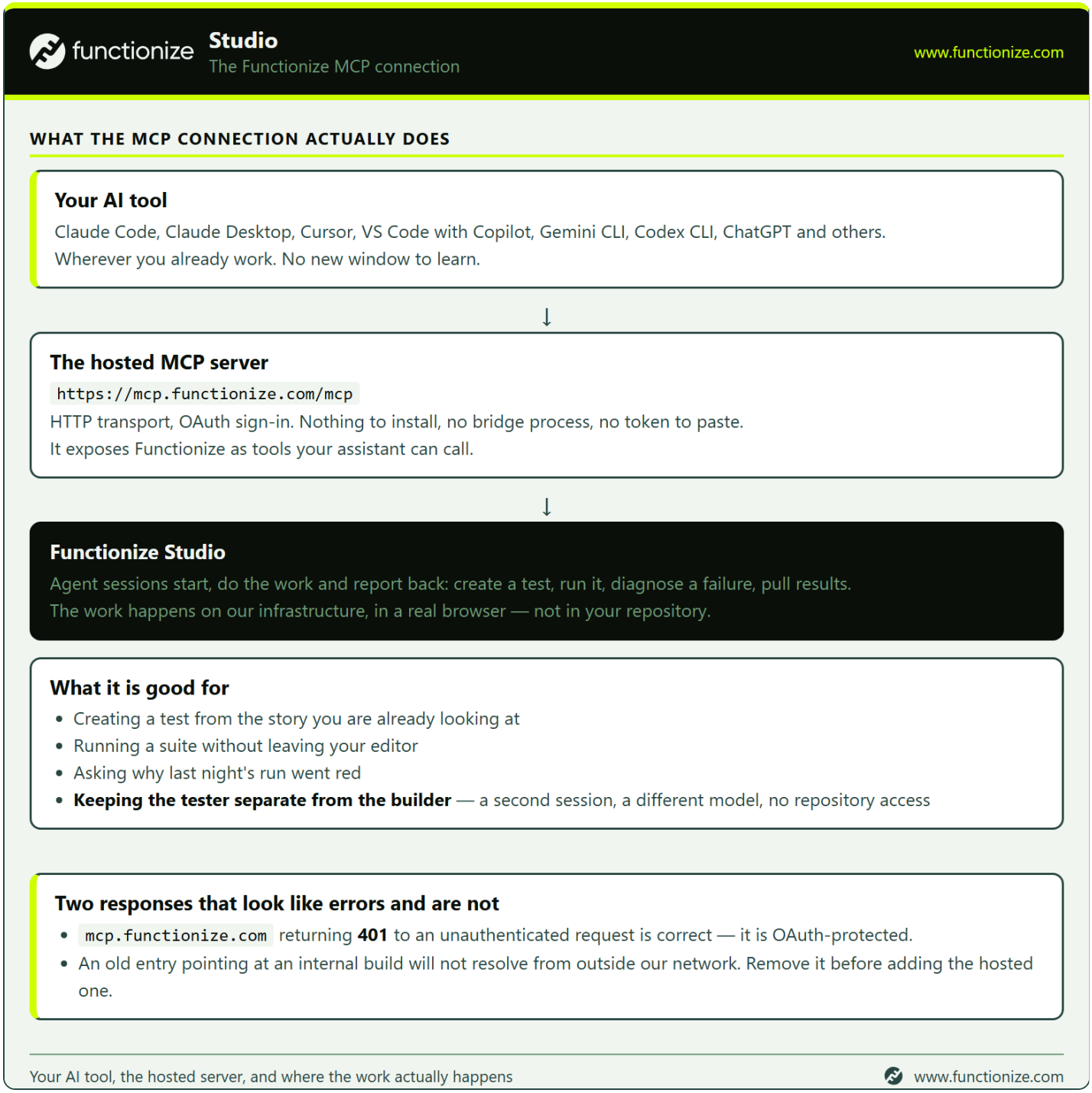
Getting Started with the Functionize MCP Server

Model Context Protocol (MCP) · https://docs.functionize.com/hc/en-us/articles/43753376165655-Getting-Started-with-the-Functionize-MCP-Server

The Functionize MCP server lets an AI client — Claude Code, Claude Desktop, Gemini CLI, or anything else that speaks the Model Context Protocol — work directly with your Functionize agent sessions. You can start a session, send it messages, stream its events, attach files and images, and stop it again, all without leaving the client you already work in.



