Les 13

Agents

Polderfest krijgt brains

Multi-step tools + externe API

Terugblik

Tool calling — wat hadden we?

Vorige les (Les 12):

- AI kreeg toegang tot tools (functies)
- Eén tool per call: bv. searchBands(genre)
- Goed voor: 'voer deze functie uit'
- Gebouwd in de Polderfest-app: chat die zoekt in Supabase

Maar wat bij complexe vragen?

"Plan een zaterdagavond met indie + techno, geen overlap,

alleen onder een dak als het regent."

Eén tool-call is niet genoeg. Drie tools + plan + combineren = AGENT.

Verschil

Tool calling vs Agent — een continuüm

Geen harde lijn — in AI SDK v6 dezelfde syntax. Verschil zit in hoe je het ontwerpt.

Aspect	Tool calling (Les 12)	Agent (Les 13)
Vraag-type	'Voer deze functie uit'	'Bereik dit doel — kies zelf hoe'
Aantal stappen	Meestal 1-3 (ook met stopWhen)	Vaak 4-10+, variabel
Wie kiest volgorde	Jouw prompt stuurt sterk	AI plant zelf, autonomer
Bronnen combineren	Per tool één bron	DB + externe API + … door elkaar
Reflectie	Nee — direct antwoord	AI checkt resultaat, beslist vervolg

Beide kunnen externe APIs — execute() is gewoon JavaScript.

Agent = mindset, niet aparte API.

Code-uiting: hogere stopWhen (8+ ipv 2-3) en bredere/opener system prompts.

Theorie

Wanneer voegt een agent écht waarde toe?

Tool calling is genoeg als:

- Je weet welke functie nodig is
- Antwoord ligt in één bron
- Pad is voorspelbaar (2-3 calls)

Agent is écht zinvol als:

- Open vraag ('plan', 'vind beste')
- Meerdere bronnen combineren
- Pad onvoorspelbaar (soms 3, soms 7)
- Resultaat triggert vervolgvraag

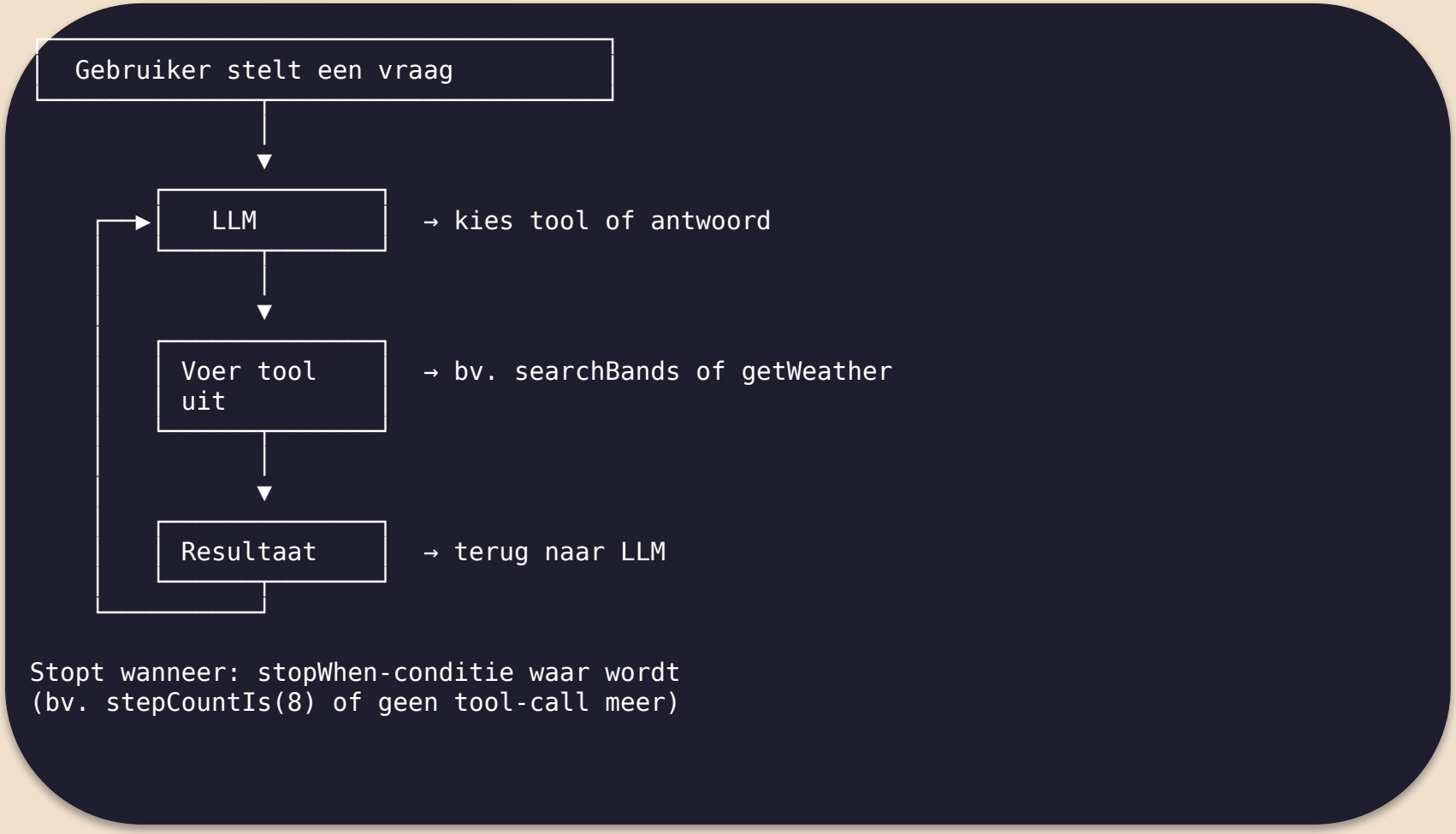
Risico's (waarom niet altijd):

