

Les 15 — Externe APIs + Cursor + Ver

Van localhost naar productie — met Cursor agents en preview deploys

Vak:	AI-Assisted Development
Opleiding:	NOVI Hogeschool Utrecht
Klas:	Klas A (3 uur fysiek, demo-driven)
Vervolg op:	Les 14 — RAG + Embeddings
Volgende les:	Les 16 — MCP servers

Leerdoelen

- Externe APIs aanroepen in Next.js Server Components met caching
- Verschil server- vs client-side fetching kennen
- Environment variabelen correct scopen (no NEXT_PUBLIC_ voor secrets)
- Cursor Composer en Background Agent inzetten op het juiste moment
- Next.js app deployen naar Vercel productie
- Preview deployments per feature branch begrijpen + gebruiken
- GitHub Actions CI workflow opzetten (lint + build)
- Branch protection toepassen op main

Inhoud

1. Externe APIs in Next.js
2. Server- vs client-side fetching
3. Environment variabelen
4. Cursor agents — Composer vs Background
5. Vercel deployments
6. Preview deployments per branch
7. GitHub Actions CI
8. Branch protection
9. Werkstroom — feature van begin tot productie
10. Productie-overwegingen
11. Bronnen

1

Externe APIs in Next.js

Een externe API is een HTTP-endpoint van iemand anders waar je data of functionaliteit kunt ophalen. Voor de meeste apps die je bouwt heb je er minstens één nodig.

Voorbeelden

- **Data-APIs** — PokéAPI, Open-Meteo, GitHub API, TMDB
- **AI-providers** — OpenAI, Anthropic, Tavily
- **Payment** — Stripe, Mollie
- **Email** — Resend, Postmark, SendGrid

Drie smaken qua authenticatie

- **Geen key** — PokéAPI, Open-Meteo. Direct fetchen.
- **API key in URL of header** — Tavily, TMDB. Key beheren als env-variabele.
- **OAuth flow** — Google, GitHub. Complexer, vaak via libraries.

Basis-pattern

```
// Server Component
async function getPokemon(name: string) {
  const res = await fetch(
    `https://pokeapi.co/api/v2/pokemon/${name}`
  );
  if (!res.ok) throw new Error("Niet gevonden");
  return res.json();
}

export default async function Page({ params }) {
  const data = await getPokemon(params.name);
  return <div>{data.name}</div>;
}
```

Drie dingen om te onthouden:

- Server Components mogen async zijn
- Errors moet je zelf afvangen (!res.ok)
- fetch in een Server Component wordt automatisch gecached door Next.js

2 Server- vs client-side fetching

Server-side

```
// app/pokemon/[name]/page.tsx
async function getPokemon(name) {
  const res = await fetch(
    `https://pokeapi.co/api/v2/pokemon/${name}`,
    { next: { revalidate: 3600 } } // cache 1 uur
  );
  return res.json();
}
```

Voordelen:

- API key blijft op de server
- Caching gratis via Next.js
- SEO-vriendelijk (HTML met data terug)
- Snellere TTFB

Wanneer: initiële data, lijsten, detail-pagina's.

Client-side

```
"use client";
import { useEffect, useState } from "react";

export function LivePokemon({ name }) {
  const [data, setData] = useState(null);
  useEffect(() => {
    fetch(`/api/pokemon/${name}`)
      .then((r) => r.json())
      .then(setData);
  }, [name]);
  return data ? <div>{data.name}</div> : <p>Loading...</p>;
}
```

Wanneer: user-interactie (search, filter), live updates.

Belangrijke regel

API keys NOOIT in client-code. Voor authenticated externe APIs: maak een API-route in app/api/... als proxy.

Cache-opties bij fetch

```
fetch(url, { cache: "force-cache" })           // permanent (default)
fetch(url, { next: { revalidate: 3600 } })      // ISR - herval na N sec
fetch(url, { cache: "no-store" })              // nooit cachen
fetch(url, { next: { tags: ["pokemon"] } })    // tag-based revalidation
```

3

Environment variabelen

Regel: alles wat geheim is — zonder NEXT_PUBLIC_ prefix.

```
// Server-only - veilig
process.env.OPENAI_API_KEY

// NEVER doen
"use client";
const key = process.env.NEXT_PUBLIC_OPENAI_KEY;
// staat in HTML response! Iedereen ziet het.
```

Vercel environment scoping

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

Per env-var kies je in welke scopes hij beschikbaar is. Zo verbruiken preview deploys niet je productie-quota.

4

Cursor agents — Composer vs Background

Cursor heeft twee soorten agents. Niet hetzelfde, niet uitwisselbaar.

Composer

- Lokaal in je editor (Cmd+I / Ctrl+I)
- Synchron — jij wacht, agent werkt, jij ziet diffs
- Multi-file edits in één keer
- Terminal-commands met approval
- Voor: pair programming, snelle changes, exploreren

Background Agent

- Runt in Cursor's **cloud sandbox**
- Asynchroon — jij geeft prompt, gaat iets anders doen
- Maakt zelf een branch + commit + opent PR
- Kan tests draaien, dependencies installeren
- Verbonden via Cursor cloud (eenmalige setup)
- Voor: welomschreven features, parallel werk, PR-prep

Aanroep

```
Cursor Composer:    Cmd+I (Mac) of Ctrl+I (Win/Linux)
Cursor Chat:        Cmd+L (kan switchen naar Agent mode)
Background Agent:   Cmd+Shift+P -> "Open Background Agent"
                    Of via Slack-integratie
```

Mentale model

Composer = pair programming. Jij + AI samen aan dezelfde plek.

