

Prompt: hooks-automazione

Sottoinsieme dei prompt del libro filtrati per tema: hooks-automazione Capitoli inclusi: Capitolo 5, Capitolo 13

Prompt 1

```
chmod +x ~/.claude/hooks/session-context.sh
chmod +x ~/.claude/hooks/rich-context.sh
```

Prompt 2

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "~/.claude/hooks/safety-check.sh",
            "timeout": 5
          }
        ]
      }
    ]
  }
}
```

Prompt 3

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Write|Edit",
        "hooks": [
          {
            "type": "http",
            "url": "http://localhost:3000/webhook/file-changed",
            "timeout": 5
          }
        ]
      }
    ]
  }
}
```

Prompt 4

```
#!/bin/bash
# Log semplice delle istruzioni caricate
echo "$(date +%H:%M:%S) loaded: $(jq -r '.file_path // "?"') >> ~/.claude/debug.log
exit 0
```

Prompt 5

```
#!/bin/bash
# Hook minimo per contesto da git
BRANCH=$(git branch --show-current 2>/dev/null)
MODIFIED=$(git status --short 2>/dev/null)
jq -n --arg ctx "Branch: ${BRANCH}\nFile modificati:\n${MODIFIED}" '{
  hookSpecificOutput: { hookEventName: "SessionStart", additionalContext: $ctx }
}'
```

Prompt 6

```
# ~/.claude/hooks/env-context.sh
ENV=${NODE_ENV:-development}
MSG="Ambiente: ${ENV}"
[ "$ENV" = "production" ] && MSG="PRODUZIONE: operare con cautela"
```

```
jq -n --arg m "$MSG" '{
  hookSpecificOutput: { hookEventName: "SessionStart", additionalContext: $m }
}'
```

Prompt 7

/loop 5m controlla se il deploy su Vercel è andato a buon fine e dimmi l'esito

Prompt 8

/loop 30m esegui npm run test e dimmi se ci sono test falliti

Prompt 9

/loop controlla se la CI è passata e se ci sono nuovi commenti di review sulla PR

Prompt 10

Controlla la PR del branch `release/next`.

- Se la CI è rossa: scarica il log del job fallito, diagnostica, e fai un push con un fix minimo
- Se ci sono nuovi commenti di review: rispondili e chiudi i thread
- Se tutto è verde e silenzioso: dì solo "tutto OK" in una riga

Prompt 11

Tra 45 minuti, controlla se i test di integrazione sono passati e dimmi l'esito.

Prompt 12

Alle 16:30, ricordami di fare il push del branch feature/auth prima che scada la finestra di deploy.

Prompt 13

Crea un task che ogni lunedì mattina alle 9:00 mi dia un riassunto dei commit sul branch main della settimana precedente.

Prompt 14

Che task schedulati ho attivi in questa sessione?

Prompt 15

Cancella il task che controlla i test ogni 30 minuti.

Prompt 16

/schedule daily PR review at 9am

Prompt 17

/schedule ogni lunedì mattina alle 8, genera un report dei commit della settimana precedente

Prompt 18

/schedule in 2 weeks, apri una PR che rimuove il feature flag di autenticazione

Prompt 19

/schedule run nome-routine

Prompt 20

/schedule domani alle 9:00, scrivi un riassunto dei commit merged nella settimana corrente e apri una PR con il file CHANGELOG.md aggiornato

Prompt 21

```
claude --worktree feature-auth
```

Prompt 22

```
claude --worktree bugfix-123
```

Prompt 23

```
claude --worktree
# Crea: worktree "bright-running-fox" (nome generato automaticamente)
```

Prompt 24

```
git remote set-head origin -a
```

Prompt 25

```
git remote set-head origin develop
```

Prompt 26

```
echo ".claude/worktrees/" >> .gitignore
```

Prompt 27

```
claude --worktree react19-upgrade
```

Prompt 28

Aggiorna React dalla versione 18 alla 19 in questo progetto.

- Aggiorna package.json con le nuove versioni
- Esegui npm install
- Risolvi tutti gli errori di tipo e i breaking change
- Esegui l'intera suite di test e correggi i test falliti
- Fammi un report finale con: le modifiche necessarie, i test che erano già falliti prima, e qualsiasi deprecation che dovrei sapere

Prompt 29

Lavora su questi tre task in parallelo, usando un worktree isolato per ognuno:

1. Implementa il modulo di autenticazione
2. Scrivi i test di integrazione per le API esistenti
3. Aggiorna la documentazione con le nuove route

Prompt 30

```
{
  "permissions": {
    "allow": [
      "Bash(npm run test)",
      "Bash(npm run lint)",
      "Bash(git add .)",
      "Bash(git commit -m *)",
      "Bash(git push origin claude/*)",
      "Read(**/*.ts)",
      "Edit(**/*.ts)"
    ],
    "deny": [
      "Bash(rm -rf *)",
      "Bash(git push --force *)",
      "Bash(git push origin main)",
      "Bash(git push origin master)",
      "Bash(curl *)",
      "Read(./env)",
      "Read(./env.*)",
    ],
    "defaultMode": "acceptEdits"
  }
}
```

