Les 14 — Lesstof

Cursor + Vercel: van leeg project naar preview-deploys

Vak: AI-Assisted Development **Opleiding:** NOVI Hogeschool Utrecht **Vorige les:** Les 13 — Agents
Volgende les: Les 15 — RAG + Embeddings

Doel van deze les

Aan het einde van deze les heb je:

- Een vers Next.js project gescaffold via `npx create-next-app`
- Een GitHub-repo met je code
- Een Vercel-project met productie-URL én preview-URL's per branch
- Vercel CLI lokaal gekoppeld + één keer `vercel env pull` gedaan
- Een werkende `.cursor/rules/general.mdc` en `AGENTS.md` in je repo
- Twee features gebouwd via Cursor in eigen branches → twee preview-URL's
- Eén Pull Request gemerged naar productie

1. Scaffolden met `npx create-next-app`

Wat het doet

`create-next-app` is het officiële scaffold-commando van het Next.js team. Eén commando, een paar interactieve vragen, en je hebt een werkende Next.js-app met sane defaults.

```
npx create-next-app@latest mijn-portfolio
```

De prompts uitgelegd

Vraag	Aanbevolen	Waarom
TypeScript?	Ja	strict typing voorkomt veel bugs
ESLint?	Ja	code-style consistency
Tailwind CSS?	Ja	utility-first styling, breed gebruikt
src/ directory?	Ja	houdt project-root schoon
App Router?	Ja	de moderne Next.js flow (server components)
Turbopack?	Ja	snellere dev-builds

Vraag	Aanbevolen	Waarom
Import alias?	@/*	@/components/foo ipv ../../components/foo

Wat je krijgt

```
mijn-portfolio/
├── .gitignore          # node_modules, .next, .env.local zijn al gegitignored
├── next.config.ts
├── package.json
├── tsconfig.json
├── src/
│   └── app/
│       ├── layout.tsx  # root layout
│       ├── page.tsx    # homepage
│       └── globals.css
└── public/
```

Direct `npm run dev` → `localhost:3000` werkt. Vanaf hier ben je productief.

2. Git + GitHub

Init en eerste push

```
git init
git add .
git commit -m "init"
git branch -M main

# Met gh CLI (snelst):
gh repo create mijn-portfolio --public --source=. --push

# Of handmatig:
# 1. github.com → New repository → naam → Create (leeg, GEEN README)
# 2. git remote add origin <url>
# 3. git push -u origin main
```

Veelgemaakte fouten

Fout	Oplossing
gh: command not found	brew install gh (Mac) / winget install GitHub.cli (Windows)
Authentication required	gh auth login doorlopen
error: src refspec main does not match any	Eerst git add . && git commit -m "init"
fatal: remote origin already exists	git remote remove origin en opnieuw

Wat zit er NIET in je commit

Check je `.gitignore` — Next.js zet er standaard de juiste regels in:

```
node_modules/
.next/
.env.local
.env*.local
```

Belangrijk: controleer altijd na je eerste push op github.com dat `.env.local` daar NIET staat. Een gelekte secret-key is dezelfde dag onveilig.

3. Vercel — drie stappen naar productie

Project importeren

1. Ga naar **vercel.com/new**
2. Klik **Import Git Repository**
3. Kies je `mijn-portfolio` repo
4. Framework auto-detect: **Next.js**
5. Klik **Deploy**

Wacht ±45 seconden. Wat Vercel doet:

- Kloont je repo
- Detecteert framework
- Draait `npm install`
- Draait `npm run build`
- Deploy't naar Edge Network
- Geeft je een productie-URL

Twee URL-types

Type	URL-format	Trigger
Productie	mijn-portfolio-<scope>.vercel.app	push naar main
Preview	mijn-portfolio-git-<branch>-<scope>.vercel.app	push naar elke andere branch

Wat je vanaf nu krijgt zonder extra werk

- Elke push naar GitHub → automatische deploy
- Elke nieuwe branch → eigen preview-URL

- Pull Request → Vercel-bot post preview-URL als comment
- Productie-URL update vanzelf bij merge naar `main`

4. Environment Variables in Vercel

Drie environments

Environment	Actief bij
Production	deploys vanaf main
Preview	alle andere branches
Development	vercel dev lokaal

Je kunt env vars in één, twee of alle drie tegelijk zetten.

