

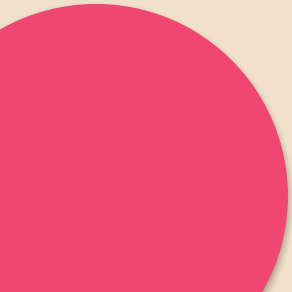


Les 16

MCP

Model Context Protocol

Eén protocol, alle AI-clients, jouw tools beschikbaar



Terugblik

Het probleem dat MCP oplost

Lessen 11-15: AI SDK, tool calling, agents, RAG, deploy.

Tools die je bouwt zitten VAST aan jouw app:

- X Wil je dezelfde tool in Cursor? Apart bouwen.
- X In Claude Desktop? Nog een keer.
- X In een script dat 's nachts draait? Drie keer.

Met MCP: één server, alle clients begrijpen het

Bouw je tool één keer, gebruik 'm in elke MCP-client.

Visual: USB-C voor AI tools — één standaard connector.

Planning

Vandaag — 180 minuten

Terugblik + waarom MCP	15 min
Theorie: MCP architectuur	20 min
Theorie: bestaande servers + eco	15 min
LIVE DEMO 1 — MCP SDK + eerste server	25 min
LIVE DEMO 2 — Laden in Cursor/Claude	20 min
Pauze	15 min
LIVE DEMO 3 — Eigen Polderfest MCP server	30 min
LIVE DEMO 4 — Resources + Prompts	20 min
MCP vs Tool Calling	10 min
Lesopdracht + Huiswerk	10 min

Theorie

Wat is MCP?

"Model Context Protocol is an open standard for connecting AI assistants to data sources and tools."

— Anthropic, november 2024

MCP Client

Cursor, Claude Desktop, custom apps

MCP Server

Jouw code: tools, resources, prompts

Protocol

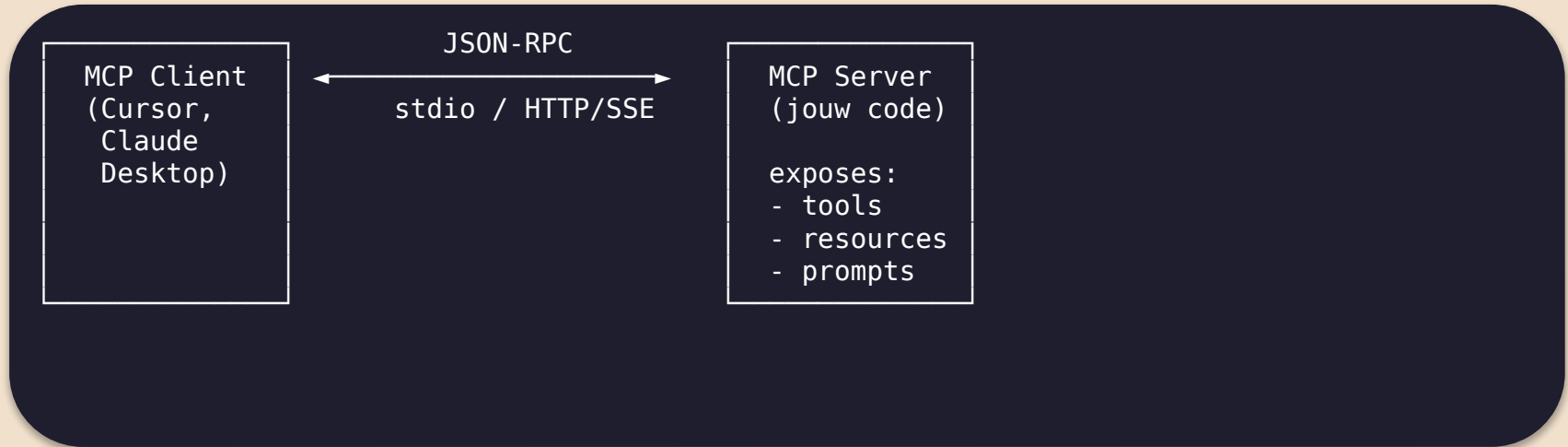
Eén taal voor de communicatie

Wat krijg je:

- Bouw je tool ÉÉN keer, gebruik 'm in elke MCP-client
- Eco van 500+ bestaande servers (filesystem, github, postgres, ...)
- Voor productie-apps: decoupling tussen AI-frontend en jouw backend

Theorie

MCP architectuur



Drie primitieven:

Tools	uitvoerbare functies (zoals tool calling)
Resources	data die de AI mag lezen (files, DB)
Prompts	herbruikbare prompt-templates

Twee transports:

stdio	lokaal proces, stdin/stdout (default)
HTTP/SSE	remote, voor cloud-deploys

Theorie

Bestaande MCP servers — het ecosysteem

Officieel (Anthropic, via npm):

@modelcontextprotocol/
server-filesystem

@modelcontextprotocol/
server-github

@modelcontextprotocol/
server-slack

@modelcontextprotocol/
server-postgres

@modelcontextprotocol/
server-puppeteer

@modelcontextprotocol/
server-memory

Community (500+): linear, notion, asana, figma, stripe, youtube, ...

Hoe gebruik je ze in Cursor:

```
// ~/.cursor/mcp.json
{
  "mcpServers": {
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": { "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_..." }
    }
  }
}
```

Restart Cursor → server actief.

