



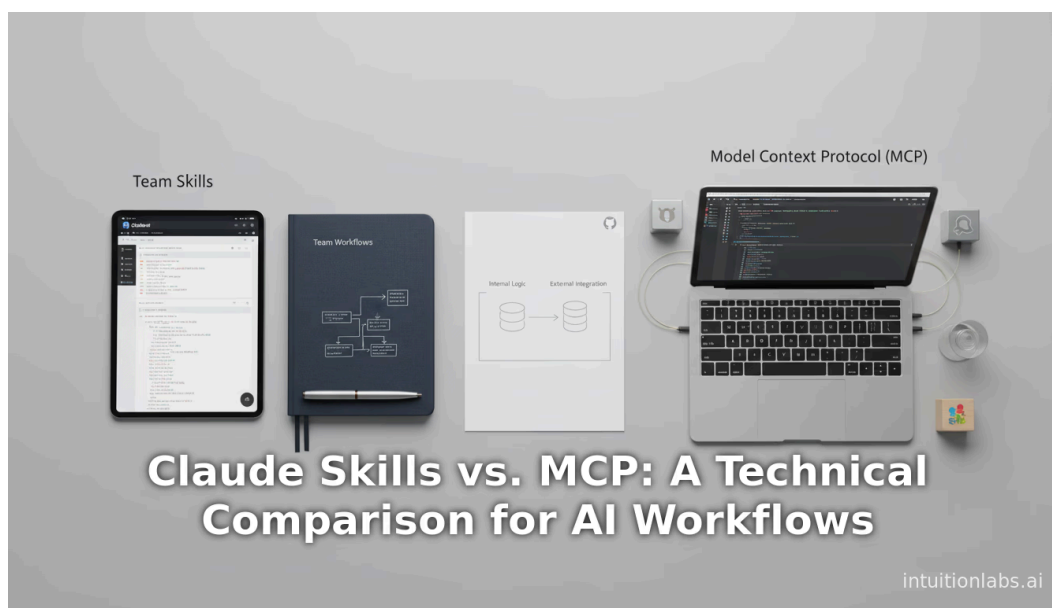
INTUITIONLABS / RESEARCH & PRACTICAL GUIDANCE

Claude Skills vs. MCP: A Technical Comparison for AI Workflows

Build practical AI for pharma and biotech with IntuitionLabs. We help life-science teams turn complex information and workflows into useful software, governed knowledge systems and AI tools.

Published by IntuitionLabs · 2025-10-27 · claude skills · model context protocol · anthropic · ai workflows · llm tools · ai customization · agentic ai · claude ai · ai

Original article: <https://intuitionlabs.ai/articles/claude-skills-vs-mcp>



In this article

1. [Claude Skills vs. MCP: A Technical Comparison for AI Workflows](#)
 2. [Executive Summary](#)
 3. [Introduction](#)
 4. [Claude Skills: Overview and Architecture](#)
 5. [Model Context Protocol \(MCP\): Overview](#)
 6. [Comparing Claude Skills and MCP](#)
 7. [Concrete Use Cases and Examples](#)
 8. [Document and Content Formatting](#)
 9. [Business Workflow Automation](#)
 10. [Comparative Summary of Use Cases](#)
 11. [Data and Evidence](#)
 12. [Perspectives and Expert Commentary](#)
 13. [Implications and Future Directions](#)
 14. [Conclusion](#)
 15. [References](#)
-

Claude Skills vs. MCP: A Technical Comparison for AI Workflows

Executive Summary

Update (October 2026): The core distinction still holds: Skills package know-how and procedures, while MCP connects a model to outside systems. Three changes since publication matter for this comparison. First, the two now overlap: MCP's official Skills extension (SEP-2640, final in September 2026) lets an MCP server publish Skills in the same SKILL.md format, which clients discover and load through MCP resources, although client support is still early ([MCP Skills extension](#)) ([MCP client matrix](#)). Second, Skills are no longer specific to Claude: the format is published as the Agent Skills open standard and is supported by agents including OpenAI Codex, Gemini CLI, GitHub Copilot and Cursor ([agentskills.io](#)). Third, MCP's context cost has dropped in practice: Claude Code now defers MCP tool definitions by default with tool search, loading only tool names and server instructions at session start ([Claude Code docs](#)). The current MCP specification is revision 2026-07-28, which makes the protocol stateless ([MCP changelog](#)).

The introduction of **Claude Skills** by Anthropic in October 2025 represents a major step in customizing AI assistants. Claude Skills are *task-specific modules*: essentially folders with a SKILL.md and optional code or resources that encode repeatable procedures and expertise for the Claude AI assistant. When a user's request matches a Skill's criteria, Claude automatically loads that Skill's instructions, allowing it to perform specialized tasks (e.g. formatting corporate documents or analyzing data) without the user having to re-explain the process each time ([simonwillison.net](#)) ([platform.claude.com](#)). In contrast, the **Model Context Protocol (MCP)**, an open standard introduced by Anthropic in late 2024 and [donated to the Linux Foundation's Agentic AI](#)

Foundation (AAIF) in December 2025, is a way of connecting an LLM like Claude to external tools and data sources in a uniform way (colinmcnamara.com) (modelcontextprotocol.wiki). MCP lets the AI invoke external APIs or services (for example, GitHub, databases, or SaaS platforms) via a standardized client-server protocol. By December 2025, MCP had grown into one of the fastest-adopted open-source AI projects, with over 97 million monthly SDK downloads and 10,000+ active servers (blog.modelcontextprotocol.io); by July 2026 the maintainers reported close to half a billion monthly downloads across MCP's Tier 1 SDKs ([MCP blog](#)).

This report provides an in-depth analysis of Claude Skills versus MCP, highlighting their design, use cases, and how they differ in practice. We survey documentation and expert commentary to explain Skills' architecture (including "progressive disclosure" loading of content) and MCP's mechanics (client/server orchestration of tool calls). We include multiple perspectives, from Anthropic's engineering blog and docs to independent developers and tech analysts, as well as concrete examples of each. For instance, Skills have already been used for tasks like formatting Excel formulas or enforcing brand style in slide decks (www.tomsguide.com) (www.tomsguide.com), yielding large time savings (one report cites ~87.5% faster completion of a finance workflow at Rakuten) (www.echofold.ai). Meanwhile, MCP has been employed to hook LLMs into systems like GitHub, CI/CD pipelines, Slack, and databases, enabling **end-to-end workflows** in engineering environments (colinmcnamara.com). We analyze their trade-offs (e.g. Skills' lightweight token use versus MCP's upfront integration cost) and discuss future implications for AI automation. The findings draw on extensive sources and examples, illustrating how Skills and MCP each excel in different domains and how they may be combined in practice (colinmcnamara.com) (simonwillison.net).

Introduction

Large language model (LLM) assistants are increasingly becoming general-purpose tools for business and creativity. A key challenge is **customizing** these assistants for specific organizations and tasks. Early approaches, such as **writing better prompts or fine-tuning models** on domain data, have limitations. More recent innovations let users add structured knowledge or tools. For example, OpenAI's "function calling" and plugins allow ChatGPT to invoke APIs; Anthropic's **Model Context Protocol (MCP)** similarly standardizes LLM-tool integration. Anthropic's latest innovation, **Claude Skills**, takes a different approach: it lets users encode procedural knowledge and preferred workflows into reusable "skills" that Claude can load on demand.

