

# 15 lessen om beter te werken met Claude Code

Wat ik leerde uit mijn eigen werkbank,  
met prompts die je direct kunt gebruiken.

---

**15** lessen uit de praktijk

---

**12** prompts om te kopiëren

---

**1** sjabloon voor je CLAUDE.md

---

# Voor wie dit is

Voor ondernemers en teams die zelf met Claude Code werken of daar binnenkort mee beginnen. Ik bouw er elke week systemen mee voor servicebedrijven. Dit zijn de gewoontes die het verschil maken tussen een model dat soms iets goeds oplevert en een model waar je op kunt bouwen.

Elke les heb ik getest in mijn eigen werk en aangevuld met de prompts die ik zelf gebruik. Wat tussen haken staat, vul je zelf in.

---

|        |                                          |               |
|--------|------------------------------------------|---------------|
| DEEL 1 | <b>Geef een goede opdracht</b>           | les 1 tot 3   |
| DEEL 2 | <b>Controleer het werk</b>               | les 4 en 5    |
| DEEL 3 | <b>Houd je context schoon</b>            | les 6 tot 10  |
| DEEL 4 | <b>Werk sneller</b>                      | les 11 en 12  |
| DEEL 5 | <b>Maak van CLAUDE.md je handleiding</b> | les 13 tot 15 |

---

## LES 01

# Laat Claude je prompt schrijven

**Waarom.** Claude schrijft inmiddels betere prompts dan de meeste mensen. Maak de prompt daarom samen, met vragen over en weer. Dat heet metaprompting.

**Zo doe je het.** Zeg wat je wilt en voor wie, en laat Claude vragen stellen tot de opdracht scherp is. Dit loont bij grote taken die veel tokens kosten. Voor een snelle dagelijkse vraag is het niet nodig.

JIJ

Ik wil X voor Y. Help me een prompt te schrijven.

CLAUDE

Voor wie is het?

Wat is een goed resultaat?

**Daarna schrijft Claude een scherpe prompt.**

Heb je een voorbeeld?

**PROMPT**

Ik wil [resultaat] voor [doelgroep]. Help me hier een goede prompt voor te schrijven. Stel me eerst één voor één vragen over voor wie het is, wat een goed resultaat is en welke voorbeelden ik heb. Schrijf de prompt pas als je genoeg weet.

## LES 02

# Beschrijf wanneer het af is

**Waarom.** Met een stappenplan van zeven punten eindig je in een sessie van honderd berichten waarin je blijft bijsturen. Claude kun je inmiddels behandelen als een vakman die je inhuurt. Een aannemer vertel je wat er moet staan en wanneer het klaar is, en hoe hij metselt laat je aan hem.

**Zo doe je het.** Geef per taak het doel en een korte lijst van wat er klopt als het af is. Zet dezelfde werkwijze in je CLAUDE.md, dan doet Claude het uit zichzelf.

## Stap voor stap hoe

1. 
2. 
3. 
4. 
5. 
6. 

LAAT WEG

## Klaar als

- ☒ Offerte staat in het CRM
- ☒ Klant krijgt een bericht
- ☒ Alle tests slagen

GEEF DIT MEE

### PROMPT

Doel: [wat er moet komen]

Klaar als:

- [controleerbaar punt]
- [controleerbaar punt]

Grenzen: [wat niet mag]

Kies zelf de aanpak en meld het als een punt niet haalbaar is.

## LES 03

# Stel eerst een diagnose

**Waarom.** Geef je problemen één voor één door, dan lost Claude steeds een symptoom op en mist hij het verband ertussen.

**Zo doe je het.** Loop door wat je hebt gemaakt en zet alles op een rij wat niet werkt of anders moet. Ik doe dat zelf hardop. Ik klik door de software en praat tegelijk tegen Claude met de spraakfunctie, zodat hij in één keer een volledig beeld heeft van wat ik anders wil.

