Member-only story

Why Agent Skills Will Transform How We Build AI

How Anthropic's new Agent Skills framework turns general-purpose AI into specialized experts — and why it changes everything about building AI agents

11 min read · 1 day ago

Reza Rezvani

Last Thursday/Friday, I hit my breaking point with Claude.

Three hours. That's how long I spent trying to get it to maintain context across a 50-file codebase. Every single change required re-explaining our database schema, our API conventions, our testing procedures. The same explanations, over and over, like teaching the same lesson to someone with amnesia.

Anthropic's Agent Skills will be the NEXT Big Thing

By hour three, I stumbled onto Agent Skills. And I'm not exaggerating when I say it felt like discovering fire.

The Demo-to-Production Gap Nobody Warns You About

Here's the thing about AI development tools: the demos are absolutely magical. The reality? Not so much.

You've watched the videos. Claude builds a full application in ten minutes. OpenAI Codex CLI debugs your gnarliest code. GitHub Copilot finishes your thoughts before you type them. It all looks effortless, almost too easy.

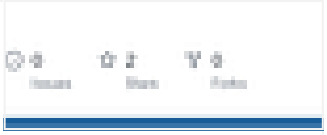
Then you bring these tools into your actual workflow — with your team's specific conventions, your company's particular quirks, your unique tech stack — and suddenly you're burning more time explaining context than you're saving.

Check out this repo I have built and published as open source on Github:

GitHub - alirezarezvani/claude-skills: A comprehensive collection of Skills for Claude Code or...

A comprehensive collection of Skills for Claude Code or Claude AI. - GitHub - alirezarezvani/claude-skills: A...

github.com



The AI isn't the problem. The problem is that even the smartest models don't inherently know YOUR stuff. They don't know your team nests database migrations three folders deep. They don't know your PR review process has twelve steps, not three. They don't know your "simple" deployment touches five interconnected systems.

A recent developer survey found teams waste 8–12 hours weekly just managing AI context and re-explaining procedures. That's not AI productivity. That's a productivity tax.

Agent Skills: Teaching AI Your Way of Working

On October 16, 2025, Anthropic released something that solves this elegantly. They call it Agent Skills, and the core insight is brilliant in its simplicity: *stop cramming everything into prompts. Instead, package your expertise into organized folders that Claude Code loads on-demand.*

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

Stop Copying Prompts. Start Building Intelligence. From Prompt Fatigue to Persistent Intelligence: Why Agent Skills Are...

alirezarezvani.medium.com

Think of Skills like onboarding a new engineer. You don't hand them the entire company wiki on day one and expect them to memorize it. You give them a table of contents, direct them to relevant sections when needed, and let them explore deeper as situations demand.

> Skills work exactly this way for Claude.

Inside a Skill: Three Layers of Intelligence

Every Skill is a folder containing three distinct types of content:

Layer one: SKILL.md — your instruction manual. It opens with YAML metadata *(name, description, usage triggers)* followed by detailed procedures. When you

mention *"PDF"* in conversation, Claude spots the PDF skill in its system prompt and loads this file.

This is an example YAML structure from my Claude AI and Claude Code Agent Skills repository, that is open source and available for free on Github:

```
---
name: API Design Excellence
description: RESTful and GraphQL API design best practices. Use when designing
---
```

Layer two: reference files — supplementary documentation like forms.md, database schemas, coding templates, or style guidelines. These load only when SKILL.md explicitly references them. A PDF skill might bundle twenty reference files, but Claude reads exclusively the ones your specific task requires.

Layer three: executable scripts — Python or bash code Claude can run. Here's the clever bit: when Claude executes a script, the code itself never enters its context window. Only the output returns. That 500-line validation script? Zero tokens consumed. Only the results count.

This architecture enables what Anthropic calls "progressive disclosure" — the secret ingredient that makes Skills actually work at scale.

Progressive Disclosure: Why This Changes the Economics of AI Development

Let me show you why this matters with actual numbers.