- Lopen langer — meer tokens, hogere kosten
- Moeilijker te debuggen door non-determinisme
- Voor simpele vragen overkill — tool calling is sneller én goedkoper

Advies: start met tool calling. Schaal naar agent als je system prompt vol stappen-recepten staat.

Theorie

Hoe ziet de agent-loop eruit?



Aanpak

Stappen naar onze Polderfest-agent

1

Bedenk de tools

searchBands (Supabase) · getStageSchedule (Supabase) · getWeather (externe API)

2

Schrijf de system prompt

Vertel de agent zijn rol, tools en werkwijze

3

Voeg stopWhen toe

Veiligheid — bv. stepCountIs(8)

4

Bouw een UI

Toon de tool-calls — anders is het magic zonder zicht

5

Test met echte vragen

"Plan een avond…", "Welke band op…?"

AI SDK

tool({ description, inputSchema, execute })

```
import { tool } from "ai";
import { z } from "zod";

searchBands: tool({
  description: "Zoek bands op genre en/of dag.",
  inputSchema: z.object({
    genre: z.string().optional(),
    day: z.enum(["vrijdag", "zaterdag", "zondag"]).optional(),
  }),
  execute: async ({ genre, day }) => {
    // Supabase query
    return data;
  },
}),
```

`description` — wat de AI hierover leest om te beslissen

`inputSchema` — Zod-schema, AI vult dit automatisch in

`execute` — wat er gebeurt als de tool wordt aangeroepen

AI SDK

stopWhen — de agent-knop

```
import { stepCountIs } from "ai";

streamText({
  model: openai("gpt-5.2"),
  messages,
  tools: { searchBands, getStageSchedule, getWeather },
  stopWhen: stepCountIs(8),   // □ dit
});
```

Zonder stopWhen:

AI doet één tool-call en stopt. Standaard chat met tools.

Met stopWhen:

AI loopt door tot conditie waar is. Een echte agent.

Combineer met hasToolCall('finishPlan') voor stop-na-specifieke-tool.

AI SDK

Externe API als tool

```
getWeather: tool({
  description: "Weer-forecast voor een festivaldag.",
  inputSchema: z.object({ date: z.string() }),
  execute: async ({ date }) => {
    const res = await fetch(
      `https://api.open-meteo.com/v1/forecast?...&start_date=${date}`
    );
    const data = await res.json();
    return {
      temperatureMaxC: data.daily.temperature_2m_max[0],
      rainChancePercent: data.daily.precipitation_probability_max[0],
    };
  },
}),
```

Open-Meteo: gratis, geen account, geen API key. Perfect voor demos.

Dit kon ook al in Les 12 tool calling — execute() is gewoon JavaScript.