Prompt 31

```
{
  "model": "claude-sonnet-4-6",
  "effortLevel": "high"
}
```

Prompt 32

```
{
  "model": "claude-sonnet-4-6",
  "effortLevel": "high",
  "permissions": {
    "allow": [
      "Bash(npm run *)",
      "Bash(npx *)",
      "Bash(git add *)",
      "Bash(git commit -m *)",
      "Bash(git push origin claude/*)",
      "Bash(git checkout -b claude/*)",
      "Read(**/*.ts)",
      "Read(**/*.tsx)",
      "Read(**/*.json)",
      "Edit(**/*.ts)",
      "Edit(**/*.tsx)"
    ],
    "deny": [
      "Bash(rm -rf *)",
      "Bash(git push --force *)",
      "Bash(git push origin main)",
      "Bash(git push origin master)",
      "Bash(curl * | bash)",
      "Bash(wget * -O- | bash)",
      "Read(./env*)",
      "Read(./secrets/**)"
    ],
    "ask": [
      "Bash(git push origin *)"
    ],
    "defaultMode": "acceptEdits"
  },
  "cleanupPeriodDays": 14
}
```

Prompt 33

```
# Aggiungere al .gitignore del progetto
echo ".claude/settings.local.json" >> .gitignore
```

Prompt 34

/schedule ogni venerdì alle 17:00, leggi tutti i commit degli ultimi 7 giorni sul branch main, raggruppa le modifiche per tipo (feature, fix, refactor), e aggiorna il file CHANGELOG.md con una nuova sezione per la settimana corrente. Committa e fai una PR con il titolo "chore: update CHANGELOG week [N]".

Prompt 35

```
/loop 2h
Controlla se il numero di issue aperte su GitHub è sceso sotto 10.
Se sì: avvisami con un messaggio "Target raggiunto: X issue aperte" e fermati.
Se no: scrivi solo quante issue ci sono e riprova tra 2 ore.
```

Prompt 36

/loop 3m controlla l'output di `vercel --logs` per eventuali errori o il messaggio "Deployment complete". Dimmi lo stato a ogni check. Quando vedi "Deployment complete" o un errore critico, fermati e avvisami.

Prompt 37

/schedule ogni giorno alle 8:30, analizza il file src/index.ts e controlla se tutte le funzioni esportate sono documentate nel README.md. Per quelle mancanti, aggiungi una sezione di documentazione minima con la firma della funzione e un esempio. Apri una PR se ci sono modifiche.

Prompt 38

/schedule ogni giorno alle 7:30, leggi tutte le issue GitHub aperte senza etichetta. Per ognuna: aggiungi l'etichetta appropriata tra "bug", "enhancement", "documentation", "question". Se non riesci a categorizzarla con certezza, aggiungi "needs-triage". Scrivi un commento breve spiegando la categorizzazione. Non chiudere nessuna issue.

Prompt 39

/schedule ogni giorno alle 22:00, leggi tutti i file .md modificati oggi nella cartella ~/note/. Crea un file ~/note/riassunti/[data]-riassunto.md con: i concetti chiave emersi oggi, le decisioni prese, e i task in sospeso che dovrò riprendere domani. Non modificare i file originali.

Prompt 40

claude --worktree ts-migration

Prompt 41

/loop 20m
Continua la migrazione dei file da JavaScript a TypeScript.
- Lavora un file alla volta nella cartella src/
- Dopo ogni file: esegui tsc --noEmit per verificare che compili
- Se ci sono errori di tipo: risolvi prima di passare al file successivo
- Fermati dopo 5 ore di lavoro totale e fammi un report di avanzamento

Prompt 42

Elenca tutti i task schedulati attivi in questa sessione con i loro ID e la prossima esecuzione pianificata.

Prompt 43

Perché il task schedulato "review-pr" ha fallito l'ultima volta?
Mostrami l'errore e suggeriscimi come configurare i permessi in settings.json per permettere questa operazione.

Prompt 44

Cancella tutti i task schedulati attivi in questa sessione.

Prompt 45

```
# Controlla se il repository ha un remote configurato
git remote -v
```

```
# Controlla lo stato dei worktrees esistenti
git worktree list
```

```
# Rimuovi worktrees orfani se necessario
git worktree prune
```

Prompt 46

Il loop che ho avviato sta modificando file che non dovrebbe toccare. Fermalo, mostrami cosa ha fatto nelle ultime 3 iterazioni, e suggerisci come riscrivere il prompt per limitare le sue azioni.

Prompt 47

Dimmi quali task schedulati sono attivi in questa sessione, qual è la frequenza di ognuno, e fai una stima del numero di chiamate API che genereranno nelle prossime 24 ore.

Prompt 48

Esegui questa operazione una volta sola e dimmi cosa faresti se fosse schedulata:
[incolla il prompt che vorresti usare nel loop]
Non fare niente – mostrami solo il piano.