Background Agent = delegeren. Je geeft taak, gaat iets anders doen, komt terug.

Wanneer welke

Scenario	Tool
Snel iets aanpassen tijdens coden	Composer
Refactor over 5 files — wil zien wat hij doet	Composer
Onbekend gebied / leren	Composer
Welomschreven feature, can-be-async	Background Agent
Parallel werken aan 3 features	3x Background Agent
Tijdens vergadering PR voorbereiden	Background Agent

Goede Background Agent prompt schrijven

Een Background Agent ziet je niet — je moet specifiek zijn.

Slecht:

```
"Voeg search toe."
```

Goed:

```
"Maak een nieuwe branch feature/search.  
Voeg een controlled input toe bovenaan app/page.tsx.  
Filter de Pokemon-lijst case-insensitive op naam terwijl  
gebruiker typt. Gebruik Tailwind. Houd stijl consistent  
met bestaande cards. Push de branch en open een PR met  
titel 'Add search bar' en korte beschrijving in PR body."
```

5 Vercel deployments

Vercel is hosting platform door Next.js team. Voor Next.js apps de simpelste deploy.

Eerste deployment

1. Push je repo naar GitHub
2. vercel.com -> Add New -> Import Git Repository
3. Kies repo, klik **Deploy**
4. ~60-90s later: live URL

Geen config nodig. Vercel detecteert Next.js automatisch.

CLI alternatief

```
pnpm i -g vercel
vercel login
vercel          # preview deploy van current dir
vercel --prod   # production deploy
vercel env pull # download .env.local van Vercel
```

6 Preview deployments per branch

Vercel maakt automatisch een preview deploy voor **elke branch en elke PR**.

```
git push origin main          -> Production deploy
                                (je-app.vercel.app)

git push origin feature/search -> Preview deploy
                                (je-app-git-feature-search-user.vercel.app)

PR open                        -> Vercel comment: "Preview: <URL>"
```

Wat krijg je

- Reviewers testen de feature live, niet alleen code-diff
- Background Agent maakt PR -> preview URL -> klikbaar
- Stakeholders zien feature voor merge
- Geen 'werkt op mijn laptop' meer

7

GitHub Actions CI

Een CI (Continuous Integration) check draait automatisch op je code. Minimaal: lint + build.

Minimaal workflow

```
# .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
```

Wat gebeurt er

1. **Trigger:** PR open of update, of push naar main
2. **Runner:** Ubuntu container op GitHub infra
3. **Stappen:** checkout, install pnpm + Node 20, install deps, lint, build
4. **Status:** groene check of rode X op de PR

Tijdsbudget: 1-2 minuten op een kleine Next.js app. Met cache: pnpm scheelt 60-80% tijd op herhaalde runs.

8

Branch protection

Standaard kan iedereen pushen naar main. Voor productie wil je dat niet.

Rules instellen op GitHub

1. Repo settings -> Branches -> Add rule
2. Branch name pattern: main
3. Check: **Require a pull request before merging**
4. Check: **Require status checks to pass** -> select CI workflow
5. Check: **Do not allow bypassing the above settings**

Effect

- Niemand kan direct naar main pushen — moet via PR
- PR kan pas merge als CI groen is
- Force-push naar main geblokkeerd
- Background Agents werken automatisch via PRs (zo bedoeld)

9

Werkstroom — feature van begin tot productie

1. Idee voor feature
2. Background Agent prompt (specifiek!)
3. Cursor opent branch + commits + opent PR
4. GitHub Actions CI draait - lint + build
5. Vercel zet preview deploy neer - URL in PR
6. Jij/team reviewt: code-diff + live preview
7. Eventueel: commentaar in PR, agent retried
8. Merge PR -> main
9. Vercel productie-deploy is automatisch
10. Klaar - live op je-app.vercel.app

Tijdsindicatie voor één feature: 10-30 minuten end-to-end.

10 Productie-overwegingen

Costs

Service	Free tier	Wanneer betalen
Vercel	Hobby (gratis persoonlijk)	Commercieel of >100GB bandwidth
GitHub Actions	2000 min/maand (public unlimited)	Bij grote teams
Cursor	Hobby (gratis), Pro (\$20)	Background Agents = Pro+
PokéAPI	Volledig gratis	n.v.t.

Security

- API keys NOOIT in client (geen NEXT_PUBLIC_)
- .env.local in .gitignore (vooraf checken!)
- Rotate keys regelmatig — vooral na PR's van Background Agents
- Vercel environment scoping om dev/prod te scheiden

Rollback

1. Vercel dashboard -> Deployments
2. Vorige werkende deploy -> 'Promote to Production'
3. Klaar, ~10 seconden — geen rebuild nodig

11 Bronnen

- Next.js fetching — nextjs.org/docs/app/building-your-application/data-fetching
- Next.js caching — nextjs.org/docs/app/building-your-application/caching
- PokéAPI — pokeapi.co
- Cursor docs — docs.cursor.com
- Cursor Background Agents — docs.cursor.com/background-agent
- Vercel docs — vercel.com/docs
- Vercel preview deployments — vercel.com/docs/deployments/preview-deployments
- Vercel environment variables — vercel.com/docs/environment-variables
- GitHub Actions — docs.github.com/en/actions
- GitHub branch protection — docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches