Les 14

Cursor + Vercel

Van leeg project naar preview-deploys

Met AI als bouwpartner

Vandaag

Wat we live gaan bouwen

- npx create-next-app — vers project
- Git init + push naar GitHub
- Vercel project koppelen → eerste deploy
- Vercel CLI: link, env pull/add
- Cursor open: Plan, Build, Context, Rules, Background
- Feature in nieuwe branch → preview-URL
- Tweede feature via Background agent

Werkwijze:

Ik bouw klassikaal. Jullie kijken mee. Na pauze doen jullie het zelf.

Scaffold

Vers Next.js project in 30 seconden

```
npx create-next-app@latest mijn-portfolio
# kies: TypeScript, Tailwind, App Router, src/ dir
cd mijn-portfolio
npm run dev
```

Wat je krijgt:

- Werkende Next.js 16 app op localhost:3000
- TypeScript + Tailwind preconfigured
- ESLint klaar
- .gitignore met .env.local, node_modules, .next

Git

Code online krijgen

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

gh repo create mijn-portfolio --public --source=. --push
```

Of handmatig:

- github.com → New repository → naam → empty
- git remote add origin <url>
- git push -u origin main

Vercel

Van GitHub-repo naar live URL

Drie stappen:

- vercel.com/new → Import Git Repository
- Kies je mijn-portfolio repo
- Framework auto-detect: Next.js → Deploy

Wat Vercel doet:

- Klonet je repo
- Draait npm install + npm run build
- Deploy't naar Edge Network
- Geeft je een productie-URL terug
- Vanaf nu: elke push → automatische deploy

URLs

Wat krijg je gratis

PRODUCTIE

```
mijn-portfolio-<scope>
.vercel.app
```

Trigger:

push naar main

PREVIEW

```
...-git-<branch>-<scope>
.vercel.app
```

Trigger:

push naar elke andere branch

Voordeel preview-URLs:

- Iedereen kan klikken zonder lokale setup
- Vercel-bot post URL automatisch in Pull Request

Env Vars

Geheimen veilig op productie

Drie environments:

Production	deploys vanaf main branch
Preview	alle andere branches
Development	vercel dev lokaal

NEXT_PUBLIC_ prefix:

- MET prefix → in browser-bundle (publiek zichtbaar)
- ZONDER prefix → server-only (echt geheim)
- Twijfel? Geen prefix.

Vercel CLI

Sneller dan klikken

Wat de CLI je geeft:

- vercel link — koppel lokale folder aan project
- vercel env pull — productie env vars → .env.local
- vercel env add / rm — vars beheren zonder dashboard
- vercel logs — bekijk runtime logs
- vercel --prod — handmatige productie-deploy

Installeren:

```
npm i -g vercel
vercel login
```

CLI demo 1

vercel link + env pull

```
cd mijn-portfolio
vercel link
# → kiest scope (org/team) + project
# → maakt .vercel/project.json aan

vercel env pull .env.local
# → trekt ALLE env vars uit gekoppelde environment
# → schrijft naar .env.local
```

Voordeel: env vars niet handmatig kopiëren.

Tip: --environment=preview voor preview-specifieke vars.

CLI demo 2

vercel env add / rm

```
# Toevoegen
vercel env add OPENAI_API_KEY production

# Voor alle 3 environments tegelijk:
vercel env add DATABASE_URL production preview development

# Verwijderen
vercel env rm OPENAI_API_KEY production

# Lijst
vercel env ls
```

Handig bij key-rotatie: drie commando's en klaar.

Cursor

Alles in één IDE

Wat is Cursor:

- VS Code-fork met AI native ingebouwd
- Niet een extensie — het IS de IDE
- Werkt met Claude, GPT, Gemini en lokale modellen

Wat je krijgt boven gewone VS Code:

- Chat met volledige codebase-context
- Plan-mode (eerst denken, dan doen)
- Build-mode (autonoom uitvoeren)
- Inline edit (Cmd+K)
- Background agents (cloud, async)
- Cursor Rules (per-project)

Cursor 3

Twee windows — Cmd+Shift+N switch

Editor Window

(classic — VS Code-stijl)

- Code schrijven + reviewen
- Cmd+L = chat (Agent mode)
- Cmd+K = inline edit
- Cmd+P = file search

Agents Window

(nieuw in Cursor 3)

- Lijst van alle agents
- Local + cloud + remote SSH
- Vanuit telefoon/Slack starten
- Async, isolated VMs

Sneltoets: Cmd+Shift+N switcht tussen beide windows

Gouden regel:

- 1 feature bouwen → Editor Window + Agent mode in chat
- Meerdere parallel → Agents Window met Background agents

Editor Window

Je hoofdwerkplek voor code

1. Cmd+Shift+N — switcht naar Editor Window
2. Layout: file explorer | editor | terminal
3. Cmd+L opent chat-paneel rechts
4. Selecteer code → Cmd+K = inline edit
5. Diff verschijnt inline in editor
6. Tab = Accept, Esc = Reject
7. Cmd+P = file search (zoals VS Code)
8. Tab = accepteer AI auto-completion

Wanneer:

Standaard coding-flow, file viewing, VS Code extensies (linters, debugger).

Agent mode

AI bouwt autonoom — in chat

1. In Editor Window: Cmd+L opent chat
2. Mode-selector bovenaan → kies 'Agent'
3. Typ HELE taak (niet 1 wijziging)
4. 'Bouw /about pagina met team-grid in huisstijl'
5. Enter → agent start, toont takenlijst
6. Edit meerdere files + runt terminal + browst web
7. GEEN limiet op tool-calls in Cursor 3
8. Eind: 'Keep' / 'Accept all' of per file

Tip:

Hou je prompt specifiek. 'About-pagina met team-grid' goed. 'Mooie site' veel te open.

Background agents

Cloud VMs in Agents Window

Agent aanmaken:

1. Cmd+Shift+N → Agents Window
2. Klik 'New agent' (of '+')
3. Kies environment: local / cloud / remote SSH
4. Beschrijf taak specifiek (als ticket)
5. Klik 'Start' → Cmd+Shift+N terug naar Editor

Resultaat ophalen:

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

Bonus: triggers vanuit telefoon / Slack / Linear ook mogelijk.

Context

@-mentions in chat

`@file:foo.tsx`	specifiek bestand
`@folder:components`	hele map
`@code:functionName`	losse functie / class
`@docs`	externe documentatie
`@web`	live web-search
`@git`	git history / blame
`@recommended`	Cursor's eigen suggestie

Voorbeeld:

