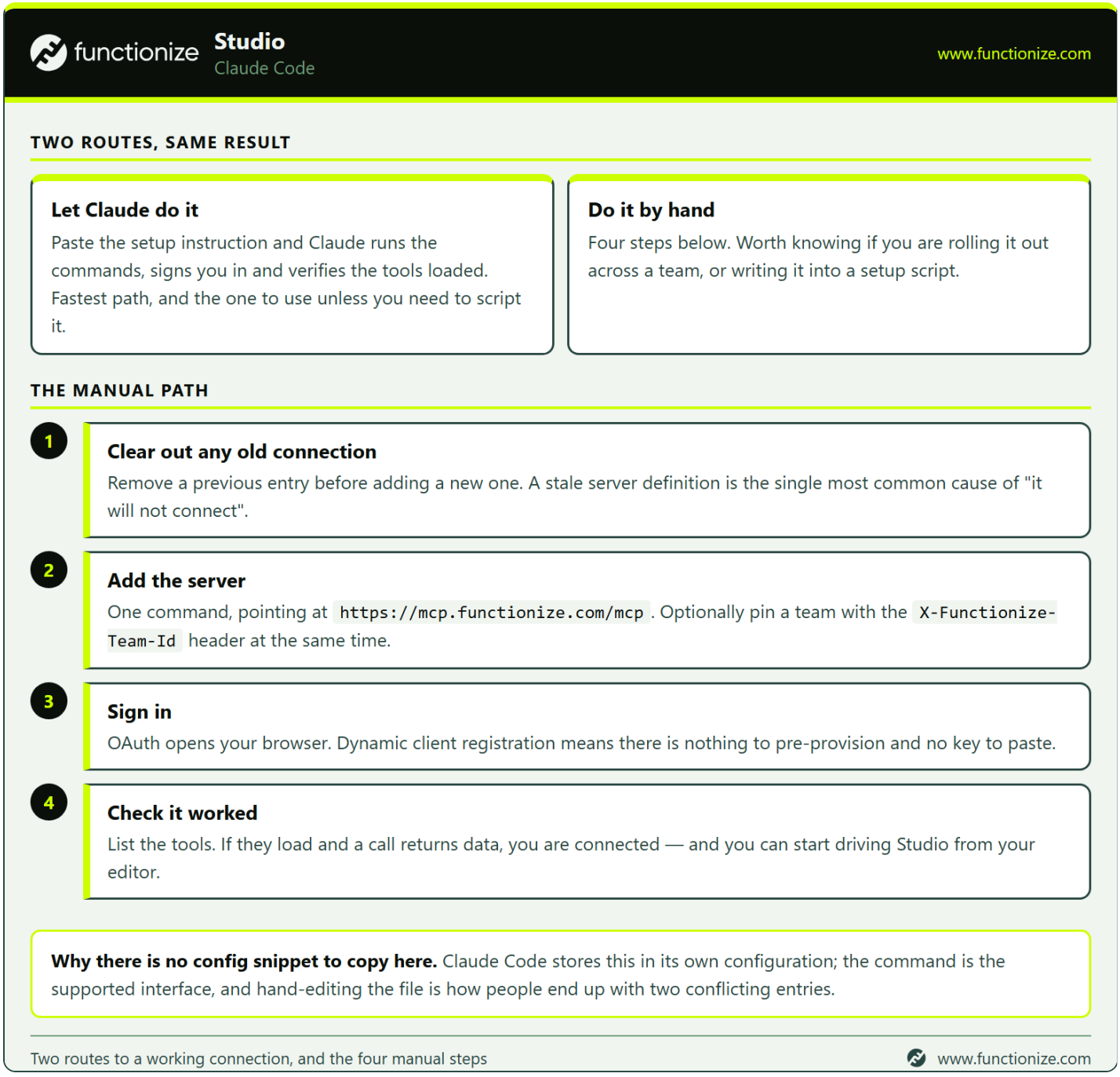
Connect Claude Code to the Functionize MCP Server

Model Context Protocol (MCP) · https://docs.functionize.com/hc/en-us/articles/43753423903511-Connect-Claude-Code-to-the-Functionize-MCP-Server

Claude Code speaks HTTP MCP and OAuth natively, so connecting it to Functionize takes one command and a sign-in. There is no bridge to install and nothing to copy and paste back into the terminal.



The recommended route is our Claude Code plugin. It installs the MCP connection for you and brings our Functionize skills with it, which the connection on its own does not. It lives at FunctionizeInc/functionize-mcp-plugin. Follow the instructions there, then skip to Check it worked.