`NEXT_PUBLIC_` prefix

Prefix	Wat gebeurt	Voorbeeld
NEXT_PUBLIC_FOO	wordt in browser-bundle gestopt	publieke API-URL, Supabase anon-key
FOO (geen prefix)	alleen server-side beschikbaar	OpenAI key, Stripe secret, DB password

Regel: zet `NEXT_PUBLIC_` ALLEEN op variabelen die echt publiek mogen zijn. Twijfel? Geen prefix.

Via het dashboard

Settings → Environment Variables → Add. Vul naam + waarde, kies environments, save.

5. Vercel CLI

Installeren

```
npm i -g vercel
vercel login          # opent browser voor auth
```

Belangrijkste commando's

```
vercel link                          # lokale folder → Vercel project
vercel env pull .env.local           # productie env vars → lokaal
vercel env pull --environment=preview .env.local

vercel env add OPENAI_API_KEY production
vercel env rm OPENAI_API_KEY production
vercel env ls                        # lijst alle vars

vercel                               # interactieve deploy
vercel --prod                        # productie deploy (handmatig)

vercel logs                          # runtime logs
vercel logs --follow                 # stream logs live
```

Typische workflow

```
# Project klaarzetten (eenmalig):
vercel link

# Bij elke nieuwe env var op productie:
vercel env pull .env.local

# Bij env var rotatie:
vercel env rm DATABASE_URL production
vercel env add DATABASE_URL production
```

6. Cursor — de IDE

Wat is Cursor

Cursor is een VS Code-fork met AI als kernfeature. Niet een extensie — het is de editor. Werkt met Claude, GPT, Gemini en lokale modellen.

Cursor 3 — twee windows

In Cursor 3 zijn er **twee hoofdvensters**:

Window	Wat je er doet	Switch met
Editor Window (classic)	Code schrijven en reviewen — als VS Code	Cmd+Shift+N → Editor
Agents Window	Meerdere agents parallel managen	Cmd+Shift+N → Agents

Sneltoets om te onthouden: `Cmd+Shift+N` switcht tussen beide.

Binnen het Editor Window:

- **Cmd+L** — chat-paneel (kies modus: Ask / Agent / Plan)
- **Cmd+K** — inline edit op selectie

- **Tab** — accepteer AI auto-completion
- **Cmd+P** — file search (zoals VS Code)

Binnen het Agents Window:

- Overzicht met lokale, cloud en remote agents
- Op één plek alle running + done agents

Gouden regel:

- Code schrijven of 1 feature bouwen → **Editor Window** + **Agent mode** in chat
- Meerdere features parallel → **Agents Window** met Background agents
- Snelle tweak in 1 file → **Cmd+K**

> **Versie-noot:** info hieronder is gebaseerd op de officiële Cursor 3 documentatie (`cursor.com/docs`). Cursor 3.7 en hoger heeft alle besproken features. Sommige UI-details kunnen per minor-versie iets verschillen.

Editor Window — stap voor stap

Je hoofdwerkplek voor code:

1. **Cmd+Shift+N** — switcht naar Editor Window (als je in Agents Window zat)
2. Layout: file explorer links, editor midden, terminal beneden (klassieke VS Code-stijl)
3. **Cmd+L** opent chat-paneel rechts
4. Voor snelle inline edits: selecteer code → **Cmd+K** → beschrijf wijziging
5. Cursor toont **diff inline** in de editor (niet in een aparte panel)
6. **Tab** om te accepteren, **Esc** om te rejecten
7. **Cmd+P** voor file search
8. **Tab** accepteert AI auto-completion suggesties

Wanneer: je standaard coding-flow, files reviewen met split screens, VS Code extensies gebruiken (linters, formatters, debugger).