Traditional approach: You need Claude's help with your codebase. You paste in 50 files — roughly 100,000 tokens. At current API pricing, that's about $1.50 per request. Multiply that across 100 daily requests from your team: $150 burned on context alone.

Skills approach: At startup, Claude sees metadata for ten skills — about 500 tokens. You mention "database," triggering the relevant skill (another 2,000 tokens). That skill references a schema file (1,000 more tokens). Total: 3,500 tokens, costing about $0.05.

That's a 97% cost reduction. But here's what matters more than money: you can now bundle unlimited organizational knowledge without hitting context limits.

How Progressive Loading Works Under the Hood

Watch what happens when you type: "Fill out this PDF form using data from users.csv"

Claude's reasoning:

> "PDF mentioned → trigger pdf skill."

It executes `cat /mnt/skills/public/pdf/SKILL.md` and reads the instructions. The SKILL.md references forms.md for form-specific logic, so Claude loads that file too. The instructions cite a validation script, prompting Claude to run `python validate_form.py users.csv form.pdf`. Only the script's output enters the context window—the code stays external.

Final context consumed: Perhaps 5,000 tokens instead of 50,000. Your PDF workflow? Already encoded in the skill. No explanation required.

Brad Abrams, Anthropic's Product Lead, demonstrated this by asking Claude Code to create a PowerPoint comparing AI models. The result?

> "A number of well-formatted slides that are easy to digest," he told The Verge.

— he told The Verge.

The magic? Claude had the PowerPoint skill pre-loaded, containing all formatting rules, layout conventions, and design best practices. Zero re-explaining necessary.

Building Your First Skill in 37 Minutes

I built my first skill last week. Total time: 37 minutes from conception to working implementation. Let me walk you through the process — including what I got wrong initially.

Step One: Identify Your Repetitive Pain Point

I chose our PR review process. Every pull request needs to pass a twelve-step checklist: test coverage verification, linting, security scanning, database migration

checks, documentation updates, and seven other items. I'd been copying this checklist into Claude Code repeatedly, asking it to validate each step.

Your trigger question: Are you performing this task more than five times weekly? Does it demand specific context? If yes to both, investing thirty minutes to build a skill probably makes sense.

Step Two: Build the Structure

I created a folder: `~/.claude/skills/pr-review-skill/`

Inside, I drafted my first `SKILL.md`:

```
---
name: pr-review-skill
description: Enforce our 12-step PR review process with automated checks. Trigg
---

# PR Review Skill
## Usage Triggers
User says: "review this PR", "check this pull request", "validate PR"
## Review Process
1. Run test coverage check (see scripts/check_coverage.sh)
2. Verify linting passes (scripts/run_lint.sh)
3. Check for security issues (scripts/security_scan.sh)
[... 9 more steps]
For complete criteria: references/review_criteria.md
```

My initial mistake: I tried cramming everything into SKILL.md. Big file, slow loading. After my first week, I refactored into the reference structure, drastically improving performance.

Final structure:

```
pr-review-skill/
  SKILL.md
  references/
    review_criteria.md
    security_guidelines.md
  scripts/
    check_coverage.sh
```

```
    run_lint.sh
    security_scan.sh
```

Step Three: Test, Break, Fix, Repeat

First test:

> "Hey Claude, review this PR for me."

Claude loaded the skill, executed the checks, identified three issues. I fixed those and retried. Second iteration caught an edge case I'd forgotten to document. Third iteration revealed my security script had outdated rules.

By iteration five, the skill was catching nuances I hadn't consciously codified — edge cases living only in muscle memory.

Within one week, our team's average PR review time dropped from 45 minutes to 12 minutes. The skill caught every mandatory checklist item, freeing human reviewers to focus on architecture decisions and logic quality instead of mechanical verification.

Watch this great explanation from Mark Kashef on Youtube *(I love his channel btw)*, who explains how to build very robust and scalable skills with Claude Code:

What This Means for AI Automation's Future

Skills aren't merely a convenient feature. They represent a fundamental architectural shift in how we construct intelligent agents.

