

Les 15 — Docenttekst (Autocue)

Fysieke les — letterlijk voorleesbaar. Slide-nummers tussen [SLIDE N].

Structuur: ±1 uur theorie + live demos → kwartier pauze → ±1,5 uur hands-on

>

Voorbereiding — accounts + tokens: - GitHub Personal Access Token (fine-grained, test-repo permissies) - Supabase Personal Access Token (`sbp\_...`) + je `project\_ref` - Vercel account (geen token vooraf — OAuth bij eerste gebruik) - Cursor + Pro Student ingelogd

>

Voorbereiding — `~/cursor/mcp.json`: - Alle 4 servers vooraf geconfigureerd - Cursor herstart — 4 indicators groen (github + supabase + vercel + polderfest) - Voor Vercel: OAuth-flow vooraf doorlopen

>

Voorbereiding — code: - `polderfest-mcp-finished` uitgepakt + `npm install` gedaan - `.env.local` werkt — `npm run dev` getest - Eén commit gedaan in `git init` (klaar voor Chat 1) - Backup productie-URL op Vercel als demo-fallback

>

Voor uitgebreide handleiding zie `Les15-Docenttekst.md` (normale versie).

BLOK 1 — Welkom + Vandaag

[SLIDE 1]

Welkom bij Les 15. Vandaag hebben we het over MCP — Model Context Protocol. Dit is de open standaard die in een jaar tijd is opgenomen door Cursor, Claude Desktop, OpenAI en veel meer. En vandaag heb jij hem ook.

[SLIDE 2]

Vandaag doen we drie blokken. Eerst kort de theorie — wat is MCP, waar komt het vandaan. Dan koppelen we live vier MCP servers — GitHub, Supabase, Vercel, en daarna onze eigen Polderfest. En in het laatste blok komt alles samen in een live demo van drie chats, waarmee we van leeg-project naar volledig gedeployd gaan zonder zelf één terminal-commando te typen.

Chat één maakt een GitHub repo aan via de GitHub MCP. Chat twee bouwt een nieuwe feature aan onze Polderfest server via de Supabase MCP. En chat drie maakt een Vercel project aan en deploy't dat naar productie.

BLOK 2 — Wat is MCP

[SLIDE 3]

Wat is MCP. Het is een open standaard die beschrijft hoe AI-clients verbinding maken met externe tools en data. Drie dingen worden vastgelegd: hoe ze contact maken, hoe servers beschrijven wat ze aanbieden, en hoe data heen-en-weer stroomt.

Anthropic heeft het eind 2024 geïntroduceerd. Open spec. En in een jaar tijd hebben Cursor, Claude Desktop, OpenAI, Windsurf en Zed het allemaal omarmd. Geen vendor lock-in.

De analogie die ik je wil laten onthouden: MCP is USB-C voor AI. Eén stekker, elke laptop.

[SLIDE 4]

Waarom is dit zo'n big deal. Stel je voor: jij hebt een handige tool gebouwd. Je wilt hem in Cursor gebruiken — Cursor heeft zijn eigen extension API. Je wilt hem in Claude Desktop — dat is een plugin systeem. Voor ChatGPT moet je een GPT maken met Custom Actions. Voor Windsurf weer iets anders.

Eén feature, vijf integraties. Allemaal anders.

Met MCP bouw je het één keer. En het werkt overal waar MCP gesproken wordt. Vijf integraties worden er één.

[SLIDE 5]

De architectuur is klein. Drie rollen om te onthouden.

De host is de app waarin je werkt — Cursor of Claude Desktop. De client is de MCP library in die host — daar hoef je niets aan te doen. En de server is wat WIJ bouwen of wat we koppelen.

Een server kan drie dingen aanbieden. Tools — functies die de AI mag aanroepen. Resources — read-only data. En prompts — voorgedefinieerde templates.

Vandaag focussen we op tools.

[SLIDE 6]

Het ecosysteem is enorm. Honderden MCP servers beschikbaar. GitHub voor issues en code. Supabase voor database. Vercel voor deployment. Filesystem, Slack, Linear, Notion, Postgres — noem maar op.

Vandaag koppelen we er vier in Cursor: GitHub, Supabase, Vercel — en onze eigen Polderfest. Met die vier kunnen we van leeg project tot live productie-URL.

BLOK 3 — Vier MCPs koppelen

[SLIDE 7]

Hier zie je de vier MCPs op een rij. Allemaal via HTTP, want dat is de moderne manier in Cursor 3. Geen losse processen meer, gewoon URLs.

GitHub gebruikt een Bearer token — dat is je Personal Access Token. Supabase ook een Bearer — je sbp-token. Vercel doet het anders en eleganter: OAuth via de browser, geen token nodig vooraf. En Polderfest is onze eigen lokale server zonder auth.

Eén patroon, vier servers. Allemaal in dezelfde mcp.json.

[SLIDE 8]

Hier zie je de complete mcp.json. Dit bestand staat in mijn home folder onder dot cursor slash mcp.json, dus globaal — werkt vanuit elke folder die ik open.

Vier entries onder mcpServers. Supabase met de mcp dot supabase punt com URL, een project ref als query parameter — read only true om veiligheid in te bouwen — en een Bearer header met mijn sbp token.

GitHub met api dot githubcopilot punt com slash mcp, Bearer met mijn ghp token.

Vercel — heel simpel — alleen type http en de URL mcp dot vercel punt com. Geen header. Cursor doet de OAuth bij eerste gebruik.

En Polderfest, lokaal op localhost drie duizend slash api slash mcp.

[SLIDE 9]

Even over de tokens. Drie tokens, één OAuth.

GitHub. Fine-grained PAT, niet classic. Op github.com onder Settings, Developer settings, Personal access tokens, Fine-grained. Permissies die je nodig hebt: Contents, Issues en Pull requests, allemaal read plus write. Token begint met ghp underscore.

Supabase. Op supabase punt com onder Account, Access Tokens, Generate new token. Belangrijk om te onthouden — dit is een account-level token, NIET de project anon key en NIET de service role key. Hij heeft toegang tot al je projecten. Begint met sbp underscore. De project ref vind je in je Supabase project URL.

Vercel. Geen token vooraf. Bij eerste gebruik in Cursor verschijnt er "needs login". Klik erop, OAuth-flow in de browser, autoriseer Cursor. Klaar.

[SLIDE 10]

Live koppelen. Vier stappen.

Eerst maak ik de dot cursor folder in mijn home directory en open mcp.json. Twee, ik plak de complete JSON van slide acht en vul mijn tokens in. Drie, ik herstart Cursor volledig met Cmd-Q en open hem opnieuw — een reload pakt mcp.json niet opnieuw in. En vier, ik open een chat en check onderaan de indicator. Vier groene servers moet ik zien.

Voor Vercel klik ik nog op "needs login" en doorloop de OAuth-flow.

(Ik laat dit live zien — open mcp.json, herstart Cursor, MCP-indicator gaat groen voor alle vier.)

BLOK 4 — Polderfest MCP

[SLIDE 11]

Tot zover bestaande servers. Nu de eigen kant.

Vandaag laat ik jullie kennismaken met onze Polderfest MCP. Zit in de finished zip die jullie straks krijgen.

Een complete Next.js app. Het meeste is gewoon Next.js zoals jullie het kennen. Het echte werk gebeurt in één bestand: app slash api slash mcp slash route.ts. Daarin staat de hele MCP server.

Vier tools zitten erin. searchBands om bands op te zoeken. getStageSchedule voor het schema van één dag. festivalStats voor totalen en verdelingen. En getWeather voor het weer via Open-Meteo.

[SLIDE 12]

Even een korte code-tour. Hoe ziet één tool eruit.

Het patroon is heel compact. Je roept server.tool aan met vier argumenten. Een naam — searchBands. Een beschrijving — let op, die leest de AI om te beslissen wanneer hij de tool gebruikt. Slechte beschrijving wordt niet aangeroepen. Het input-schema in Zod — die kennen jullie uit Les 12 en 13. En

een async functie die de logica doet.

Wat nieuw is ten opzichte van Les 12 en 13: het return-formaat. Geen plat object meer, maar een object met een content-array. Elk item heeft een type en een tekst. Dat is MCP-specifiek.

(Ik open even de finished route.ts en wijs de structuur aan.)

[SLIDE 13]

Lokaal draaien is drie commando's plus de env-vars.

Uitpakken, cd erin, npm install. Copy env.example naar env.local en vul de Supabase-keys in — die hebben jullie nog uit Les 11 en 12. Daarna npm run dev.

Op localhost drie duizend zie je een landingspagina. Het MCP-endpoint zit op slash api slash mcp. En polderfest stond al in mijn mcp.json, dus na herstart van Cursor zie ik vier groene indicators in plaats van drie.

(Ik draai dit live. Open localhost. Test in Cursor met "Welke jazz-bands spelen op zondag" — Polderfest MCP roept searchBands aan.)

BLOK 5 — LIVE: De drie chats workflow

[SLIDE 14]

Nu komt waar we naartoe hebben gewerkt. De climax van de demo.

Ik open drie aparte chats in Cursor. Elke chat heeft één doel. Chat één gebruikt GitHub MCP om een repo aan te maken en als remote te koppelen. Chat twee gebruikt Polderfest MCP plus Supabase MCP om een nieuwe feature te bouwen. Chat drie gebruikt Vercel MCP om een project aan te maken en te deployen.

Aan het eind: complete deploy van een nieuwe feature, vanaf mijn editor, zonder dat ik een terminal-commando of een browser hoeft te openen voor GitHub of Vercel.

[SLIDE 15]

Chat één. GitHub MCP.

Ik open een nieuwe Cursor chat — Cmd-L, vers begin. En ik typ:

"Maak een nieuwe public GitHub repo polderfest-mcp aan via GitHub MCP. Voeg toe als remote origin aan deze folder. Push de main branch."

Wat ik vervolgens zie. GitHub MCP roept create-onderstreep-repository aan. Ik krijg de URL terug. Cursor draait vervolgens git remote add origin met die URL in mijn terminal. En git push min-u origin main.

Ik switch naar github punt com om te bevestigen — de repo staat er, met onze code.

Geen browser. Geen gh repo create zelf typen. Cursor heeft het via MCP gedaan.

[SLIDE 16]

Chat twee. Polderfest features.

Nieuwe Cursor chat — Cmd-L, opnieuw vers. Ik typ de hele taak:

"Ik wil een review-systeem voor de Polderfest MCP. Eerste stap: vraag Supabase MCP om een tabel band-onderstreep-reviews te maken — id, band-id als foreign key, score één tot tien, comment, en created-at. Tweede stap: voeg twee tools toe aan at-file app slash api slash mcp slash route.ts — addReview met bandName score en comment, plus getReviews met bandName. Gebruik dezelfde patroon als bestaande tools."

Wat er gebeurt: Supabase MCP genereert de migration en draait hem, vraagt om bevestiging, ik zeg ja, tabel staat. Daarna gaat Cursor zelf de tool-code genereren in route.ts. Ik accept all.

Test in dezelfde chat — twee vragen die nu via Polderfest MCP gaan: "Geef Maud and The Cathedrals een review van 9, episch." Daarna: "Toon alle reviews voor Maud and The Cathedrals."

Drie MCPs werken samen in één chat: Supabase MCP voor de tabel, Cursor zelf voor de code, Polderfest MCP voor het testen.

[SLIDE 17]

Chat drie. Vercel MCP.

Nieuwe Cursor chat. Ik typ:

"Maak via Vercel MCP een nieuw Vercel-project voor mijn GitHub repo polderfest-mcp. Voeg toe als env vars in productie: SUPABASE underscore URL en SUPABASE underscore SERVICE underscore ROLE underscore KEY — ik plak ze hier. Daarna deploy naar productie en geef me de URL terug."

Vercel MCP gaat aan de slag. Project wordt aangemaakt, gekoppeld aan de GitHub repo. Env vars worden gezet. Deploy wordt getriggerd — ongeveer 45 seconden. Ik krijg de productie-URL terug.

Laatste actie van vandaag: ik vervang in mijn mcp.json het polderfest entry — de localhost URL wordt nu de Vercel productie-URL. Herstart Cursor. Vraag in chat: "Welke indie-bands spelen er op zaterdag?" Polderfest werkt, maar nu via productie, niet meer via mijn lokale dev-server.

Dat is wat we vandaag voor elkaar hebben gekregen.

BLOK 6 — Lesopdracht + Extra werk

[SLIDE 18]

Tot zover de demo. Na de pauze gaan jullie hetzelfde doen.

Acht stappen. Pak de finished zip uit. npm install plus env vars. npm run dev. Voeg vier MCPs toe aan mcp.json — de template staat in de lesbestanden, met placeholders waar jullie tokens moeten. Herstart Cursor — vier indicators groen.

Dan de drie chats. Chat één: GitHub MCP voor de repo. Chat twee: test Polderfest MCP met een paar vragen. Chat drie: Vercel MCP voor het project en de deploy. Eindig met polderfest in mcp.json wijzen naar de productie-URL.

Geen deliverables. Anderhalf uur. Ik loop rond.

[SLIDE 19]

Voor wie sneller is — niet verplicht. Doe wat ik in chat twee deed, voor je eigen feature.

Het patroon is hetzelfde. Vraag Supabase MCP om een nieuwe tabel met seed-data. Vraag Cursor om de bijbehorende tools in route.ts. Push naar GitHub — Vercel deploy't automatisch want het project is al gekoppeld. Test in chat.

Een paar ideeën: reviews zoals de demo. Een wishlist voor favoriete bands. Stage ratings. Of notes per band. Of iets totaal anders. Wat zou jij willen kunnen vragen aan je Polderfest data?

Geen deliverables.

[SLIDE 20]

Volgende week Les 16. RAG en embeddings. We bouwen een PDF Q&A; app from scratch. Embeddings met pgvector — dat is een Postgres-extensie voor vector search. Context-injectie naar de LLM.

Voor de volgende les heb je drie dingen nodig. Polderfest MCP staat live op Vercel — dat doen jullie vandaag. Vier MCPs gekoppeld in Cursor — github, supabase, vercel, polderfest. En je Supabase project blijft draaien, want we breiden het uit met pgvector.

Tot volgende week.