This report examines **Claude Skills**, introduced by Anthropic on Oct 16, 2025, and compares them to **MCP** (introduced Nov 2024, donated to the **Agentic AI Foundation** in Dec 2025). We draw on primary sources (official docs and engineering articles) and commentary from developers and analysts. We cover how Skills are structured, how they operate, and what concrete problems they solve, contrasted with MCP's design and typical uses. Key questions include: *How do Skills achieve context efficiency and ease of use? In what ways are MCP servers and Skills complementary or competing? What real-world tasks illustrate each?* Throughout, we use examples (from user stories and technical blogs) and data (e.g. reported productivity gains) to ground our analysis.

Claude Skills are a late-2025 release, building on Anthropic's platform (Claude.ai and Claude Code) and enabling "skill-based" AI. They aim to turn Claude into a *specialist* for a user's workflows (platform.claude.com). By contrast, MCP was conceived as a *protocol* connecting any LLM to outside systems

like databases, SaaS tools, and custom APIs (colinmcnamara.com) (colinmcnamara.com). This report situates both in context: Skills as primarily about **instructions and context** for specific tasks, and MCP as about **expanding the LLM's reach** via external APIs. We also discuss related concepts (e.g. subagents, agents) only insofar as they illuminate the Skills vs. MCP distinction.

We begin with a detailed look at Claude Skills (how they are defined, loaded, and used) and at MCP's architecture. We then directly compare their features (see Table 1 below). Next we survey real-world use cases for each: from business automation and document processing to coding and analysis. Finally, we discuss implications for the **future of AI workflows**: how Skills and MCP might evolve, the impact on productivity, and how organizations will adopt these technologies.

Claude Skills: Overview and Architecture

Definition: Claude Skills are *folders* containing a SKILL.md text file (with YAML frontmatter metadata) plus any associated scripts, documents, or assets. Each Skill “packages” a specific workflow or organizational guideline. For example, Anthropic's docs describe Skills as “filesystem-based resources” that provide Claude with domain-specific expertise, effectively capturing the best practices or standard operating procedures for a task (platform.claude.com). Unlike a one-off prompt, a Skill explicitly tells Claude how to handle a class of tasks, so the user does not have to re-explain it every time (platform.claude.com) (simonwillison.net).

Content and Progressive Loading: The SKILL.md file contains human-readable instructions, templates, example outputs, and metadata (like the skill's name, description, and “apply” conditions). Anthropic's engineering blog explains that at session startup, Claude pre-loads only the metadata (name and description) from each installed Skill (claude.com). This lightweight initial step is crucial: it consumes about 100 tokens per skill, according to Anthropic's documentation ([Claude Platform Docs](https://claude.com/docs)), allowing Claude to recognize which skills might apply to the user's request without flooding the prompt with all details (simonwillison.net) (claude.com). When Claude detects that a particular Skill is relevant (based on its description or embedded examples), it will then *load* the full instructions and any linked files into its context (simonwillison.net) (claude.com). This “progressive disclosure” means Skills can be stored in large numbers without overwhelming the model. In practice, Claude only adds the content of a skill to its working context when needed (claude.com) (www.echofold.ai).

Execution and Code: Skills are not just static text. They can include executable scripts (e.g. Python programs) to perform tasks. For instance, in a community example, a “slack-gif-creator” skill included Python code that used the PIL library to draw frames and create an animated GIF (simonwillison.net). Official docs confirm that Skills can bundle code for deterministic operations or integrations: one snippet defines an `analyze_logs(log_file)` function within a skill's folder (colinmcnamara.com). Runtime and network access depend on the Claude product surface. API Skills run in a sandboxed container without network access, while Claude Code Skills run in the user's local environment and have the same network access as other programs on that computer ([Claude Platform Docs](https://claude.com/docs)). In short, a Skill can be entirely code-free (just textual instructions) or include executable code.

Portability and Scope: Skills are *portable* across Claude's interfaces. A Skill folder written by a user can be loaded in Claude.ai, Claude Code (IDE environment), the Claude Agent SDK, or via the Claude API (howaiworks.ai) (platform.claude.com). Anthropic provides both pre-built Agent Skills (for PowerPoint, Excel,

Word, and PDF processing) and supports custom Skills uploaded via the `/v1/skills` API endpoints or installed by dropping files into `.claude/skills/` directories (platform.claude.com). The API supports up to 20 Skills per request, and custom Skills on the Claude API are shared workspace-wide ([Claude Platform Docs](#)). Crucially, once defined, a Skill is reusable across chats and projects: it encapsulates ongoing business rules. For example, Anthropic's official docs say Skills transform a general-purpose agent into a specialist by encoding workflows and context that apply domain-wide (platform.claude.com).

Use Cases and Examples: In practice, Skills have already shown promise in many domains. Tech articles note users building skills for formatting and compliance tasks: Tom's Guide reports that Skills can "format Excel formulas, apply brand standards in presentations, and adhere to legal policies" automatically (www.tomsguide.com). Another article notes users creating Skills for marketing content: for example, a single Skill can generate a newsletter or press release in AP style, or restyle slide decks according to brand language (www.tomsguide.com). In business workflows, Skills have been applied to summarizing sales calls into a CRM system, drafting proposals from meeting notes, or generating research briefs, tasks that normally require repeated human effort (sider.ai) (sider.ai). For instance, one business blog outlines a "sales call-to-CRM auto-summarizer" Skill that ingests a transcript, extracts key points, and maps them into structured CRM fields (sider.ai). Another example is a "proposal/SOW drafter" Skill that takes input like call notes and pricing matrices and outputs a full draft Statement of Work (sider.ai). These illustrate how Skills can automate multi-step business processes without manual prompting.

Benefits: Proponents highlight several advantages of this design. Because Skills load only as needed, they impose minimal token overhead (simonwillison.net) (www.echofold.ai). Once a Skill is authored, Claude effectively "remembers" that procedure, eliminating redundant prompting and reducing errors. In Anthropic's launch announcement, Rakuten said a finance workflow that once took a day now takes an hour ([Anthropic](#)). Anthropic's documentation describes Skills as reusable, on-demand resources for packaging domain-specific workflows and best practices ([Claude Platform Docs](#)). Specific enterprise ROI figures should be evaluated against direct, independently verifiable case-study evidence before being used to estimate expected savings.

Limitations: Skills require an agent environment with file system access (in Claude, the agent framework and its file system). They are no longer unique to Anthropic's platform: the format is published as the Agent Skills open standard and is supported by other agents, including OpenAI Codex, Gemini CLI, GitHub Copilot and Cursor (agentskills.io). Anthropic has also [open-sourced a public Skills repository on GitHub](#) with example Skills. Skills also rely on the model having tool execution enabled (code execution). Custom Skills do not sync across surfaces: Skills uploaded to Claude.ai must be separately uploaded to the API, and Claude Code Skills are filesystem-based and separate from both. On the Claude API, Skills cannot make external network calls or install runtime packages; only pre-installed packages are available (platform.claude.com). They may not handle tasks that require real-time external data unless combined with other mechanisms (see MCP below). The user must author the skill carefully; otherwise, Claude might not trigger it correctly. Nonetheless, Skills offer a new, human-friendly way to "program" the AI by writing Markdown and examples, which many see as simpler than complex prompt engineering.