**HARDOP DOORLOPEN**

- Opslaan faalt bij lege velden
- Datum in verkeerd formaat
- Lijst laadt traag

**EERST DE OORZAAK**

- 1 Wat gaat er mis
- 2 Waarom gaat het mis
- 3 Hoe los je het op

**PROMPT**

Hieronder staat alles wat nu niet werkt. Los nog niets op. Zoek per punt eerst de oorzaak en geef me een overzicht van wat er misgaat, waarom en hoe je het zou oplossen. Wacht daarna op mijn akkoord.  
[jouw lijst of ingesproken notities]

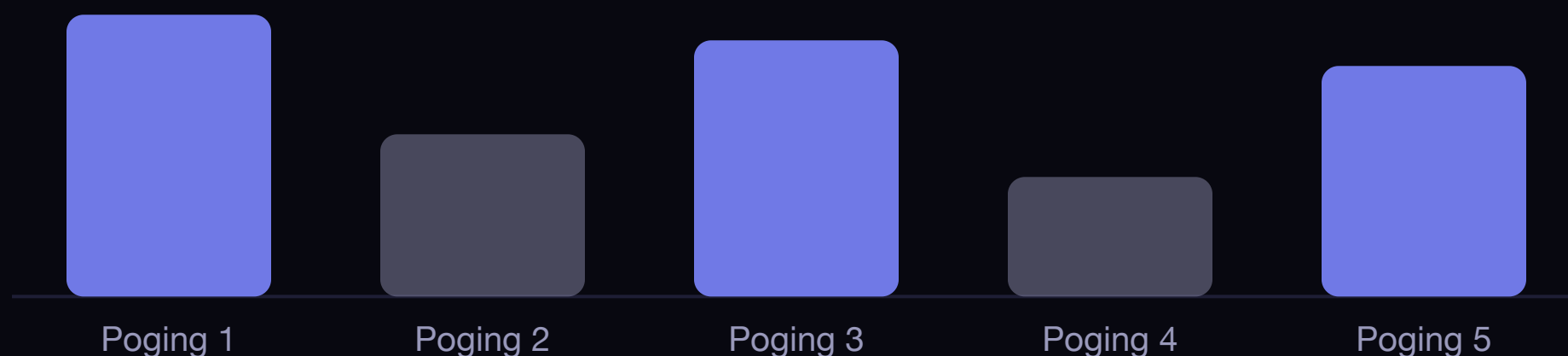
## LES 04

# Vertrouw de eerste uitkomst niet

**Waarom.** Een goede eerste uitkomst kan geluk zijn. Voer je dezelfde opdracht nog een keer uit, dan kan het resultaat flink anders uitvallen.

**Zo doe je het.** Draai belangrijke opdrachten 3 tot 10 keer en leg de uitkomsten naast elkaar. Dan zie je wat steeds goed gaat en wat toeval was. Dit heet evals, een woord dat je vaak tegenkomt als je met taalmodellen werkt.

