MCP DEV SUMMIT BENGALURU 2026

From SSE to Streamable HTTP

What actually changed in MCP's transport layer
and why your servers need to catch up.

$WHOAMI

> DevRel Engineer @ Harness, Building DBDevOps

> CNCG Noida Chapter Lead

> Active OSS Contributor

> Tinker >> Builder

AGENDA · 6 ACTS

What We'll Cover

01 **Why I'm talking about this**

How a confusing transport bug sent me down the rabbit hole

02 **Name things correctly**

What “SSE” really means and the raw format

03 **The transport timeline**

stdio → HTTP+SSE → Streamable HTTP

04 **Why the old architecture hurt**

Sticky sessions, made visible

05 **Migrating: before vs after**

A code walkthrough the repo is yours to keep

06 **Gotchas, decisions & what’s next**

Compat traps, real deadlines, the road ahead

It Started With a Very Confusing Bug

> *“The MCP server works perfectly in Claude Desktop. Deploy it behind a load balancer and it **dies silently on every other request**. No errors. No logs. Just… nothing.”*

Three days of debugging later:

- The architecture required sticky sessions : GET /sse and POST /message had to hit the same instance.
- The spec had deprecated this two-endpoint architecture two months earlier. I hadn't noticed.
- Half the SDK versions out there still defaulted to the old pattern, with no warnings.

TERMINOLOGY

Let's Name Things Correctly

SSE wasn't deprecated. The two-endpoint **HTTP+SSE architecture** was. SSE the format is still alive Streamable HTTP uses it for streaming responses today.

HTTP+SSE transport

DEPRECATED

Two endpoints:
GET /sse (persistent)
+ POST /message.
Required sticky sessions.

Streamable HTTP transport

CURRENT SPEC

One endpoint that handles both POST and GET.
Responses can be JSON or a stream.

SSE (the format)

STILL ALIVE

text/event-stream MIME type.
Used inside Streamable HTTP as one of the response modes for streaming.

When I say "SSE → Streamable HTTP," I mean the architecture shift : not that SSE the format went anywhere.

What an SSE Response Actually Looks Like

```
GET /sse HTTP/1.1
Host: api.example.com
Accept: text/event-stream

HTTP/1.1 200 OK
Content-Type: text/event-stream
Cache-Control: no-cache
Connection: keep-alive

event: message
data: {"jsonrpc":"2.0","id":1,"result":{...}}

event: message
data: {"jsonrpc":"2.0","method":".../progress"}

: keep-alive comment
```

text/event-stream

The MIME type that says “this is a stream : keep reading.”

data:

Each line carries one event payload : JSON-RPC rides right inside.

event: / id: / retry:

Optional fields for typing events and resuming after a drop.

Still here today

Streamable HTTP sends exactly this when it chooses to stream.

TRANSPORT EVOLUTION

The MCP Transport Timeline

Three phases : each architectural, not protocol-level.

stdio

Nov 2024

Local only
Process I/O
Claude Desktop

DEPRECATED

HTTP+SSE (2 endpoints)

Nov 2024

Remote servers
GET /sse + POST /message
Sticky sessions required

CURRENT

Streamable HTTP

Mar 2025

Single endpoint
Stateless by default
Uses SSE for streaming

Notice: “HTTP+SSE” had two endpoints; “Streamable HTTP” has one. That's the architectural difference.

THE DUAL-ENDPOINT PROBLEM

Why the Old Architecture Caused Headaches

HTTP+SSE ARCHITECTURE

1. GET /sse → opens persistent connection
2. Server sends endpoint URL via SSE event
3. Client POSTs to /message endpoint
4. Response arrives back over SSE stream
5. Both requests MUST hit the same server

REAL-WORLD PAIN POINTS

Load balancers

Sticky sessions required. Most LBs don't do this by default.

Connection drops

Network hiccup = lost session. No recovery path.

Auth complexity

Two endpoints = two auth surface areas to secure.

