

Hoe een senior developer jouw setup zou bouwen

Jamal, 14-8-2026: "Stap in de rol van een senior developer met extreem veel ervaring op het gebied van AI, code en automatisering, en kom terug met een plan zoals een developer head het zou aanpakken."

Dit is dat plan. Gebaseerd op wat er vandaag echt is gemeten, niet op algemene adviezen.

Het eerste wat zo iemand zou vragen

Niet "hoe is het gebouwd". Maar: **wat levert het op?**

Ik heb alle 69 geplande taken op de MacBook ingedeeld:

SOORT WERK	AANTAL	AANDEEL
Dient de salon direct (blogs, mail, klanten, social, SEO)	8	11%
Houdt het systeem zelf draaiend (wachters, backups, tokens, geheugen)	33	47%
Overig of onduidelijk	28	42%

Bijna de helft van al het automatische werk bestaat om het systeem zelf overeind te houden. Een negende deel doet iets voor de zaak.

Dat is de kern. Bij een gezond systeem is die verhouding omgekeerd. Dit is geen kritiek op wat er gebouwd is, het is de normale uitkomst van twee jaar organisch groeien zonder één keer terugkijken.

Een developer head zou hier stoppen met alle technische vragen en eerst dit op tafel leggen.

De diagnose in één zin

Je hebt geen slecht systeem. Je hebt geen architectuur.

Dat is iets anders. Elk onderdeel is op zichzelf verstandig gebouwd, met goede redenen, vaak na

een echte storing. Maar niemand heeft ooit de laag eronder gelegd: afspraken over hoe dingen

draaien, meten, melden en configureren.

Zonder die laag krijgt elk nieuw probleem zijn eigen oplossing. Dat is precies wat er gebeurd is:

- **5 verschillende manieren om iets te laten draaien:** LaunchAgents (Mac), cron (VPS), Scheduled Tasks (Windows), pm2 (VPS), en hooks (Claude zelf).
- **44 verschillende manieren om te meten of iets nog werkt.**
- **Minstens 4 plekken waar instellingen en sleutels staan:** `.claude.json`, de Keychain, losse `.env`-bestanden en de plists zelf. Dat leverde gisteren nog een gBrain-token op die op de ene plek werkte en op de andere dood was.
- **Geen enkele plek waar staat wat er allemaal draait.** Dat moest ik vandaag met de hand bij elkaar zoeken over vijf machines.

Wat een ervaren developer NIET zou doen

Alles opnieuw bouwen.

Dat is de klassieke fout, en de duurste. Bij een herbouw gooi je ook alles weg wat wél werkt, inclusief de tientallen kleine correcties die er na echte fouten in zijn geslopen. Die zitten niet in de documentatie. Die zitten in de code.

Concreet voorbeeld van vandaag: ik dacht dat 143 van je 309 scripts dood waren. Na beter meten

bleken er **zes** echt dood. De rest is gereedschap dat gebruikt wordt. Een herbouw had 137 werkende dingen gesloopt.

Wat zo iemand wel doet heet **de wurgvijg**: je zet een nieuwe laag naast de oude, laat werk één voor één overlopen, en pas als alles over is haal je het oude weg. Nooit een grote knal.

Het plan in vijf fasen

Fase 0 — Zichtbaar maken (grotendeels af)

Je kunt niet verbeteren wat je niet kunt zien. Dit was de blinde vlek: er was geen enkele manier

om te zien of een van de 44 wachters was weggefallen.

Vandaag gedaan: de vloot-wachter draait, met de controles in een lijst in plaats van in code.

Nog te doen: Coco, Julio en Lenovo meenemen. Zolang die buiten beeld blijven, is elk cijfer een schatting.

Klaar als: je op één plek kunt zien wat er op alle zeven machines draait en of het nog leeft.

Fase 1 — Eén fundament bouwen (2 tot 3 weken)

Niet nieuwe functies. Alleen afspraken, met code die ze afdwingt. Vier stukken:

1. Eén manier om een taak te draaien.

Nu vijf mechanismen. Er blijft er één per machinesoort, met dezelfde vorm: elke taak schrijft bij elke run een stempel weg, logt op dezelfde plek, en heeft een eigenaar en een vervaldatum.

Dat stempelen lost meteen de fout op die ik vandaag zelf maakte: ik las een oud logbestand als

stilstand, terwijl die scripts alleen loggen als er iets mis is. Met een stempel bij elke run is ouderdom wél een geldige maat.

2. Eén manier om te melden.

Nu roept elk script zelf Jamal aan. Straks meldt alles aan één plek, en die bundelt. Eén

bericht

per dag in plaats van twintig losse.

3. Eén bron voor instellingen en sleutels.

De Keychain wordt de enige plek voor geheimen. Alles wat ze nu ergens anders heeft staan, gaat

daarheen. Dubbele waarden zijn dan onmogelijk in plaats van moeilijk te vinden.