DEZELFDE OPDRACHT, VIJF KEER UITGEVOERD

**PROMPT**

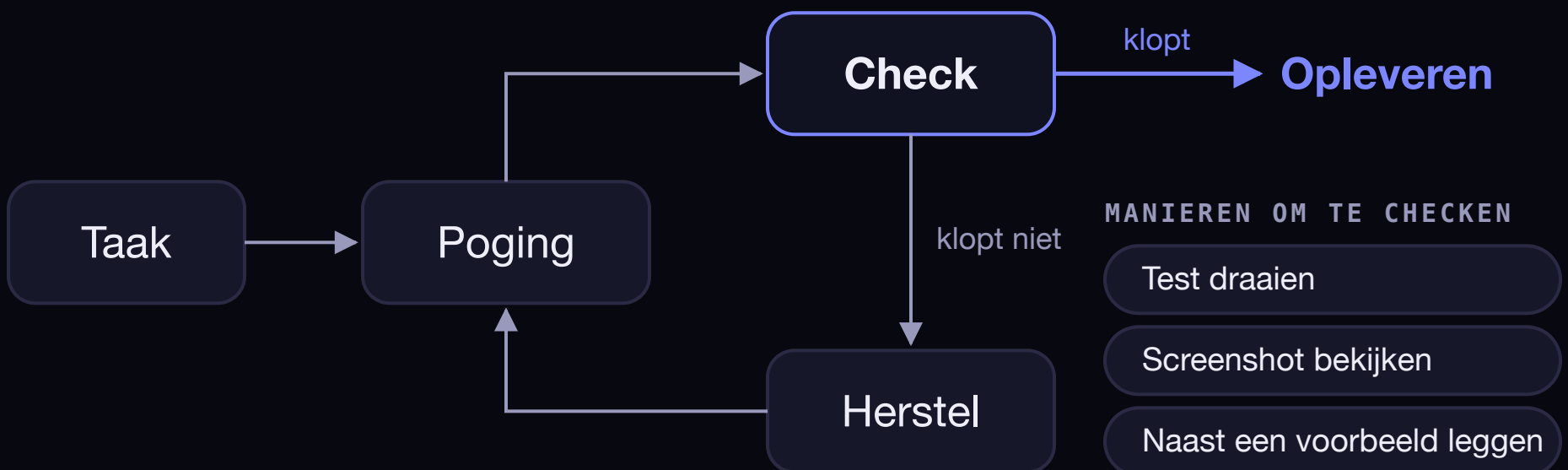
Voer deze opdracht vijf keer uit met dezelfde input, elke keer in een aparte agent. Zet de uitkomsten naast elkaar, geef aan waar ze verschillen en welke het best is, met de reden.

## LES 05

# Geef Claude een manier om zijn werk te checken

**Waarom.** Zonder controle bouw je iets en hoop je dat het werkt. Met een controle herhaalt Claude zijn werk tot het klopt.

**Zo doe je het.** Een taak gaat dan rond van poging naar controle en herstel, en gaat pas de deur uit als de controle slaagt. Is er nog geen manier om te checken, laat Claude er dan eerst een opzetten.

**PROMPT**

Bedenk voordat je begint hoe je controleert dat dit werkt. Is daar nog niets voor, bouw dan eerst een controle, zoals een test, een screenshot die je bekijkt of een vergelijking met `[voorbeeld]`. Draai de controle na elke poging en lever pas op als hij slaagt.

LES 06

# Kijk wat je context verbruikt

**Waarom.** Nog voor je iets typt, is een deel van je context al gevuld. Hoe voller je context, hoe duurder elke stap en hoe minder scherp Claude nadenkt.

**Zo doe je het.** Kijk aan het begin van een grote taak wat je context vult en zet uit wat je in dit project niet nodig hebt. Deze drie commando's helpen daarbij.



|                 |                                                                                          |
|-----------------|------------------------------------------------------------------------------------------|
| <b>/context</b> | laat in een raster zien wat je context vult, met tips om ruimte te maken                 |
| <b>/doctor</b>  | toont welke skills, koppelingen en plugins je niet gebruikt en wat ze aan context kosten |
| <b>/compact</b> | vat een lange sessie samen en maakt ruimte vrij                                          |

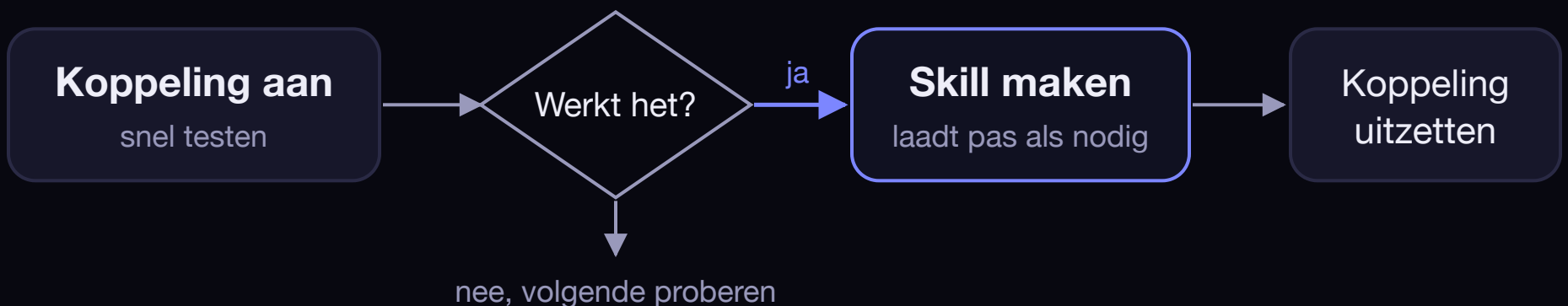
De verdeling in de schets is een voorbeeld.

