Kimi K3 in de vloot

Dit is het plan om Kimi K3 gericht in te zetten binnen je botvloot. Niet overal, maar precies waar zijn twee sterke punten tellen: een enorm geheugen (1 miljoen tokens) en een lagere prijs dan Opus.

De rode draad: Kimi wordt je tweede brein voor controle en grote-tekst-klussen. Conductor levert, Kimi controleert onafhankelijk, Conductor zet het live.

Waarom Kimi, en waarom niet overal

Sterk:

- 1 miljoen tokens context. Hele boeken, alle verslagen of een complete codebase passen in één keer in zijn hoofd.
- Goedkoper dan Opus voor zwaar werk.
- Onafhankelijk brein. Een tweede paar ogen dat niet dezelfde denkfout maakt als Conductor.

Niet inzetten voor:

- Klein en simpel werk. Daar blijft Sonnet of een lokaal model goedkoper.
- Haarvisie klant-teksten en merk. Daar blijft Claude de baas, want jouw regels en toon zitten in Claude.
- Standaard "voor de zekerheid". Kimi kost nu geld (betaald plan). Alleen inzetten waar het meetbaar verschil maakt.

Dit sluit aan op je vaste regel: het goedkoopste model dat goed genoeg is, en het dure model alleen waar het telt.

Wat er al staat

Goed nieuws: de basis is er al.

- Kimi draait als bot op de mini-pc in Spanje (Kimi K3, via de Moonshot-motor).
- Hij leest de bot-wachtrij (queue-poller) en hangt aan het gedeelde geheugen (gBrain).
- Betaald plan is actief, dus geen credit-stops meer.
- Bewezen in de praktijk: hij checkte 850 Appel-vragen na en vond echte fouten.

Er hoeft dus weinig gebouwd te worden. Het gaat vooral om Kimi slim en vast inzetten.

De drie toepassingen

1. Kimi als feit-checker (hoogste waarde)

Een vaste controle-stap vóór iets live gaat.

- **Blogs (haartips.nl):** Kimi checkt feiten en vaktermen vóór publicatie, bovenop de bestaande controle.
- **Appel/Dani:** Kimi controleert toetsen, nulmeting en flashcards (al bewezen).
- **Verslagen:** Kimi leest een verslag na op fouten voordat het naar jou gaat.

Patroon: Conductor schrijft → Kimi controleert → Conductor verbetert en zet live.

2. Grote-tekst-klussen (1 miljoen tokens)

Alles wat te groot is voor een gewoon model.

- Dani's 93 boeken in één keer verwerken voor Appel.
- Alle verslagen samen doorzoeken of samenvatten.
- Een complete bot-codebase in één keer nakijken op bugs.
- Lange Plaud-opnames in één keer verwerken.

3. Goedkoop bulk-werk

Grote batches nakijken of samenvatten waar Opus te duur is maar Sonnet net niet genoeg.

Hoe we het implementeren (stappen)

Stap 1 — Basis stevig maken (klein)
De queue-poller draait als vaste taak, met 60 minuten tijd per klus. Controleren dat hij altijd meedraait.

Stap 2 — Een vaste "Kimi-check"-vorm
Een duidelijk formaat waarin Conductor een controle-opdracht aan Kimi geeft: wat controleren, en in welke vorm het antwoord terugkomt (bijvoorbeeld: alleen de zekere fouten, met de juiste versie erbij). Zo blijft het antwoord bruikbaar.

Stap 3 — Inbouwen in je bestaande werk
Een optionele "laat Kimi checken"-stap toevoegen aan:
- de blog-workflow
- de Appel-deploy
- de verslag-pijplijn

Stap 4 — Kosten in beeld
Kimi's verbruik loggen, zodat je altijd ziet wat het kost. Zo blijft het bewust.

Waar beginnen

Mijn voorstel voor de eerste twee, want die geven direct waarde:

1. **Blog-feitcheck.** Hoogste frequentie, duidelijke winst: geen feitfouten meer live.
2. **Appel-verificatie.** Al bewezen deze week; Dani's toetsen blijven kloppen.

De rest (grote-tekst-klussen, verslag-review) bouwen we daarna stap voor stap uit.

Samengevat

- Kimi = tweede brein voor controle en grote teksten.
- De koppeling staat al; het gaat om slim en vast inzetten.

- Claude blijft de baas voor merk en klant; Kimi is de controleur en de zwaargewicht-lezer.
- Begin met blog-feitcheck en Appel-verificatie.

Zeg maar of je akkoord bent met deze richting, dan begin ik met stap 1 en 2.