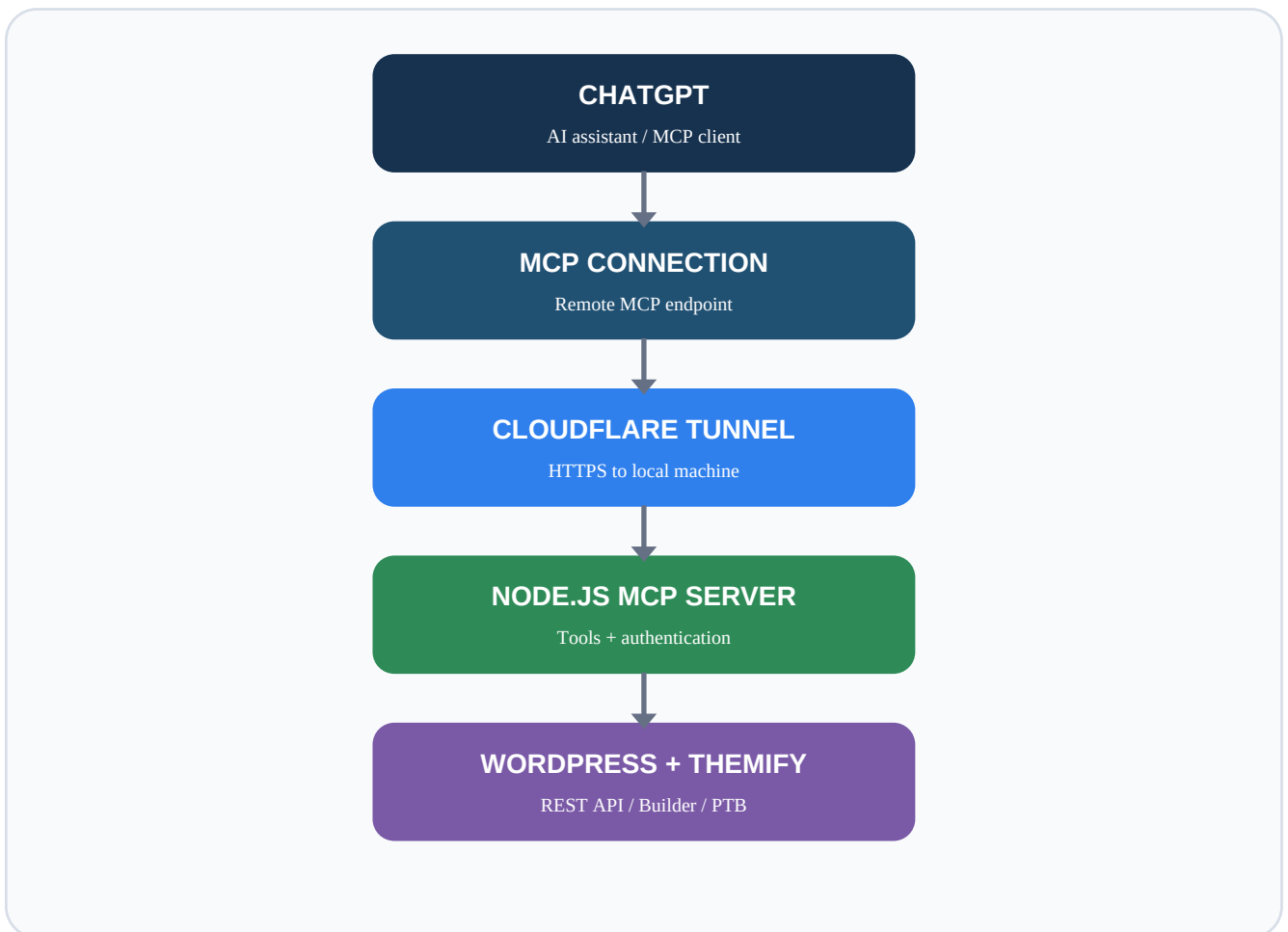


Connect Themify WordPress to ChatGPT Using MCP

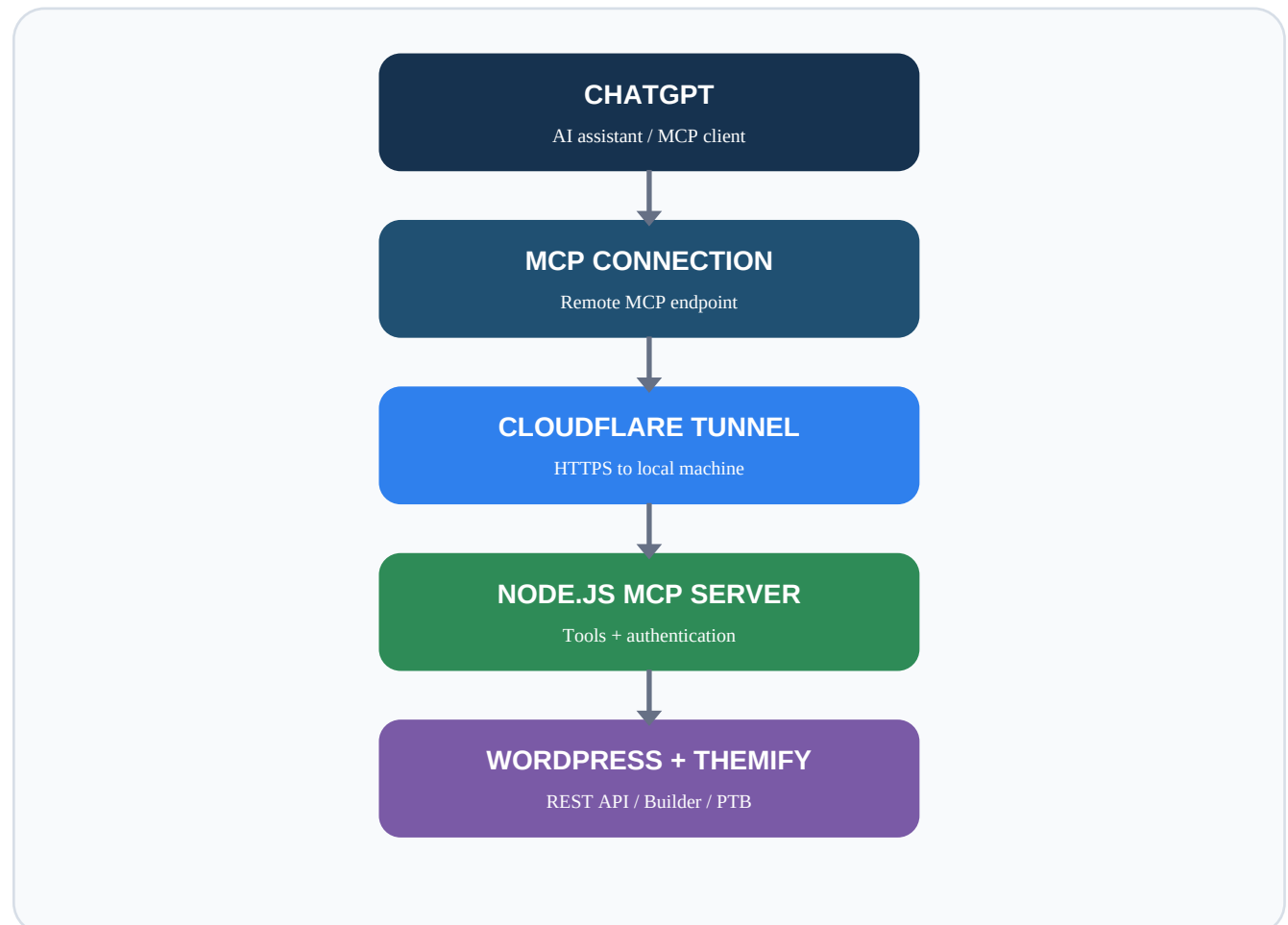
A practical guide with diagrams, setup steps, testing, security, and examples of what an AI assistant can do with Themify.



Goal: Build a read-only bridge from ChatGPT to a Themify WordPress site using an MCP server and HTTPS tunnel. Write operations can be added later.