## LES 07

# Test met een koppeling, schaal met een skill

**Waarom.** Een koppeling, ook wel MCP genoemd, zet je met één klik aan, dus daarmee zie je snel of iets werkt. Een skill maken kost meer tijd, maar een skill laadt pas volledig als hij nodig is en kost daardoor minder context.

**Zo doe je het.** Test een nieuwe toepassing eerst via een koppeling. Werkt het, maak er dan een skill van en zet de koppeling uit. De tijd die je in de skill steekt is nooit weggegooid, want je weet al dat het werkt.

**PROMPT**

Deze koppeling doet wat ik wil. Maak er een skill van die hetzelfde doet, met alleen de onderdelen die ik in dit project gebruik. Laat daarna zien hoe ik de koppeling uitzet.

LES 08

# Laat de code de bron zijn

**Waarom.** Een skill of notitie verandert niet mee als je code verandert. Na een paar versies beschrijft hij iets wat niet meer klopt, terwijl de code altijd laat zien hoe het nu werkt.

**Zo doe je het.** Zet in skills en in je CLAUDE.md alleen wat je niet uit de code kunt halen, zoals keuzes, afspraken en valkuilen. Voor de rest laat je Claude de code lezen. /doctor helpt daarbij, want hij haalt uit je CLAUDE.md wat Claude ook uit de code kan afleiden.

|       | VERSIE 1 | VERSIE 2  | VERSIE 3  |
|-------|----------|-----------|-----------|
| Code  | actueel  | actueel   | actueel   |
| Skill | actueel  | verouderd | verouderd |

PROMPT

Lees de code en vergelijk die met mijn CLAUDE.md en mijn skills. Geef aan welke stukken niet meer kloppen met hoe het nu werkt en wat Claude ook zelf uit de code kan halen. Stel voor wat eruit kan en verwijder nog niets.

## LES 09

# Stel tussendoor vragen met /btw

**Waarom.** Je ziet iets voorbijkomen wat je niet snapt, terwijl Claude midden in een taak zit. Een gewone vraag komt in je gesprek terecht en vult je context.

**Zo doe je het.** Typ /btw met je vraag erachter, ook terwijl Claude nog bezig is. De vraag onderbreekt de taak niet en het antwoord komt niet in je gesprek. Claude antwoordt alleen uit wat al in de sessie staat, dus vraag naar iets wat hij eerder heeft gelezen of besloten.



● Claude werkt aan de koppeling met de boekhouding

> /btw waarom koos je net voor een wachtrij?

Omdat de koppeling maximaal tien verzoeken per seconde aankan.

KOMT NIET IN JE GESPREK

## VOORBEELDEN

/btw hoe heette dat configuratiebestand ook alweer?

/btw waarom koos je net voor deze aanpak?

/btw welke bestanden heb je al aangepast?

## LES 10

# Begin fris met een handoff note

**Waarom.** Lange sessies worden minder scherp, net zoals jij na een lange werkdag.

**Zo doe je het.** Vraag aan het eind van een lange sessie om een handoff note en begin daarmee een nieuwe sessie. Claude weet dan meteen waar je gebleven was, met een lege context.

**Goed moment.** Als Claude dingen vergeet die je eerder zei, of als /context laat zien dat je context grotendeels vol zit.

