Agent Skills

Agent Skills are modular capabilities that extend Claude's functionality. Each Skill packages instructions, metadata, and optional resources (scripts, templates) that Claude uses automatically when relevant.

Copy page

Why use Skills

Skills are reusable, filesystem-based resources that provide Claude with domain-specific expertise: workflows, context, and best practices that transform general-purpose agents into specialists. Unlike prompts (conversation-level instructions for one-off tasks), Skills load on-demand and eliminate the need to repeatedly provide the same guidance across multiple conversations.

Key benefits:

- **Specialize Claude**: Tailor capabilities for domain-specific tasks
- **Reduce repetition**: Create once, use automatically
- **Compose capabilities**: Combine Skills to build complex workflows

For a deep dive into the architecture and real-world applications of Agent Skills, read our engineering blog: Equipping agents for the real world with Agent Skills.

Using Skills

Anthropic provides pre-built Agent Skills for common document tasks (PowerPoint, Excel, Word, PDF), and you can create your own custom Skills. Both work the same way. Claude automatically uses them when relevant to your request.

Pre-built Agent Skills are available to all users on claude.ai and via the Claude API. See the Available Skills section below for the complete list.

Custom Skills let you package domain expertise and organizational kno
available across Claude's products: create them in Claude Code, upload them via the API,

Claude Docs

Get started:

- For pre-built Agent Skills: See the quickstart tutorial to start using PowerPoint, Excel, Word, and PDF skills in the API
- For custom Skills: See the Agent Skills Cookbook to learn how to create your own Skills

How Skills work

Skills leverage Claude's VM environment to provide capabilities beyond what's possible with prompts alone. Claude operates in a virtual machine with filesystem access, allowing Skills to exist as directories containing instructions, executable code, and reference materials, organized like an onboarding guide you'd create for a new team member.

This filesystem-based architecture enables **progressive disclosure**: Claude loads information in stages as needed, rather than consuming context upfront.

Three types of Skill content, three levels of loading

Skills can contain three types of content, each loaded at different times:

Level 1: Metadata (always loaded)

Content type: Instructions. The Skill's YAML frontmatter provides discovery information:

```
---
name: pdf-processing
description: Extract text and tables from PDF files, fill forms, merge documen
---
```

Claude loads this metadata at startup and includes it in the system prompt. This lightweight approach means you can install many Skills without context penalty; Claude only knows each Skill exists and when to use it.

Level 2: Instructions (loaded when triggered)

Content type: Instructions. The main body of SKILL.md contains procedural knowledge: workflows, best practices, and guidance:

````
## Quick start

Use pdfplumber to extract text from PDFs:

```python
import pdfplumber

with pdfplumber.open("document.pdf") as pdf:
    text = pdf.pages[0].extract_text()
```

For advanced form filling, see [FORMS.md](FORMS.md).
````

When you request something that matches a Skill's description, Claude reads SKILL.md from the filesystem via bash. Only then does this content enter the context window.

Level 3: Resources and code (loaded as needed)

Content types: Instructions, code, and resources. Skills can bundle additional materials:

```
pdf-skill/
├── SKILL.md (main instructions)
├── FORMS.md (form-filling guide)
├── REFERENCE.md (detailed API reference)
└── scripts/
    └── fill_form.py (utility script)
```

Instructions: Additional markdown files (FORMS.md, REFERENCE.md) containing specialized guidance and workflows

Code: Executable scripts (fill_form.py, validate.py) that Claude runs via bash; scripts provide deterministic operations without consuming context

Resources: Reference materials like database schemas, API documentation, templates, or examples

