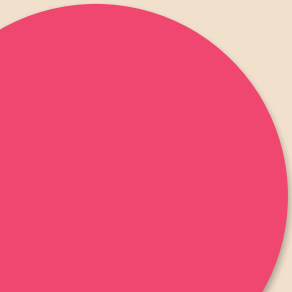




Les 13

Externe APIs + Cursor + Vercel

Van localhost naar productie — met Cursor agents en preview deploys



Terugblik

Van localhost naar de wereld in

Lessen 11-12:

- Les 11: AI SDK basics + Polderfest
- Les 12: Tool Calling

Tot nu toe: alles op localhost. Geen wereld eromheen.

Vandaag:

- Externe APIs aanroepen (geen LLM nodig)
- App naar productie met Vercel
- Feature branches met Cursor (Composer + Background Agent)
- GitHub Actions CI (lint + build per PR)

Planning

Vandaag — 180 minuten

Welkom + Terugblik	10 min
Theorie: Externe APIs in Next.js	15 min
Theorie: Cursor — Composer vs Background	15 min
Theorie: Vercel + GitHub Actions CI	15 min
LIVE DEMO 1 — Pokédex met Composer	30 min
LIVE DEMO 2 — Deploy naar Vercel	15 min
Pauze	15 min
LIVE DEMO 3 — Feature branch met Background Agent	25 min
LIVE DEMO 4 — GitHub Actions CI	15 min
Composer vs Background — wanneer welke?	10 min
Lesopdracht + Huiswerk	10 min
Vragen + Afsluiting	5 min

Theorie 1

Wat is een externe API?

Het idee: HTTP endpoints van iemand anders. `fetch()` → JSON terug.

Voorbeelden:

PokéAPI

Pokemon data — géén key

Tavily

Web search — key in header

Open-Meteo

Weer — géén key

GitHub API

Repos, issues, users

Stripe

Betalingen — key + secret

In Next.js — twee plekken om te fetchen:

Server-side

- Server Component / API route
- Key veilig, automatisch gecached
- Voor initiële data

Client-side

- `useEffect` + `fetch`
- Voor user-interactie
- Live updates, search, filter

Theorie 1

Externe API — Server Component code

```
// app/pokemon/[name]/page.tsx
async function getPokemon(name: string) {
  const res = await fetch(
    `https://pokeapi.co/api/v2/pokemon/${name}`,
    { next: { revalidate: 3600 } } // cache 1 uur
  );
  if (!res.ok) throw new Error("Pokemon niet gevonden");
  return res.json();
}

export default async function PokemonPage({ params }) {
  const pokemon = await getPokemon(params.name);
  return (
    <div>
      <h1>{pokemon.name}</h1>
      <img src={pokemon.sprites.front_default} alt="" />
    </div>
  );
}
```

Drie dingen om te onthouden:

1. async Server Component
2. next: { revalidate } cache
3. Error handling met !res.ok

Theorie 2

Cursor — twee soorten agents

Composer

Cmd+I

- Lokaal in editor
- Synchroon — jij wacht
- Multi-file diffs
- Jij accepteert per file
- Pair programming
- Snel iteraten op UI

Background Agent

Cmd+Shift+P → Background

- Cursor cloud sandbox
- Async — jij elders
- Opent branch + PR zelf
- Kan tests/deps draaien
- Delegeren
- Parallel meerdere features

Composer = pair programming. Background = delegeren.

Theorie 3

Vercel preview deploys — URL per branch

Hoe het werkt:

	push naar main	→ <code>je-app.vercel.app</code>	(productie)
	push naar feature/X	→ <code>je-app-git-feature-X-user.vercel.app</code>	(preview)
	PR open	→ Vercel comment in PR met preview URL	(klikbaar)

Environment scopes:

Production	Deploys vanaf main	Echte API key
Preview	Alle andere branches	Test API key
Development	Lokaal (.env.local)	Dev API key

Background Agent PR + Vercel preview = klikbare live demo voor reviewer.

Theorie 3

GitHub Actions CI — lint + build per PR

```
# .github/workflows/ci.yml
name: CI
on:
  pull_request:
    branches: [main]
  push:
    branches: [main]

jobs:
  check:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: pnpm/action-setup@v3
        with: { version: 9 }
      - uses: actions/setup-node@v4
        with: { node-version: 20, cache: pnpm }
      - run: pnpm install --frozen-lockfile
      - run: pnpm lint
      - run: pnpm build
```

Waarom?

