

Prompt: mcp

Sottoinsieme dei prompt del libro filtrati per tema: mcp Capitoli inclusi: Capitolo 5, Capitolo 9

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

Sostituisci ``/percorso/cartella`` con il percorso reale della cartella a cui vuoi dare accesso. Esempi:

- Mac/Linux: ``/home/nomeutente/Documenti`` o ``~/Documenti``
- Windows (WSL): ``/mnt/c/Users/nomeutente/Documents``
- Windows (nativo): usa la versione Claude Desktop che gestisce i percorsi Windows

Se vuoi dare accesso a più cartelle, aggiungi più percorsi separati da spazio alla fine del comando:

Prompt 8

****Autenticazione****

Nessuna. Il controllo di accesso è il percorso stesso: specificare una cartella significa che Claude può leggere e scrivere solo in quella cartella e nelle sue sottocartelle.

****Esempio di utilizzo****

Immagina di avere una cartella con cinquanta file di testo – appunti presi negli ultimi sei mesi, in modo disordinato, senza una struttura precisa. Con filesystem puoi dire:

Prompt 9

Claude leggerà i file uno per uno, estrarrà le informazioni richieste, e creerà il file di riepilogo – tutto senza che tu debba aprire un singolo documento.

Prompt 10

[PRATICA] Apri il terminale, installa filesystem con il comando sopra puntando alla tua cartella Documenti, poi chiedi a Claude di listarti i dieci file modificati più di recente. Verifica che risponda con i nomi reali dei tuoi file. Se i nomi corrispondono a file che conosci, il MCP funziona correttamente. Poi chiedi di aprire il più recente e fare un breve riassunto del suo contenuto – questo testa sia la lettura della lista che la lettura del contenuto di un singolo file.

****[DIETRO] Come funziona l'accesso ai file****

Quando Claude usa filesystem, invia comandi al server MCP locale (il processo npx che hai avviato). Questo server ha accesso al filesystem reale del tuo computer, ma solo nella cartella che hai specificato. Claude non vede il file system completo – vede solo quello che il server MCP decide di esporre, e il server MCP espone solo la cartella che gli hai indicato.

Questo è il meccanismo di sicurezza fondamentale: la "sandbox" non è gestita da Claude ma dal server MCP che fa da intermediario. Se dici a Claude "leggi /etc/password" e hai dato accesso solo a ~/Documenti, il server MCP rifiuterà l'operazione.

****Trappole****

- [!] Non dare accesso a ``/`` (radice del filesystem Mac/Linux) o a ``C:\`` (radice Windows) – limita sempre il percorso a una cartella specifica. Dare accesso all'intero disco significa che Claude potrebbe leggere file di sistema, password, configurazioni sensibili.
- [!] filesystem non ha cestino – i file eliminati tramite Claude Code spariscono definitivamente (non vanno nel Cestino del sistema operativo). Prima di chiedere operazioni di cancellazione massiva, fai un backup.
- [!] Le operazioni di scrittura sono immediate. "Modifica tutti i .doc in questo formato" cambia i file senza passare da "salva come bozza". Chiedi sempre prima un'anteprima delle modifiche.

****Una sessione di lavoro reale con filesystem****

Ecco come si presenta una sessione tipica dopo aver installato filesystem. Il contesto: hai una cartella ~/Clienti con sottocartelle per ciascun cliente, e ogni sottocartella contiene proposte, contratti, e comunicazioni salvate come file di testo.

Utente: "Guarda la cartella ~/Clienti e dimmi quali clienti non hanno avuto aggiornamenti negli ultimi 30 giorni."

Claude accede al filesystem, lista le cartelle, controlla le date di modifica dei file, e risponde con un elenco dei clienti "inattivi". Poi puoi continuare:

"Per ciascuno di questi clienti, leggi l'ultima comunicazione e proponi una email di follow-up personalizzata."

Claude legge i file, estrae il contesto dell'ultima interazione, e genera bozze di email personalizzate per ciascun cliente – il tutto senza che tu abbia dovuto aprire un singolo file. Questa è la differenza concreta che filesystem introduce nel lavoro quotidiano.