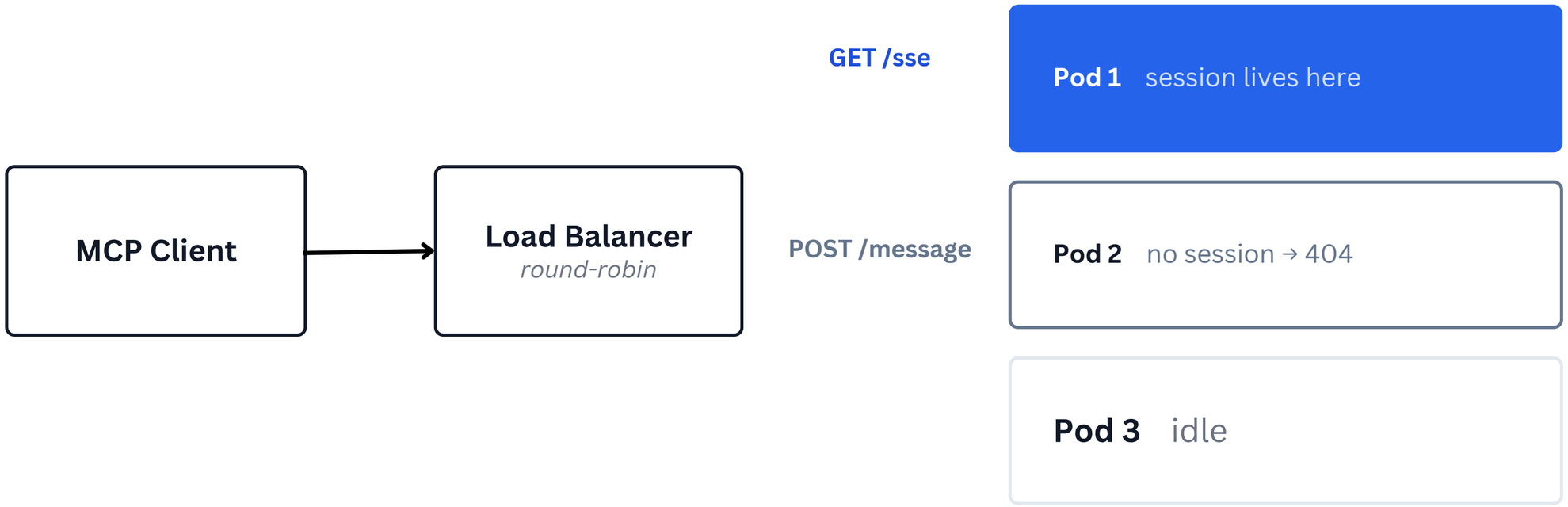
Serverless?

Forget it. Persistent connections and serverless don't mix.

THE FAILURE MODE

Sticky Sessions, Visually

Why GET and POST landing on different pods silently kills the session.



The session lives on Pod 1; Pod 2 has never heard of it.
Old fix: sticky sessions (pin the client to one pod).

MARCH 2025 SPEC

What the March 2025 Spec Actually Changed

BEFORE · HTTP+SSE

- GET /sse : persistent connection
- POST /message : separate endpoint
- Server MUST track session state
- Client waits for SSE endpoint event
- Two auth surfaces to protect
- Sticky sessions for LB required

AFTER · STREAMABLE HTTP

- POST /mcp : single unified endpoint
- Response: JSON body or SSE stream
- Stateless by default (sessions optional)
- Immediate response, no handshake
- One auth surface, standard Bearer
- Works with any standard LB

“JSON body OR SSE stream” , the server picks the response mode per request. The SSE format lives on, inside HTTP.

CODE · BEFORE

The Old Dual-Endpoint Pattern

```
// Old pattern : HTTP + SSE dual-endpoint (deprecated)
const server = new McpServer({ name: 'mcp-demo', version: '1.0.0' });
const transport = new SSEServerTransport('/message', res);

app.get('/sse', async (req, res) => {
  // opens persistent SSE stream : sticky session required!
  await server.connect(transport);
});

app.post('/message', async (req, res) => {
  // completely separate endpoint : must hit same process
  await transport.handlePostMessage(req, res);
});
```

× Two endpoints. Session state. Sticky LB. Breaks on serverless.

CODE · AFTER

The New Streamable HTTP Pattern

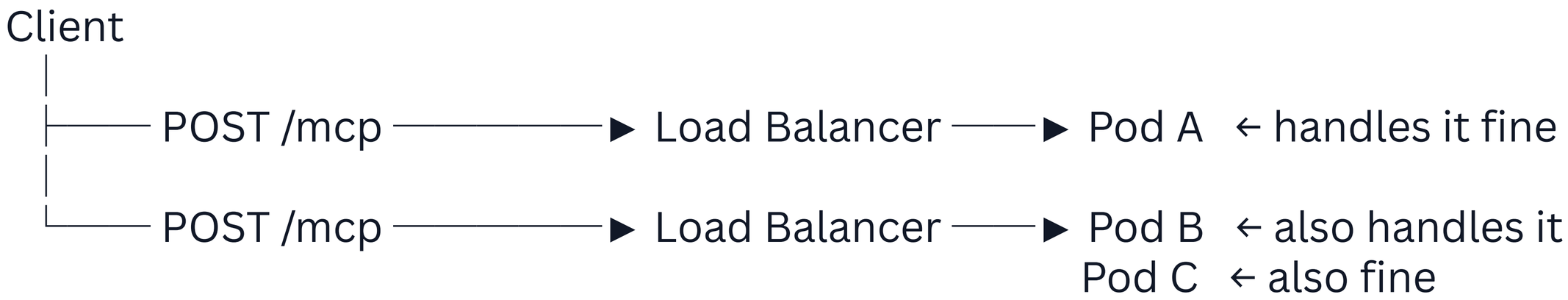
```
// New pattern : Streamable HTTP (March 2025 spec)
const server = new McpServer({ name: 'mcp-demo', version: '1.0.0' });
const sessions = new Map(); // optional : only if you need state

app.post('/mcp', async (req, res) => {
  // single endpoint : server decides JSON vs SSE per request
  const sessionId = req.headers['mcp-session-id'];
  let transport = sessions.get(sessionId)
    ?? new StreamableHTTPServerTransport({
         sessionIdGenerator: () => randomUUID() });
  if (!sessions.has(sessionId)) sessions.set(transport.sessionId, transport);
  await server.connect(transport);
  await transport.handleRequest(req, res, req.body);
});
```

