Agent Skills

Use agent skills to extend Codex with task-specific capabilities. A skill packages instructions, resources, and optional scripts so Codex can follow a workflow reliably. You can share skills across teams or with the community. Skills build on the open agent skills standard.

Skills are available in both the Codex CLI and IDE extensions.

Agent skill definition

A skill captures a capability expressed through Markdown instructions in a `SKILL.md` file. A skill folder can also include scripts, resources, and assets that Codex uses to perform a specific task.

<FileTree class="mt-4" tree={[{ name: "my-skill/", open: true, children: [{ name: "SKILL.md", comment: "Required: instructions + metadata", }, { name: "scripts/", comment: "Optional: executable code", }, { name: "references/", comment: "Optional: documentation", }, { name: "assets/", comment: "Optional: templates, resources", },], },]} />

Skills use **progressive disclosure** to manage context efficiently. At startup, Codex loads the name and description of each available skill. Codex can then activate and use a skill in two ways:

1. **Explicit invocation:** You include skills directly in your prompt. To select one, run the `/skills` slash command, or start typing `$` to mention a skill. Codex web and iOS don't support explicit invocation yet, but you can still ask Codex to use any skill checked into a repo.

```
<img src="https://developers.openai.com/images/codex/skills/skills-
selector-cli-light.webp"
  alt=""
  class="block w-full lg:h-64 rounded-lg border border-default my-0
object-contain bg-[#F0F1F5] dark:hidden"
/>
<img src="https://developers.openai.com/images/codex/skills/skills-
selector-cli-dark.webp"
```

```
  alt=""
  class="hidden w-full lg:h-64 rounded-lg border border-default my-0
object-contain bg-[#1E1E2E] dark:block"
/>

<img src="https://developers.openai.com/images/codex/skills/skills-
selector-ide-light.webp"
  alt=""
  class="block w-full lg:h-64 rounded-lg border border-default my-0
object-contain bg-[#E8E9ED] dark:hidden"
/>
<img src="https://developers.openai.com/images/codex/skills/skills-
selector-ide-dark.webp"
  alt=""
  class="hidden w-full lg:h-64 rounded-lg border border-default my-0
object-contain bg-[#181824] dark:block"
/>
```

2. **Implicit invocation:** Codex can decide to use an available skill when your task matches the skill's description.

In either method, Codex reads the full instructions of the invoked skills and any extra references checked into the skill.

Where to save skills

Codex loads skills from these locations. A skill's location defines its scope.

When Codex loads available skills from these locations, it overwrites skills with the same name from a scope of lower precedence. The list below shows skill scopes and locations in order of precedence (high to low).

Skill Scope	Location	Suggested Use
REPO	`$CWD/.codex/skills` Current working directory: where you launch Codex.	If you're in a repository or code environment, teams can check in skills relevant to a working folder. For example, skills only relevant to a microservice or a module.
REPO	`$CWD/../.codex/skills` A folder above CWD when you launch Codex inside a Git repository.	If you're in a repository with nested folders, organizations can check in skills relevant to a shared area in a parent folder.
REPO	`$REPO_ROOT/.codex/`	If you're in a repository with nested fold-

Skill Scope	Location	Suggested Use
	`skills` The topmost root folder when you launch Codex inside a Git repository.	ers, organizations can check in skills relevant to everyone using the repository. These serve as root skills that any subfolder in the repository can overwrite.
`USER`	`$CODEX_HOME/skills` (macOS and Linux default: `~/.codex/skills`) Any skills checked into the user's personal folder.	Use to curate skills relevant to a user that apply to any repository the user may work in.
`ADMIN`	`/etc/codex/skills` Any skills checked into the machine or container in a shared, system location.	Use for SDK scripts, automation, and for checking in default admin skills available to each user on the machine.
`SYSTEM`	Bundled with Codex.	Useful skills relevant to a broad audience such as the skill-creator and plan skills. Available to everyone when they start Codex and can be overwritten by any layer above.

Codex supports symlinked skill folders and follows the symlink target when scanning these locations.

Create a skill

To create a new skill, use the built-in `$skill-creator` skill in Codex. Describe what you want your skill to do, and Codex will start bootstrapping your skill.

If you also install `$create-plan` (experimental) with `$skill-installer install the create-plan skill from the .experimental folder`, Codex will create a plan for your skill before it writes files.

For a step-by-step guide, see Create custom skills.

You can also create a skill manually by creating a folder with a `SKILL.md` file inside a valid skill location. A `SKILL.md` must contain a `name` and `description` to help Codex select the skill:

```
---
name: skill-name
description: Description that helps Codex select the skill
metadata:
```

```
  short-description: Optional user-facing description
---

Skill instructions for the Codex agent to follow when using this
        skill.
```

Codex skills build on the agent skills specification. Check out the documentation to learn more.

Install new skills

To install more than the built-in skills, you can download skills from a curated set of skills on GitHub using the `$skill-installer` skill:

```
$skill-installer install the linear skill from the .experimental
        folder
```

You can also prompt the installer to download skills from other repositories.

After installing a skill, restart Codex to pick up new skills.

Skill examples

Plan a new feature

`$create-plan` is an experimental skill that you can install with `$skill-installer` to have Codex research and create a plan to build a new feature or solve a complex problem:

```
$skill-installer install the create-plan skill from the .experimental
        folder
```

Access Linear context for Codex tasks

```
$skill-installer install the linear skill from the .experimental
        folder
```

Have Codex access Notion for more context

```
$skill-installer notion-spec-to-implementation
```