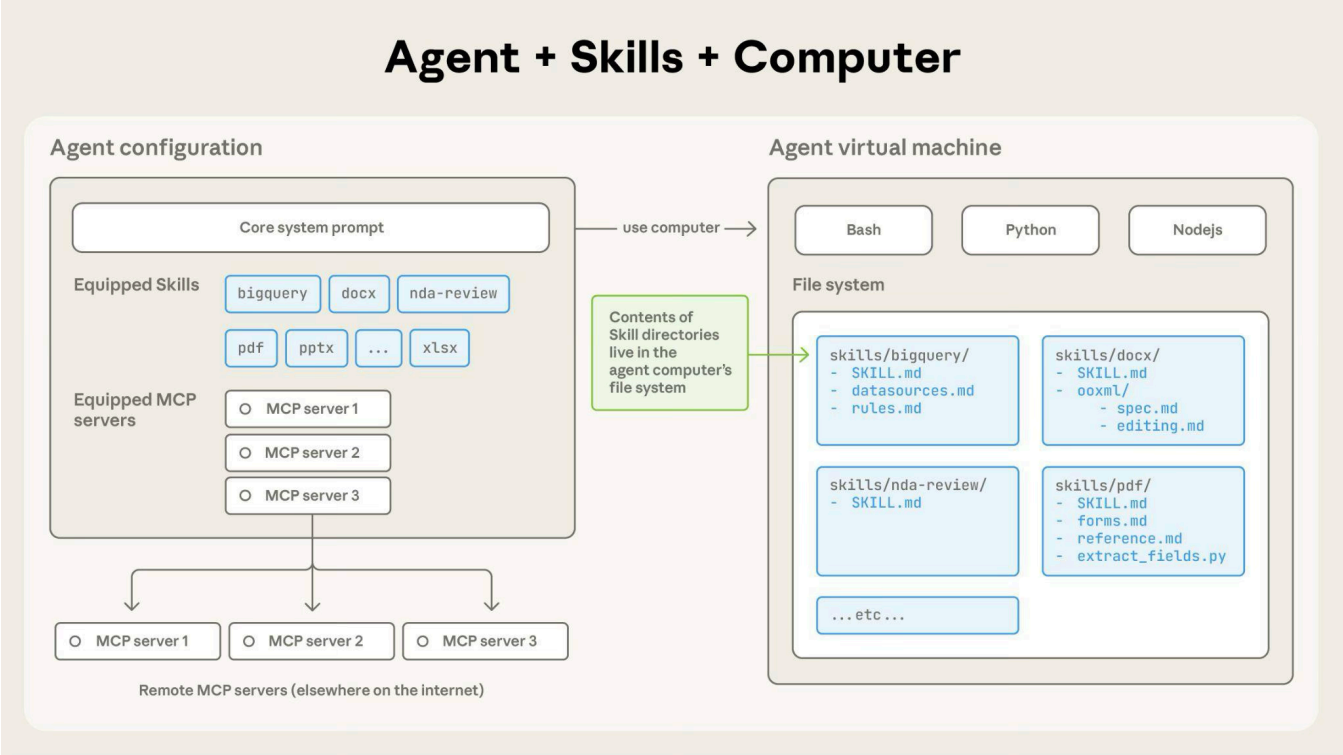
Claude accesses these files only when referenced. The filesystem model means each content type has different strengths: instructions for flexible guidance, code for reliability, resources for factual lookup.

Metadata	startup)	Skill	frontmatter
Level 2: Instructions	When Skill is triggered	Under 5k tokens	SKILL.md body with instructions and guidance
Level 3+: Resources	As needed	Effectively unlimited	Bundled files executed via bash without loading contents into context

Progressive disclosure ensures only relevant content occupies the context window at any given time.

The Skills architecture

Skills run in a code execution environment where Claude has filesystem access, bash commands, and code execution capabilities. Think of it like this: Skills exist as directories on a virtual machine, and Claude interacts with them using the same bash commands you'd use to navigate files on your computer.



How Claude accesses Skill content:

When a Skill is triggered, Claude uses bash to read SKILL.md from the filesystem, bringing its instructions into the context window. If those instructions reference other files (like FORMS.md or a database schema), Claude reads those files too using additional bash commands. When instructions mention executable scripts, Claude runs them via bash and receives only the output (the script code itself never enters context).

On-demand file access: Claude reads only the files needed for each specific task. A Skill can include dozens of reference files, but if your task only needs the sales schema, Claude loads just that one file. The rest remain on the filesystem consuming zero tokens.