Verschil bij agents: meerdere bronnen door elkaar combineren in één antwoord.

Vandaag extra

Drie agent-features bovenop Les 12

prepareStep

model per stap

```
Stap 0 → krachtig (gpt-5.2)
Stap 1+ → goedkoop (gpt-5-
mini)
Lagere kosten, zelfde
kwaliteit
```

hasToolCall + finishPlan

stop op signaal

```
stopWhen: [stepCountIs(8),
  hasToolCall("finishPlan")]
Forceer gestructureerd
eindplan
```

addToFavorites

write-tool

```
Tool schrijft naar Supabase
Agent verandert de wereld,
niet alleen lezen
```

Combinatie: agent plant met krachtig model, voert uit met goedkoop, schrijft favoriet naar DB, sluit af met gestructureerd plan.

Dit is écht agent-werk — niet zomaar tool calling met meer stappen.

Roadmap

Polderfest → Polderfest-met-agent

Vandaag bouwen we ÉÉN extra route bovenop de bestaande Polderfest-app.

Al bekend uit Les 11/12

- ✓ tool() — description + schema + execute
- ✓ stopWhen + stepCountIs
- ✓ Supabase client opzetten
- ✓ searchBands / getDaySchedule
- ✓ Multi-step tool calling
- ✓ Bestaande tabellen: bands + user_favorites

Nieuw vandaag

STEP 7	getWeather (externe API)
STEP 8	addToFavorites (WRITE-tool)
STEP 9	finishPlan (structured output)
STEP 10	hasToolCall stop-conditie
STEP 11	prepareStep (model per stap)
STEP 13	UI tool-call rendering

Komende slides: 6 code-stappen — per stap zoek je op // STEP N in route.ts of page.tsx.

STAP 7

getWeather — externe API als tool

Zoek in editor (Cmd+F):

`STEP 7 — getWeather` app/api/agent/route.ts

```
getWeather: tool({
  description: "Weer-forecast voor een festivaldag.",
  inputSchema: z.object({ date: z.string() }),
  execute: async ({ date }) => {
    const res = await fetch(
      `https://api.open-meteo.com/v1/forecast?...&start_date=${date}`,
    );
    return await res.json();
  },
}),
```

execute() is gewoon JavaScript — fetch werkt. Open-Meteo: gratis, geen account, geen API-key. Externe APIs konden ook al in Les 12.

STAP 8

addToFavorites — eerste WRITE-tool

Zoek in editor (Cmd+F):

`STEP 8 — addToFavorites` app/api/agent/route.ts

```
addToFavorites: tool({
  description: "Voeg een band toe aan favorieten.",
  inputSchema: z.object({ bandName: z.string() }),
  execute: async ({ bandName }) => {
    // 1. Zoek band op naam → krijg band_id
    const { data: band } = await supabase
      .from("bands").select("id, name")
      .ilike("name", `%${bandName}%`).limit(1).single();
    if (!band) return { error: `Band niet gevonden` };

    // 2. Insert in bestaande user_favorites tabel
    const { data, error } = await supabase
      .from("user_favorites")
      .insert({ user_email: "demo@polderwave.app",
                band_id: band.id })
      .select().single();
    return { added: true, band: band.name };
  },
})
```

NIEUW vs Les 12: tool die DATA SCHRIJFT. Gebruikt bestaande user_favorites tabel (FK naar bands). Productie: echte auth + RLS (Les 10).

STAP 9

finishPlan — structured eindplan

Zoek in editor (Cmd+F):

`STEP 9 — finishPlan` app/api/agent/route.ts

```
finishPlan: tool({
  description: "Roep dit ALS LAATSTE aan met gestructureerd eindplan.",
  inputSchema: z.object({
    summary: z.string(),
    items: z.array(z.object({
      time: z.string(),
      band: z.string(),
      stage: z.string(),
      reason: z.string(),
    })),
  }),
  execute: async ({ summary, items }) => ({ summary, items, ok: true }),
}),
```

NIEUW: Zod-schema dwingt structuur af. UI rendert dit als nette timeline-kaart. Geen vrije-vorm-tekst meer die je moet parsen.