****Troubleshooting filesystem****

Se ricevi errore "ENOENT: no such file or directory": il percorso che hai specificato nel comando di installazione non esiste o è scritto in modo sbagliato. Verifica il percorso con `ls ~/Documenti` (Mac/Linux) o aprendo Esplora file (Windows).

Se ricevi errore "Permission denied": hai tentato di accedere a un file protetto dal sistema. filesystem rispetta i permessi del filesystem del sistema operativo – se un file non è leggibile dal tuo utente, Claude non può leggerlo.

Se Claude dice "non ho accesso a quella cartella": il percorso che stai chiedendo non è incluso nella cartella che hai specificato all'installazione. Puoi reinstallare il MCP aggiungendo il nuovo percorso:

Prompt 11

****Alternativa se filesystem non funziona****

Claude Code ha tool file integrati (Read, Write, Edit) che operano sulla directory corrente del progetto. Non richiedono configurazione MCP ma hanno scope più limitato: funzionano bene per progetti con una struttura di cartelle definita, meno bene per accedere a file sparsi in luoghi diversi del computer.

9.2 puppeteer

****Cosa fa****

puppeteer apre un browser Chrome reale – invisibile, controllato da Claude – e gli permette di navigare pagine web, fare clic su elementi, compilare form, catturare screenshot, ed eseguire JavaScript. È il MCP per tutto quello che richiede "guardare una pagina web" e interagire con essa come farebbe un essere umano con un mouse e una tastiera.

Le pagine web che richiedono interazione (clic su tab, scorrimento, login) sono inaccessibili ai normali strumenti di scraping perché il loro contenuto viene caricato dinamicamente dopo che l'utente interagisce. puppeteer supera questo limite perché usa un browser vero.

Usalo per: fare screenshot di siti per poi analizzarli, compilare form ripetitivi, testare se una pagina funziona correttamente, estrarre dati da siti che richiedono interazione, automatizzare flussi web che esegui manualmente ogni settimana.

[VER 2026-05-03 – <<https://github.com/modelcontextprotocol/servers>>] Il pacchetto npm è `@modelcontextprotocol/server-puppeteer`. Nota: il progetto modelcontextprotocol lo classifica come "reference implementation" – un esempio di come costruire server MCP, non un prodotto enterprise. In pratica funziona bene per usi personali.

****Installazione****

Prompt 12

Il primo avvio richiede qualche minuto perché Puppeteer scarica automaticamente una versione di Chromium (circa 200 MB). Questo accade una sola volta.

****Autenticazione****

Nessuna. Se devi accedere a siti protetti da login, puoi usare puppeteer per fare il login interattivamente (fornendo le credenziali nel prompt) ma questa operazione resta sotto la tua responsabilità e va verificata con i termini di servizio del sito.

****Esempio di utilizzo****

Un caso d'uso immediato: hai un portale web aziendale o un gestionale che aggiorni manualmente ogni settimana con gli stessi dati. Con puppeteer puoi automatizzare:

Prompt 13

Sostituisci `la-tua-api-key` con la chiave che hai copiato dalla dashboard firecrawl.dev. La chiave ha il formato `fc-xxxxxxxxxxxxxxxxxxxxxxxxxxxx`. Se la dimentichi, puoi sempre generarne una nuova dalla dashboard – le chiavi precedenti vengono disattivate automaticamente.

****Autenticazione****

[VER 2026-05-03 – <<https://firecrawl.dev>>] API key gratuita su firecrawl.dev. Il piano free offre 500 crediti al mese. Una pagina singola costa 1 credito, un crawling di un sito piccolo (10-20 pagine) costa 10-20 crediti. Per uso personale moderato, il piano gratuito è ampiamente sufficiente.

****Esempio di utilizzo****

Ricerca e analisi competitiva:

Prompt 14

Monitoraggio di contenuti:

Prompt 15

Crawling di un intero sito per documentazione:

Prompt 16

[PRATICA] Scegli un sito web che leggi regolarmente (un blog, un quotidiano, una newsletter). Chiedi a Claude di estrarre i titoli degli ultimi 10 articoli con un breve sommario per ciascuno. Confronta il tempo impiegato con quello che ci vorresti tu manualmente.