```
Refactor @file:UserCard.tsx zodat het @code:useUser uit @folder:hooks gebruikt
```

Cursor Rules

Per-project instructies

Wat zijn rules:

- Markdown in .cursor/rules/
- Worden aan ELKE prompt toegevoegd
- Per-bestand-glob mogelijk

Voorbeeld .cursor/rules/general.mdc:

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

- TypeScript strict mode
- Tailwind voor styling — geen CSS modules
- Server components by default
- Imports met @/ alias
- Geen any-types
```

AGENTS.md

High-level project context

CURSOR RULES

Instructies — doe het zo

- Per-prompt regels
- Style + naming
- Auto-apply

AGENTS.md

Documentatie — zo zit het

- Tech stack + waarom
- Architectuur
- Hoe runnen/deployen

Mooi detail:

AGENTS.md wordt ook gelezen door Claude Code en GitHub Copilot.

Eén bestand, meerdere AI-tools.

Cursor Tips

Power-user shortcuts

1. Cmd+I voor Composer (multi-file)
2. @-recommended laat Cursor suggesties doen
3. Plan + Apply per stap (niet alles in 1)
4. alwaysApply:false voor opt-in rules
5. Auto-mode: beste model per taak
6. Cmd+Shift+Enter voor diff-preview
7. Notepads = herbruikbare snippets

Anti-pattern: alles in 1 gigantische chat. Hou focused.

Feature flow

Van idee naar preview-URL

1	**Plan**	Beschrijf feature in Cursor
2	**Branch**	git checkout -b feature/x
3	**Build**	Apply plan, evt. inline tweaks
4	**Commit + Push**	Vercel start preview-build
5	**Preview-URL**	Test, geef feedback, itereer

Cyclus ≈ 10 minuten per feature, inclusief AI denkwerk.

Live demo 1

Hero section via Plan + Build

Wat ik live demonstreer:

- git checkout -b feature/hero
- Cursor Chat met Plan-toggle
- Prompt: 'Plan hero met gradient, titel, 2 CTAs'
- Plan reviewen → Apply
- Cmd+K voor inline tweaks (kleuren, spacing)
- git add . && git commit -m 'feat: hero section'

Resultaat: nieuwe hero in feature-branch, klaar voor push.

Push → preview

Vercel doet de rest

```
git push -u origin feature/hero
```

Binnen 30–60 seconden:

- Vercel start preview-build
- Preview-URL verschijnt in dashboard
- (Bij open PR) Vercel-bot plakt URL als comment

Open de preview-URL:

Je nieuwe hero staat live op een unieke URL,
gedeeld met iedereen die de link heeft.

Live demo 2

About page via Background agent

Wat ik live demonstreer:

- git checkout -b feature/about
- Cursor → 'Open background agent'
- Prompt: '/about page met team-grid, stijl als hero'
- Agent werkt in eigen tab — IK BLIJF DOORWERKEN
- Inline edit op een ander bestand intussen
- Agent klaar → review diff → apply
- Commit + push → tweede preview-URL

Effect: twee features tegelijk, beide met eigen preview.

PR → productie

Cyclus afsluiten

Op GitHub:

1. Beide branches → PR's openen
2. Reviewen in preview-URLs
3. Mergen naar main
4. Vercel detecteert merge → productie-deploy
5. Productie-URL update vanzelf

Mentaliteit:

main is altijd deploy-baar. Feature branches = experimenteerruimte.

Recap

Wat we vandaag hebben gedaan

- ✓ npx create-next-app — vers Next.js project
- ✓ Git init + push naar GitHub
- ✓ Vercel project + eerste deploy
- ✓ Vercel CLI: link, env pull, add/rm
- ✓ Cursor: Chat, Plan, Build, Inline, Rules, AGENTS.md
- ✓ Twee features in branches → twee preview-URLs
- ✓ PR's gemerged → productie

Cursor + Vercel = je standaard-werkbank voor alles wat komt.

Nu jullie

Lesopdracht — bouw je eigen flow

Wat je gaat doen (zie PDF voor details):

1. npx create-next-app — eigen project

2. Push naar je eigen GitHub

3. Vercel project + eerste deploy

4. vercel link + vercel env pull

5. .cursor/rules/general.mdc + AGENTS.md

6. Feature 1 — Plan + Build via Cursor

7. Feature 2 — via Background agent

8. PR + merge naar productie

Te delen in de chat:

- Productie-URL
- Tweede preview-URL + PR-URL

Volgende les

Les 15 — RAG + Embeddings

Wat we gaan bouwen:

- PDF Q&A-app from scratch
- Embeddings + vector search (pgvector)
- Context-injectie naar de LLM
- Antwoord op basis van eigen documenten

Voorbereiding:

- Je Next.js-app staat live op Vercel
- Minstens 2 PR's gemerged met preview-URLs
- AGENTS.md + .cursor/rules/ zijn ingevuld

Tot volgende week!