4. Eén register van wat er draait.

Een lijst die zichzelf bijhoudt: elke taak, op welke machine, wat hij doet, wie hem gemaakt heeft, wanneer hij mag vervallen. Staat iets niet in het register, dan hoort het er niet.

Klaar als: een nieuwe taak toevoegen betekent één regel in het register plus het echte werk. Geen eigen meldcode, geen eigen instellingen, geen eigen wachtter.

Fase 2 — Overzetten (4 tot 6 weken, in de achtergrond)

Per taak, in kleine stappen, met dubbeldraaien. Volgorde op risico:

1. De negen meetwachters (loopt al)
2. De vijf rapporteurs die eigenlijk geen wachters zijn: naar de ochtendbriefing
3. De vijf herstarters (deze grijpen in, dus als laatste en met de meeste zorg)
4. De 57 hooks. Dit is het grootste onzichtbare risico: ze draaien bij **elke** actie, en er staan er vier tussen die stil kunnen falen zonder ooit een fout te geven.

Klaar als: de oude versie een week uit heeft gestaan zonder dat iemand iets miste.

Fase 3 — Opruimen (1 week)

Pas nu, en niet eerder. Opruimen vóór je zicht hebt is gokken.

- 6 echt dode scripts weg
- 68 backupbestanden in de scripts-map archiveren
- De 1,1 GB `gstack`-skill: dat is 99% van je hele skills-map, met 704 MB aan bibliotheken. Gebruik je hem niet, dan is dat in één klap weg.
- Dubbelingen samenvoegen: noodstop-wachtter staat drie keer, Paco heeft twee Suno-wachters.

Fase 4 — Zorgen dat het niet terugkomt (doorlopend)

Dit is het deel dat een developer head het belangrijkste vindt, en dat meestal wordt overgeslagen.

Elk nieuw ding moet langs vier vragen:

1. Wat gaat er kapot als dit er niet is?
2. Bestaat er al iets dat dit doet?
3. Hoe merk ik dat het stilvalt?
4. Wanneer mag het weer weg?

Geen antwoord op één van de vier betekent: niet bouwen.

Was die toets er geweest, dan waren er nooit 44 wachters gekomen, en ook geen twaalf marketing-tools. Ik heb hem gisteren en vandaag zelf twee keer overgeslagen.

De vier regels die een senior dev zou opleggen

1. Eén ding doet één ding, en er is er maar één van.

Twee Suno-wachters op één machine is geen extra zekerheid, het is twee keer half onderhouden.

2. Elk automatisch ding meldt dat het geleefd heeft.

Niet alleen dat er iets mis is. Ook dat het gedraaid heeft. Stilte moet verdacht zijn, niet geruststellend.

3. Instellingen staan op één plek, code op een andere.

Wat er gecontroleerd wordt hoort in een lijst. Hoe er gecontroleerd wordt hoort in code. Dat scheelt bij elke wijziging een programmeur.

4. Alles heeft een vervaldatum.

Niet om het weg te gooien, maar om het één keer per jaar tegen het licht te houden.

De onaangename vraag

Een developer head met dertig jaar ervaring zou dit ook zeggen, en het is de belangrijkste zin

van dit verslag:

Je bent kapper, geen softwarebedrijf.

Elke regel code die je erbij krijgt, is een verplichting die er nooit meer afgaat. 627 scripts zijn 627 dingen die kapot kunnen. Dat is te veel voor twee salons.

De vraag bij elk volgend voorstel, ook van mij, is niet "kan dit". Het antwoord daarop is bijna altijd ja. De vraag is: **hoeveel knippen levert dit op, en wat kost het aan onderhoud?**

Bij 47% van je automatische werk is het antwoord op de eerste helft: niets. Dat werk bestaat alleen omdat er ander werk is.

Dat is niet erg zolang je het weet. Het wordt erg zodra het onzichtbaar is.

Wat ik voorstel

1. **Fase 0 afmaken:** Coco, Julio en Lenovo in beeld. Daar heb ik toegang voor nodig.
2. **Fase 1 bouwen:** het fundament. Twee tot drie weken, en het levert op zichzelf niets zichtbaars op. Dat is normaal en het is de investering waar alles daarna op rust.
3. **Beslis over gstack:** 1,1 GB, gebruik je hem?
4. **Neem de vier vragen aan als vaste toets.** Ook voor mij, vooral voor mij.
5. **En kies één ding dat de salon dient** om parallel af te maken, zodat deze hele operatie niet alleen over zichzelf gaat. Mijn voorstel: de blogketen, want daar staat alles al klaar en die ligt sinds 13 juni stil.

Dat laatste punt is precies het punt van dit hele verslag. Een systeem dat alleen zichzelf onderhoudt, is geen systeem meer maar een hobby.