Skills in pratica ✦

Tre skill complete da copiare oggi: il riassunto del diff con contesto dinamico, le convenzioni di team che si caricano da sole, il tool con script incluso.

Aggiornato: luglio 2026 · Fonte: documentazione ufficiale Anthropic (code.claude.com/docs) · Vol. 9 (extra) della serie — teoria nel vol. 2.

1. L'anatomia in 60 secondi

una directory, un SKILL.md

Una skill è una directory con dentro un `SKILL.md`: frontmatter YAML tra `---` (dice a Claude **quando** usarla) e corpo markdown (le istruzioni che segue **quando** la usa). Il nome della directory diventa il comando che digiti; la `description` decide quando Claude la carica da solo. Il corpo entra in contesto solo all'invocazione: materiale lungo non costa quasi nulla finché non serve. I vecchi custom command sono stati **fusi nelle skill**: `.claude/commands/deploy.md` e `.claude/skills/deploy/SKILL.md` creano entrambi `/deploy` (vol. 11).

Livello	Path	Vale per
Personal	`~/.claude/skills/<nome>/SKILL.md`	Tutti i tuoi progetti
Project	`.claude/skills/<nome>/SKILL.md`	Solo quel progetto (committalo: lo usa il team)
Plugin	`<plugin>/skills/<nome>/SKILL.md`	Dove il plugin è abilitato, come `/plugin:nome`
Enterprise	managed settings	Tutta l'organizzazione (vince sugli altri)

Le directory sono **osservate**: skill nuove o modificate valgono subito, senza riavvio. A parità di nome: enterprise > personal > project, e una skill tua sostituisce quella bundled omonima.

2. Esempio 1: il riassunto del diff

contesto dinamico con ! `comando`

```
# ~/.claude/skills/summarize-changes/SKILL.md
---
description: Riassume le modifiche non committate
  e segnala i rischi. Usala quando l'utente chiede
  cosa è cambiato, vuole un commit message o una
  review del suo diff.
---

## Modifiche correnti

!`git diff HEAD`

## Istruzioni

Riassumi le modifiche qui sopra in due o tre punti,
poi elenca i rischi che noti: error handling
mancante, valori hardcoded, test da aggiornare.
Se il diff è vuoto, dì che non ci sono modifiche.
```

Perché funziona

- La riga `!`git diff HEAD`` è **dynamic context injection**: Claude Code esegue il comando *prima* che Claude legga la skill e sostituisce la riga con l'output — Claude riceve il diff reale, non il comando. Multi-riga: blocco recintato aperto con ```` ```! ````.
- La `description` contiene le frasi che diresti davvero ("cosa è cambiato", "commit message"): è ciò su cui Claude matcha per l'invocazione automatica.

Provala in 2 modi

Modifica un file qualsiasi e chiedi "*Cosa ho cambiato?*" (invocazione automatica) oppure digita `/summarize-changes` (diretta). Stessa skill, due inneschi.

3. Esempio 2: conoscenza che si carica da sola

user-invocable: false + paths

```
# .claude/skills/api-conventions/SKILL.md
---
name: api-conventions
description: Pattern di design API di questo
  repo. Si applica quando si scrivono o
  modificano endpoint.
user-invocable: false
paths:
  - "src/api/**"
---

Quando scrivi endpoint API:
- Naming RESTful delle risorse
```

```
- Formato errori consistente
- Validazione delle richieste in ingresso
```

Il pattern "reference content"

- È conoscenza di fondo, non un'azione: `user-invocable: false` la nasconde dal menu `/` — solo Claude la invoca, quando serve.
- `paths` limita l'attivazione automatica ai file che matchano i glob: le convenzioni API non inquinano il contesto mentre lavori sul CSS.
- Alternativa a CLAUDE.md: quando una sezione di CLAUDE.md è diventata una *procedura* più che un fatto, spostala in una skill — si paga solo quando si usa. E tieni il corpo asciutto: una volta invocata **resta in contesto** per tutta la sessione.

4. Esempio 3: la skill con lo script dentro

supporting files + ${CLAUDE_SKILL_DIR}

```
# ~/.claude/skills/codebase-visualizer/
#   ├─ SKILL.md
#   └─ scripts/visualize.py
---
name: codebase-visualizer
description: Genera una vista ad albero
  interattiva del codebase. Usala per esplorare
  un repo nuovo o trovare i file grossi.
allowed-tools: Bash(python3 *)
---

Esegui lo script dalla root del progetto:

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

Crea `codebase-map.html` e lo apre nel browser.
```

Il pattern "script bundled"

- Lo script fa il lavoro pesante, Claude orchestra. Funziona per qualsiasi output visuale: grafi di dipendenze, coverage, schemi DB.
- `${CLAUDE_SKILL_DIR}` si espande nella directory della skill: il path dello script è giusto ovunque sia installata.
- `allowed-tools: Bash(python3 *)` pre-approva solo ciò che serve; tutto il resto segue i permessi normali.
- Altri file utili: `reference.md`, `examples/`, template — citali da SKILL.md così Claude sa quando aprirli. SKILL.md sotto le 500 righe.

5. Il frontmatter che conta

tutto opzionale, description raccomandata

Campo	Effetto
`description` / `when_to_use`	Come Claude decide se caricarla da solo. Caso d'uso chiave all'inizio: in listing il testo combinato è troncato a 1.536 caratteri.
`disable-model-invocation` / `user-invocable`	Il primo a `true` = solo tu (deploy, commit — decidi tu il momento); il secondo a `false` = solo Claude (conoscenza di fondo, via dal menu `/`).
`allowed-tools` / `disallowed-tools`	Pre-approva tool mentre la skill è attiva / li toglie dal pool (fino al tuo prossimo messaggio).
`argument-hint` / `arguments`	Hint in autocomplete (es. `[issue-number]`) / nomi per gli argomenti posizionali (vol. 11).
`context: fork` + `agent`	Esegue la skill in un subagente isolato; `agent` sceglie quale (es. `Explore`). Solo per skill con un compito esplicito.
`model` / `effort` / `paths` / `hooks`	Override di modello o effort per il turno / glob che limitano l'attivazione automatica / hook scoped alla skill.

▶ Se qualcosa non va

Non si attiva? Metti nella description le parole che l'utente direbbe davvero; con troppe skill il listing viene accorciato — `/doctor` dice quali description sono tagliate. YAML rotto = skill senza metadata: `/nome` funziona, l'auto-invocazione no. **Si attiva troppo?** Description più specifica o `disable-model-invocation: true`. Per misurarla sul serio: `/plugin install skill-creator@claude-plugins-official` fa eval A/B con e senza skill.

Fonti: code.claude.com/docs — skills (getting started, frontmatter reference, advanced patterns, troubleshooting) · Standard aperto: agentskills.io