## SESSIE 1



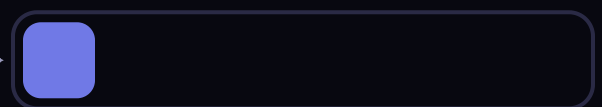
vol en minder scherp

**Handoff note**

afgerond  
nog te doen  
doel  
bestanden



## SESSIE 2



fris, en weet waar je was

**PROMPT**

Maak een handoff note die ik kan meenemen naar een nieuwe sessie. Zet erin wat we hebben afgerond, wat er nog moet gebeuren, wat het doel van de nieuwe sessie is en welke bestanden daarvoor belangrijk zijn.

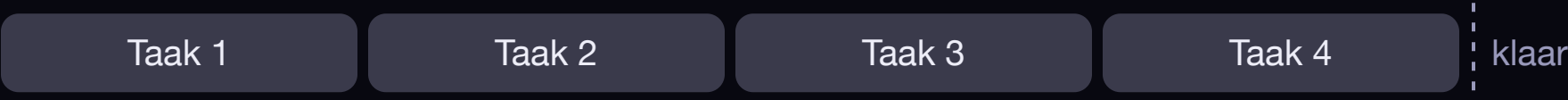
LES 11

# Laat taken tegelijk lopen

**Waarom.** Een taak geven, wachten, de volgende geven en weer wachten is waar de meeste tijd heen gaat.

**Zo doe je het.** Geef in één bericht vier tot zes losse taken mee en laat Claude ze verdelen over agents die tegelijk werken. Claude doet dat steeds vaker uit zichzelf, en je kunt er ook om vragen. Houd elke taak afgebakend, zodat de agents elkaar niet in de weg zitten.

NA ELKAAR



TEGELIJK



PROMPT

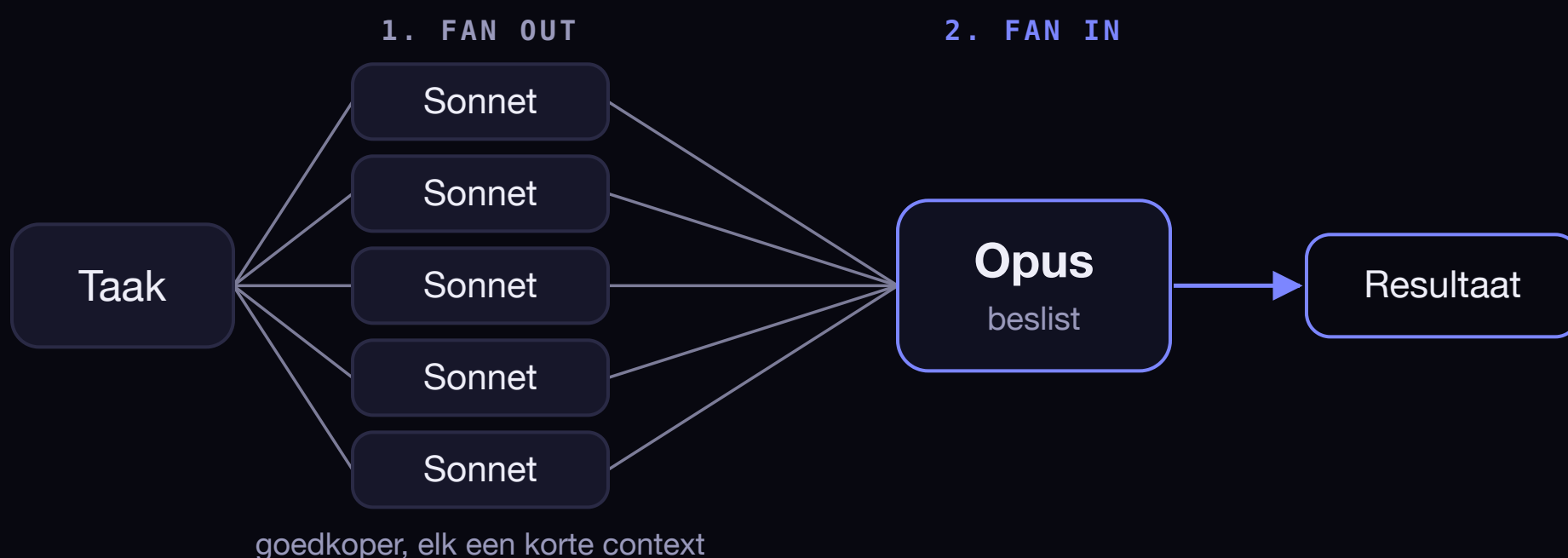
Ik heb [aantal] losse taken. Voer ze tegelijk uit met aparte agents en voeg de resultaten daarna samen.  
Taak 1: [wat], klaar als [punt]  
Taak 2: [wat], klaar als [punt]  
Taak 3: [wat], klaar als [punt]  
Laat elke agent alleen de bestanden openen die hij voor zijn taak nodig heeft.

