Les 2 — Lesstof

OpenCode — AI Code Assistants in de terminal

Vak:	AI Fundamentals
Cursus:	AI Developer
Volgende les:	Les 3 — Cursor Basics

Vak: AI Fundamentals **Opleiding:** NOVI Hogeschool Utrecht **Vorige les:** Les 1 — Introductie AI + LLMs **Volgende les:** Les 3 — Cursor Basics

Inhoud

1. Het idee — AI als pair-programmer

In Les 1 zag je hoe AI je kan helpen prototypes maken (v0.dev → website in 10 minuten). Vandaag een ander gebruik van AI: **in je terminal, naast je code, als pair-programmer**.

Dit type AI-tool noemt men een **coding agent**. Verschil met chat-AI:

- Chat-AI (ChatGPT): typ prompt → antwoord in browser. Jij implementeert handmatig.
- Coding agent: typ prompt → AI **bewerkt direct je files**, runt commando's, ziet errors, fixt zelf.

OpenCode is zo'n coding agent — open-source, in je terminal, werkt met elk model (OpenAI, Anthropic, Gemini, lokale modellen).

2. Wat is OpenCode

OpenCode is een open-source CLI coding agent. Geïnstalleerd via npm, je runt 'm in je project-folder en chat met AI die direct in je codebase werkt.

Kernfeatures

- **Multi-model** — kies OpenAI, Anthropic, Google, OpenRouter, Ollama (local)
- **Tool calling** — AI kan files lezen/schrijven, terminal commands runnen, tests draaien
- **AGENTS.md** — een markdown-file in je repo die AI als project-context gebruikt
- **Plan mode** — AI maakt eerst een plan voor je akkoord geeft (veiliger)
- **Open-source** — gratis, audit-baar, geen vendor lock-in

Wat krijg je niet (vs. paid tools)

- Geen visuele editor zoals Cursor
- Geen IDE-integratie (werkt in terminal)
- Geen background agents zoals Cursor's cloud agents

Voor: developers die in terminal werken, willen experimenteren met models, of geen subscription willen.

3. OpenCode vs Cursor vs Claude Code

Drie populaire coding-agents in 2025. Allemaal anders.

	OpenCode	Cursor	Claude Code
Type	CLI (terminal)	Full IDE (VS Code fork)	CLI (terminal)
Cost	Gratis (API-cost zelf)	$20/mnd Pro	$20/mnd Pro
Model	Elk (provider keuze)	Cursor's models + own keys	Alleen Claude
Best voor	Backend, scripts, experimenteren	Frontend, IDE-flow, leerlingen	Diepere code-tasks, agents
Curve	Steiler — terminal	Vlakker — IDE	Steiler — terminal

In deze leerlijn:

- Les 2: OpenCode (begrijpen wat een coding agent doet, terminal-flow)
- Les 3: Cursor (full IDE-ervaring)
- Later: combineer beide naar smaak

Veel pro-devs gebruiken **beide** — Cursor voor IDE-werk, OpenCode/Claude Code voor swift terminal-tasks.

4. Installatie + setup

Installeren

Eén command:

```
npm install -g opencode-ai
```

Of via brew:

```
brew install opencode-ai/tap/opencode
```

API key

OpenCode heeft een model nodig. Twee opties:

Optie A — OpenRouter (aanbevolen voor leerlingen)

- Account op openrouter.ai
- $5 credit toevoegen — werkt met alle modellen, prepaid
- Eén key voor OpenAI, Anthropic, Google, etc.

Optie B — Direct provider

- OpenAI key, Anthropic key, etc. — directe billing

Configureren

```
opencode auth login
# kies provider, plak key
```

Of via env:

```
export OPENROUTER_API_KEY=sk-or-v1-...
```

Starten

In je project-folder:

```
cd ~/projects/mijn-app
opencode
```

Terminal opent een chat-interface. Type je vraag.

5. Basis-workflow in OpenCode

Workflow 1 — Quick file edit

```
[opencode] > Voeg een README.md toe met basic install instructions

AI: [maakt README.md aan]
Je accepteert.
```

Workflow 2 — Multi-file feature

```
[opencode] > Voeg een /about pagina toe in Next.js. Match de styling
             van de homepage.

AI: [leest app/page.tsx voor stijl, maakt app/about/page.tsx]
Diff getoond. Je accepteert per file.
```

Workflow 3 — Bug fix

```
[opencode] > De Send button werkt niet. Console toont een TypeError.

AI: [leest console output, vindt de bug, fixt 'm]
```

Slash commands

- `/init` — genereer AGENTS.md voor dit project
- `/clear` — clear conversation history
- `/model` — switch tussen models
- `/plan` — toggle plan mode

Approval-flow

OpenCode vraagt approval voor:

- File-edits (per file, diff zichtbaar)
- Terminal commands (per command)

Goeie gewoonte: lees diff voordat je accepteert.

6. AGENTS.md — context geven

AGENTS.md = een markdown file in je project-root die AI gebruikt als "project context". Vergelijkbaar met `.cursorrules` (Cursor) of `CLAUDE.md` (Claude Code).

Wat zet je erin?

```
# AGENTS.md

## Project
- Naam: QuickPoll
- Stack: Next.js 16, TypeScript, Tailwind, Supabase
- Hosting: Vercel

## Conventies
- Use Tailwind classes, geen CSS-files
- Server Components by default, "use client" alleen waar nodig
- Maak geen comments tenzij gevraagd
- Run `pnpm lint` na elke file-edit

## Belangrijke files
- `app/` — Next.js App Router pages
- `lib/supabase.ts` — DB client
- `types/database.ts` — generated DB types
```

Effect

AI gebruikt deze context bij elke prompt. Geen "ik gebruik Next.js met App Router" herhalen — AI weet het al.

Best practice

Genereer eerst met `/init`, edit daarna handmatig. Hou 'm tussen 50-200 regels.

7. Plan mode vs Act mode

OpenCode heeft twee werkmodes.

Act mode (default)

AI doet direct wat je vraagt. Snelle iteratie. Voor: kleine taken, je weet wat je wilt.

Plan mode

AI maakt eerst een **plan** in tekst — welke files, welke wijzigingen, welke risk. Jij keurt af/goed. Pas daarna doet AI de wijzigingen.

```
[opencode] > /plan
[plan-mode] > Refactor /lib/auth.ts naar gebruik van Supabase auth

AI Plan:
1. Lees lib/auth.ts huidige logic (~80 regels)
2. Vervang door @supabase/ssr client
3. Update import in 3 files: app/login, app/api, middleware
4. Genereer types via supabase gen types

Akkoord? (y/n)
```

Voor: complexe refactors, onbekend gebied, code je niet wilt breken.

Vuistregel: plan mode bij twijfel. Kost je 30 seconden extra, voorkomt uren herstelwerk.

8. Slim werken — context-management

AI heeft een **context window** — wat het tegelijk kan onthouden. Bij OpenCode meestal 128k-200k tokens. Voelt veel, maar in een groot project ben je er snel doorheen.

Tips

Start nieuwe sessie per feature — `/clear` na elke voltooide feature voorkomt context-pollution.

Geef expliciet de juiste files — AI hoeft niet je hele repo te lezen. "Werk in app/auth/login.tsx" is beter dan "fix de login".

Houd AGENTS.md compact — meer is niet beter. AI leest dit bij elke prompt.

Skip large generated files — node_modules, .next/, dist/. OpenCode doet dit by default maar check.

Wanneer crasht context

- Mega-files (>5000 regels)
- Veel back-and-forth in één sessie (>50 berichten)
- Veel terminal-output (test-failures, log-dumps)

Symptoom: AI vergeet eerdere instructies, hallucineert files die niet bestaan. → `/clear`, herstart, kleinere scope.

9. Limitaties + valkuilen

AI maakt fouten

- Verzint imports die niet bestaan
- Gebruikt verouderde APIs (Next.js 14 syntax in een Next.js 16 project)
- Hallucineert function-signatures

Maatregel: lint na elke change, draai tests, lees diff.

AI is overtuigd van fout

Een fout antwoord klinkt soms net zo overtuigend als een goed. AI presenteert beide met evenveel zelfvertrouwen.

Maatregel: vraag "weet je dit zeker?", "geef bronnen", "test dit met de runtime".

Over-engineering

AI maakt soms te veel — wrapper-classes, abstracties, type-gymnastics. Voor productie OK, voor leren niet.

Maatregel: "houd het simpel, kortste werkende versie".

Onveilige commands

AI kan `rm -rf` voorstellen, of `git push --force`. Standaard moet je goedkeuren — maar dat is jouw veiligheidsmechanisme.

Maatregel: lees commands voor accepteren. Bij twijfel: weiger.

10. Productie-workflow tips

Tip 1 — Klein per commit

AI kan een uur aan code in 10 minuten schrijven. Verleidelijk om alles tegelijk te doen. **Commit per logische eenheid** — kleinere PRs, makkelijker review, makkelijker rollback.

Tip 2 — Test direct

"AI heeft het gemaakt, maar werkt het?" — runnen. Lint, build, unit tests. AI ziet eigen fouten via test-output en kan ze fixen — als je 'm de output geeft.

Tip 3 — Lees diffs

Sla geen diffs over door snelheid. Aandachtig lezen voorkomt dat verkeerde wijzigingen in main belanden.

Tip 4 — AGENTS.md als living doc

Update AGENTS.md als project evolueert. Nieuwe afspraken, nieuwe libraries, nieuwe gotchas → erin. AI-output verbetert.

Tip 5 — Combineer met handmatig denken

AI is sneller, maar jij snapt de business. Architectuurkeuzes, productdecisies, security-tradeoffs — daar ben jij verantwoordelijk. AI is een hulp, geen vervanger.

Bronnen

- **OpenCode docs:** https://opencode.ai/docs
- **OpenCode GitHub:** https://github.com/sst/opencode
- **OpenRouter:** https://openrouter.ai
- **AGENTS.md spec:** https://github.com/openai/agents.md
- **Cursor rules vs AGENTS.md:** https://docs.cursor.com/context/rules
- **Anthropic — Building effective agents:** https://www.anthropic.com/research/building-effective-agents