- Voor merge: lint clean, types kloppen, build OK
- Background Agent maakt PR → CI draait → groen/rood
- Voorkomt 'werkte op mijn laptop'
- ~1-2 min op kleine Next.js app
- Caching met pnpm scheelt 60-80% tijd

Vandaag bouwen we

Pokédex — PokéAPI + Cursor + Vercel

Stack:

- Next.js 16 + TypeScript + Tailwind + Shadcn
- PokéAPI (géén key, gratis)
- Cursor (Composer + Background Agent)
- GitHub (repo + Actions)
- Vercel (productie + preview deploys)

Features:

- Pokédex list (eerste 151)
- Detail page per Pokemon
- Search met filter
- Type-filter chips



LIVE DEMO 1

Pokédex met Composer — ~30 min

- 1 `pnpm create next-app pokedex`
- 2 Open in Cursor — `Cmd+I` voor Composer
- 3 Prompt: bouw Pokédex-startpagina met 151 Pokémon
- 4 Composer maakt `app/page.tsx` + cards + sprites
- 5 `pnpm dev` — werkt lokaal
- 6 Tweede prompt: detail-pagina `/pokemon/[name]`
- 7 Polish: 'stats te dun, paarse gradient'

Composer is een TAB in Cursor, niet hetzelfde als Chat.

Multi-file diffs zichtbaar — accepteer per file.

LIVE DEMO 2

Deploy naar Vercel productie — ~15 min

- 1 GitHub: new repo 'pokedex' (public)
- 2 Lokaal: git init + add + commit + push
- 3 Vercel: Add New → Import Git Repository → kies pokedex
- 4 Klik Deploy — wachten ~60-90s
- 5 Open productie URL — werkt!
- 6 Wijziging maken → git push → 2e deploy in ~30s

Geen config nodig — Vercel detecteert Next.js automatisch.

Push naar main = deploy. Geen extra commando.



Pauze

15 minuten

LIVE DEMO 3

Feature branch + Background Agent — ~25 min

- 1 Cursor: Cmd+Shift+P → 'Open Background Agent'
- 2 Verbind repo aan Cursor cloud (eenmalig)
- 3 Specifieke prompt: branch + feature + push + PR
- 4 Wacht 3-5 min — volg agent in Cursor panel
- 5 PR verschijnt op GitHub + Vercel preview URL
- 6 Test feature live op preview URL
- 7 Code review op GitHub → merge → productie

Specifieke prompts — bestandsnamen, gedrag, output-formaat.

Parallel: 2 Background Agents tegelijk = 2 features parallel.

LIVE DEMO 4

GitHub Actions CI — ~15 min

1 Composer: 'voeg .github/workflows/ci.yml toe' (lint + build)

2 Push naar branch chore/add-ci → PR open

3 GitHub Actions tab → workflow draait → groen

4 Demo failing: introduceer typo, push → rode X

5 Fix met Composer → push → groen

6 Branch protection: main alleen via PR met groene CI

Background Agent kan typo's maken — CI vangt dat op vóór merge.

Geen 'werkt op mijn laptop' discussies meer.

Reflectie

Composer vs Background Agent — wanneer welke?

Snelle wijziging tijdens coden

Composer

Complex refactor — wil meekijken

Composer

Onbekend gebied / leren

Composer

Welomschreven feature, async

Background

Parallel werken aan 3 features

3x Background

Tijdens vergadering PR voorbereiden

Background

Mentale model:

Composer = pair programming — jij + AI samen

Background = delegeren — jij elders, AI levert PR

Lesopdracht + Huiswerk

Lesopdracht (30 min)

- Eigen Pokédex repo opzetten
- 1 feature met Composer
- Push naar GitHub
- Connect Vercel + deploy
- Productie URL werkt

Huiswerk (~2 uur, voor Les 15)

- A: Feature via Background Agent
- B: GitHub Actions CI workflow
- C: Branch protection op main
- D: DEPLOY.md (5 secties)
- Bonus: 2e API / env scoping

Beoordeling (10 pt — voldoende 6+):

- | | |
|--|------|
| • A — Background Agent feature + preview URL | 3 pt |
| • B — CI werkt + bewijs van groen + rood | 2 pt |
| • C — Branch protection actief | 1 pt |
| • D — DEPLOY.md compleet (5 secties) | 3 pt |
| • Productie URL werkt | 1 pt |

Afsluiting

Vragen?

Vandaag gezien:

- ✓ Externe API integratie in Server Components
- ✓ Cursor Composer voor synchrone changes
- ✓ Cursor Background Agent voor async features
- ✓ Vercel productie + preview per branch
- ✓ GitHub Actions CI (lint + build)
- ✓ Branch protection op main

Volgende les (Les 14): RAG + Embeddings

- Wat is MCP (Model Context Protocol)
- Bestaande MCP servers gebruiken in Cursor/Claude Desktop
- Eigen MCP server bouwen + laden
- Daarna Les 17 (Externe APIs deep), Les 18 (Auth + RLS)

Vragen? Feedback?