Model Context Protocol (MCP): Overview

Definition and Purpose: The Model Context Protocol (MCP) is an *open standard* for connecting LLMs to external applications and data sources. Originally promoted by Anthropic's team, MCP was **donated to the Agentic AI Foundation (AAIF)**, a directed fund under the Linux Foundation co-founded by Anthropic, Block, and OpenAI, in December 2025 (linuxfoundation.org). Platinum members of the AAIF include Amazon Web Services, Anthropic, Block, Bloomberg, Cloudflare, Google, Microsoft, and OpenAI, signaling broad industry endorsement. MCP abstracts tool invocation so that an AI assistant can use any connected service via a uniform interface, much like a universal adapter (colinmcnamara.com) (www.ema.co). Officially, MCP is described as "the open protocol connecting AI to external tools and data sources" (colinmcnamara.com). Its tagline on community pages evokes a "USB-C for AI applications", implying a plug-and-play ethos (modelcontextprotocol.wiki).

Architecture: MCP works via *servers* (for each tool) and *clients* (the LLM environment). Each MCP server "exposes" one or more tools or data interfaces. For example, Anthropic and community developers have built MCP servers for GitHub (code and PRs), databases (PostgreSQL), WordPress, Slack, etc. In Claude Code, project-scoped MCP server configurations can be stored in a `.mcp.json` file; other MCP clients use their own configuration mechanisms. MCP servers expose tools that clients can discover and call. The protocol does not mandate a particular user-interaction pattern: a client may let a model discover and invoke tools automatically, while the client sends any resulting `tools/call` request to the server and receives the result.

A significant protocol update in March 2025 (spec version 2025-03-26) introduced **Streamable HTTP** as the standard transport, **replacing the older HTTP+SSE transport**. Streamable HTTP supports chunked transfer encoding and progressive message delivery over a single HTTP connection, enabling MCP servers to be deployed on serverless infrastructure like AWS Lambda without long-lived connections. That revision also added JSON-RPC batching, but MCP removed support for batching in the 2025-06-18 specification revision ([MCP 2025-03-26 changelog](#); [MCP 2025-06-18 changelog](#)). The current revision, 2026-07-28, removes protocol-level sessions and the `initialize` handshake so that each request is self-contained, moves Tasks into an official extension, and adds an `extensions` capability field for optional extensions ([MCP 2026-07-28 changelog](#)).

In January 2026, MCP introduced **MCP Apps**, an extension allowing MCP tools to return interactive UI components (dashboards, forms, visualizations) that render directly in the conversation (blog.modelcontextprotocol.io). ChatGPT, Claude, Goose, and Visual Studio Code announced support at launch, and the MCP extension support matrix now also lists clients such as Cursor and Microsoft 365 Copilot ([MCP client matrix](#)). This expands what MCP integrations can deliver beyond text responses.

Importantly, MCP is meant to be **model-driven**: the LLM itself decides which tools to invoke and how, guided by instructions. The system is often envisioned as an LLM agent talking to other agents. For example, a guide describes the architecture for a CI/CD monitoring agent: an MCP server connects to GitHub and CircleCI for repo and build info, and Claude (the client) queries it via MCP calls (colinmcnamara.com). In another tutorial, a client-side MCP component can ask the LLM for help ("elicitation/sampling"), while each tool's abilities (resources, API schemas) are registered in MCP servers (colinmcnamara.com) (colinmcnamara.com).

Design Goals: MCP was introduced to address the “spaghetti” of ad-hoc AI-tool integrations. Instead of custom code stubs for each API, MCP proponents say, teams can adopt a uniform framework. As one analyst explains, MCP “aims to standardize the way LLMs interact with external applications”, replacing brittle one-off integrations with a structured protocol (www.ema.co). The MCP specification includes not just functions but also other contexts (prompts, resources, sampling rules) that can guide the LLM’s use of those tools (www.hckrnws.com) (www.hckrnws.com). MCP provides optional authorization capabilities for HTTP-based transports and recommends that clients give people clear visibility and confirmation for tool invocations. It does not, by itself, guarantee that every server has authorization, least-privilege permissions, or a controlled deployment; those safeguards depend on the client and server implementation.

Limitations and Critiques: Despite its promise, MCP has drawbacks. In practice, registering an MCP server often requires loading extensive metadata (tool documentation, function definitions) into the LLM’s context. Analyst Simon Willison notes that GitHub’s official MCP definition alone can consume “tens of thousands of tokens” (simonwillison.net), leaving little room in the prompt. This high context cost is one of the main criticisms. Clients have since reduced it: Claude Code now defers MCP tool definitions until Claude needs them, loading only tool names and server instructions at session start ([Claude Code docs](#)), and the MCP roadmap lists progressive discovery of tools as a priority ([MCP roadmap](#)). An enterprise review also warns that MCP is not yet a turnkey solution for complex workflows: it struggles with long sequences, app-specific data models, real-time triggers, and human-in-the-loop scenarios (www.ema.co). Essentially, MCP can connect tools, but orchestrating multi-step business processes still requires additional layering (planning, agent logic). Security has also emerged as a key concern: a [CoSAI security white paper](#) and the discovery of remote code execution flaws in widely used MCP servers have highlighted the need for careful vetting of MCP integrations.

From a usage standpoint, MCP tends to require more technical setup than Skills. Developers must run or install MCP servers (often Node or Python packages) and configure JSON metadata. The learning curve can be steep: early adopters have noted that many developers “don’t understand MCP at all” and that its broad feature set can be confusing (www.hckrnws.com). In contrast to Skills’ “Markdown + optional script” approach (simonwillison.net) (claude.com), MCP often involves writing or generating API schemas and prompts. That said, the ecosystem is rapidly maturing: the [MCP Registry](#) (previewed September 2025 and still labeled a preview as of October 2026, per the [registry documentation](#)) aims to make discovering and installing MCP servers much simpler. On the plus side, MCP is now *vendor-neutral under Linux Foundation governance* (not tied to Claude) and supports many languages; any LLM client with an MCP integration can use it, including ChatGPT, Gemini, and many IDE-based agents.

Use Cases: MCP is best suited for enabling LLMs to query or control existing systems. For example, a sales agent could use an MCP server to query a company’s CRM database, or a CI bot could fetch GitHub issues via MCP. In work by Colin McNamara, he shows a concrete scenario: an LLM agent uses MCP clients to connect to GitHub, CircleCI, Slack, and a Postgres database, thereby aggregating project data (colinmcnamara.com). Separately, a marketing chatbot might call an MCP server that interfaces with a Tweet scheduler (to post social updates) or a Shopify inventory API (to check stock). MCP essentially turns any HTTP API (or database) into a callable tool for the LLM. Because it is standardized, once defined, those tools can be invoked with natural language by any model that understands the protocol.