1. What Are We Building?

MCP (Model Context Protocol) allows an AI application to use approved tools and data from an external system. Here, a Node.js MCP server sits between ChatGPT and WordPress, while a WordPress plugin exposes selected WordPress and Themify data through REST endpoints.



Recommended approach: start read-only. First prove that ChatGPT can safely read site, PTB, and Builder information. Add write tools only after the read-only system is stable.

2. What Can It Do?

Search WordPress content: Find exposed posts, pages, categories, and other content.

Read Themify Builder: Inspect Builder rows, columns, modules, text, images, spacing, colors, and responsive settings.

Read PTB: Retrieve selected Post Type Builder/custom-post data.

Explain layouts: Turn raw Builder data into a readable page structure.

Troubleshoot: Use actual Builder settings to investigate responsive, spacing, visibility, and layout issues.

Later: make changes: Add carefully scoped write tools after authentication and permissions are proven.

Example questions

- What modules are on my Home page?
- Find the section containing "About Us".
- What mobile padding is configured for this section?
- Find all posts in the Featured category.

- Show me the structure of the Home page.

3. Prerequisites

- **LocalWP** with a local WordPress site such as `http://themify.local/`.
- **Themify Builder** and optionally Post Type Builder (PTB).
- **Node.js + npm** for the MCP server.
- **Homebrew** on macOS.
- **cloudflared** for a development HTTPS tunnel.
- **ChatGPT MCP access** for the final registration step.

4. Install the WordPress MCP Bridge

Install the bridge plugin from WordPress Dashboard → Plugins → Add New → Upload Plugin, then activate it.

The bridge exposes selected endpoints:

```
/wp-json/themify-mcp/v1/site  
/wp-json/themify-mcp/v1/posts  
/wp-json/themify-mcp/v1/ptb/{id}  
/wp-json/themify-mcp/v1/builder/{id}
```

5. Create WordPress Authentication

Create a WordPress Application Password for the MCP user. Keep it private. If it is exposed, revoke it and create a new one.

6. Create the MCP Server

A typical project contains:

```
mcp-server/  
■■■ server.js  
■■■ package.json  
■■■ .env  
■■■ .env.example
```

Install dependencies:

```
cd "/path/to/mcp-server"  
npm install
```

Configure .env:

```
WP_URL=http://themify.local  
WP_USERNAME=your-wordpress-username  
WP_APPLICATION_PASSWORD=your-application-password  
MCP_API_KEY=your-mcp-api-key  
PORT=3000
```

7. Start and Test the MCP Server

```
npm start
```

Expected: Themify MCP server listening on port 3000

Test:

```
curl http://localhost:3000/health  
Expected: {"ok":true,"service":"themify-mcp"}
```

8. Test the WordPress Bridge

Example:

```
curl -u themify-mcp http://themify.local/wp-json/themify-mcp/v1/site
```

A successful response should contain site/theme information. Exact fields depend on the bridge implementation.

9. Test Themify Builder Data

Find a WordPress page ID, then call:

```
curl -u themify-mcp "http://themify.local/wp-json/themify-mcp/v1/builder/2319"
```

The response contains the Builder data for that page, including whatever modules, content, layout, and responsive settings the bridge exposes.

10. Install Cloudflare Tunnel

On macOS:

```
brew install cloudflared
```

Verify:

```
cloudflared --version
```

11. Create a Development Tunnel

Keep the MCP server running. In another Terminal tab:

```
cloudflared tunnel --url http://localhost:3000
```

Cloudflare prints a temporary HTTPS address ending in trycloudflare.com.

Important: Quick Tunnels are for development/testing. For production, use a named/stable tunnel and appropriate access controls.

12. Test the Public Tunnel

If Cloudflare gives you <https://example.trycloudflare.com>, test:

```
curl https://example.trycloudflare.com/health
```

Expected: `{"ok":true,"service":"themify-mcp"}`

Your MCP endpoint is:

```
https://example.trycloudflare.com/mcp
```

13. Connect to ChatGPT

If your ChatGPT account/workspace exposes custom MCP/developer functionality, register the remote MCP endpoint using the /mcp URL, not /health.