## LES 12

# Goedkoop verkennen, duur beslissen

**Waarom.** Informatie verzamelen kan prima met een goedkoper model. De beslissing en het echte werk leg je bij het sterkste model.

**Zo doe je het.** Vraag om een fan out en fan in aanpak. Een paar agents op een goedkoper model zoals Sonnet zoeken elk één onderdeel uit en houden hun context kort en scherp. Daarna neemt een sterker model zoals Opus met hun samenvattingen de beslissing. In mijn ervaring is dat sneller, goedkoper en beter.

**PROMPT**

Werk in twee stappen. Laat eerst een paar agents op Sonnet de informatie voor deze taak verzamelen, elk over één onderdeel, en laat ze een korte samenvatting teruggeven. Neem daarna met Opus op basis van die samenvattingen de beslissing en voer de taak uit.

## LES 13

# Stel je CLAUDE.md in

**Wat het is.** CLAUDE.md is een bestand in je project dat Claude aan het begin van elke sessie leest. Zie het als het inwerkdocument voor een nieuwe collega.

**Zo doe je het.** Typ `/init` in je project, dan schrijft Claude zelf het eerste deel met wat waar staat. Vul de rest aan met je voorkeuren, wat Claude zonder vragen mag doen en de lessen die je onderweg leert. Op de volgende pagina staat een sjabloon.

**Houd hem kort.** Alles wat erin staat, kost in elke sessie context.



# Sjabloon voor je CLAUDE.md

Kopieer dit naar CLAUDE.md in de hoofdmap van je project en vul in wat tussen haken staat.

## # CLAUDE.md

### ## Samenvatting

[Laat /init dit deel schrijven, met wat dit project is en wat waar staat.]

### ## Mijn voorkeuren

- Schrijf in het Nederlands.
- Geef altijd volledige bestandspaden.
- [taal of tool die je wilt, bijvoorbeeld TypeScript]

### ## Wat je mag doen

- Bestanden in dit project aanpassen en tests draaien zonder te vragen.
- Vraag eerst voordat je iets verwijdt, installeert of naar buiten stuurt.

### ## Zo werken we

- Ik geef per taak een doel en een lijst Klaar als. Kies zelf de aanpak.
- Bedenk voordat je begint hoe je controleert dat het werkt, en lever pas op als die controle slaagt.
- Verdeel losse taken over agents die tegelijk werken.

### ## Lessen

- [Zet hier een regel als iets twee keer misgaat.]
- Vraag jezelf na elke grote taak hoe het sneller had gekund met minder tokens, en zet het antwoord hier als één regel.

LES 14

# Ruim oude regels op en laat hem leren

**Waarom.** Veel regels met NOOIT en ALTIJD zijn littekens van oudere modellen. Het huidige model heeft ze vaak niet meer nodig, en ze kunnen hem zelfs in de weg zitten.

**Zo doe je het.** Haal ze weg en zet alleen terug wat echt opnieuw misgaat. /doctor prompt-audit zoekt verouderde en tegenstrijdige instructies voor je op. Maakt Claude steeds dezelfde fout, stel hem dan een vraag waar hij van leert, zoals hoe het sneller had gekund met minder tokens, en laat hem het antwoord als les in je CLAUDE.md zetten.

