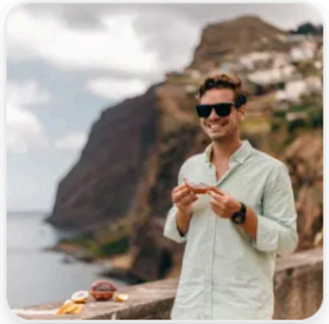
Combine Skills and MCP To Close the Context Gap

As AI agents become more capable, their biggest limitation is no longer reasoning — it's **context**.

MCP Dev Summit NA 2026 • New York City • Pedro Rodrigues • Supabase

Hi, I'm Pedro 👋

Full time Software Engineer, Part time Sports Lover



AI Tooling Engineer @ Supabase
MCP Dev Summit Europe 2025 Keynote Speaker
MCP enthusiast

🇵🇹 **Co-founder of Lisbon AI Week**
Co-hosted MCP Conference Lisbon 2025

Connect on LinkedIn

pedro-neves-rodrigues @Rodriguespn23 Rodriguespn

What is MCP?

An open standard for connecting AI agents to external systems. Like USB-C for AI applications.

PRIMITIVES

Tools
Functions agents invoke to act

Resources
Data sources for context

Prompts
Reusable interaction templates

OTHER MCP FEATURES

- **OAuth 2.1**
  Credential isolation & governance
- **Transports**
  Stdio + Streamable HTTP
- **Sampling**
  Server-initiated LLM completions
- **Elicitation**
  Request user confirmation

Source: modelcontextprotocol.io

What are Skills?

Folders containing instructions, scripts, and resources that agents discover and load dynamically when relevant to a task.

pdf-processing/SKILL.md

```
---
name: pdf-processing
description: Extract PDF text, fill
  forms, merge files. Use when
  handling PDFs.
---

## When to use this skill
Use this skill when the user
needs to work with PDF files ...

## How to extract text
1. Use pdfplumber for text ...

## How to fill forms
...
```

Frontmatter

Name and description — agents use this lightweight index (~100 tokens) to discover and select the right skill

Instructions

Procedural knowledge, best practices, and domain expertise in natural language — loaded on demand (<5k tokens)

Bundled Resources

Optional scripts/, references/, and assets/ — executable code, docs, and templates the agent uses during execution

Source: agentskills.io/what-are-skills

MCP Tools vs Skill Scripts

RPC-boundary functions vs executable code bundled with instructions.

MCP tools

- + typed arguments with JSON schema
- + process & credential isolation
- + remote execution
- − schema bloat at scale [1]
- − higher ceremony — restart on changes, JSON-RPC overhead

Skill scripts

- + zero schema overhead until invoked
- + instant iteration — edit, save, done
- + agents can self-author scripts
- − requires shell access — broader security surface
- − portability issues across environments

[1] Solutions: Claude Code Tool Search Tool (platform.claude.com) · Code Mode MCP (blog.cloudflare.com)

MCP Resources vs Skill References

Same goal — contextual data for agents. Different distribution models.

MCP resources

- + standardized protocol across all servers
- + remote access, no local filesystem needed
- + dynamic content — updates without client changes
- − under-defined in spec — "application driven" is vague
- − inconsistent client adoption

Skill references

- + co-located with instructions, git-friendly
- + no protocol overhead — just read a file
- + works offline, no server dependency
- − local only — no remote distribution
- − no dynamic updates without re-downloading

MCP Prompts vs SKILL.md

Who loads the context — the user or the agent?

MCP prompts

- + user-invoked — explicit, predictable
- + structured parameters with server-side context
- − requires manual invocation every time
- − no autonomous discovery

SKILL.md

- + agent-pulled — autonomous discovery
- + rich content: workflows, conditional logic, hundreds of lines
- − no standardized remote distribution (yet)
- − no built-in versioning

Different Problems. Different Solutions.

The primitives overlap on the surface. But they solve fundamentally different problems.

MCP solves the N×M problem

N agents × M data sources = unsustainable custom integrations. MCP provides a standard protocol so every agent speaks the same language to every data source.

Skills solve context saturation

80 tools can burn **30k+ tokens** before any user input — up to **72%** of the context window. [1] Skills use progressive disclosure: lightweight index upfront, full content on demand.

[1] MCP vs CLI, Hugging Face, huggingface.co/blog/nielsr/mcp-vs-cli

Different Problems. Different Solutions.

The primitives overlap on the surface. But they solve fundamentally different problems.

MCP solves the N×M problem

N agents × M data sources = unsustainable custom integrations. MCP provides a standard protocol so every agent speaks the same language to every data source.

Skills solve context saturation

80 tools can burn **30k+ tokens** before any user input — up to **72%** of the context window. [1] Skills use progressive disclosure: lightweight index upfront, full content on demand.

🏎️ Think of it this way

MCP is the **driver** — executing actions, turning the wheel, hitting the gas.

Skills are the **co-pilot**— reading the map, calling out turns, providing context for what's ahead.

> *"You would not fire the driver just because the co-pilot knows the way."*

[1] MCP vs CLI, Hugging Face, huggingface.co/blog/nielsr/mcp-vs-cli

Our Experience at Supabase

Supabase is the Postgres development platform. Postgres database, Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings, all in one platform.

Our MCP server

20+ tools including database, edge functions, branching, storage, and more. Highlighting the three most relevant:

- `execute_sql` Run queries against the database
- `get_advisors` Scan for database performance and security improvements
- `search_docs` Search Supabase documentation

A Supabase skill

A skill that teaches agents **how** to use those tools:

Search docs before acting
Verify against current docs, not training data

Security checklist
security_invoker on views, RLS on every table, search_path on functions

Opinionated workflows
Scaffold migrations, run advisors, check the diff

Same prompt. Different context.

We asked two agents the same thing: create a `team_summary_view` for a collaborative task board. One had MCP tools. The other had MCP tools **+ a Supabase skill**.

PROMPT

"Create a team_task_summary view so team members can see how their team is doing."

MCP only

Claude Sonnet 4.6 — Supabase MCP server, no skill

```
create view team_summary_view
as select ...
```

Missing `security_invoker`. Team members could see **all teams' data**, not just their own.

MCP + Skill

Claude Sonnet 4.6 — same MCP server + Supabase skill with security checklist

```
create view team_summary_view
with(security_invoker = true)
as select ...
```

The agent found the RLS policies page with the view disclaimer. Each team member only sees **their own team's data**.

This pattern repeats

We ran 6 Supabase scenarios across Claude Code and Codex with three conditions: no tools (baseline), MCP tools only, and MCP + Skill. Scored by LLM judge on Braintrust.

● Supabase-specific tasks (3) ● Postgres general tasks (3)

Agent	Tasks	Baseline	MCP Only	MCP + Skill
Claude Code (Opus 4.6)	Supabase	58%	50%	67%
	Postgres	58%	50%	67%
Claude Code (Sonnet 4.6)	Supabase	50%	50%	67%
	Postgres	42%	67%	75%
Codex (GPT-5.4)	Supabase	75%	67%	88%
	Postgres	67%	75%	88%
Codex (GPT-5.4 Mini)	Supabase	42%	58%	75%
	Postgres	42%	67%	67%

AVG TURNS **Baseline** 7.8 → **MCP+Skill** 11.7 (~1.5x)

AVG TOKENS **Baseline** 8.8k → **MCP+Skill** 9.4k (+7%)

Score = avg Task Completeness (0–100%) · —= not yet run · Efficiency metrics averaged across completed latest baseline and MCP+Skill runs