Les 15

MCP

Model Context Protocol

Superkrachten voor je AI via standard-tools

Vandaag

Drie blokken + climax

Wat we vandaag doen:

1	**Theorie**	wat is MCP, waarom (kort)
2	**Vier MCPs koppelen**	GitHub, Supabase, Vercel, Polderfest
3	**Drie chats workflow**	alles samen aan het werk

Climax — drie chats:

- Chat 1: GitHub MCP → repo aanmaken + remote
- Chat 2: Polderfest features bouwen via Cursor + Supabase MCP
- Chat 3: Vercel MCP → project + deploy

MCP

Open standaard voor AI-tools

- Hoe AI-clients verbinding maken met tools en data
- Hoe servers tools, resources en prompts beschrijven
- Hoe data heen-en-weer stroomt over een transport

Open spec — geen vendor lock-in:

- Gestart door Anthropic (november 2024)
- Cursor, Claude Desktop, OpenAI, Windsurf, Zed ondersteunen
- Analogie: USB-C voor AI — één stekker, elke laptop

Voor MCP

1 feature × 5 clients = 5 integraties

VOOR MCP

- Cursor: extension API
- Claude Desktop: plugins
- ChatGPT: GPTs / Actions
- Windsurf: eigen API
- Zed: eigen API
- = 5 integraties

MET MCP

- Eén MCP-server
- Cursor: werkt
- Claude Desktop: werkt
- ChatGPT: werkt
- Windsurf: werkt
- = 1 integratie

Dezelfde tool, één keer bouwen, overal beschikbaar.

MCP

Drie rollen + drie primitives

Drie rollen:

Host

Cursor, Claude Desktop

Client

MCP library

Server

GitHub / Polderfest / ...

Drie primitives:

Tools

functies

Resources

read-only data

Prompts

templates

Vandaag focus op TOOLS.

Ecosysteem

Honderden servers beschikbaar

GitHub

issues, PRs, repos

Supabase

DB queries, schema

Vercel

projects, deploys

Filesystem

lees/schrijf folder

Slack / Linear

tickets + posts

Brave / Notion

search + pages

Vandaag koppelen we vier:

GitHub, Supabase, Vercel + onze eigen Polderfest.

Vier MCPs

Het overzicht

MCP	Type	Auth
GitHub	`HTTP (remote)`	Bearer (PAT)
Supabase	`HTTP (remote)`	Bearer (sbp_-token)
Vercel	`HTTP (remote)`	OAuth (browser)
Polderfest	`HTTP (lokaal)`	geen

Alle vier in één mcp.json — zelfde patroon, andere URLs.

Pad: ~/.cursor/mcp.json (globaal, werkt overal).

mcp.json

~/.cursor/mcp.json — de complete config

```
{
  "mcpServers": {
    "supabase": {
      "type": "http",
      "url": "https://mcp.supabase.com/mcp?project_ref=<ref>&read_only=true",
      "headers": { "Authorization": "Bearer sbp_..." }
    },
    "github": {
      "type": "http",
      "url": "https://api.githubcopilot.com/mcp/",
      "headers": { "Authorization": "Bearer ghp_..." }
    },
    "vercel": {
      "type": "http",
      "url": "https://mcp.vercel.com"
    },
    "polderfest": {
      "type": "http",
      "url": "http://localhost:3000/api/mcp"
    }
  }
}
```

Patroon: type http + url + optioneel headers. Vercel doet OAuth.

Tokens

Drie tokens, één OAuth

GitHub PAT (fine-grained):

- github.com → Settings → Developer settings → PAT → Fine-grained
- Permissions: Contents + Issues + PR (read+write)
- Token begint met ghp_...

Supabase Personal Access Token:

- supabase.com → Account → Access Tokens → Generate
- ACCOUNT-level — NIET project anon key / service role
- Token begint met sbp_... — project_ref uit project URL

Vercel: geen token vooraf

Bij eerste gebruik in Cursor → "Needs login" → OAuth in browser

Live koppelen

Vier stappen

1. mkdir -p ~/.cursor && nano ~/.cursor/mcp.json
2. Plak de complete JSON + vul je tokens in
3. Cmd+Q Cursor → opnieuw openen (geen reload!)
4. Open chat → MCP-indicator: 4 servers groen

Voor Vercel:

- Klik "Needs login" naast vercel-indicator
- OAuth-flow in browser → autoriseer Cursor
- Indicator wordt groen

Polderfest MCP

Onze eigen server

```
polderfest-mcp/
├── app/
│   ├── api/mcp/route.ts   ← MCP server zelf
│   ├── lib/supabase.ts    ← Supabase client
│   ├── page.tsx           ← landingspagina
│   └── layout.tsx
├── package.json
├── .env.example
└── README.md
```

