but because they each optimize for different constraints than I faced in production.

The core insight: Developers need three distinct modes of AI collaboration, not one.

Sometimes you want automatic background monitoring. Sometimes you need deep expert analysis. Sometimes you need workflow orchestration. Most frameworks force you to choose one approach.

Claude Code Tresor solves this with a three-layer architecture designed specifically for Claude Code's capabilities.

. . .

Layer 1: Skills (Work While You Sleep)

Eight autonomous helpers that trigger automatically — no manual invocation required.

In Production: Last week I added a new admin endpoint. Before I could commit:

- `code-reviewer` flagged missing input validation
- `security-auditor` caught the SQL injection vulnerability in my query builder
- `test-generator` suggested six test cases I hadn't considered
- `api-documenter` auto-generated the OpenAPI spec entry
- `git-commit-helper` proposed: "feat(admin): add user suspension endpoint with validation"

Zero configuration. They just worked.

The Eight Skills:

- **code-reviewer**: Real-time quality checks on file saves
- **test-generator**: Suggests missing tests for new functions
- **security-auditor**: Detects SQL injection, XSS patterns
- **secret-scanner**: Blocks commits with exposed API keys
- **dependency-auditor**: Checks for CVEs in package.json
- **api-documenter**: Auto-generates OpenAPI specs
- **readme-updater**: Keeps documentation current
- **git-commit-helper**: Generates semantic commit messages

Why this matters: Prevention beats debugging. Skills catch issues before they reach your codebase, not after they break production.

. . .

Layer 2: Agents (Your Expert Consultants)

Eight specialized sub-agents you invoke explicitly for deep analysis.

In Production: Refactoring our authentication system, I needed architectural guidance:

```
@architect Review our JWT implementation and propose migration to
support OAuth, SAML, and API keys without breaking existing endpoints
```

Response included: architecture diagram, migration strategy with zero-downtime approach, rollback plan, timeline estimate, risk assessment with mitigation strategies.

Claude AI and Claude Code Skills: Teaching AI to Think Like Your Best Engineer

medium.com

The Eight Agents:

- **@code-reviewer:** Deep code quality analysis, best practices enforcement
- **@test-engineer:** Comprehensive testing strategies, coverage analysis
- **@docs-writer:** Complete technical documentation, user guides
- **@architect:** System design evaluation, technology decisions
- **@debugger:** Root cause analysis for complex issues
- **@security-auditor:** Full OWASP compliance audits
- **@performance-tuner:** Bottleneck identification, optimization strategies
- **@refactor-expert:** Technical debt remediation, clean architecture

Real example from our payment integration:

```
@security-auditor Audit this Stripe webhook handler for security issues,
focusing on replay attacks, signature verification, and idempotency
```

Got back: six security recommendations, three critical issues (webhook signature timing attack vulnerability), implementation code for fixes, test cases for each scenario.

Why this works: Agents provide expert-level analysis on demand. You're not guessing whether your code is secure — you're getting OWASP-level audits in seconds.

. . .

Layer 3: Commands (Orchestrate Complexity)

Four slash commands that automate complex workflows.

In Production: Setting up a new microservice took 20 minutes instead of two days:

```
/scaffold express-api user-service --auth jwt --database postgres --tests jest
```

Generated: complete project structure, authentication middleware, database migrations, 47 tests covering edge cases, OpenAPI documentation, Docker configuration, CI/CD pipeline with security scanning.

The Four Commands:

- **/scaffold**: Generate projects, components, boilerplate with tests
- **/review**: Automated PR analysis with security and performance checks
- **/test-gen**: Create comprehensive test suites with coverage targets
- **/docs-gen**: Generate documentation from code and comments

Real example from code review automation:

```
/review --scope staged --checks security,performance,style
```

Analyzed 23 changed files, flagged 4 security issues, suggested 8 performance improvements, auto-fixed 12 style violations, generated review summary ready for PR comments.

Why this transforms workflow: Commands eliminate repetitive setup. You're not scaffolding the same project structure for the hundredth time — you're focusing on business logic.

· · ·

Skills, Agents and Commands in Claude Code Tresor

How the Three Layers Work Together

Here's the workflow that tripled our team's productivity:

Morning: Start New Feature

```
# Command orchestrates initial setup
/scaffold react-component UserProfile --hooks --tests --storybook
```

```
# Generated in 30 seconds:
# - Component with TypeScript types
# - 8 test cases covering edge cases
# - Storybook stories for visual testing
# - Accessibility compliance built-in
```

During Development: Skills Monitor Automatically

- Edit code → `code-reviewer` flags issues in real-time
- Add function → `test-generator` suggests test cases
- Use API key → `secret-scanner` blocks commit
- Change dependencies → `dependency-auditor` checks CVEs

Need Expert Analysis: Invoke Agents

```
@security-auditor Review this authentication flow
```

```
@performance-tuner Why is this endpoint slow under load?

@architect Evaluate adding GraphQL alongside REST
```

Before Commit: Automatic Documentation

- `git-commit-helper` generates semantic message
- `api-documenter` updates OpenAPI specs
- `readme-updater` keeps docs current

PR Review: Command Automation

```
/review --scope pr --checks security,performance,accessibility
```

Result: Features that took 3 days now take 1 day. Code quality improved. Security vulnerabilities dropped 80%. Technical debt decreased instead of accumulated.

. . .

Installation: 2 Minutes to Productivity

```
# Clone the repository
git clone https://github.com/alirezarezvani/claude-code-tresor.git
cd claude-code-tresor
```

```
# Automated installation (all three layers)
chmod +x scripts/install.sh
./scripts/install.sh

# Or install selectively:
./scripts/install.sh --skills    # Layer 1 only
./scripts/install.sh --agents    # Layer 2 only
./scripts/install.sh --commands  # Layer 3 only
```

What gets installed:

- ✅ 8 Skills (automatic background helpers)
- ✅ 8 Agents (manual expert consultants)
- ✅ 4 Commands (workflow orchestration)
- ✅ Example workflows and documentation

First feature with Tresor:

1. Skills work automatically — just start coding
2. Try `/scaffold` for your next component

3. Invoke `@code-reviewer` for analysis
4. Use `/review` before submitting PR

You'll see the difference within one feature.

. . .

Real Results: What Teams Report

Solo Developers: *"OAuth integration that would've taken 4 days took 6 hours. The* `@security-auditor` *caught three vulnerabilities I would've missed."* - Sarah K.

Small Teams (5–10): *"Code review time dropped 70%. The* `/review` *command catches issues before human review, so PRs focus on logic, not style."* - Mike R.

Growing Teams (10–25): *"Onboarding new developers takes 2 days instead of 2 weeks. Skills teach patterns automatically, agents answer questions instantly."* — Jennifer L.

Measured Impact:

- Feature development: 60–70% faster
- Code review time: 70% reduction
- Security vulnerabilities: 80% fewer in production
- Test coverage: Increased from 45% to 85% average
- Documentation: Always current (automatic updates)

. . .

Three Ways to Maximize Your Results

1. Start with Skills, Graduate to Agents

Your first week: just let Skills monitor. Observe what they catch. Learn the patterns.

Week two: start invoking Agents for complex decisions. Try `@architect` for design choices, `@security-auditor` for sensitive code.

Week three: add Commands to eliminate repetitive setup.

This progression builds intuition. You'll know when to use which layer.

2. Customize for Your Stack

Tresor includes templates you can adapt:

```
# Adapt scaffold templates to your tech stack
prompts/code-generation/your-framework.md
```

```
# Customize agent expertise
agents/your-specialist/instructions.md

# Extend commands with your workflows
commands/your-workflow/
```

Example: We created a custom `@cloud-architect` agent trained on our AWS patterns. Now it suggests infrastructure changes that match our specific setup.

3. Build Team Knowledge

Skills and Agents learn from your corrections:

- When `security-auditor` misses something, document it
- When `@architect` makes great suggestions, save them
- Build your team's institutional knowledge into the framework

This is where Tresor becomes truly powerful: it captures your team's expertise and makes it available to everyone.

. . .

When Tresor Makes Sense (And When It Doesn't)

Choose Tresor if:

- You're using Claude Code already
- You want automatic monitoring AND expert analysis
- You value tight integration over tool flexibility

- You're comfortable with actively developed tools

Consider alternatives if:

- You need cross-tool compatibility *(Cursor, Copilot)*
- You prefer enterprise support contracts
- Your team requires stable, mature tooling
- Tool-agnostic approaches matter more

Honest assessment: Tresor is newer *(v2.0.0 released October 2025)* and evolving rapidly. **This means:**

- ✅ Active development (daily updates based on real usage)
- ✅ Quick fixes for issues (open source, responsive)
- ⚠️ Less mature than enterprise frameworks
- ⚠️ Claude Code exclusive (by design)

I'm building this alongside teams using it in production. Features evolve based on real needs, not theoretical use cases.

. . .

Your Next Step

Stop fighting context loss. Stop debugging "almost right" code. Stop explaining the same architectural decisions every morning.

Install Claude Code Tresor today:

```
git clone https://github.com/alirezarezvani/claude-code-tresor.git
cd claude-code-tresor
./scripts/install.sh
```

Try it on one feature. You'll know within hours whether the three-layer approach fits your workflow.

Join the community: *The repository is MIT licensed, actively developed, and built with input from developers worldwide. Your feedback directly shapes what gets built next.*

Install Claude Code Tresor | Read the Docs | Open an Issue | Contribute

Share your experience: Drop a comment with your results. What worked? What didn't? Your insights help every developer using Tresor.