Where the server lives

The server is hosted by us at `https://mcp.functionize.com/mcp`. It is an always-on HTTPS endpoint, so there is nothing to install and nothing to keep running on your own machine. You sign in with your Functionize account in the browser, and all ten tools are available over the public internet — no VPN, Tailscale or network allow-listing.

What you get

Once connected, ten tools become available to your client:

- `start_agent_session` — begin a new agent session
- `send_agent_message` — send a message into a session
- `get_agent_session` — read a session's current state
- `get_agent_session_events` — read a session's event log
- `stream_agent_session_events` — follow events as they happen
- `list_agent_sessions` — list the sessions on your team
- `list_agent_teams` — list the teams you belong to
- `stop_agent_session` — cancel the turn a session is working on. The session keeps its history and can continue.
- `upload_session_file` — attach a file or image to a session
- `delete_session_file` — remove an uploaded file before it is sent with a message

Note that `list_agent_sessions` is team-wide. On a shared team it returns sessions started by your colleagues as well as your own.

Before you start

Everyone

- A Functionize account, and the ability to sign in to it in a browser.
- An internet connection. The server is a public HTTPS endpoint, so no VPN, Tailscale, or network allow-listing is required.

Signing in

Sign in with the email and password you use for Functionize Studio. If you registered with Google, set a password via **Forgot password** on the Studio login page first. If your company signs in to Functionize through SSO, contact your Functionize representative.

Only if you need the bridge fallback

Claude Desktop connects natively through **Settings** → **Connectors** → **Add custom connector**, which needs nothing installed. **Node.js 18 or newer** is required only if you fall back to the `mcp-remote` bridge — for example on an older build. Claude Code and the other clients never need it.

```
node --version
```

If that reports a version below 18, install Node from nodejs.org, or on macOS `brew install node`.

Choose your guide

Claude Code	The simplest path. Claude Code speaks HTTP MCP and OAuth natively — two commands, no bridge, nothing to copy and paste. Works the same on macOS and Windows.
Claude Desktop	Add it under **Settings** → **Connectors** → **Add custom connector** — nothing to install. A configuration-file fallback using the `mcp-remote` bridge is covered too, for macOS and Windows.

If you use both, you can connect both. They are independent.

A note on teams

By default the connection acts as **your default team**. If you belong to more than one team and want a connection pinned to a specific one, each guide shows how, using the `X-Functionize-Team-Id` header.

Two things to know about pinning:

- **Switching teams inside Studio does not move the MCP connection.** The connection stays on your default team, or on whatever team the header pins it to, until you change the configuration.
- `list_agent_teams` **echoes a pinned team ID back without checking it.** Seeing the ID in its output does not confirm the pin worked. To confirm it, list sessions and check you are seeing the sessions you expect for that team.

To find your team IDs, connect first and then ask your client to list your Functionize teams.

Pick your client

- Connect Claude Code to the Functionize MCP Server — the CLI
- Connect Claude Desktop to the Functionize MCP Server — the desktop app, via a connector
- Connect Cursor to the Functionize MCP Server
- Connect VS Code and GitHub Copilot to the Functionize MCP Server
- Connect Gemini CLI to the Functionize MCP Server
- Connect ChatGPT to the Functionize MCP Server — needs developer mode
- Connect Devin Desktop, Cline, Zed, JetBrains and Goose to the Functionize MCP Server — five more, in one place
- Connect Any MCP Client to Functionize — anything else — the reference
- Troubleshooting the Functionize MCP Connection — when it does not work