Tip: kun je Cursor laten starten in Editor Window als default? Ja. Settings → "Open Agents Window on startup" → uit. Of start vanuit terminal met `cursor --classic`.

Agent mode (binnen chat in Editor Window) — stap voor stap

De AI-modus *binnen* het chat-paneel:

1. **Cmd+L** opent chat-paneel rechts (in Editor Window)
2. Bovenaan chat: **mode-selector** → kies **"Agent"**
3. Typ de **hele taak** (niet een specifieke kleine wijziging)
4. *"Bouw een /about pagina met team-grid in onze huisstijl"*
5. **Enter** — agent start zelfstandig

6. Cursor toont een lopende **takenlijst** + voortgang per stap
7. Agent edit meerdere files, runt terminal-commando's (vraagt vooraf), kan zelfs **web browsen**
8. **Geen limiet** op aantal tool-calls per taak
9. Aan het einde: samenvatting → **Keep / Accept all** of per file

Wanneer: grotere features die meerdere bestanden raken, refactors, nieuwe pagina's of API-routes.

Belangrijke tip: hou je prompt specifiek. *"Bouw een about-pagina met team-grid"* = goed. *"Bouw een mooie website"* = veel te open.

Background agents (Agents Window) — stap voor stap

Async cloud-based agents in geïsoleerde VMs:

Cmd+Shift+N → Agents Window opent.

Een agent aanmaken:

1. In Agents Window: klik **New agent** (of +)
2. Kies environment: **local / cloud / remote SSH**
3. Beschrijf de taak — specifiek, als een ticket
4. Optioneel: branch-naam (`feature/about`)
5. Klik **Start**
6. **Cmd+Shift+N** terug naar Editor Window — agent babysit zichzelf

Resultaat ophalen:

1. **Cmd+Shift+N** naar Agents Window
2. Klare agents staan op **Done** in de agents-lijst
3. Klik agent → zie **diff + commit message + alle stappen**
4. **Open in PR** → automatische PR op GitHub
5. Vercel maakt preview-URL → review → merge

Triggers van buitenaf: Background agents kun je in Cursor 3 ook starten vanuit je telefoon, Slack, GitHub of Linear — handig tijdens vergaderingen.

Wanneer: 2–4 parallelle features, lange refactors die je niet wilt babysitten, taken vanaf mobiel.

Context — `@`-mentions

In chat type je @ voor een autocomplete. Wat je kunt mention'en:

Tag	Wat het doet
@file:foo.tsx	inclusief één specifiek bestand
@folder:components	hele map als context
@code:functionName	losse functie of class

Tag	Wat het doet
@docs	externe documentatie
@web	live web-search
@git	git history / blame
@recommended	Cursor suggereert zelf relevante bestanden

Tip: hou de scope klein. Veel mentions in één chat verslechtert de output, niet verbetert.

Cursor Rules

Markdown-bestanden in `.cursor/rules/` die aan elke prompt worden toegevoegd als systeem-instructie.

Voorbeeld `.cursor/rules/general.mdc`:

```
---
description: Algemene project-conventies
alwaysApply: true
---

- TypeScript strict mode
- Tailwind voor styling — geen CSS modules
- Server components by default; "use client" alleen waar nodig
- Imports met @/ alias
- Geen any-types
```

YAML front-matter opties:

- `alwaysApply: true/false` — actief in elke prompt
- `globs: ["**/*.test.ts"]` — alleen actief op matching bestanden
- `description: "..."` — wat de rule doet

Project context — `AGENTS.md`

Eén markdown-bestand in je repo-root dat beschrijft:

- Tech stack + waarom die keuzes
- Architectuur en belangrijke patronen
- Hoe lokaal draaien, hoe deployen
- Pointers naar belangrijke bestanden

`AGENTS.md` wordt door Cursor, Claude Code én GitHub Copilot gelezen. Eén file, meerdere tools.

Background agents — async cloud

Tweede Cursor-sessie die asynchroon op een feature werkt, in een eigen Git branch, in een aparte tab.

Workflow:

1. Beschrijf feature → klik "open in background"
2. Agent werkt in eigen branch, jij blijft in main editor
3. Agent meldt zich klaar → review diff → apply/iterate/discard

Wanneer: voor parallelle features. Drie of vier agents tegelijk is normaal.

Top 7 tips voor power-users

1. **Cmd+I** voor Composer met multi-file context (niet alleen Cmd+L)
2. `@-recommended` laat Cursor zelf relevante bestanden voorstellen
3. **Plan + Apply per stap** — niet alles in één klap accepteren
4. `alwaysApply: false` rules voor opt-in conventies
5. **Auto-mode** — Cursor kiest beste model per taak
6. **Cmd+Shift+Enter** voor diff-preview vóór accept
7. **Notepads** — herbruikbare prompt-snippets per project

7. De feature-workflow

Vijf stappen, één feature

```
1. Plan      → Cursor chat: beschrijf de feature, vraag plan
2. Branch    → git checkout -b feature/x (Cursor kan dit ook)
3. Build     → Apply het plan, eventueel inline tweaks
4. Commit+   → git add . && git commit -m "feat: x" && git push
   Push
5. Preview   → Vercel maakt preview-URL, test & itereer
```

Eén feature ≈ 10 minuten

Inclusief AI denkwerk. Wat dit versnelt is niet dat AI mooier code schrijft — het is dat de **cycle korter wordt**.

Anti-patterns

- Geen branch maken, direct op `main` werken (geen preview, geen rollback)
- 5 features in één commit ("WIP" commits)
- Plan niet lezen → Apply → het deugt niet → terugkrabbelen
- Background agents starten zonder duidelijke prompt → bagger output

8. Production checklist

Voordat je een app "klaar" beschouwt:

Check	Waarom
Productie-URL werkt voor alle routes	Anders krijgen gebruikers 404
Env vars staan goed (alle environments)	Anders crashen API routes
.env.local staat in .gitignore	Anders lekken keys op GitHub
Eerste feature-branch + PR gedaan	Bewijs dat preview-flow werkt
.cursor/rules/general.mdc is ingevuld	Anders weet Cursor je conventies niet
AGENTS.md is ingevuld	Anders weet AI de architectuur niet
Build slaagt zonder warnings	Anders begin je met technical debt
404 en error.tsx pages bestaan	Voor nette foutmeldingen

9. Verder lezen

- **Cursor docs** — `docs.cursor.com` — features, shortcuts, Rules-syntax
- **Vercel docs** — `vercel.com/docs` — alle CLI-commando's, env vars, domains
- **Next.js docs** — `nextjs.org/docs` — App Router, server components, routing
- **AGENTS.md spec** — `agentsmd.org` — community-gedragen format
- **Pro Git** (gratis online) — `git-scm.com/book` — definitieve Git-referentie

Samenvatting

Vandaag heb je geleerd:

- Scaffolden met `npx create-next-app` — alle defaults uitgelegd
- Git init + GitHub push (via gh CLI of handmatig)
- Vercel project koppelen — productie + preview URLs gratis erbij
- Vercel CLI — `link`, `env pull/add/rm`, `logs`, `--prod`
- Environment variables — drie environments, `NEXT_PUBLIC_` prefix
- Cursor: chat, plan, build, inline edit, rules, AGENTS.md, background agents
- Feature workflow: plan → branch → build → push → preview
- Top tips voor Cursor power-users

Volgende les bouwen we hier RAG bovenop — een PDF Q&A-app; met embeddings.