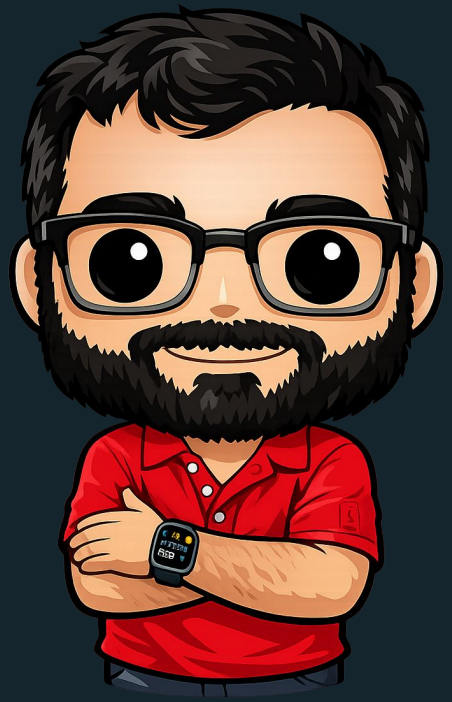
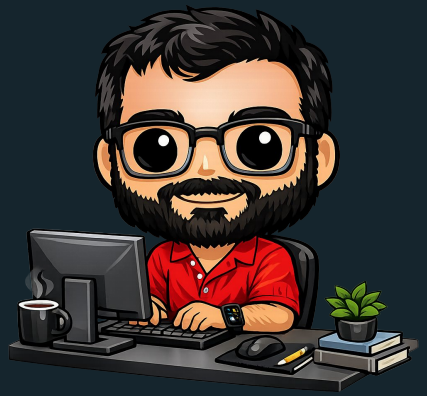

APOLLO
April 2, 2026
Your #1 Docs Audience Isn't Human:
Dev Ed's MCP Strategy at Apollo