Efficient script execution: When Claude runs `validate_form.py`, the script's code never loads into the context window. Only the script's output (like "Validation passed" or specific error messages) consumes tokens. This makes scripts far more efficient than having Claude generate equivalent code on the fly.

No practical limit on bundled content: Because files don't consume context until accessed, Skills can include comprehensive API documentation, large datasets, extensive examples, or any reference materials you need. There's no context penalty for bundled content that isn't used.

This filesystem-based model is what makes progressive disclosure work. Claude navigates your Skill like you'd reference specific sections of an onboarding guide, accessing exactly what each task requires.

Example: Loading a PDF processing skill

Here's how Claude loads and uses a PDF processing skill:

1. **Startup**: System prompt includes: `PDF Processing - Extract text and tables from PDF files, fill forms, merge documents`
2. **User request**: "Extract the text from this PDF and summarize it"
3. **Claude invokes**: `bash: read pdf-skill/SKILL.md` → Instructions loaded into context
4. **Claude determines**: Form filling is not needed, so FORMS.md is not read
5. **Claude executes**: Uses instructions from SKILL.md to complete the task

The diagram shows:

1. Default state with system prompt and skill metadata pre-loaded
2. Claude triggers the skill by reading SKILL.md via bash
3. Claude optionally reads additional bundled files like FORMS.md as needed
4. Claude proceeds with the task

This dynamic loading ensures only relevant skill content occupies the context window.

Where Skills work

Skills are available across Claude's agent products:

Claude API

The Claude API supports both pre-built Agent Skills and custom Skills. Both work identically: specify the relevant `skill_id` in the `container` parameter along with the code execution tool.

Prerequisites: Using Skills via the API requires three beta headers:

- `code-execution-2025-08-25` - Skills run in the code execution container
- `skills-2025-10-02` - Enables Skills functionality
- `files-api-2025-04-14` - Required for uploading/downloading files to/from the container

organization-wide.

To learn more, see Use Skills with the Claude API.

Claude Code

Claude Code supports only Custom Skills.

Custom Skills: Create Skills as directories with SKILL.md files. Claude discovers and uses them automatically.

Custom Skills in Claude Code are filesystem-based and don't require API uploads.

To learn more, see Use Skills in Claude Code.

Claude Agent SDK

The Claude Agent SDK supports custom Skills through filesystem-based configuration.

Custom Skills: Create Skills as directories with SKILL.md files in `.claude/skills/`. Enable Skills by including `"Skill"` in your `allowed_tools` configuration.

Skills in the Agent SDK are then automatically discovered when the SDK runs.

To learn more, see Agent Skills in the SDK.

Claude.ai

Claude.ai supports both pre-built Agent Skills and custom Skills.

Pre-built Agent Skills: These Skills are already working behind the scenes when you create documents. Claude uses them without requiring any setup.

Custom Skills: Upload your own Skills as zip files through Settings > Features. Available on Pro, Max, Team, and Enterprise plans with code execution enabled. Custom Skills are individual to each user; they are not shared organization-wide and cannot be centrally managed by admins.

To learn more about using Skills in Claude.ai, see the following resources in the Claude Help Center:

- What are Skills?
- Using Skills in Claude
- How to create custom Skills

Skill structure

Every Skill requires a `SKILL.md` file with YAML frontmatter:

```
---
name: your-skill-name
description: Brief description of what this Skill does and when to use it
---

# Your Skill Name

## Instructions
[Clear, step-by-step guidance for Claude to follow]

## Examples
[Concrete examples of using this Skill]
```

Required fields: `name` and `description`

Field requirements:

`name` :

- Maximum 64 characters
- Must contain only lowercase letters, numbers, and hyphens
- Cannot contain XML tags
- Cannot contain reserved words: "anthropic", "claude"

`description` :

- Must be non-empty
- Maximum 1024 characters
- Cannot contain XML tags

The `description` should include both what the Skill does and when Claude should use it. For complete authoring guidance, see the best practices guide.

Security considerations

We strongly recommend using Skills only from trusted sources: those you created yourself or obtained from Anthropic. Skills provide Claude with new capabilities through

stated purpose.