****[DIETRO] Come firecrawl gestisce i siti difficili****

I siti web moderni usano spesso JavaScript per caricare contenuti dopo il rendering iniziale: le notizie, le liste prodotti, i commenti arrivano tramite chiamate API successive all'apertura della pagina. Un parser HTML semplice vede solo lo scheletro vuoto. firecrawl risolve questo problema usando browser headless nel suo backend – ma tu non devi gestire niente di tutto questo, avviene automaticamente. In più, firecrawl ha sistemi per aggirare le protezioni anti-bot più comuni (non per violare termini di servizio, ma per leggere contenuti pubblicamente accessibili).

La differenza rispetto al WebFetch integrato in Claude Code è sostanziale: WebFetch legge HTML grezzo (veloce ma limitato), firecrawl renderizza la pagina completa e restituisce solo il testo pulito (più lento ma molto più preciso con siti dinamici).

****Trappole****

- [!] Il piano free ha 500 crediti mensili. Un crawling di un sito grande può consumare molti crediti rapidamente. Prima di fare crawling, stima le pagine: usa `firecrawl\_map` per vedere l'elenco degli URL, poi decidi se procedere.
- [!] Alcuni contenuti dinamici che si caricano solo dopo interazione specifica (clic su un tab, compilazione di un form) non vengono estratti. Per questi casi usa puppeteer.
- [!] firecrawl può accedere solo a contenuti pubblicamente visibili (o accessibili con autenticazione che fornisci). Non è adatto per siti che richiedono login complessi con 2FA.

****Rassegna stampa automatizzata: firecrawl in azione****

Uno dei workflow più utili per professionisti che devono tenersi aggiornati: la rassegna stampa giornaliera automatizzata.

Crea un file ~/Rassegna/fonti.txt con i siti che segui (uno per riga). Poi ogni mattina puoi usare filesystem + firecrawl insieme:

Prompt 17

Non richiede configurazione aggiuntiva. Il piano gratuito ha rate limiting (poche centinaia di richieste al giorno), sufficiente per uso personale.

****Autenticazione****

Nessuna. context7 funziona senza account o API key per l'uso base.

****Esempio di utilizzo****

Documentazione su versioni specifiche:

Prompt 18

Confronto tra strumenti:

Prompt 19

Debug con documentazione aggiornata:

Prompt 20

I dati vengono salvati localmente in un file JSON nella cartella dei dati di sistema. Non vengono mai inviati a server esterni.

****Autenticazione****

Nessuna. I dati sono salvati localmente sul tuo computer.

****Esempio di utilizzo****

La prima sessione: costruisci il tuo profilo.

Prompt 21

Aggiornamento della memoria:

Prompt 22

[PRATICA] Installa memory e crea il tuo profilo di base: nome, professione, 3-4 clienti o progetti principali, le tue preferenze di comunicazione con Claude. Chiudi completamente Claude Code (non solo la finestra – termina l'applicazione), riapri in una nuova sessione, e chiedi "Cosa sai di me?". Verifica che ricordi le informazioni che hai inserito. Questo è il test fondamentale di persistenza – se funziona, la memoria sopravvive alla chiusura della sessione.

****memory nel tempo: come evolve il grafo****

La memoria non è statica – deve evolvere con te. Dopo il primo mese di utilizzo, la tua memoria dovrebbe contenere:

- Il tuo profilo di base (chi sei, cosa fai)
- I tuoi clienti o collaboratori principali con note aggiornate
- I tuoi progetti in corso con stato e scadenze
- Le tue preferenze e il tuo stile di lavoro con Claude

Dopo tre mesi, aggiungi:

- Pattern che hai osservato ("i meeting del lunedì mattina sono sempre urgenti")
- Risorse che usi spesso ("il mio fornitore preferito per X è Y")
- Informazioni su tools e processi consolidati

Una volta al mese è utile fare una pulizia: chiedi a Claude "Cosa c'è nella mia memoria che è probabilmente obsoleto?" Claude elencherà entità e osservazioni con date vecchie o potenzialmente superate, e potrai decidere cosa cancellare.