✓ One endpoint. Streams via SSE when needed. Standard auth. Works on any LB.

THE SUCCESS MODE

GO Stateless



BACKWARD COMPATIBILITY

The Gotchas Migration Guides Skip

SDK version mismatch

@modelcontextprotocol/sdk < 1.10 doesn't support Streamable HTTP. Check your lock file, not just package.json.

Clients still hitting GET /sse

Some older clients still initiate via the old dual-endpoint pattern. Add a compat shim or return 405 with a clear error.

Auth header forwarding

Old SSE GET requests often had auth in query params. Streamable HTTP expects standard Authorization: Bearer.

Session cleanup leaks

If you use a sessions Map(), you need an eviction strategy. In-memory state + horizontal scaling = state loss.

DECISION GUIDE

Choosing Your Transport

Is this a local tool (same machine as the client)?

Yes → **Use stdio** local process I/O

No

Do you need real-time streaming responses?

Yes → **Streamable HTTP** streams via SSE format

No

Do you have session-state requirements?

No → **Streamable HTTP** stateless : simplest

Yes

Streamable HTTP + session Map / Redis

Three branches, one destination: Streamable HTTP is the answer for any remote server today.

DEPRECATION DEADLINES

This Isn't Hypothetical

Keboola	**April 1, 2026**	Removing the SSE endpoint
Atlassian Rovo	**June 30, 2026**	mcp.atlassian.com/v1/sse → /v1/mcp
More vendors	**Throughout 2026**	Following the same pattern

If your server still uses the dual-endpoint pattern, your integrations will start breaking in 2026. This is not a “someday” migration anymore.

CASE STUDY

The Migration in Production

Atlassian's Rovo MCP server, a real deprecation already in motion.

THE REAL DRIVER

- **Endpoint move :** mcp.atlassian.com/v1/sse → /v1/mcp
- **Deadline :** SSE kept for backward compat only until 30 Jun 2026
- **Why :** SSE fragile on drops (“no connection for request id”); align with the MCP spec
- **Already shipped :** /v1/mcp is GA : 46+ tools, OAuth 2.1, same access controls

WHAT THE SWITCH TAKES

~1	file changed : the transport swap
2 → 1	endpoints to operate
0	sticky-session LB rules

FUTURE OF MCP TRANSPORTS

What About WebSockets?

Short answer: proposed (SEP-1287), but the official roadmap keeps two transports and evolves Streamable HTTP instead.

THE PROPOSAL · SEP-1287

A WebSocket transport for true full-duplex, session-resilient messaging : attractive for sampling, elicitation, and server push. Still a community proposal, not a standard transport.

THE OFFICIAL CALL · TRANSPORT WG, DEC 2025

Just two official transports remain : stdio (local) and Streamable HTTP (remote). Need something else? Use a Custom Transport. The real work is making Streamable HTTP stateless.

WHERE IT'S ACTUALLY HEADING

Stateless protocol (drop the initialize handshake) · **sessions in the data model** (cookie-like) · **routing via HTTP headers** · **Server Cards** at /.well-known/mcp.json

Per the MCP Transport Working Group roadmap (Dec 2025): SEPs targeted for Q1 2026, next spec release tentatively June 2026.

ACTION ITEMS

What To Do Monday Morning

1

Remember what actually changed

The architecture, not the protocol. SSE format still lives inside Streamable HTTP for streaming.

2

Check your SDK version

npm ls @modelcontextprotocol/sdk : you need ≥ 1.10.0 for Streamable HTTP support.

3

Migrate the transport

SSEServerTransport → StreamableHTTPServerTransport. One-file change in most cases; drop sticky-session LB config.

4

Add a compat shim

Return 405 with a clear message for legacy GET /sse requests : don't just 404 silently.

Thank You

Questions? Find me after the talk, or ping me on LinkedIn.

MCP spec (transports)	modelcontextprotocol.io/specification
Future of transports (WG post)	blog.modelcontextprotocol.io
Harness MCP server	github.com/harness/mcp-server

@sonichigo · Developer Relations Engineer · Harness Inc