~~NOOIT X gebruiken~~

~~NOOIT Y aanraken~~

ALTIJD eerst vragen

~~NOOIT Z draaien~~

~~ALTIJD uitleggen~~

N00IT tests overslaan

WEG

ERMEE

HOUDEN

PROMPT

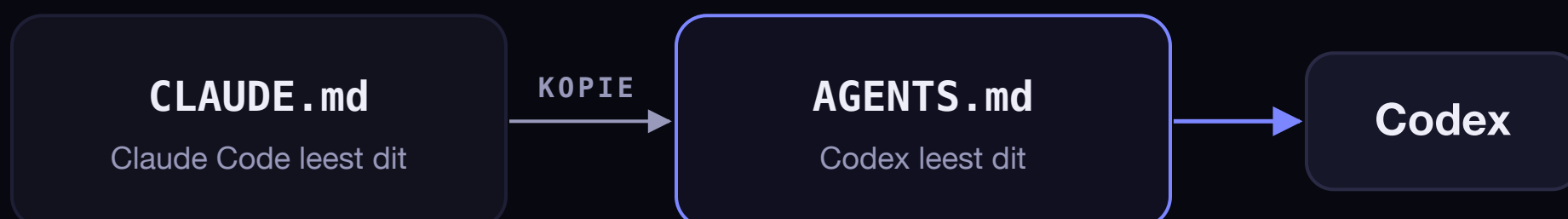
Lees mijn CLAUDE.md en mijn skills. Zet elke regel met NOOIT of ALTIJD in een lijst en geef per regel aan of het huidige model hem nog nodig heeft. Stel voor wat weg kan en verwijder nog niets.

## LES 15

# Houd een reservetool achter de hand

**Waarom.** Soms ligt een dienst eruit of valt de kwaliteit tijdelijk tegen. Dan wil je niet stilstaan.

**Zo doe je het.** Leer de basis van een tweede tool, bijvoorbeeld Codex van OpenAI. Je hoeft hem niet tot in detail te kennen, als je er maar mee verder kunt. Kopieer je CLAUDE.md naar een bestand AGENTS.md in dezelfde map. Codex leest dat bestand, dus je werkt daar meteen met dezelfde afspraken.

**PROMPT**

Maak van mijn CLAUDE.md een AGENTS.md voor Codex. Neem alle afspraken over en laat weg wat alleen voor Claude Code geldt, zoals slash commando's.

# De 15 lessen op een rij

- 01 Laat Claude je prompt schrijven
- 02 Beschrijf wanneer het af is
- 03 Stel eerst een diagnose
- 04 Vertrouw de eerste uitkomst niet
- 05 Geef Claude een manier om zijn werk te checken
- 06 Kijk wat je context verbruikt
- 07 Test met een koppeling, schaal met een skill
- 08 Laat de code de bron zijn
- 09 Stel tussendoor vragen met /btw
- 10 Begin fris met een handoff note
- 11 Laat taken tegelijk lopen
- 12 Goedkoop verkennen, duur beslissen
- 13 Stel je CLAUDE.md in
- 14 Ruim oude regels op en laat hem leren
- 15 Houd een reservetool achter de hand

# Liever dat het gewoon voor je draait?

Alles in dit document kun je zelf toepassen. Er zit geen geheim onder dat ik heb weggelaten.

Wat je hier niet ziet, is het werk erna. Een systeem dat elke dag draait voor je bedrijf, ook als jij er niet naar kijkt. Dat bouw ik met ScaleOS voor servicebedrijven.

Wil je zien hoe dat er voor jouw bedrijf uitziet, dan bouw ik gratis een proefmodule voor het proces dat jou nu de meeste tijd kost.

**Stuur me een bericht op LinkedIn of kijk op [thedalindustry.nl](https://thedalindustry.nl)**

Roïn Dalstra · ScaleOS