Comparing Claude Skills and MCP

The following table contrasts *Claude Skills* and the *Model Context Protocol* across several dimensions:

Feature / Aspect	Claude Skills	Model Context Protocol (MCP)
Purpose	Encapsulate human workflows and domain knowledge as reusable instructions and scripts (simonwillison.net) (platform.claude.com). Tailor Claude to specific tasks by providing specialized guidance.	Provide a uniform interface for LLMs to call external tools, services, or data (colinmcnamara.com) (www.ema.co). Standardize how the model maps requests to APIs.
Activation / Trigger	Skills are <i>automatically</i> detected by Claude when relevant. Claude scans skill metadata at start-up and dynamically loads a skill if it matches the user's query (simonwillison.net) (howaiworks.ai). No explicit "call" needed.	MCP functions are <i>invoked explicitly</i> by the agent/LLM via protocol calls. The model formulates a request (e.g. <code>tool(...args)</code>) and sends it to the relevant MCP server (colinmcnamara.com) (howaiworks.ai).
Configuration / Setup	Writing a skill involves creating a folder with a <code>SKILL.md</code> (YAML frontmatter + instructions) and optional code/files. Skills can be uploaded via the Claude UI (as zip files), placed in a directory (<code>~/.claude/skills/</code> or <code>.claude/skills/</code>), or managed through the <code>/v1/skills</code> API endpoints. The API supports up to 20 Skills per request (platform.claude.com) (Claude Platform Docs). No runtime service is needed.	Requires running or deploying MCP servers and a client configuration (<code>.mcp.json</code>). Each external system needs a server component (e.g. <code>@modelcontextprotocol/server-github</code>). The developer must install these, register them, and ensure an MCP client (Claude Code or agent SDK) points to them (colinmcnamara.com) (colinmcnamara.com). The official, community-driven MCP Registry supports discovery of available MCP servers (Anthropic).
Context Usage	Progressive disclosure: initially only names/descriptions of skills enter context (few tokens). Full skill content is only loaded when needed (claude.com) (simonwillison.net). This keeps prompt sizes small.	Generally requires pre-defining tool interfaces and permissions in a prompt or system context. In practice, MCP calls often involve loading API docs, schemas, or parameter descriptions into context (which can be thousands of tokens) (simonwillison.net) (www.ema.co). Claude Code's tool search now defers MCP tool definitions until they are needed, which reduces this overhead (Claude Code docs).
Ease of Use	Designed for end users/teams. Skills can be written in natural-language Markdown; basic skills require little or no coding (www.tomsguide.com) (platform.claude.com). Anthropic provides a guided Skill-creator.	More technical setup: users must understand the MCP spec, run servers, and write config. Analysts note many users initially find it complex. Often geared toward developers and engineers who manage AI tool integrations (www.ema.co) (www.hckrnws.com).
Runtime Behavior	Runtime depends on the product surface. API Skills run in a network-isolated code-execution container; Claude Code Skills run in the user's local environment and may have the same network access as other programs on that computer; claude.ai access varies by user or administrator settings (Claude Platform Docs).	MCP servers function as external utilities. When Claude issues an MCP call, the external server executes (e.g. calling a web API or querying a DB) and returns results to Claude. Whether sensitive operations require authorization, confirmation, or scoped permissions depends on the server and client implementation.

Feature / Aspect	Claude Skills	Model Context Protocol (MCP)
Performance / Efficiency	Highly efficient in terms of tokens: because of selective loading, multiple skills can coexist without bloating context (simonwillison.net) (www.echofold.ai). Claude activates only the minimal info needed for the task, preserving throughput and speed.	Potentially less efficient: parsing heavy MCP definitions can consume large parts of the token budget (simonwillison.net). However, once set up, MCP can handle large external data (e.g. full database queries) without loading all data into the LLM at once.
Capability Scope	Extends Claude's abilities via <i>instructions and code that you provide</i> . Limited by what you encode in the skill files. Excels at contextual, procedural tasks (text formatting, data wrangling) and can be combined sequentially.	Extends Claude's abilities by <i>granting access to external systems</i> . It can fetch real-time data, leverage legacy systems, or execute complex APIs not built into Claude. In principle, any tool with an MCP adapter can be used (from cloud services to IoT devices).
Sharing & Versioning	Skills are essentially files/folders. They can be version-controlled (e.g. via Git), shared with collaborators, or packaged for reuse. Anthropic suggests sharing skills as teams manage them.	MCP endpoints and schemas can likewise be shared, and organizations may operate their own servers. The MCP Registry provides a centralized discovery service, though server deployment and trust decisions remain with clients and operators.
Security Model	Skills are executable content and should be obtained only from trusted sources. Their access to network and system resources depends on the Claude product surface, so bundled instructions and code should be reviewed before use (Claude Platform Docs).	MCP can support authorization and scoped access for protected HTTP servers, but authorization is optional and is not a universal access-control guarantee. Clients and server operators must implement, configure, and review authentication, authorization, tool confirmation, and server trust boundaries.
Ecosystem & Standards	Anthropic's product implementations are platform-specific, but Anthropic published Agent Skills as an open standard for cross-platform portability in December 2025; the format is now supported by agents such as OpenAI Codex, Gemini CLI and GitHub Copilot (agentskills.io). Anthropic also maintains an open-source Skills repository on GitHub and supports sharing through Claude Code Plugins (Anthropic).	An open standard now governed by the Agentic AI Foundation under the Linux Foundation, with platinum members including Anthropic, OpenAI, Google, Microsoft, AWS, Cloudflare, Block, and Bloomberg. Over 10,000 active MCP servers and 97M+ monthly SDK downloads as of December 2025. Since September 2026, an official Skills extension lets MCP servers distribute Skills (MCP Skills extension). Designed to work with any LLM that implements the MCP client spec (modelcontextprotocol.wiki) (colinmcnamara.com).

Table 1: Comparison of Claude Skills versus Model Context Protocol.

This comparison highlights key trade-offs. Skills focus on **embedded expertise and lean prompts**, whereas MCP focuses on **connecting to external tools via a standard protocol**. Skills are simple to author and quick to run, but their domain of action is what you explicitly encode. MCP requires more setup but gives Claude virtually unlimited reach (any service can be wired in). Both can co-exist: as Colin McNamara suggests, in a complex workflow one might use MCP servers to fetch data (GitHub, CI, etc.) while using *skills* to interpret that data or produce summaries (colinmcnamara.com) (simonwillison.net).

In Claude Code, a common way to combine the two for internal services is to register each internal API as an MCP server (for a team, in a project-scoped `.mcp.json` committed to the repository) and to write a project Skill in `.claude/skills/` that tells Claude when and how to use those tools. A Skill's `allowed-tools` frontmatter can name MCP tools (in the `mcp__<server>__<tool>` form) so they run without a permission prompt during

the turn that invokes the Skill, and a Skill folder can also be packaged as a plugin that bundles its own MCP servers, so the instructions and the integration ship together ([Claude Code skills docs](#)) ([Claude Code MCP docs](#)).