The Composability Revolution

Here's where this gets genuinely exciting: skills compose naturally. You can invoke multiple skills within a single task. Need to extract data from a PDF, validate it against a database schema, then generate a formatted report? That's three skills collaborating seamlessly, each handling its domain, without requiring any explanation from you.

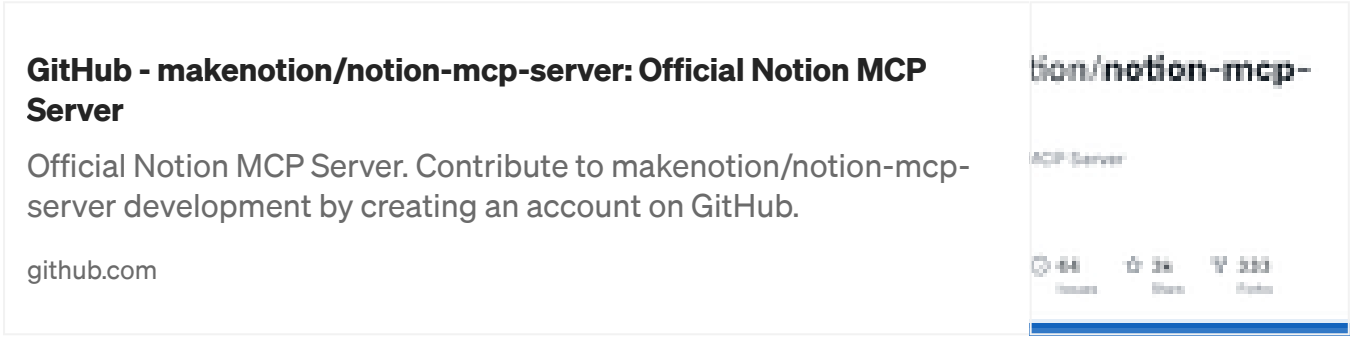
This is what Anthropic means calling Skills *"modular, scalable, and portable."* Build once, deploy everywhere. Share across teams. Version control through Git. Update when processes evolve.

Why Skills Beat Simple Prompts: Prompts are ephemeral instructions that disappear after each conversation. Skills are permanent knowledge repositories that persist across all sessions. Prompts require re-explaining. Skills require building once.

The Model Context Protocol Connection

Anthropic has signaled deeper integration with **Model Context Protocol (MCP) servers** coming soon. For the unfamiliar, MCP enables AI agents to connect with external tools and data sources. Skills will orchestrate these connections.

Envision this: A skill leveraging MCP (Model Context Protocol) to pull data from your Notion workspace, process it through Python scripts, then push updates to your Slack channels and GitHub issues automatically. The skill encodes your complete workflow; MCP provides the external connectivity. Together, they make genuinely autonomous agents practical — not just possible.



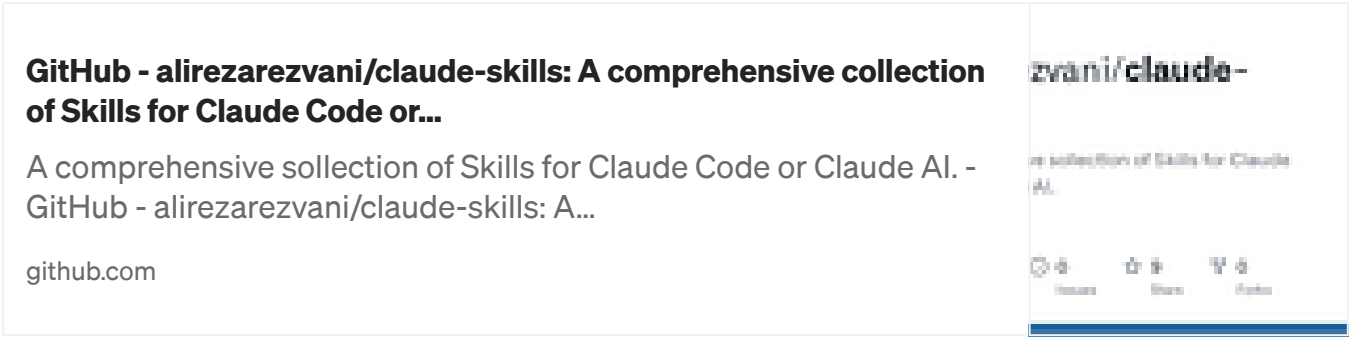
The Self-Improving Agent Endgame

Anthropic's long-term vision? Agents that generate their own skills autonomously. After Claude performs the same task ten times, it proposes: *"I notice you repeat this frequently. Should I create a skill for it?"*

