

Skills in practice ✨

Three complete skills to copy today: the diff summary with dynamic context, team conventions that load themselves, the tool with a bundled script.

Updated: July 2026 · Source: official Anthropic documentation (code.claude.com/docs) · Vol. 9 (extra) of the series — theory in vol. 2.

1. The anatomy in 60 seconds

one directory, one SKILL.md

A skill is a directory containing a `SKILL.md`: YAML frontmatter between `---` (tells Claude **when** to use it) and a markdown body (the instructions it follows **when** it uses it). The directory name becomes the command you type; the `description` decides when Claude loads it on its own. The body enters context only on invocation: long material costs almost nothing until it's needed. The old custom commands have been **merged into skills**: `.claude/commands/deploy.md` and `.claude/skills/deploy/SKILL.md` both create `/deploy` (vol. 11).

Level	Path	Applies to
Personal	<code>~/.claude/skills/<name>/SKILL.md</code>	All your projects
Project	<code>.claude/skills/<name>/SKILL.md</code>	That project only (commit it: the team uses it)
Plugin	<code><plugin>/skills/<name>/SKILL.md</code>	Wherever the plugin is enabled, as <code>/plugin:name</code>
Enterprise	managed settings	The whole organization (wins over the others)

The directories are **watched**: new or edited skills take effect immediately, no restart. On a name clash: enterprise > personal > project, and a skill of yours replaces the bundled one with the same name.

2. Example 1: the diff summary

dynamic context with `!`command``

```
# ~/.claude/skills/summarize-changes/SKILL.md
---
description: Summarizes uncommitted changes and
  flags the risks. Use it when the user asks what
  changed, wants a commit message or a review
  of their diff.
---

## Current changes

!`git diff HEAD`

## Instructions

Summarize the changes above in two or three
points, then list the risks you notice: missing
error handling, hardcoded values, tests to update.
If the diff is empty, say there are no changes.
```

Why it works

- The line `!`git diff HEAD`` is **dynamic context injection**: Claude Code runs the command *before* Claude reads the skill and replaces the line with the output — Claude receives the real diff, not the command. Multi-line: fenced block opened with ````!`.
- The `description` contains the phrases you would actually say ("what changed", "commit message"): that's what Claude matches on for automatic invocation.

Try it 2 ways

Edit any file and ask *"What did I change?"* (automatic invocation) or type `/summarize-changes` (direct). Same skill, two triggers.

3. Example 2: knowledge that loads itself

user-invocable: false + paths

```
# .claude/skills/api-conventions/SKILL.md
---
name: api-conventions
description: API design patterns for this
  repo. Applies when writing or modifying
  endpoints.
user-invocable: false
paths:
  - "src/api/**"
---

When writing API endpoints:
- RESTful resource naming
```

- Consistent error format
- Validation of incoming requests

The "reference content" pattern

- It's background knowledge, not an action: `user-invocable: false` hides it from the `/` menu — only Claude invokes it, when needed.
- `paths` limits automatic activation to files matching the globs: API conventions don't pollute the context while you work on CSS.
- Alternative to CLAUDE.md: when a CLAUDE.md section has become a *procedure* more than a fact, move it into a skill — you pay only when it's used. And keep the body lean: once invoked it **stays in context** for the whole session.

4. Example 3: the skill with the script inside

supporting files + `{CLAUDE_SKILL_DIR}`

```
# ~/.claude/skills/codebase-visualizer/
#   ├── SKILL.md
#   └── scripts/visualize.py
---
name: codebase-visualizer
description: Generates an interactive tree
  view of the codebase. Use it to explore a
  new repo or find the large files.
allowed-tools: Bash(python3 *)
---

Run the script from the project root:

```bash
python3 ${CLAUDE_SKILL_DIR}/scripts/visualize.py .
```

Creates `codebase-map.html` and opens it in the browser.
```

The "bundled script" pattern

- The script does the heavy lifting, Claude orchestrates. Works for any visual output: dependency graphs, coverage, DB schemas.
- `${CLAUDE_SKILL_DIR}` expands to the skill's directory: the script path is right wherever it's installed.
- `allowed-tools: Bash(python3 *)` pre-approves only what's needed; everything else follows normal permissions.
- Other useful files: `reference.md`, `examples/`, `templates` — cite them from SKILL.md so Claude knows when to open them. Keep SKILL.md under 500 lines.

5. The frontmatter that matters

all optional, description recommended

| Field | Effect |
|--|--|
| <code>description / when_to_use</code> | How Claude decides whether to load it on its own. Key use case first: in the listing the combined text is truncated at 1,536 characters. |
| <code>disable-model-invocation / user-invocable</code> | The first set to <code>true</code> = only you (deploy, commit — you pick the moment); the second set to <code>false</code> = only Claude (background knowledge, out of the <code>/</code> menu). |
| <code>allowed-tools / disallowed-tools</code> | Pre-approves tools while the skill is active / removes them from the pool (until your next message). |
| <code>argument-hint / arguments</code> | Hint in autocomplete (e.g. <code>[issue-number]</code>) / names for positional arguments (vol. 11). |
| <code>context: fork + agent</code> | Runs the skill in an isolated subagent; <code>agent</code> picks which one (e.g. <code>Explore</code>). Only for skills with an explicit task. |
| <code>model / effort / paths / hooks</code> | Model or effort override for the turn / globs that limit automatic activation / hooks scoped to the skill. |

► If something goes wrong

Not triggering? Put in the description the words the user would actually say; with too many skills the listing gets shortened — `/doctor` tells you which descriptions are cut. Broken YAML = skill without metadata: `/name` works, auto-invocation doesn't.

Triggering too much? More specific description or `disable-model-invocation: true`. To measure it properly: `/plugin install skill-creator@claude-plugins-official` runs A/B evals with and without the skill.

Sources: code.claude.com/docs — skills (getting started, frontmatter reference, advanced patterns, troubleshooting) · Open standard: agentskills.io