Example endpoint:

```
https://example.trycloudflare.com/mcp
```

Availability of custom MCP registration and write permissions depends on the current ChatGPT plan/workspace and OpenAI rollout. Check current OpenAI documentation before production use.

14. Recommended MCP Tools

Tool	Purpose
get_site_info	Read WordPress/theme/site information
search_posts	Search exposed WordPress posts
get_ptb_post	Read an exposed PTB/custom-post record
get_themify_builder_data	Read Builder data for a page
find_page	Find a page by title or slug
get_builder_modules	Return a simplified Builder module tree
search_builder_text	Find text inside Builder data
get_responsive_settings	Inspect responsive values

15. What You Can Ask

Site: "What WordPress theme is this site using?"

Pages: "Find the Home page."

Builder: "Show me the structure of the Home page."

Text: "Find the section containing "About Us"."

Responsive: "What mobile padding is configured for this section?"

PTB: "Find all excursions in Goa."

Troubleshooting: "Check Builder settings that could affect this section's width."

16. Read-Only First, Write Later

A safe development path is: Read → Analyze → Controlled write → Automation.

Phase	Focus	Examples
1	Read	Site info, posts, PTB, Builder data
2	Analyze	Module tree, text search, responsive settings
3	Controlled write	Update selected text or spacing
4	Automation	Larger WordPress workflows

17. Security Checklist

- Keep WordPress Application Passwords private.
- Keep MCP API keys private.
- Use read-only access while testing.
- Expose only the WordPress data/tools that are needed.
- Use HTTPS for remote MCP connections.
- Use a stable, secured tunnel for production.
- Add confirmation and permission checks before write operations.
- Review important write actions.

18. Troubleshooting

MCP server won't start: Check Node.js, npm install, .env, and port 3000.

Health endpoint fails: Make sure npm start is still running.

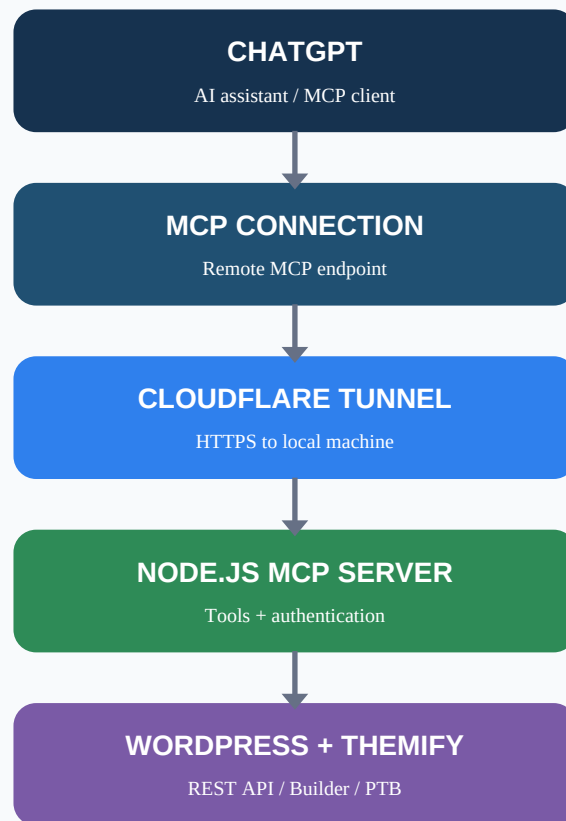
WordPress request fails: Check LocalWP, the site URL, and credentials.

Builder data is empty: Verify the page ID and Builder data.

Cloudflare URL fails: Keep cloudflared running and point it at localhost:3000.

ChatGPT cannot connect: Check the /mcp URL, authentication, and account/workspace MCP support.

19. Complete Architecture



The WordPress bridge controls what leaves WordPress. The Node.js MCP server exposes approved information as AI tools. Cloudflare provides an HTTPS path during development. ChatGPT is the user-facing AI client.

20. Final Result

The result is a foundation for a Themify-aware AI assistant that can inspect WordPress, Builder, and PTB data. Later, carefully controlled editing tools can be added.

References: OpenAI MCP documentation and Cloudflare Tunnel documentation should be checked for the latest capabilities and production requirements.