Daniel Abdelsamed

Staff Software Engineer

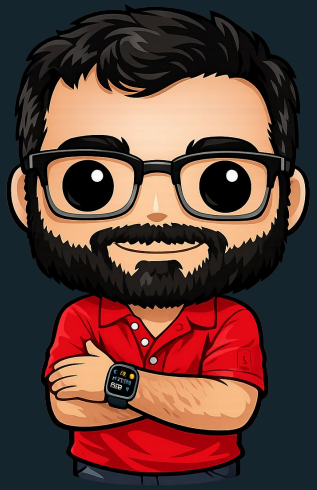
Education @ Apollo GraphQL



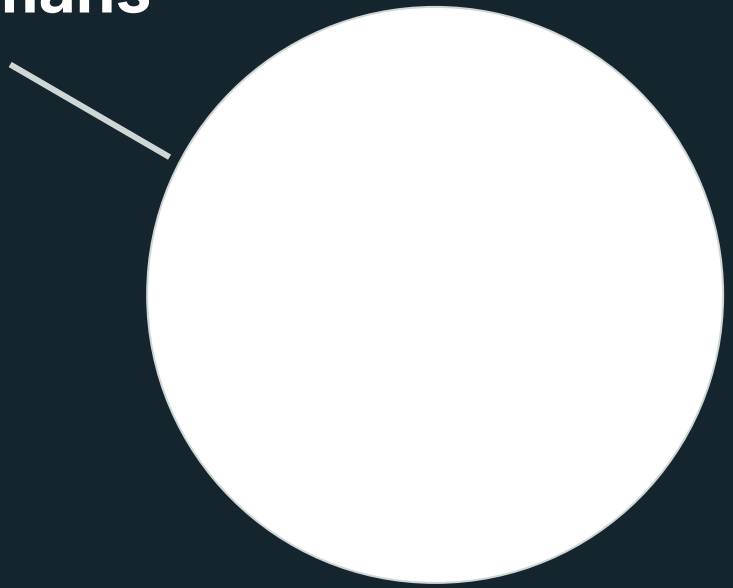
Authors

APOLLO DOCS

Readers

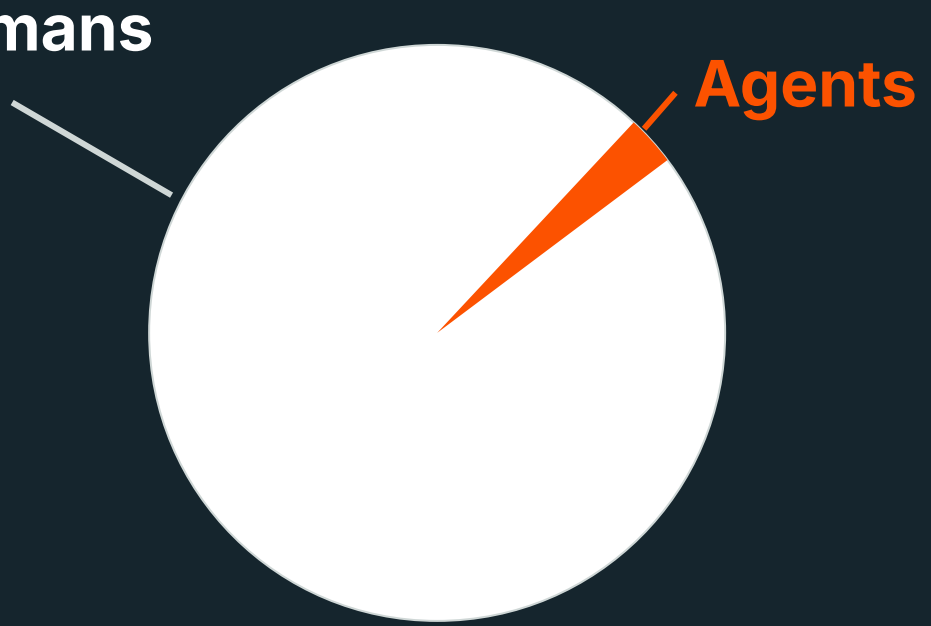


Humans

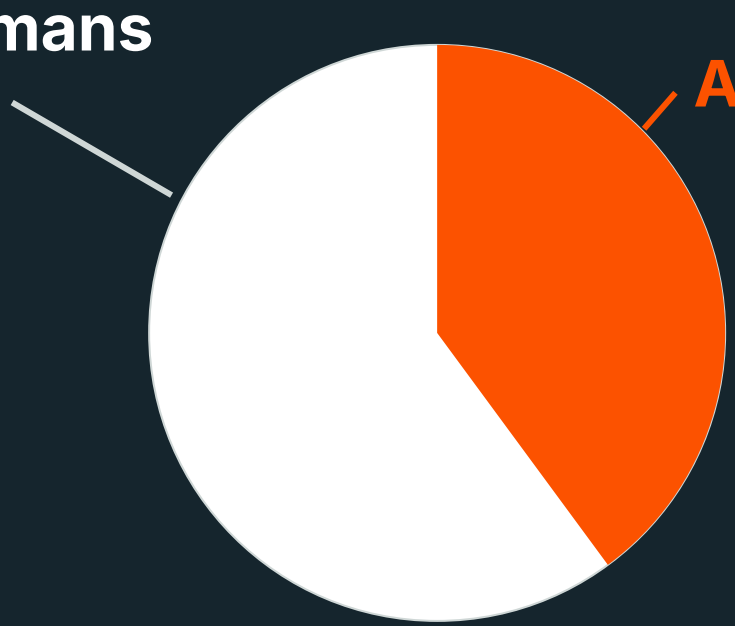


Audience

2022



Audience 2024



Audience 2026

>40%

traffic is agentic

The dual audience dilemma

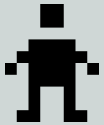
AGENTS

Learn from patterns

Dry, to the point content

Strict do’s and don’t

Code examples above all



HUMANS

Explanations

Context for understanding

Narratives

Desire to explore

How do we serve both agents and humans, without losing quality for either?

Agents

Tools

Our Journey

01 The first MCP tooling

02 A purpose-built agentic content source

03 How we tested improvement

Existing capabilities as MCP

`search_docs`

`read_docs`

search_docs

query: Apollo Router

```
[{
  page: “react/get-started”
  snippet: “ ... use read_docs(react/get-started)”
}, {
  page: “react/data/queries”
  snippet: “ ... ”
}]
```

read_docs

```
slug: react/get-started
section: 0
```

```
{
  slug: “react/get-started”,
  content: “ ... read_docs(graphos/routing, 1)”
  section: 0,
  total_sections: 10,
}
```

search_docs: 500 tokens

read_docs: 2k tokens

read_docs(10): 20k tokens

search_docs: 500 tokens

read_docs: 2k tokens

25k Tokens.

One answer.
Wasted calls.
Wasted tokens.
Wasted time.

Context is courtesy.

Needed a new tool with

- Low token usage
- Fast responses
- Accurate answers
- Single tool call for answers

This time, start with the interface

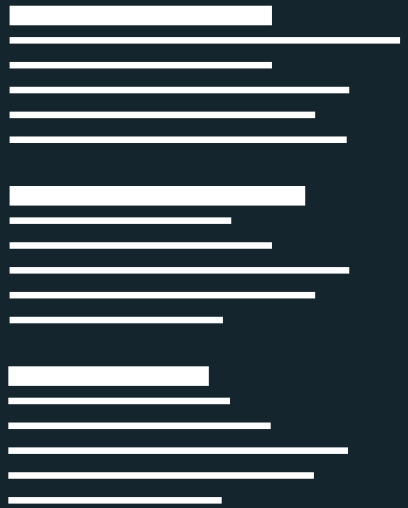
```
research_docs
questions: string[]
context: string
```

```
[{
  page: “graphos/routing”
  chunk: “Exact section of documentation”
}, ...]
```

Content written
for *people*,
not for *machines*

Building the research pipeline

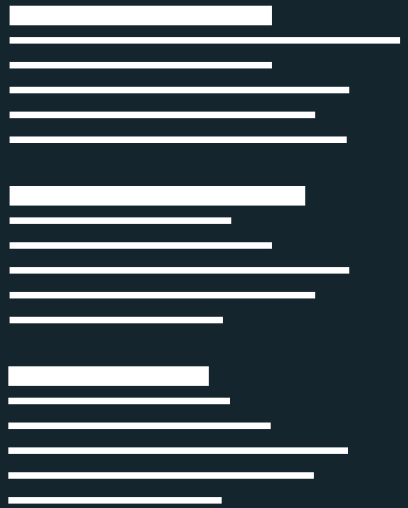
Chunk | Store | Retrieve | MCP



Where to define chunk boundaries?

- ➔ Headers
- ➔ Word count
- ➔ Token/character count
- ➔ Code blocks

Chunk | Store | Retrieve | MCP



Utilize AI to generate intelligent chunks!

- ➔ Headers
- ➔ Word count
- ➔ Token/character count
- ➔ Code blocks

Chunk | **Store** | Retrieve | MCP

How do we store this information semantically as opposed to structurally?

I am getting a cache error in Apollo Client. How can it be fixed?