> ⚠ If you must use a Skill from an untrusted or unknown source, exercise extreme caution and thoroughly audit it before use. Depending on what access Claude has when executing the Skill, malicious Skills could lead to data exfiltration, unauthorized system access, or other security risks.

Key security considerations:

- **Audit thoroughly**: Review all files bundled in the Skill: SKILL.md, scripts, images, and other resources. Look for unusual patterns like unexpected network calls, file access patterns, or operations that don't match the Skill's stated purpose
- **External sources are risky**: Skills that fetch data from external URLs pose particular risk, as fetched content may contain malicious instructions. Even trustworthy Skills can be compromised if their external dependencies change over time
- **Tool misuse**: Malicious Skills can invoke tools (file operations, bash commands, code execution) in harmful ways
- **Data exposure**: Skills with access to sensitive data could be designed to leak information to external systems
- **Treat like installing software**: Only use Skills from trusted sources. Be especially careful when integrating Skills into production systems with access to sensitive data or critical operations

Available Skills

Pre-built Agent Skills

The following pre-built Agent Skills are available for immediate use:

- **PowerPoint (pptx)**: Create presentations, edit slides, analyze presentation content
- **Excel (xlsx)**: Create spreadsheets, analyze data, generate reports with charts
- **Word (docx)**: Create documents, edit content, format text
- **PDF (pdf)**: Generate formatted PDF documents and reports

These Skills are available on the Claude API and claude.ai. See the quickstart tutorial to start using them in the API.

For complete examples of custom Skills, see the Skills cookbook.

Limitations and constraints

Understanding these limitations helps you plan your Skills deployment effectively.

Cross-surface availability

Custom Skills do not sync across surfaces. Skills uploaded to one surface are not automatically available on others:

- Skills uploaded to Claude.ai must be separately uploaded to the API
- Skills uploaded via the API are not available on Claude.ai
- Claude Code Skills are filesystem-based and separate from both Claude.ai and API

You'll need to manage and upload Skills separately for each surface where you want to use them.

Sharing scope

Skills have different sharing models depending on where you use them:

- **Claude.ai**: Individual user only; each team member must upload separately
- **Claude API**: Workspace-wide; all workspace members can access uploaded Skills
- **Claude Code**: Personal (`~/.claude/skills/`) or project-based (`.claude/skills/`); can also be shared via Claude Code Plugins

Claude.ai does not currently support centralized admin management or org-wide distribution of custom Skills.

Runtime environment constraints

The exact runtime environment available to your skill depends on the product surface where you use it.

- **Claude.ai**:
  - **Varying network access**: Depending on user/admin settings, Skills may have full, partial, or no network access. For more details, see the Create and Edit Files support article.
- **Claude API**:

cannot install new packages during execution.

  - **Pre-configured dependencies only**: Check the code execution tool documentation for the list of available packages

- **Claude Code**:
  - **Full network access**: Skills have the same network access as any other program on the user's computer
  - **Global package installation discouraged**: Skills should only install packages locally in order to avoid interfering with the user's computer

Plan your Skills to work within these constraints.

Next steps

Get started with Agent Skills

Create your first Skill

</>

API Guide

Use Skills with the Claude API

>_

Use Skills in Claude Code ↗

Create and manage custom Skills in Claude Code

Use Skills in the Agent SDK

Use Skills programmatically in TypeScript and Python

Claude Docs

Claude Docs

Solutions

AI agents

Code modernization

Coding

Customer support

Education

Financial services

Government

Life sciences

Partners

Amazon Bedrock

Google Cloud's Vertex AI

Learn

Blog

Catalog

Courses

Use cases

Connectors

Customer stories

Engineering at Anthropic

Events

Powered by Claude

Service partners

Startups program

Company

Anthropic

Careers

Economic Futures

Research

News

Responsible Scaling Policy

Security and compliance

Transparency

Help and security

Availability

Status

Claude Docs

Terms and policies

Privacy policy

Responsible disclosure policy

Terms of service: Commercial

Terms of service: Consumer

Usage policy