****[DIETRO] Knowledge graph vs. semplici note****

memory non salva testo grezzo ma un grafo strutturato. La differenza è importante: se salvi "Bruno lavora con Azienda Alpha" come nodo, memory sa che "Bruno" è una persona, "Azienda Alpha" è un'organizzazione, e tra loro esiste una relazione di tipo "cliente". Quando chiedi "chi sono i miei clienti?", memory restituisce tutte le entità collegate a te dalla relazione "cliente" – non cerca parole chiave in un testo.

Questo rende le query molto più precise: "quali sono i progetti aperti?" restituisce solo le entità di tipo progetto con stato "aperto", non tutte le volte che hai menzionato la parola "progetto" in qualsiasi contesto.

La struttura a grafo permette anche relazioni indirette: se sai che "Evento Gamma è cliente di Bruno" e "Bruno usa Lightroom per gli eventi", puoi inferire che "Evento Gamma è associato a workflow Lightroom" anche senza averlo scritto esplicitamente.

****Trappole****

- [!] La memoria è locale – non si sincronizza tra computer diversi. Se lavori su laptop e desktop, hai due knowledge graph separati. Puoi copiarli manualmente (sono file JSON) ma non c'è sincronizzazione automatica.
- [!] Informazioni obsolete rimangono in memoria finché non le cancelli esplicitamente. Se un cliente termina il contratto, devi dirlo esplicitamente a Claude perché aggiorni il grafo. Una pulizia periodica (ogni tre mesi) mantiene la memoria accurata.
- [Tip] Inizia semplice: aggiungi solo le informazioni che usi spesso. Un grafo sovraffollato diventa rumoroso. Aggiungi più dettagli man mano che capisci cosa ti è utile ricordare.

****Costruire il tuo knowledge graph: cosa ricordare****

Non tutto merita di essere nella memoria. Un knowledge graph efficace contiene informazioni che:

- Cambiano raramente (il tuo nome, il tuo settore, i tuoi strumenti principali)
- Ci vorresti tempo a rispiegare ogni volta (la struttura dei tuoi progetti, i tuoi processi standard)
- Servono spesso come contesto (chi sono i tuoi clienti, quali sono le tue tariffe)

Non mettere in memoria:

- Dettagli che cambiano spesso (stato di avanzamento di un task specifico)
- Informazioni che già hai in un documento organizzato
- Tutto – un grafo troppo denso diventa rumoroso

Un buon punto di partenza per chiunque:

Prompt 23

Sostituisci ``IL_TUO_GITHUB_PAT`` con il token che hai copiato.

****Autenticazione****

[VER 2026-05-03 – <<https://github.com/settings/tokens>>] Personal Access Token (PAT) generato dalla sezione Settings del tuo account GitHub. Usa i "Fine-grained tokens" – permettono di limitare l'accesso ai soli repository che ti servono e ai soli permessi necessari. Questo è più sicuro dei "Classic tokens" che danno accesso a tutto.

****Esempio di utilizzo****

Analisi e miglioramento di documentazione:

Prompt 24

Revisione di un file di configurazione:

Prompt 25

[PRATICA] Installa il MCP github, poi chiedi a Claude di mostrarti la lista dei tuoi repository con il nome e la data dell'ultimo commit. Verifica che veda i repository del tuo account (pubblici e privati con il PAT configurato correttamente).

****github come strumento di documentazione aziendale****

Molte piccole imprese e liberi professionisti che collaborano con team tecnici scoprono che GitHub è diventato il loro sistema di documentazione "de facto" – non per scelta, ma perché lo sviluppatore lo usa e lì finisce tutto. Con il MCP github puoi interagire con quella documentazione senza imparare l'interfaccia GitHub.

Prompt 26

Questo ti dà una panoramica della documentazione esistente senza doverti loggare su GitHub, navigare la struttura delle cartelle, aprire ogni file e leggerlo uno per uno.