Concrete Use Cases and Examples

Below we discuss concrete scenarios illustrating where Claude Skills and MCP are applicable. In many real-world solutions, they are actually complementary: MCP brings in outside information, and Skills apply specialized processing or compliance logic.

Document and Content Formatting

- **Brand-Compliant Document Creation:** A common use of Claude Skills is ensuring documents follow company guidelines. For example, one skill might apply a corporate style to slide decks or newsletters. A tech article notes that Claude Skills can “apply brand standards in presentations” and “generate press releases in AP style” (www.tomsguide.com). Another example is a “branded guidelines scale” skill that rewrites content in a company’s voice.

Skill example: A user creates a skill that given a bullet-plain text prompt, outputs a formatted document (PowerPoint or PDF) with the company’s logo, colors, and approved phrasing. Once set up, Claude automatically uses this skill whenever formatting is requested.

MCP analog: To achieve something similar with MCP, one could connect Claude via MCP to an internal CMS or document management API that holds style templates. The LLM would retrieve the template and fill it. However, MCP alone wouldn’t know *how* to apply wording guidelines; it would just fetch a resource. Skills encapsulate the styling instructions explicitly, which MCP lacks.

- **Spreadsheet and PDF Automation:** Skills have also been used to automate office tasks. Tom’s Guide mentions Skills that can format Excel formulas or work with PDF contents (www.tomsguide.com). In fact, Anthropic’s own release of multi-type document editing (for .xlsx, .docx, .pdf, .pptx) was implemented via Skills (simonwillison.net). For instance, a “spreadsheet-cleaner” Skill could ingest a messy CSV, apply custom transformations (remove nulls, format dates), and output a polished table.

Skill example: A Skill processes an Excel file in the workspace: user uploads data, and Claude runs the skill to fix column formats, apply filters, or generate charts as needed. The logic (e.g. “pivot this table, highlight these cells”) is encoded in the skill’s YAML/script.

MCP analog: Alternatively, MCP could connect to an external spreadsheet service (like Google Sheets or an internal database) and run queries/calculations via their API. This is powerful for data retrieval. But the *procedural knowledge* of how to clean or format the sheet would have to come from the model’s prompts or a separate memory; Skills provide an explicit template for that.

Business Workflow Automation

- **Sales and CRM Workflows:** Business blogs have detailed multi-step sales workflows handled by Skills. One example skill, a “sales call auto-summarizer”, takes a call transcript, extracts the prospect’s needs, objections, timeline, budget, etc., and then populates those into a CRM system ([sider.ai](#)). Another skill automatically drafts proposals or Statements of Work (SOW) from a meeting’s notes and pricing info ([sider.ai](#)). These Skills encapsulate the logic of those repetitive tasks (and even define approved boilerplate and disallowed phrases) to ensure consistency.

Skill example: After a sales meeting, the rep provides the call recording or notes to Claude, which activates a CRM-summarizer Skill and returns mapped CRM fields plus a short summary note. Submitting those fields to the CRM requires an authorized MCP or other configured integration, or a Claude Code/local environment with the necessary credentials and tooling; API Skills have no external network access ([Claude Platform Docs](#)).

MCP analog: MCP could link Claude to external systems like a transcription service and the CRM database. For instance, an MCP server could fetch the transcript and another could write to the CRM’s API. However, even if the model can call those systems, it still needs instructions on how to interpret the transcript. The Skill provides that domain logic (what fields go where). Without a Skill, one could use MCP to get raw data, but would have to instruct Claude manually to map it.

- **Customer Support and Operations:** Another use case is supporting helpdesk or HR processes. A “ticket triage” Skill might classify incoming support tickets, query an FAQ, and draft a response adhering to company guidelines ([sider.ai](#)). Similarly, HR teams could use a Skill to screen candidates: one example skill “flags indemnity clauses” in an NDA or evaluates resumes for key criteria. These automate where humans normally would spend time.

Skill example: An HR agent pastes a candidate’s resume and a short prompt (“screen for role X, highlight concerns”) and Claude loads an HR policies skill that checks the resume against role requirements. The output lists matches and flags issues.

MCP analog: To involve MCP, one could connect Claude to the company’s HR database or knowledge base via MCP servers. For example, an MCP server to the ticketing system (Zendesk/Jira) and one to the solution knowledge base. Then Claude, acting as an agent, might call these via MCP. Indeed, Darshan Joshi (in [EMA.ai](#)) describes an “AI employee builder” scenario where an AI uses MCP to look up customer tickets and take actions like “create a Zendesk ticket” ([www.ema.co](#)). However, again the specialized knowledge of how to comply with support protocols or legal language would have to come from either Skills or additional instructions.

- **CI/CD and Engineering Dashboards:** A compelling example combining MCP and Skills comes from a development context. McNamara outlines building a CI/CD dashboard: an LLM agent uses MCP servers to connect to GitHub (for code/PR data), CircleCI (for build status), Slack, and a metrics database. This satisfies the “external integration” need ([colinmcnamara.com](#)). Then, Skills are created to analyze those data streams; e.g. one Skill might “analyze build trends” or “generate release status reports” using the data already fetched ([colinmcnamara.com](#)). Thus, MCP pulls in the raw telemetry; Skills perform insight and summarization.

Skill example: After MCP servers collect the latest test coverage metrics and failure logs, Claude activates a “trend analyzer” skill that computes pass rates over time and summarizes hotspots. This Skill encapsulates the analytic routine.

MCP role: Without MCP, the model would have no automatic way to query GitHub or CI. With MCP alone, the model could fetch issues or logs, but to interpret them (e.g. “identify why build failed last week”), Skills or prompt engineering would still be needed. The combination illustrated here shows one workflow: MCP handles data processes; Skills handle domain logic.

- **Data Analysis and Reporting:** Businesses can also use Skills for internal analytics. A finance team could use a Skill to standardize the generation of a budget-variance report from quarterly data. The handling of sensitive models and data depends on the product surface and its configured environment; the Claude API’s Skills container has no network access, while Claude Code Skills run in the user’s local environment ([Claude Platform Docs](#)). MCP could connect the Skill to secure data sources (via an internal database server) if needed, but the Skill contains the actual logic of analysis.

Comparative Summary of Use Cases

While the above examples illustrate Skills heavily, MCP has its own domain for use:

- **API-Driven Tasks:** Any task that requires calling a third-party service, like sending email via Gmail API, querying external news, or triggering cloud workflows, can leverage MCP. For instance, an LLM could use MCP to post a tweet by connecting to Twitter’s API or check live currency rates through a financial data API. These are applications where the model needs *external data or action* that Skills alone cannot provide.
- **Multi-System Orchestration:** Workflows that span multiple systems naturally rely on MCP. For example, a supply-chain agent might oversee inventory (database), shipment (logistics API), and sales (ERP). MCP allows one coherent MCP client to speak to each. A Skill does not inherently provide a connection to an inventory system; access requires MCP or another configured integration, or a Claude Code/local environment with the necessary credentials and tooling ([Claude Platform Docs](#)).
- **Legacy Tools:** If a company has a legacy application or on-prem system, MCP can connect the LLM to it via a custom server. On the other hand, encoding those legacy workflows as Skills could substitute some manual steps, but full integration likely requires MCP adapters.