STAP 10

hasToolCall — stop op signaal

Zoek in editor (Cmd+F):

`STEP 10 — hasToolCall` app/api/agent/route.ts

```
stopWhen: [
  stepCountIs(8),                  // STAP 5 — veiligheid
  hasToolCall("finishPlan"),       // STAP 10 — klaar-signaal
],
```

NIEUW: combineer stop-condities. Agent stopt direct zodra finishPlan is aangeroepen — gegarandeerd nette afsluiting, ongeacht resterende stappen.

STAP 11

prepareStep — model per stap

Zoek in editor (Cmd+F):

`STEP 11 — prepareStep` app/api/agent/route.ts

```
prepareStep: ({ stepNumber }) => {
  if (stepNumber === 0) {
    return { model: openai("gpt-5.2") };       // planning
  }
  return { model: openai("gpt-5-mini") };       // uitvoering
},
```

NIEUW: stap 0 = krachtig model voor begrip + planning. Stap 1+ = goedkoper model voor uitvoering van tool-calls. Significant lagere kosten en latency, zelfde eindresultaat.

STAP 13

UI — tool-calls visualiseren

Zoek in editor (Cmd+F):

`STEP 13 — Tool-call rendering` `app/agent/page.tsx`

```
{m.parts.map((p, i) => {
  if (p.type === "tool-finishPlan") {
    return <PlanCard result={p.result} />;  // nette timeline
  }
  if (p.type === "tool-addToFavorites") {
    return <FavoriteBadge name={p.input.bandName} />;
  }
  if (p.type.startsWith("tool-")) {
    return <ToolDetails part={p} />;  // expand-able json
  }
})}
```

Loop door m.parts en herken tool-* parts. Speciaal renderen voor finishPlan (kaart) en addToFavorites (⭐ badge) maakt het magisch.

Stap voor stap

Polderfest → Polderfest-met-agent

1. Polderfest-original uitpakken
2. Nieuwe route app/api/agent/route.ts aanmaken
3. Eerste tool: searchBands (Supabase)
4. stopWhen: stepCountIs(8) toevoegen
5. Tweede tool: getStageSchedule
6. Derde tool: getWeather (externe API)
7. System prompt schrijven
8. UI app/agent/page.tsx met tool-call-weergave
9. Testen in browser
10. Pushen naar Vercel

Volledig stap-voor-stap met code:

`Les13-Stap-voor-stap.md in lesbestanden — jullie referentie`

polderfest-original.zip = startpunt · polderfest-with-agent.zip = referentie

LIVE DEMO

Polderfest-agent in actie

Drie prompts in de browser:

1. **"Plan een avond met indie en techno voor zaterdag."**

   → searchBands × 2 + getStageSchedule

2. **"Wat speelt er op zondag en hoe is het weer dan?"**

   → getWeather (externe API) + searchBands

3. **"Welke band moet ik NIET missen op vrijdag?"**

   → creatieve combinatie, agent vormt mening

Daarna in de code:

`app/api/agent/route.ts` — drie tools + stopWhen

`app/agent/page.tsx` — UI met tool-call-weergave

NU JULLIE

Bouw zelf een Polderfest-agent

Materialen in lesbestanden/:

`polderfest-original.zip`	startpunt — uitpakken of eigen Polderfest gebruiken
`polderfest-with-agent.zip`	referentie als je vastloopt — mag je openen
`Les13-Stap-voor-stap.md`	elf concrete stappen — volg één voor één

Stappenplan:

1. Unzip starter OF gebruik je eigen Polderfest uit Les 12
2. Volg Les13-Stap-voor-stap.md — werk in eigen tempo
3. Test in browser met drie voorbeeld-prompts
4. Deploy naar Vercel (env vars niet vergeten!)
5. Deel je productie-URL in de chat

Vragen? Hand op of roep me — ik loop rond.

Afsluiting

Wat we vandaag hebben gedaan

- Verschil tool calling vs agent — stopWhen is de knop
- Drie nieuwe AI SDK functies: tool(), stepCountIs(), multi-step loops
- Externe APIs als tool — Open-Meteo, geen key nodig
- Polderfest-app uitgebreid: chat-met-context → echte agent
- Live demo + zelf gebouwd

Volgende les (Les 14): Externe APIs + Cursor + Vercel deploy

Pokédex bouwen met Cursor Composer + Background Agent, naar productie met preview deploys.

Afronding naar productie-niveau development.

Vragen?