Incorrect type policies

`possibleTypes` incorrectly configured

Embedding Models

aka. LLM mind reading

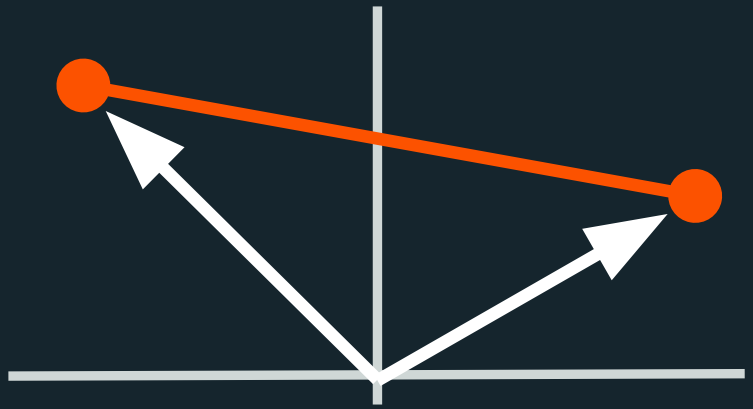
```
I am getting a cache
error in Apollo Client.  ──►  [0.123, … 3071 more]
How can it be fixed?

Incorrect type policies  ──►  [0.456, … 3071 more]
```

Cosine Similarity

I am getting a cache error in Apollo Client. How can it be fixed?

[0.123, ...]

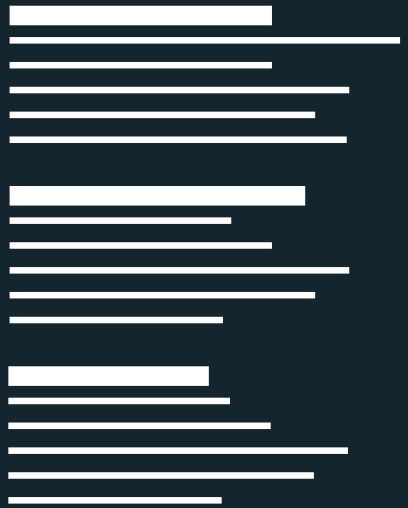


Incorrect type policies

[0.456, ...]

Apollo client and incorrect type policies actually aren't that similar...

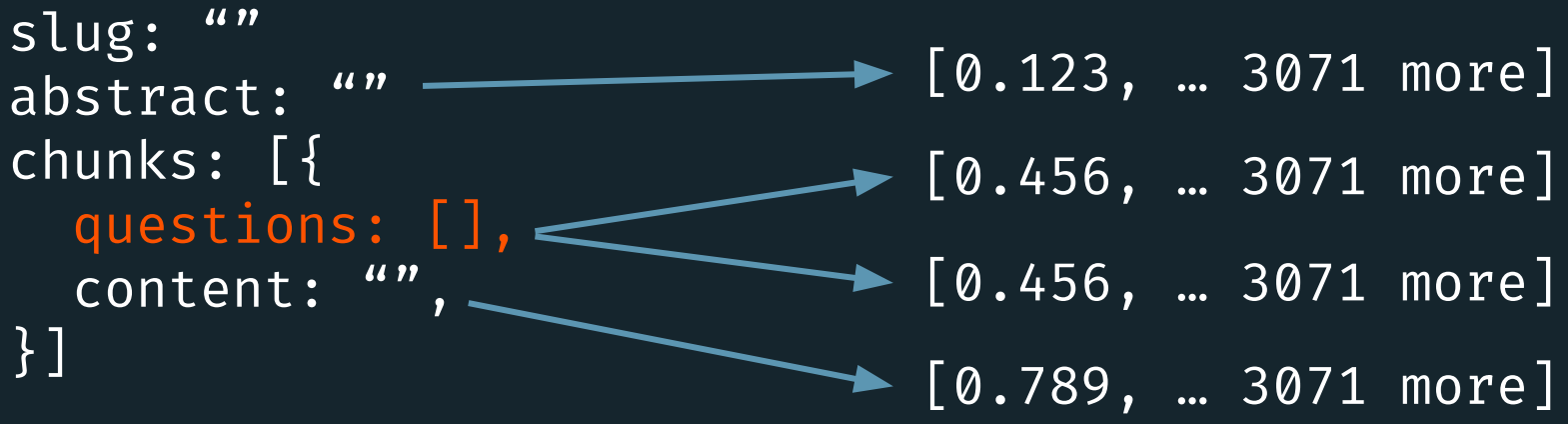
Chunk | Store | Retrieve | MCP



```
{
  abstract: “”
  chunks: [{
    questions: [],
    content: “”,
  }]
}
```