****[DIETRO] HTTP transport vs. stdio per GitHub****

GitHub offre il suo MCP server come servizio HTTP remoto – non devi installare niente sul tuo computer oltre all'autenticazione. Quando Claude invoca un tool github, la richiesta viaggia da Claude Code al server MCP di GitHub (via HTTP), che esegue le operazioni sull'API GitHub e restituisce il risultato. Il token PAT nell'header è la tua firma digitale che dice a GitHub "questo è Bruno, ha i permessi X".

Il vantaggio del transport HTTP rispetto alla vecchia versione stdio è che il server è mantenuto da GitHub stesso, aggiornato quando l'API GitHub cambia, e non richiede Node.js locale o dipendenze npm.

****Trappole****

- [!] Non mettere il PAT direttamente in chiaro in un file di configurazione se quel file potrebbe finire in un repository pubblico. Usa variabile d'ambiente: prima esegui ``export GITHUB_PAT=il-tuo-token`` nel terminale, poi aggiungi il MCP con ``--header "Authorization: Bearer ${GITHUB_PAT}"``.
- [!] I PAT hanno una scadenza. Quando scadono, il MCP smette di funzionare senza preavviso (vedrai errori di autenticazione). Quando crei il token, imposta una scadenza di 6-12 mesi e annotala subito in calendario.
- [!] I "Fine-grained tokens" limitano l'accesso ai repository che selezioni. Se crei un nuovo repository dopo aver configurato il token, dovrai aggiornare le permission del token per includerlo.

****github per chi non scrive codice: documentazione e gestione issue****

Il MCP github è spesso percepito come uno strumento per sviluppatori. In realtà, se la tua organizzazione o il tuo team usa GitHub per qualsiasi scopo – versionare documenti, tracciare task, gestire progetti – github ti apre le stesse possibilità.

Esempio per chi gestisce un sito web con un team tecnico: il tuo sviluppatore usa GitHub per il codice del sito. Tu non sai leggere codice, ma hai bisogno di capire cosa è cambiato di recente e segnalare problemi.

Prompt 27

Autenticazione: nessuna.

****Quando usarlo****

Non tutti i problemi richiedono sequential-thinking. È particolarmente utile per:

- Decisioni con molte variabili e vincoli (scegliere tra alternative, pianificare)
- Analisi di situazioni complesse con molti fattori interconnessi
- Problemi dove il "perché" è importante quanto il "cosa" – vuoi capire il ragionamento

- Casi in cui hai dubbi che Claude possa semplificare troppo
- Prima di usare /plan per task di sviluppo complessi
- Quando vuoi condividere il ragionamento con altri – il formato strutturato è più comunicabile di una risposta fluida

Non serve per: domande dirette con risposta univoca, recupero di informazioni semplici, conversazioni veloci, compiti creativi dove il pensiero strutturato può limitare l'originalità.

****Esempio di utilizzo****

Decisione strategica:

Prompt 28

Passo 2 – Avvia Claude Code e autenticati:

Prompt 29

Vedrai i server gmail e google-calendar nell'elenco con stato "pending auth". Selezionali e segui il flusso OAuth nel browser: accedi al tuo account Google con le credenziali normali, concedi i permessi richiesti (leggi attentamente cosa stai autorizzando – puoi concedere solo lettura se preferisci iniziare in modo conservativo), e il browser ti reindirizzerà a Claude Code.

[VER 2026-05-03 – <<https://code.claude.com/docs/en/mcp>>] I token OAuth sono memorizzati in modo sicuro nel tuo sistema. Vengono rinnovati automaticamente senza che tu debba rifare il login. Se il browser non si apre automaticamente, copia l'URL fornito da Claude Code e aprilo manualmente.

****Autenticazione****

[VER 2026-05-03 – <<https://developers.google.com/workspace/guides/configure-mcp-servers>>] OAuth 2.0 con account Google. Scope richiesti: Gmail (leggi e scrivi email, gestisci etichette), Calendar (leggi e scrivi eventi, gestisci calendari). Puoi controllare e revocare l'accesso in qualsiasi momento da myaccount.google.com/security → "App con accesso al tuo account".

****Esempi di utilizzo****

Triage giornaliero delle email: