AIFAIS

MYELIN

Myelin

Een Zelflerend AI Agent Systeem met Bewijsgebaseerd Kennisbeheer, Deterministische Evaluatie en Autonome Taakuitvoering

Architectuurdocument 2027 — Bijgewerkt april 2026 met productiedata van 250+ geautomatiseerde commits, 14 pre-commit tests en een signaaldichtheid-IQ van 155.

155
BREIN IQ

14
TESTS

250+
COMMITS

Component	Implementatie
Runtime	Headless Claude Code op Linux VPS
Orkestratie	Cron + bash taakrunner met lockfiles + flock
State	Git repository (append-only kennisbank)
Evaluatie	14 pre-commit tests + 2-laags semantische verificatie
Communicatie	Discord bot (tweerichtings) + webhooks
Leren	Observeren → reflecteren → zelfverbeteren → evo-gate
Dashboard	Autogegenereerd brain.html (graaf + missiecontrole)

De 10 Stadia van AI-Evolutie

Van praten met AI tot volledig autonome systemen. Elk stadium geeft de AI meer context en meer autonomie.

#	Stadium	Status	Beschrijving
01	Conversationele AI	VOLTOOID	Mens typt, AI antwoordt. Geen geheugen, geen tools, geen autonomie.
02	AI-Ondersteund Coderen	VOLTOOID	AI vult code aan in je editor. Mens stuurt, AI suggereert.
03	CLI Agent	VOLTOOID	AI voert meerstaps-taken uit vanuit terminal. Bestanden, code, commits.
04	Persistente Agent	VOLTOOID	AI draait continu met geheugen en communicatiekanalen (Discord, e-mail).
05	Gestructureerd Kennisbrein	VOLTOOID	Gecureerde kennisbank met betrouwbaarheidsscores en bewijstracking.
06	Zelfverbeterende Agent	HUIDIG ←	AI draait 24/7, observeert, leert, verbetert eigen prompts. Evo-gate voorkomt drift.
07	Autonome Laagrisico-Acties	VOLGENDE	AI maakt PRs voor laagrisico-fixes. Tests valideren, mens merget.
08	Cross-Project Intelligentie	GEPLAND	Patroon in project A detecteren en toepassen op project B.
09	Agent-Gestuurde Strategie	GEPLAND	AI analyseert data en stelt strategische richting voor.
10	Volledig Autonome Operatie	VISIE	Mens als eigenaar, niet operator. Het Rick Rubin-model: smaak boven uitvoering.

Kernarchitectuur

Ontwerpprincipes

- **Prompt = contract, geen suggestie** — Alles wat niet expliciet staat, wordt creatief geïnterpreteerd.
- **Bewijs boven vertrouwen** — Elke claim moet bewijs citeren: HTTP-status, bestandspad, git hash.
- **Faal luid, niet fout** — Pipelines stoppen bij fouten. Alerts gaan af. Geen stille slechte data.
- **Deterministisch waar mogelijk** — Bash-checks vóór semantische checks. Vertrouw AI nooit om zichzelf te evalueren.
- **Signaaldichtheid boven volume** — Slim = nuttige kennis toegepast, niet bestanden opgestapeld.

Repositorystructuur

```
brain/
    patterns/ # Herbruikbare technische patronen (SEO, frameworks)
    playbook/ # Causale ketens: als X → dan Y → dan Z
    mental-models/ # Beslisframeworks en denkpatronen
    clients/ # Bedrijfscontext per klant/project
    decisions/ # Architectuurbeslissingen met rationale
    reference/ # Verwijzingen naar externe systemen
    data/ # Taakoutput: JSON, eval logs, baselines
    claims/ # Ongevalideerde hypotheses (wachten op data)
    inbox/ # Onverwerkte items (auto-opgeschoond)
    server/
        prompts/ # Eén .md per taak (≤50 regels)
        tests/ # 14 pre-commit tests
        config.sh # Single source of truth
        run-task.sh # pull → claude → eval → commit → push
```

Hoe het systeem leert

Het systeem vormt een gesloten feedbacklus. Agent-output wordt geëvalueerd, evaluatieresultaten voeden terug in het geheugen, en geheugen vormt toekomstig gedrag. Onafhankelijk gevalideerd door Anthropic's nog niet uitgebrachte "Kairos"-feature (gelekt april 2026).

Fase	Wat Er Gebeurt	Output
1. Uitvoeren	Agent draait taak vanuit promptbestand	data/*.json
2. Evalueren L1	Deterministische bash-checks (geen AI)	eval-log.jsonl
3. Evalueren L2	Steekproef van 3 claims, verificatie met bron	reflection-*.json
4. Categoriseren	Faaltaxonomie (6 typen)	nikola-memory.md
5. Onthouden	Blinde vlekken + terugkerende fouten bijwerken	nikola-memory.md
6. Aanpassen	Morgen-acties schrijven voor volgende cyclus	reflection-*.json
7. Zelfverbeteren	Eigen prompts muteren (evo-gate gevalideerd)	evo-log.jsonl
8. Bewaken	Pre-flight blokkeert als slagingspercentage < 60%	preflight lock

IQ-Score: Signaaldichtheid boven Volume

Het IQ van het brein meet bruikbaarheid, niet omvang. Ruis verwijderen verhoogt het IQ. Ongevalideerde notities toevoegen verandert het nauwelijks.

As	Gewicht	Wat Het Meet
Signaaldichtheid	50/140	Bewijs per bestand, betrouwbaarheid van gevalideerde bestanden, validatieratio
Toepassing	50/140	Recentheid (laatste 30 dagen), diep bewijs (2+ keer bevestigd)
Structuur	40/140	Categoriedekking, onderwerpdichtheid, minimale levensvatbare omvang
Nauwkeurigheid	Straf	-3 per tegengesproken bestand (lage betrouwbaarheid + bewijs)

Evaluatie & Testen

14 Pre-Commit Tests

Test	Wat Het Vangt
prompt-lint	Prompts die 50 regels overschrijden
prompt-structure	Hardgecodeerde secrets (API-sleutels, tokens)
config-drift	Prompts niet in sync met canonieke config
cron-integrity	Ontbrekende promptbestanden, dubbele schema's
data-deps	Config ↔ generator ↔ crontab inconsistenties
contracts	Ontbrekende JSON-outputvelden, kapotte pipeline-afhankelijkheden
evo-gate	Pad-regex regressie, beschermd-bestandsschendingen
learn-signals	Find-syntaxbugs, intervalmismatches
brain-html	Generator crash, ontbrekende nodes, ongeldige graaf-JSON
iq-health	IQ-regressie > 5 punten, massale bewijsafname
regressions	6 specifieke eerdere bugs die nooit mogen terugkeren
learning-loop	Geheugensecties ontbreken, reflectie kapot
output-quality	Taakoutputs van gisteren ontbreken of ongeldig
eval-trend	7-daags slagingspercentage onder 60%-drempel

Evolutieveiligheid: evo-gate

De agent kan zijn eigen prompts muteren om te verbeteren. Maar zelfmodificatie zonder vangrails leidt tot catastrofale drift. De evo-gate is een deterministisch bash-script (geen AI) dat elke mutatie valideert vóór commit:

Check	Wat Het Valideert
Beschermde bestanden	reflection.md, evo-gate.sh, config.sh kunnen niet zelf-bewerkt worden
Regelaantal	Gemuteerde prompt moet ≤50 regels blijven
Outputformaat	JSON-templateblok moet behouden blijven
Structuur	Sectiekoppen (##) mogen niet vernietigd worden
Datumvariabelen	JJJJ-MM-DD patronen moeten behouden blijven
Padvalidatie	Alle gerefereerde mappen moeten bestaan

De Stack

Bewust eenvoudig. Geen frameworks, geen orkestratielagen, geen databases. Elk component is vervangbaar en debugbaar met standaard Unix-tools.

Laag	Tool	Waarom
AI Runtime	Claude Code (headless)	Tool use, bestandstoegang, git, web
Orkestratie	cron + bash	Beproefd, nul afhankelijkheden
State	Git (lokaal + GitHub)	Geversioneerd, diffbaar, gedistribueerd
Evaluatie	bash + python	Deterministisch, geen AI in eval-lus
Alerting	Discord (tweerichtingsbot)	Gratis, instant, bidirectioneel
Dashboard	brain.html (gegenereerd)	Graaf + missiecontrole, geen server nodig
Zoeken	QMD (semantisch)	Vectorzoekopdrachten over kennisbank
Locking	flock + lockfiles	Voorkomt gelijktijdige git pushes
Config	Enkel bash-bestand	Eén bron van waarheid, feature flags
Testen	14 bash-testscripts	Pre-commit + dagelijkse pre-flight

Kerncijfers (april 2026)

Metriek	Waarde
Dagelijkse geautomatiseerde taken	10-16 (cron-ingepland)
Commits (eerste 4 weken)	250+
Pre-commit tests	14 (was 8 bij lancering)
Kennisbestanden	56 (51 gevalideerd, IQ 155)
Leerlus-interval	Elke 2 uur (7 runs/dag)
Reflectienauwkeurigheid	9/9 L1, 3/3 L2 (laatste)
Dagen zonder Mark-correctie	6+ (lopend)
Feature flags	3 (zelfverbetering, hypothesetesten, impacttracking)

Veelgestelde Vragen

Hoe weet de AI wat goed en fout is?

De AI weet dat niet zelf. Daarom werkt het systeem in stappen die elkaar controleren:

1. AI genereert een bevinding
2. Een aparte AI-stap opent het daadwerkelijke bronbestand en controleert: klopt dit?
3. Resultaat wordt beoordeeld als VALID, PARTIALLY VALID of INVALID — met bewijs

Geen claim zonder bewijs. Niet "ik denk dat dit ontbreekt" maar "ik heb bestand X gelezen op regel Y en daar staat Z." Een hallucinatie valt op omdat het bewijs ontbreekt of niet klopt.

Hoe weten we dat IQ 155 echt 155 is?

Het IQ is geen absoluut getal — het is een nauwkeurigheidspercentage gebaseerd op signaaldichtheid.

Het meet: van alle kennis in het systeem, hoeveel is gevalideerd, recent toegepast en bevestigd door meerdere bronnen? Elke week worden bevindingen onafhankelijk gecontroleerd. Stijgt de nauwkeurigheid → het systeem leert. Daalt het → er is een probleem. Springt het ineens 30 punten → alert. Dat is de sanity check.

Wat als een mens foute input geeft?

Menselijke input is niet automatisch verifieerbaar. Daarom behandelt het systeem het als hypothese, niet als waarheid.

Nieuwe input begint met een lage betrouwbaarheidsscore. Die stijgt alleen als de data het bevestigt — meer clicks, betere posities, bevestiging bij andere klanten. De persoon die input geeft is niet de persoon die valideert. De data valideert.

Uitzondering: een expert met jarenlange ervaring. "Ik heb dit bij 30 klanten gezien" is bewijs. Dat begint met hoge confidence en wordt direct opgeslagen als expert rule.

Wat als de expert en de AI het oneens zijn?

De AI is geen ja-knikker en geen betweter. Er zijn drie scenario's:

1. **Objectief fout:** AI corrigeert direct. Geen discussie.
2. **Expert weet meer:** Specifieke ervaring wint van generieke trainingsdata. Expert rule, hoge confidence.
3. **Grijze zone:** AI geeft zijn perspectief met reden en vraagt waarom de expert anders denkt. Geen van beiden wint automatisch. Bij onopgelost meningsverschil: opslaan als spanning, monitoren bij volgende cases.

Het eerlijkste systeem durft te zeggen: "Ik ben het niet met je eens, en dit is waarom."