Vier tools:

searchBands	zoek op query/genre/dag	**getStageSchedule**	schema voor 1 dag
festivalStats	totalen + verdeling	**getWeather**	Open-Meteo forecast

Code

Eén tool — anatomie

```
server.tool(
  "searchBands",
  "Zoek bands op query, genre, of dag.",
  {
    query: z.string().optional(),
    genre: z.string().optional(),
    day: z.enum(["vrijdag","zaterdag","zondag"]).optional(),
  },
  async ({ query, genre, day }) => {
    let q = supabase.from("bands").select("*").limit(50);
    if (query) q = q.ilike("name", `%${query}%`);
    if (day) q = q.eq("day", capitalizeDay(day));
    const { data } = await q;
    return { content: [{ type: "text", text: JSON.stringify(data) }] };
  },
);
```

Bekend: Zod. Nieuw: server.tool() + content-array.

Lokaal draaien

Drie commando's

```
unzip polderfest-mcp-finished.zip
cd polderfest-mcp-finished
npm install
cp .env.example .env.local
# vul SUPABASE_URL + SERVICE_ROLE_KEY
npm run dev
```

Resultaat:

- Landingspagina: localhost:3000
- MCP endpoint: localhost:3000/api/mcp
- Polderfest zit al in mcp.json → herstart Cursor → 4 groene indicators

Drie chats

Demo overview — alles samen

Chat 1 — `GitHub MCP`

Repo aanmaken + remote

Chat 2 — `Polderfest + Supabase`

Nieuwe feature bouwen

Chat 3 — `Vercel MCP`

Project + deploy + env vars

Geen `git`, `gh` of `vercel` zelf typen. Cursor + MCPs doen het.

Chat 1

GitHub MCP: repo + remote

Nieuwe Cursor chat — prompt:

```
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 er gebeurt:

- GitHub MCP → create_repository (URL terug)
- Cursor draait `git remote add origin <url>`
- Cursor draait `git push -u origin main`
- Open github.com → repo staat er

Geen browser. Geen `gh repo create` zelf.

Chat 2

Polderfest features — Cursor + Supabase MCP

```
Ik wil een review-systeem voor de Polderfest MCP.

Stap 1: vraag Supabase MCP om tabel band_reviews
  (id, band_id FK, score 1-10, comment, created_at)

Stap 2: voeg twee tools toe aan @file:app/api/mcp/route.ts:
- addReview(bandName, score, comment)
- getReviews(bandName)
```

Test in dezelfde chat:

- "Geef Maud and The Cathedrals een review: 9, episch concert"
- "Toon alle reviews voor Maud and The Cathedrals"

Drie MCPs samen: Supabase + Cursor + Polderfest.

Chat 3

Vercel MCP: project + deploy

```
Maak via Vercel MCP een nieuw Vercel-project voor mijn
GitHub repo `polderfest-mcp`.

Voeg env vars (production):
- SUPABASE_URL = <plak>
- SUPABASE_SERVICE_ROLE_KEY = <plak>

Daarna: deploy + geef me de URL.
```

Resultaat:

- Productie-URL: polderfest-mcp-<scope>.vercel.app/api/mcp
- Update polderfest in mcp.json → productie-URL
- Herstart Cursor → werkt zonder lokale dev-server

Lesopdracht

Doe de hele flow zelf — ±1 uur

1. Pak finished zip uit, install, env vars, npm run dev
2. Voeg 4 MCPs toe aan ~/.cursor/mcp.json
3. Cursor herstart → 4 indicators groen
4. Chat 1: GitHub MCP → repo + remote
5. Chat 2: test Polderfest met een paar vragen
6. Chat 3: Vercel MCP → project + deploy
7. Update mcp.json met productie-URL → herstart

Geen deliverables. Gewoon doen tot het werkt.

Extra werk

Voeg een eigen feature toe

Voor wie sneller is. Niet verplicht.

Doe Chat 2 nog een keer voor je eigen idee:

1. Bedenk feature die nieuwe data nodig heeft
2. Supabase MCP → tabel + seed-data via Cursor
3. Cursor → genereer nieuwe MCP tool(s)
4. Push → Vercel deploy't automatisch
5. Test in Cursor chat

Ideeën:

- addReview + getReviews (zoals demo)
- addToWishlist + getMyWishlist
- rateStage + getStageRatings
- addNote + getNotes

Volgende les

Les 16 — RAG + Embeddings

Wat we gaan bouwen:

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

Voorbereiding:

- Polderfest MCP staat live op Vercel
- Vier MCPs gekoppeld in Cursor
- Supabase project blijft draaien — we breiden uit met pgvector

Tot volgende week!