Table 2 below gives a flavor of how Skills and MCP might be applied across common domains:

Domain/Task	Claude Skills Example	MCP Example
Marketing & Branding	Skill to generate brand-compliant newsletters and presentations (www.tomsguide.com). E.g. “rewrite this pitch email in our brand voice” skill.	Connect to CMS/brand assets via MCP server; query style guidelines API. LLM must still decide phrasing.
Document Processing	Skill for automated Excel cleanup or PDF filling (simonwillison.net) (www.tomsguide.com).	MCP to Google Sheets or internal DB; LLM issues queries (e.g. “aggregate sales by region”) to get data.

Domain/Task	Claude Skills Example	MCP Example
Sales/CRM	Call-summarizer Skill that extracts and maps transcript details to CRM fields.	MCP or another configured integration to the CRM and transcription APIs; the integration, rather than the Skill alone, performs CRM updates (Claude Platform Docs).
Customer Support	Ticket classification & response skill (sider.ai).	MCP to ticketing system and knowledgebase. LLM routes actions (“close ticket”, “escalate ticket”) via MCP.
Dev/Ops Automation	Build-log analysis skill (e.g. “summarize test failures”).	MCP to GitHub/CI/CD/SLA tools as in (colinmcnamara.com); orchestrate deployment tasks.
Finance/Analytics	Budget report generator skill (spreadsheet in → analysis out).	MCP to ERP or data warehouse; retrieve raw data then prompt do analysis.
Miscellaneous Tools	Image processing skill (convert product images to thumbnails).	MCP to image-processing API or cloud function; LLM passes image data via server.

Table 2: Examples of Claude Skills vs MCP approaches in different domains. (Skills examples are drawn from user stories and documentation; MCP approaches are potential analogs.)

In summary, **Claude Skills** tend to shine when automating well-defined, internally-specifiable tasks (corporate style, SOP-driven workflows, data formatting, routine analyses, etc.). **MCP** shines when the task requires interacting with external systems or APIs outside Claude’s native environment. Notably, a given solution may use both: MCP to gather data and Skills to act on it.

Data and Evidence

Public documentation establishes how Skills package reusable instructions, resources, and optional code, but it does not establish a general productivity benchmark for Skills ([Claude Platform Docs](#)). Reported business outcomes should be treated as organization-specific unless supported by a direct, methodologically described case study.

In contrast, MCP has primarily been tested in developer environments. Its effectiveness is often measured in terms of flexibility and standard compliance, not minutes saved. Analysts like Darshan Joshi argue that MCP offers a structured path but warn that it isn’t a silver bullet: many enterprise needs (multi-step workflows, data schemas) remain opaque to MCP alone ([www.ema.co](#)). For example, an MCP-based finance query might retrieve the wrong field if the LLM doesn’t know the database schema. Thus, Skills (which can encode exact schema mappings in their instructions) may yield more correct outcomes in domain tasks.

No formal surveys or benchmark studies of Skills vs MCP yet exist. However, developer experience suggests each tool has stochastic cost/benefit: Skills allow rapid gains with minimal coding (a 10-minute skill setup can automate hours of work ([www.tomsguide.com](#))), whereas MCP investment is front-loaded (spinning up servers, writing configs). Over time, as more MCP servers become available, the cost to add new tools may drop.

Perspectives and Expert Commentary

The viewpoints of various experts highlight the perceived roles of Skills and MCP. Simon Willison, a developer and early Claude user, enthuses that Skills bring “context-window documents” that are immediately useful and testable (www.hckrnws.com) (www.hckrnws.com). He contrasts this with traditional documentation, noting Skills create a short feedback loop: one can “write up a context markdown, ask Claude to do something, and iterate in minutes” (www.hckrnws.com). In his own blog post, Willison explicitly compares Skills to MCP: he observes that while MCP got a lot of hype, it suffers from heavy token use (e.g. GitHub's MCP definition) and that “almost everything I might achieve with an MCP can be handled by a CLI tool instead” (simonwillison.net). He argues Skills capture that functionality without even coding a CLI, just by dropping a Markdown file describing the task (simonwillison.net).

Conversely, others note MCP's value in standardization. Some developers emphasize that MCP goes beyond simple function calls: it can expose prompt templates and iterative “elicitation” strategies (www.hckrnws.com) (www.hckrnws.com). The open standard nature means multiple LLM platforms (not just Claude) can interoperate with shared tools. Darshan Joshi (EMA.ai) points out that businesses originally migrated to MCP as part of their “AI strategy” to avoid brittle one-off systems, and it remains the best abstraction for mapping user requests to enterprise APIs (www.ema.co). His view is that while Skills are great for many tasks, large organizations still need a standardized integration layer for existing software stacks.

In practical terms, many forecast that Skills and MCP will coexist. Technologists note that complex agents often use both: MCP servers supply live data, while skills inject business logic. Indeed, Colin McNamara's example shows an LLM agent using both kinds: “Use MCP to connect to GitHub/CI, then create Skills for analyzing build trends and generating reports” (colinmcnamara.com). Even Simon Willison quips that “you don't need MCP for prompt templates and context resources – those are skills” (www.hckrnws.com), suggesting the one-sided view that prompt/context customization can be done via Skills without MCP.

Implications and Future Directions

The emergence of Claude Skills (and similar concepts) signals a maturing of AI workflow automation. By allowing users to encode reusable routines into the AI's “memory”, Skills reduce the need for constant prompt engineering. This is already changing how organizations manage knowledge: writing Skills has become part of operational documentation, akin to writing unit tests or procedures for human staff. Anthropic has launched a [public Skills repository on GitHub](#), and a growing community on platforms like [awesome-claude-skills](#) curates shared Skills for common tasks. Claude Code now also supports Skills via Plugins and auto-loads skills from added directories, further lowering the barrier to adoption.

For MCP, the transition to Linux Foundation governance under the Agentic AI Foundation marks a pivotal moment. With backing from Anthropic, OpenAI, Google, Microsoft, AWS, Cloudflare, Block, and Bloomberg, MCP is now industry infrastructure rather than a single vendor's project (linuxfoundation.org). By December 2025 the ecosystem counted more than 10,000 active public MCP servers ([Anthropic](#)); Colin McNamara's guide walks through examples of available servers (colinmcnamara.com) (colinmcnamara.com). The introduction of **MCP Apps** in January 2026, enabling interactive UI components like dashboards and forms within the conversation, blurs the line between Skills and MCP, as Skills can now trigger rich UI delivered by

MCP servers (blog.modelcontextprotocol.io). Integration between Skills and MCP is already a reality: a skill can internally leverage MCP servers as part of its operations, and vice versa. Since September 2026 the overlap is also part of the protocol: MCP's Skills extension lets a server publish Skills that clients discover and load over MCP ([MCP Skills extension](#)).

Competitive forces continue to shape the landscape. OpenAI has joined the Agentic AI Foundation and adopted MCP, and Anthropic lists Gemini among the products that have adopted MCP ([Anthropic](#)). The convergence around MCP as a shared standard, combined with each platform's unique approach to Skills-like features (e.g., ChatGPT's custom GPTs and memory modules, Google's agent frameworks), once suggested that only the integration layer would standardize. That is changing: the Agent Skills format is now an open standard supported across many agents (agentskills.io), and MCP's Skills extension defines how servers distribute Skills. The key difference persists: Skills are at the **prompt/knowledge layer**, whereas MCP is at the **integration layer**. Both are needed for an AI agent to be truly useful in the real world.

In terms of adoption, indicators are strongly positive. At launch, Box said it uses Skills to turn stored files into PowerPoint, Excel and Word documents, and Canva said it plans to use Skills to customize its agents ([Anthropic](#)) (www.tomsguide.com). Organization-specific productivity or return-on-investment claims should be assessed against direct, methodologically described evidence before being generalized. In January 2026, Anthropic brought interactive versions of workplace tools such as Slack, Figma, and Asana into Claude, built on MCP Apps ([Claude blog](#)) (blog.mean.ceo). On the other hand, challenges remain: maintaining a large library of skills requires governance (version control, team curation, security vetting, and ensuring skills stay updated). Managing MCP servers raises its own considerations: security vulnerabilities discovered in popular MCP servers underscore the need for careful auditing (adversa.ai). Organizations will need best practices for both, and the growing body of documentation from Anthropic, the AAIF, and the developer community is rapidly filling that gap.

Future Research: From a research perspective, these developments invite study. For example, one could measure how much a Skill improves task accuracy versus a zero-shot prompt on the same task. Or benchmark the token-efficiency of Skills vs similar functions via function calls. The broader impact on productivity and economic value also merits study. On the technical frontier, exploring automated synthesis of Skills (e.g. training an agent to write SKILL.md files from examples) could be a frontier. The current MCP roadmap (last updated August 22, 2026) prioritizes agentic messaging primitives, HTTP-native transport unification, agent identity and enterprise security, improved primitives including progressive tool discovery, and SDK developer experience; it does not commit to a specific next-release date ([Model Context Protocol roadmap](#)). The convergence of Skills and MCP Apps, where Skills can trigger rich interactive UI provided by MCP servers, represents a particularly promising area for investigation, as it may fundamentally reshape how users interact with AI agent workflows.

Conclusion

Claude Skills represent a significant evolution in how we teach AI assistants to handle complex tasks. By letting users package procedures and expertise as reusable, on-demand resources, Skills can reduce repeated prompt instructions across conversations ([Claude Platform Docs](#)). In contrast, the Model Context Protocol offers a complementary capability: the ability to reach outside the model's context into enterprise

tools. Our analysis shows that Skills and MCP differ in motivation and mechanics: Skills streamline **internal** processes with minimal token cost, while MCP standardizes **external** integrations (albeit with more upfront effort). Each has its domain of strength.

Ultimately, both contribute to the vision of "AI agents" that can operate autonomously on business workflows. Claude Skills act as the AI's internal playbook, while MCP servers are the AI's nervous system connecting it to the outside world. The rapid maturation of both technologies through late 2025 and 2026 (Skills now supported across the Claude API, Claude Code, Claude Agent SDK, and Claude.ai; MCP now under Linux Foundation governance with broad industry backing) demonstrates that using both in tandem can unlock powerful automation (e.g. engineering dashboards, financial reporting, workplace integrations via Slack and Figma). The introduction of MCP Apps further deepens this synergy by allowing Skills to surface interactive UI components delivered through MCP servers.

As of October 2026, the trajectory is clear: Skills and MCP are not competitors but complementary layers of an agentic AI stack. Companies that embrace Skills as part of their AI infrastructure gain AI assistants that truly "know" their organization's processes and context, without being reminded each time. Meanwhile, MCP is an open standard under the Agentic AI Foundation; Anthropic reported in December 2025 that ChatGPT, Gemini, Microsoft Copilot, and Visual Studio Code were among the AI products that had adopted it. The combination is already delivering on the promise of iterative, no-code AI workflows as an integral part of how businesses operate.

References

1. Anthropic – Claude Engineering Blog. *Equipping agents for the real world with Agent Skills*. (Oct 16, 2025) (claude.com).
2. Claude Platform Docs – *Agent Skills Overview* (updated 2026) (platform.claude.com).
3. Simon Willison's Weblog – "*Claude Skills are awesome, maybe a bigger deal than MCP*" (Oct 16, 2025) (simonwillison.net) (simonwillison.net).
4. Simon Willison's Weblog – "*Claude Skills are awesome, maybe a bigger deal than MCP*" (Oct 16, 2025) (simonwillison.net) (simonwillison.net).
5. Colin McNamara – *Understanding Skills, Agents, Subagents, and MCP in Claude Code* (2025) (colinmcnamara.com) (colinmcnamara.com).
6. Model Context Protocol – *Key Changes: 2025-03-26* (modelcontextprotocol.io).
7. Tom's Guide – Rick Broida, "*Claude just got customizable 'Skills'...*" (Oct 16, 2025) (www.tomsguide.com) (www.tomsguide.com).
8. Tom's Guide – Staff Writer, "*Claude Skills are here - the smartest AI feature...*" (Oct 19, 2025) (www.tomsguide.com) (www.tomsguide.com).
9. Echofold News – "*Claude Skills: Revolutionising AI Automation with Composable Workflows*" (Oct 2025) (www.echofold.ai) (www.echofold.ai).
10. Bionic Business (newsletter) – Sam Woods, "*How To Use Claude Skills (Save Time, Make Money)*" (Oct 23, 2025) (www.bionicbusiness.com) (www.bionicbusiness.com).
11. Ema.co / Engineering in AI – Darshan Joshi, "*The State of MCP in the Enterprise*" (May 2025) (www.ema.co) (www.ema.co).