The agent would analyze the pattern, generate the SKILL.md structure, write necessary scripts, test the implementation, and integrate it into your skills library. Automatically. Without human intervention.

We haven't reached that point yet. But the foundation exists. Skills provide agents a mechanism to codify their own behavioral patterns into reusable capabilities. That's the pathway to genuine AI autonomy.

Download the set of curated & Ready-to-Use Agent skills for Claude Code and Claude AI skills that I was able to build since the release of Anthropic's Claude AI and Claude Agents Skills by clicking on the link on my Github repository:



The Security Conversation We Need to Have About Agent Skills

Before you rush off to build fifty skills, we need to address the uncomfortable truth: *security.*

Skills execute code. Malicious skills execute malicious code. If someone deceives you into installing a skill that exfiltrates sensitive data or compromises your system, Claude will run it without question — it can't distinguish malicious intent from legitimate functionality.

Anthropic emphasizes this explicitly: **install skills only from trusted sources.** Specifically:

- Skills you authored personally
- Official skills from Anthropic
- Skills from teammates you trust completely

- Open source skills you've personally audited line-by-line

Downloading random skills from the internet without code review is asking for disaster. Read every script. Examine every SKILL.md. Watch for unexpected network calls, suspicious file access patterns, or operations misaligned with the stated purpose.

This isn't hypothetical paranoia — it's fundamental security hygiene. Treat skills exactly like you'd treat any code executing with your permissions. Because that's precisely what they are.

Think of it this way: would you `curl | bash` a random script from the internet? Then don't casually install unvetted skills either.

Your 30-Day Skills Adoption Roadmap

Ready to implement this in your workflow? Here's a realistic timeline:

Week 1: Learn and Identify

Start by spending 2–3 hours exploring Anthropic's existing skills. Test the document skills — PDF manipulation, DOCX creation, PPTX generation. Observe how they trigger.

Then audit your own workflows. What tasks do you perform more than five times weekly? Which require specific context? Document 3–5 candidates.

I discovered my top candidates weren't what I initially expected. My code review process topped the list, but my "annoying" deployment process? Actually fine — it just needed better documentation.

Week 2: Build Your First Skill

Choose your simplest candidate — not your most impactful. You're learning the pattern, not solving your biggest problem yet.

Create the folder structure. Write your SKILL.md. Test it ten times minimum. Share with one trusted colleague for feedback. Iterate based on real usage, not theoretical perfection.

Your goal: one working skill by week's end. Not perfect. Just functional.

Week 3: Scale Thoughtfully

Build your second skill, but this time consider composition. Can it integrate with your first skill? Should it?

Train 2–3 additional team members on both skills. Start measuring concrete impact: time saved, errors reduced, quality consistency improvements.

This is where you'll discover whether your team actually wants this or if you're building in isolation. Pay attention to resistance — it often signals valid concerns, not just change aversion.

Week 4: Plan Your Strategy

Map out your next ten skills. Assign skill owners — yes, skills need maintainers, just like any infrastructure. Create an internal skills repository. Set quarterly goals that actually matter to your team.

Think about skills as infrastructure — comparable to your CI/CD pipeline, but for organizational intelligence and process knowledge.

By month's end, you'll have 2–3 production skills, genuine team buy-in (or clarity about obstacles), and a realistic roadmap for scaling. More importantly, you'll understand how to package expertise in formats AI agents can genuinely utilize.

The "Last Mile" Problem — Solved

Here's what keeps striking me: Skills solve AI adoption's notorious *"last mile"* problem.