Chunk | **Store** | Retrieve | MCP

Embedding Our Content



Chunk | Store | **Retrieve** | MCP

Content

```
[0.123, …]
[0.456, …]
[0.456, …]
[0.789, …]
```

Input

```
questions: []
context: “”
```

```
[0.123, …]
[0.456, …]
```

Output

```
[{
  slug: “”
  abstract: “”
  chunks: [{
    content: “”,
    score: 0.85
  }]
}]
```

Chunk | Store | Retrieve | MCP

research_docs

```
questions: string[]
context: string
```

```
[{
  slug: ""
  abstract: ""
  chunks: [{
    content: "",
    score: 0.85
  }]
}]
```

How to test in a world of non-determinism and constant change?

MCP Testing 101

- Utilize Agents!
- Tests that model user behavior
- Isolate variables
- And one more thing...

E2E ~~Docs~~ MCP Testing

Executor Agent

Scoring Agent

Shared Sandbox

Test Suite

Initial Prompts

Connected Tools

Static Binary Rubric

Executor Agent

- Built like the user’s environment
  - Same model
  - Same tools
- Turn down the temperature
- Use `seed` if available
- Don’t steer with the prompt

Scoring Agent

No executor knowledge

Read-only access

Pre-written binary rubric

Produces final score

Use REAL Statistics

Imagine This:

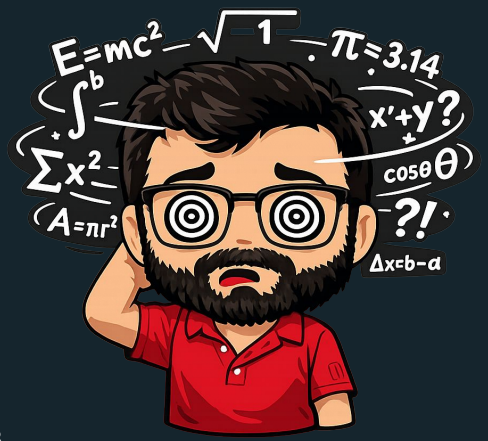
1. Brand new site deployed
2. First user checks out
3. Has cart total 300% avg.

New site increases cart value by **3x... right?**

Use REAL Statistics

AI is expensive so use Fieller's Theorem!

$$(m_L, m_U) = \frac{1}{(1-g)}\left[\frac{a}{b} - \frac{g\nu_{12}}{\nu_{22}} \mp \frac{t_{r,\alpha}\, s}{b}\sqrt{\nu_{11} - 2\frac{a}{b}\nu_{12} + \frac{a^2}{b^2}\nu_{22} - g\left(\nu_{11} - \frac{\nu_{12}^2}{\nu_{22}}\right)}\right]$$



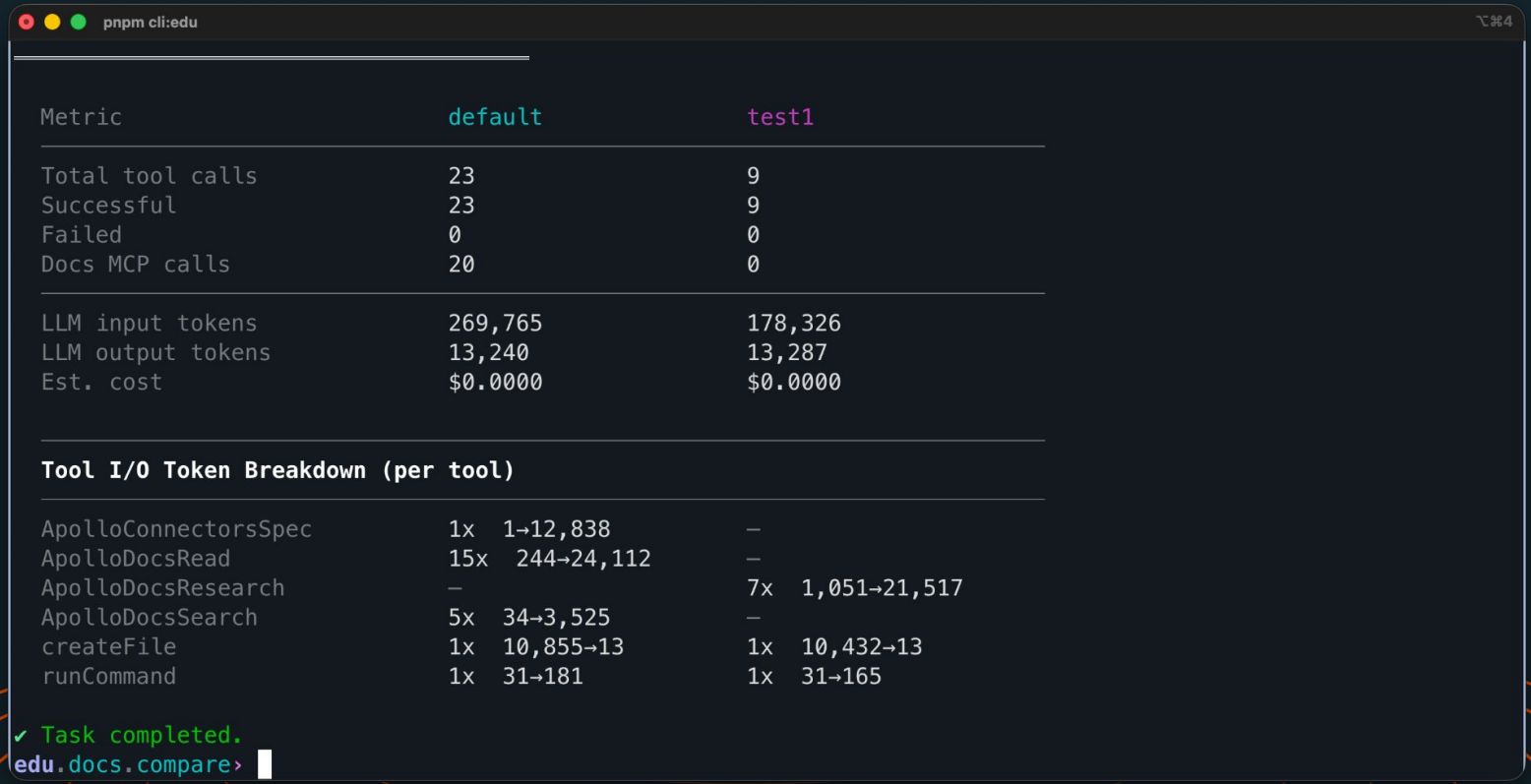
"My MCP server uses 30% less tokens on average, how confident am I with a sample size of 12"

Research vs Search/Read

70% less tool calls

50% less tool tokens

[demo]


pnpm cli:edu
⌥⌘4
Metric default test1
Total tool calls 23 9
Successful 23 9
Failed 0 0
Docs MCP calls 20 0
LLM input tokens 269,765 178,326
LLM output tokens 13,240 13,287
Est. cost $0.0000 $0.0000
Tool I/O Token Breakdown (per tool)
ApolloConnectorsSpec 1x 1→12,838 –
ApolloDocsRead 15x 244→24,112 –
ApolloDocsResearch – 7x 1,051→21,517
ApolloDocsSearch 5x 34→3,525 –
createFile 1x 10,855→13 1x 10,432→13
runCommand 1x 31→181 1x 31→165
✔ Task completed.
edu.docs.compare›

What we learned!

01 Started with existing capabilities, iterated from there

02 Harness math and AI for accurate responses

03 Use statistics for confident E2E MCP testing

Thank you!

daniel-abdelsamed

Booth G3