4. Configurazione Claude Code

Permessi, comandi, skill e hook.

- Permessi, comandi, skill e hook

Permessi, comandi, skill e hook

?? Permessi in `.claude/settings.json`

Definisce cosa Claude Code può fare senza chiedere e cosa non deve mai toccare.

```
{
  "permissions": {
    "allow": [
      "Bash(dotnet build:*)",
      "Bash(dotnet test:*)",
      "Bash(dotnet format:*)"
    ],
    "deny": [
      "Read(./.env)",
      "Edit(./migrations/**)",
      "Edit(./appsettings.Production.json)"
    ]
  },
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [ { "type": "command", "command": "dotnet format --verify-no-changes" } ]
      }
    ]
  }
}
```

⚠ Gli esempi sono indicativi: adattare comandi e percorsi al progetto e verificare la sintassi aggiornata nella documentazione ufficiale di Claude Code.

?? Comandi personalizzati — `.claude/commands/`

Un file markdown per ogni prompt ripetuto. Esempio `review.md`, richiamabile con `/review`:

```
Esegui una code review delle modifiche non ancora committate.
Controlla: convenzioni in docs/conventions.md, query SQL parametrizzate,
test mancanti. Elenca i problemi per gravità, senza modificare file.
```

? Skill — `.claude/skills/`

Procedure a più passaggi. Esempio `.claude/skills/nuovo-modulo/SKILL.md`:

```
---
name: nuovo-modulo
description: Aggiunge un nuovo modulo seguendo la struttura e le convenzioni del progetto.
---
1. Crea la cartella in src/ seguendo docs/architecture.md
2. Aggiungi i test in tests/
3. Esegui scripts/test.ps1 e correggi finché è verde
```

? Hook

Eseguono comandi in automatico (formattazione, test) dopo le modifiche: vedi la sezione `hooks` nell'esempio sopra.

? CLAUDE.md nelle sottocartelle

Un `CLAUDE.md` in una sottocartella viene caricato solo quando l'agente lavora lì: utile nei monorepo.