We've had powerful language models for two years. They're legitimately impressive. Yet most teams struggle transitioning from *"impressive demo"* to *"materially improves my daily work."* The gap? Organizational context — the unwritten knowledge about how your specific team operates.

Skills bridge that gap. They provide a mechanism to package the tribal knowledge, the *"this is how we do things here"* conventions, the hard-won lessons from expensive mistakes. And they accomplish this in a format that's portable, version-controllable, and doesn't require costly fine-tuning or model retraining.

This isn't just another feature. It's a different paradigm for constructing AI systems.

The teams recognizing this early — starting to build their skills libraries now — will establish an 18-month advantage over everyone else. They'll possess AI agents that

genuinely understand their workflows. They'll have codified their best practices into executable knowledge. They'll have infrastructure for organizational intelligence that compounds over time.

That's not hype. That's just how institutional knowledge works when you finally have a good format for capturing it.

What's Your First Skill Going to Be?

I'm genuinely curious: which workflow are you most eager to transform into a skill?

Your code review process? Your deployment checklist? Customer support triage? Data pipeline validation? Meeting notes template? Bug report formatting?

Whatever it is, you now have a proven method to teach it to Claude once and leverage it forever. That's the core promise of Agent Skills. Based on my experience so far, it's a promise they're actually delivering.

Drop a comment below sharing your first skill idea. I'm genuinely curious what problems people are tackling with this. And if you build something valuable, please share it — the entire ecosystem improves when we learn from each other's solutions.

. . .

Resources to Get Started:

- Anthropic Skills Documentation — Official guide
- Skills GitHub Repository — Example skills and templates
- Claude Agent SDK Guide — Build custom agents
- Engineering Blog: Agent Skills Deep Dive — Technical architecture

. . .

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 | **Explore:** GitHub

Continue Learning

Related Articles:

- *Building Production-Grade Claude Code Workflows*
- *From Tribal Knowledge to Organizational Assets: Documentation Patterns That Work*
- When the Ground Shifts: Leading Engineering Teams Through the Anxiety We All Feel

Ai Productivity | Ai Automation | Claude Skills | Anthropic Claude | Ai Agent

Follow

Written by Reza Rezvani

849 followers · 67 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

Reza Rezvani

The Complete Claude Code 2.0

Claude Code 2.0 isn't just an update — it's a complete reimagining of how AI assists development. Here's everything it can do.

Sep 30 315 11

Reza Rezvani

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 my entire workflow.

Sep 18 277 13

In nginity by Reza Rezvani

The Flutter Architecture That Saved Our Team 6 Months of Rework

Sep 14 389 14

Reza Rezvani

The Claude Code Guide That Saved 40 Hours Per Sprint: 7 Battle-Tested Tips for Engineering Teams

TL;DR: After a nightmare debugging session nearly killed our sprint, I discovered 7 Claude Code workflows that slashed our debugging time...

Sep 16 64 3

See all from Reza Rezvani

Recommended from Medium

aakash

Perplexity Just Unleashed 10 FREE AI Agents That Do Your Entire Job (The “Comet” Shortcut)

Stop what you are doing. The era of the simple AI search engine is officially over.

Oct 7 559 15

In The Context Layer by Jannis

Claude’s New “Skills” Show How Anthropic Is Layering Intelligence on Top of MCP

3d ago 202 1

In UX Collective by Adrian Levy

What Perplexity's AI browser reveals about UX's future

I stopped typing URLs after 3 days.

Oct 12 510 16

In Towards AI by Gowtham Boyina

Why Your Software Development Life Cycle Will Not Work for Your AI Agents (And How to Change That)

Based on IBM's Guide to Architecting Secure Enterprise AI Agents with MCP—Verified by Anthropic (Oct 2025)

Oct 12 235

Riccardo Tartaglia

5 Essential MCP Servers Every Developer Should Know

I've been experimenting with Model Context Protocol servers for a few months now, and I have to say, they've changed the way I work.

Oct 11 18 2

In Generative AI by Thomas Reid

Google puts another nail in the RAG coffin with URL Context Grounding

Eliminate model hallucinations when processing online data

Oct 2 355 16