Theorie

Eigen MCP server — TypeScript SDK

Package:

```
pnpm add @modelcontextprotocol/sdk
```

Skeleton:

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StdioServerTransport } from "@modelcontextprotocol/sdk/server/stdio.js";
import { z } from "zod";

const server = new McpServer({ name: "polderfest", version: "1.0.0" });

server.tool("searchBands", "Zoek bands",
  { day: z.string().optional() },
  async ({ day }) => {
    const bands = await queryDb(day);
    return { content: [{ type: "text", text: JSON.stringify(bands) }] };
  }
);

await server.connect(new StdioServerTransport());
```

Drie regels: maak server, registreer tool, connect transport.

Vandaag bouwen we

Polderfest MCP server

Doel: MCP server die Polderfest-data exposeert. Werkt in Cursor, Claude Desktop, anywhere.

Wat zit erin:

Tool	searchBands	Filter op dag/stage/genre
Tool	getBandStats	Aggregate counts
Tool	addFavorite	Write — voeg toe aan favorites
Resource	bands://list	Volledige bandenlijst (read-only)
Prompt	dailyRecap	"Geef recap van dag X" template

Demo: eerst in dev (stdio), daarna geladen in Cursor + Claude Desktop.

LIVE DEMO 1

MCP SDK + eerste server — ~25 min

- 1 pnpm init + add @modelcontextprotocol/sdk zod
- 2 src/index.ts — minimal server skeleton (15 regels)
- 3 Eerste tool: searchBands met mock-data
- 4 pnpm build → dist/index.js
- 5 MCP Inspector: npx @modelcontextprotocol/inspector dist/index.js
- 6 Inspector UI: tool aanroepen → JSON resultaat

MCP Inspector is jouw vriend — debug zonder Claude/Cursor nodig.

LIVE DEMO 2

Laden in Cursor + Claude Desktop — ~20 min

Cursor (~/.cursor/mcp.json):

```
{  
  "mcpServers": {  
    "polderfest": {  
      "command": "node",  
      "args": ["/full/path/dist/index.js"]  
    }  
  }  
}
```

Claude Desktop: zelfde JSON, andere file.

- 1 Edit config, restart Cursor
- 2 Settings → MCP → groen vinkje
- 3 Chat: 'Zoek bands op zaterdag' — tool wordt aangeroepen
- 4 Idem Claude Desktop
- 5 Eén server, twee clients, geen extra werk



Pauze

15 minuten

LIVE DEMO 3

Echte Polderfest server — ~30 min

- 1 Server koppelen aan Supabase (env vars via mcp.json)
- 2 searchBands echt naar database
- 3 getBandStats toevoegen (counts per groep)
- 4 addFavorite write-tool — let op user-intent in description
- 5 Errors als data: { isError: true }
- 6 Live test in Cursor — echte data uit DB komt terug
- 7 Logging via console.error (stdout = protocol)

Belangrijk: MCP server = gewoon Node-proces. Gebruik dezelfde gewoonten.

LIVE DEMO 4

Resources + Prompts — ~20 min

Resources — read-only data die AI mag laden:

```
server.resource("bands-list", "bands://list", async () => ({
  contents: [{
    uri: "bands://list",
    mimeType: "application/json",
    text: JSON.stringify(allBands),
  }],
}));
```

In Cursor: @polderfest:bands-list → resource in context

Prompts — herbruikbare templates:

```
server.prompt("daily-recap", "Recap van dag", { day: z.string() },
({ day }) => ({
  messages: [{ role: "user", content: {
    type: "text", text: `Geef recap van ${day}...`
  } }],
})
);
```

Reflectie

MCP vs Tool Calling — wanneer welke?

Tool Calling (Les 12)

- In jouw Next.js app
- Alleen jouw chat-users
- Setup-complexiteit laag
- Snel, lokaal
- Voor: app-features

MCP (Les 16)

- Aparte server, los proces
- Alle MCP-clients
- Distributie via npm
- Server proces met state
- Voor: herbruikbare tools

Mentale model:

Tool calling = function in mijn app

MCP server = library die elke AI-client kan gebruiken

Lesopdracht + Huiswerk

Lesopdracht (30 min)

- MCP server skeleton
- 1 tool met mock-data
- MCP Inspector test
- Laden in Cursor of Claude
- Chat-vraag werkt

Huiswerk (~2 uur)

- A: 3 read-tools + 1 write naar Supabase
- B: 1 resource toevoegen
- C: Laden in beide clients
- D: MCP.md (5 secties)
- Bonus: prompt-template / npm publish

Beoordeling (10 pt — voldoende 6+):

- | | |
|-----------------------------|------|
| • A — tools werken | 3 pt |
| • B — resource werkt | 1 pt |
| • C — geladen in 2 clients | 2 pt |
| • D — MCP.md compleet | 3 pt |
| • Server runt zonder errors | 1 pt |

Afsluiting

Vragen?

Vandaag gezien:

- ✓ MCP = open protocol voor AI ↔ tools/data
- ✓ Drie primitieven: tools, resources, prompts
- ✓ Twee transports: stdio + HTTP/SSE
- ✓ Eigen server in <50 regels code
- ✓ Werkt in Cursor + Claude Desktop + custom clients

Volgende les (Les 17): Externe APIs in diepte

- OAuth flows (Google, GitHub login)
- Webhooks ontvangen + verifiëren
- Paid APIs (Stripe checkout, Resend email)
- Rate-limit + retry patterns

Daarna: Les 18 — Supabase Auth + RLS multi-user (de laatste!)

Vragen?