12. Sider.ai Blog – “Claude Skills at Work: 15 Real Business Workflows...” (updated Oct 17, 2025) ([sider.ai](#)) ([sider.ai](#)).
13. Tom’s Guide – Rick Broida, “Claude just got customizable ‘Skills’...” (Oct 16, 2025) ([www.tomsguide.com](#)).
14. ModelContextProtocol.io / Wiki – MCP Intro and Documentation. (2024–2026) ([colinmcnamara.com](#)) ([modelcontextprotocol.wiki](#)).
15. Simon Willison’s Weblog – further excerpts on MCP and Skills (Oct 16, 2025) ([simonwillison.net](#)) ([simonwillison.net](#)).
16. Colin McNamara – Understanding Skills, Agents, Subagents, and MCP in Claude Code: code example of a skill with a Python function ([colinmcnamara.com](#)).
17. Claude Platform Docs – Agent Skills Overview ([platform.claude.com](#)).
18. Model Context Protocol – MCP Registry ([modelcontextprotocol.io](#)).
19. HowAIWorks.ai – Anthropic Introduces Agent Skills: Customizable AI Capabilities (Oct 17, 2025) ([howaiworks.ai](#)) ([howaiworks.ai](#)).
20. Model Context Protocol – Key Changes: 2025-06-18 ([modelcontextprotocol.io](#)).
21. Linux Foundation – Agentic AI Foundation (AAIF) Announcement (Dec 9, 2025) ([linuxfoundation.org](#)).
22. Anthropic – Donating the Model Context Protocol and Establishing the Agentic AI Foundation (Dec 2025) ([anthropic.com](#)).
23. MCP Blog – MCP Apps: Bringing UI Capabilities to MCP Clients (Jan 26, 2026) ([blog.modelcontextprotocol.io](#)).
24. MCP Blog – Exploring the Future of MCP Transports (Dec 2025) ([blog.modelcontextprotocol.io](#)).
25. Adversa AI – Top MCP Security Resources – February 2026 ([adversa.ai](#)).
26. GitHub – Anthropic Skills Repository ([github.com/anthropics/skills](#)).
27. MCP Specification – Roadmap (2026) ([modelcontextprotocol.io](#)).
28. Model Context Protocol – Key Changes: 2026-07-28 ([modelcontextprotocol.io](#)).
29. MCP Blog – The 2026-07-28 Specification (Jul 28, 2026) ([blog.modelcontextprotocol.io](#)).
30. Model Context Protocol – Skills extension (SEP-2640) ([modelcontextprotocol.io](#)) and Extension Support Matrix ([modelcontextprotocol.io](#)).
31. Agent Skills – Agent Skills Overview (open standard) ([agentskills.io](#)).
32. Claude Code Docs – Connect Claude Code to tools via MCP (tool search) ([code.claude.com](#)) and Extend Claude with skills ([code.claude.com](#)).
33. Claude Platform Docs – Using Agent Skills with the API ([platform.claude.com](#)).
34. Anthropic – Introducing Agent Skills (Oct 16, 2025; updated Dec 18, 2025) ([claude.com](#)).
35. Claude Blog – Interactive connectors and MCP Apps (Jan 26, 2026) ([claude.com](#)).
36. Model Context Protocol – The MCP Registry (preview) ([modelcontextprotocol.io](#)).

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://modelcontextprotocol.io/extensions/skills/overview>
2. <https://modelcontextprotocol.io/extensions/client-matrix>
3. <https://agentskills.io/>

4. <https://code.claude.com/docs/en/mcp>
5. <https://modelcontextprotocol.io/specification/2026-07-28/changelog>
6. <https://simonwillison.net/2025/Oct/16/claude-skills/>
7. <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>
8. <https://www.anthropic.com/news/donating-the-model-context-protocol-and-establishing-of-the-agentic-ai-foundation>
9. <https://colinmcnamara.com/blog/understanding-skills-agents-and-mcp-in-claude-code>
10. <https://modelcontextprotocol.wiki/en/introduction>
11. <http://blog.modelcontextprotocol.io/posts/2025-12-09-mcp-joins-agentic-ai-foundation/>
12. <https://blog.modelcontextprotocol.io/posts/2026-07-28/>
13. <https://www.tomsguide.com/ai/claude-just-got-customizable-skills-heres-how-they-could-supercharge-your-workflow>
14. <https://www.tomsguide.com/ai/claude-skills-are-here-and-they-might-be-the-smartest-ai-feature-youre-not-using-yet>
15. <https://www.echofold.ai/news/claude-skills-announcement>
16. <https://intuitionlabs.ai/articles/agentic-ai-temporal-orchestration>
17. <https://intuitionlabs.ai/articles/what-is-context-engineering>
18. <https://www.linuxfoundation.org/press/linux-foundation-announces-the-formation-of-the-agentic-ai-foundation>
19. <https://intuitionlabs.ai/articles/zapier-vs-n8n-ai-workflows>
20. <https://claude.com/blog/equipping-agents-for-the-real-world-with-agent-skills>
21. <https://howaiworks.ai/blog/anthropic-agent-skills-announcement>
22. <https://platform.claude.com/docs/en/build-with-claude/skills-guide>
23. <https://sider.ai/th/blog/ai-tools/claude-skills-at-work-15-real-business-workflows-that-don-t-waste-your-time>
24. <https://claude.com/blog/skills>
25. <https://github.com/anthropics/skills>
26. <https://www.ema.co/blog/engineering-in-ai/the-state-of-mcp-in-the-enterprise-what-works-and-what-comes-next>
27. <https://blog.fka.dev/blog/2025-06-06-why-mcp-deprecated-sse-and-go-with-streamable-http/>
28. <https://modelcontextprotocol.io/specification/2025-03-26/changelog>
29. <https://modelcontextprotocol.io/specification/2025-06-18/changelog>
30. <http://blog.modelcontextprotocol.io/posts/2026-01-26-mcp-apps/>
31. <https://www.hckrnws.com/stories/45619537>
32. <https://modelcontextprotocol.io/development/roadmap>
33. <https://adversa.ai/blog/top-mcp-security-resources-february-2026/>
34. <https://modelcontextprotocol.io/registry>
35. <https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>
36. <https://code.claude.com/docs/en/skills>
37. <https://github.com/travisvn/awesome-claude-skills>
38. <https://claude.com/blog/interactive-tools-in-claude>
39. <https://blog.mean.ceo/anthropic-claude-news-february-2026/>
40. <https://www.bionicbusiness.com/p/issue-86-claude-skills-how-to-business>

41. <http://blog.modelcontextprotocol.io/posts/2025-12-19-mcp-transport-future/>

THE PUBLISHER

About IntuitionLabs

IntuitionLabs is an AI consulting, custom software development and data engineering firm serving pharmaceutical, biotechnology, medical-device and other life-science organizations. We work with clinical, regulatory, medical-affairs, commercial, quality and IT teams to connect technology decisions with the work people need to accomplish.

AI consulting and adoption

Our AI enablement services cover readiness assessments, use-case selection, governance and policies, team workshops, adoption measurement and ongoing advisory support. We help organizations structure the information layer behind AI: source material, context, permissions and maintained knowledge that make generated answers useful and reviewable. Private LLM inference and hosted AI options support teams evaluating how to operate AI with appropriate control over their data and infrastructure.

Software, data and life-science workflows

IntuitionLabs develops custom software for pharma and biotech, integrates enterprise systems, and builds data engineering and business intelligence solutions. Areas of focus include AI agents, regulatory research, medical writing, medical affairs, CMC information, competitive intelligence and clinical-document workflows. Our eTMF intelligence work includes cross-system reconciliation and inspection-readiness support.

Enterprise platforms and regulated delivery

We provide Veeva services, application support, managed services, integrations and custom applications, alongside enterprise content work involving platforms such as Egnyte. For regulated workflows, our services include GxP enablement, computer-system validation and software development addressing 21 CFR Part 11 requirements. The applicable controls, validation responsibilities and acceptance criteria are defined for each engagement.

Work with IntuitionLabs

Explore [AI enablement](#), [pharma and biotech software development](#), [data engineering and BI](#), and [Veeva services](#). [Contact IntuitionLabs](#) to discuss your workflow, information sources and implementation needs.

IntuitionLabs publishes educational research to help life-science teams make informed technology decisions. Coverage of a product or organization does not imply a client relationship, endorsement or partnership.

Website: <https://intuitionlabs.ai>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.