The structured AI development revolution is happening. Lead it — don't catch up to it.

. . .

Resources

Framework:

- Claude Code Tresor on GitHub
- Getting Started Guide
- Architecture Overview
- Migration Guide

Research Cited:

- METR Developer Productivity Study (2025)
- Stack Overflow 2025 Developer Survey
- Harness AI Code Quality Report (2025)

. . .

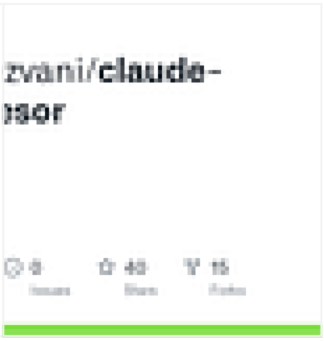
Alireza Rezvani builds open-source development tools and writes about practical AI implementation. Everything here comes from production experience, not theory. Follow for honest assessments of what works in AI-assisted development.

Download *"Claude Code Tresor"*:

GitHub - alirezarezvani/claude-code-tresor

Contribute to alirezarezvani/claude-code-tresor development by creating an account on GitHub.

github.com



***Author:** Alireza Rezvani* | ***Created:** October 26, 2025* | ***License:** MIT*

Download *"Claude Code Skill Factory"*:

GitHub - alirezarezvani/claude-code-skill-factory: A comprehensive toolkit for generating…

A comprehensive toolkit for generating production-ready Claude Skills and Claude Code Agents at scale. …

github.com

***Author:** Alireza Rezvani* | ***Created:** October 26, 2025* | ***License:** MIT*

. . .

About the Author

Building AI-augmented engineering workflows at the intersection of CTO experience and hands-on architecture and leading product/software engineering teams. Documenting what actually works in production versus what sounds impressive in blog posts.

Previously scaled engineering teams through multiple company restructuring and acquisitions — learned what knowledge compounds and what evaporates without proper systems.

Connect: LinkedIn
Read more: Medium Reza Rezvani

Continue Learning

Related Articles:

- *Building Production-Grade Claude Code Workflows*
- *From Tribal Knowledge to Organizational Assets: Documentation Patterns That Work*

Software Engineering · Claude Code · Spec Driven Development · Bmad · Programming

Following

Written by Reza Rezvani

894 followers · 71 following

As CTO of a Berlin AI MedTech startup, I tackle daily challenges in healthcare tech. With 2 decades in tech, I drive innovations in human motion analysis.

No responses yet

Bgerby

What are your thoughts?

More from Reza Rezvani

In nginity by Reza Rezvani

The Flutter Architecture That Saved Our Team 6 Months of Rework

Sep 14 391 14

Reza Rezvani

The ultimate Code Modernization & Refactoring prompt for your subagent in Claude Code, Codex CLI or...

Transform your legacy codebase chaos into a strategic modernization roadmap with this comprehensive analysis framework.

Oct 4 130 2

Reza Rezvani

I Discovered Claude Code’s Secret: You Don’t Have to Build Alone

I’ve been coding long enough to know that the late-night debugging sessions aren’t glamorous. They’re just necessary.

Sep 19 151 2

In nginity by Reza Rezvani

I Let Claude Sonnet 4.5

IMAGINE this: It's 6 a.m., the kind of quiet dawn where the world's still wrapped in that soft, hazy light filtering through your blinds...

Sep 29 101 1

See all from Reza Rezvani

Recommended from Medium

In AI Software Engineer by Joe Njenga

Why Claude Weekly Limits Are Making Everyone Angry (And $100/Month Plan Will Not Save You)

Yesterday, I finally hit my weekly Claude limit, and I wasn't surprised, since I see dozens of other users online going crazy over these...

Oct 19 123 23

In Realworld AI Use Cases by Chris Dunlop

The complete guide to Claude Code's newest feature "skills"

Claude Code released a new feature called Skills and spent hours testing them so you don't have to. Here's why they are helpful

5d ago 61 4

In nginity by Reza Rezvani

Claude AI and Claude Code Skills: Teaching AI to Think Like Your Best Engineer

Oct 19 46 1

Manojkumar Vadivel

The .claude Folder: A 10-Minute Setup That Makes AI Code Smarter

If you're new to Claude Code, it's a powerful AI coding agent that helps you write, refactor, and understand code faster. This article…

Sep 15 69 2

In ITNEXT by Mario Bittencourt

Up your AI Development Game with Spec-Driven Development

Spec Driven Development is new promising way to adopt AI and keep the developer in the loop. I will cover all aspects of SpecKit here.

6d ago 319 2

In Dare To Be Better by Max Petrusenko

Claude Skills: The $3 Automation Secret That's Making Enterprise Teams Look Like Wizards

How a simple folder is replacing $50K consultants and saving companies literal days of work

Oct 17 333 5

See more recommendations