If you would rather add the server by hand, the rest of this page covers that. One thing to know first: **a hand-added entry pointing at the same URL hides the plugin's entry**, so do one or the other rather than both.

The fastest way: let Claude do it

Claude Code can set this up itself. Open Claude Code in any project and paste the prompt below exactly as it is. It will run the commands and tell you when to sign in.

```
Connect me to the hosted Functionize MCP server.

1. Run: claude mcp list
   Remove any entry whose URL contains internal-dev.functionize.com, and any
   other Functionize-named entry I no longer recognize.

2. Run: claude mcp add --scope user --transport http functionize-hosted \
          https://mcp.functionize.com/mcp

3. Confirm it was added with: claude mcp list

4. Then STOP and tell me to run /mcp myself, select functionize-hosted, and
   authenticate in the browser.

Run the commands, do not just print them.
```

When it tells you to, run `/mcp`, pick `functionize-hosted`, and sign in. Then skip to Check it worked.

If you would rather do it by hand, or the prompt does not work in your setup, the manual steps follow.

Manual setup

Step 1 — Clear out any old connection

If you connected to an earlier internal build of this server, remove that entry first. An entry pointing at `internal-dev.functionize.com` will not work from outside our network, and having both is confusing.

```
claude mcp list
```

Remove anything whose URL contains `internal-dev.functionize.com`, or any Functionize-named entry you no longer recognize:

```
claude mcp remove <name>
```

There is nothing else to clear out. The old `@functionize/mcp` stdio package has been retired; if you still have an entry for it, remove that too.

Step 2 — Add the server

```
claude mcp add --scope user --transport http functionize-hosted \
  https://mcp.functionize.com/mcp
```

`--scope user` **matters.** Without it the entry is added at the default scope, which loads the server in the current project only — so the connection appears to vanish the moment you open Claude Code somewhere else. Add it at user scope and it is available in every project.

That writes an `http` server entry into your Claude Code configuration. If you want to see what it produced, the equivalent JSON under `mcpServers` looks like this:

```
"functionize-hosted": {
  "type": "http",
  "url": "https://mcp.functionize.com/mcp"
}
```

The configuration file is at `~/.claude.json` on macOS and `%USERPROFILE%\.claude.json` on Windows, though you should not normally need to edit it by hand.

Optional: pin a team

If you belong to more than one Functionize team and want this connection to act as a specific one, add the header when you create the entry:

```
claude mcp add --scope user --transport http functionize-hosted \
  https://mcp.functionize.com/mcp \
  --header "X-Functionize-Team-Id: 982"
```

Replace `982` with your team's numeric ID. Omit the header entirely to use **your default team**. Switching teams inside Studio does not move the MCP connection.

`list_agent_teams` echoes a pinned team ID back without checking it, so seeing the ID in its output does not confirm the pin worked. Confirm it by listing sessions and checking you see the sessions you expect for that team.

Step 3 — Sign in

Run:

```
/mcp
```

Select `functionize-hosted` from the list and choose to authenticate. Claude Code then opens your browser to sign in with your Functionize account, captures the response automatically, and stores the tokens for you.

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.

The MCP server has no separate account and no separate password.

The sign-in page is hosted at `functionize.us.auth0.com`. If you are ever prompted for Functionize credentials on a different domain, do not enter them.

You will be prompted to re-authenticate automatically whenever the token expires.

Step 4 — Check it worked

Ask Claude Code:

> list my Functionize agent sessions

It should call `list_agent_sessions` and return your team's sessions. Remember that the list is team-wide, so on a shared team you will see your colleagues' sessions too.

All ten tools should now be available. You can confirm with `/mcp`.

Why there is no copy and paste here

You may have seen MCP connections that ask you to copy an authorization code back into the terminal. That is not needed here, and it is worth knowing why so you do not go looking for it.

Our OAuth broker permits loopback redirect URIs, so Claude Code can start a listener on a local port and have the authorization code redirected straight back to it. The client captures the code itself.

So sign-in needs a browser on the same machine as Claude Code. If you see a prompt to paste an authorization code, something else is handling the connection — our server does not offer a copy-and-paste sign-in flow.

If something goes wrong

See Troubleshooting the Functionize MCP Connection, which covers sign-in problems, stale tokens, and connections that appear but return nothing.

Related

- Getting Started with the Functionize MCP Server — what the server can do, once connected
- Connect Any MCP Client to Functionize — the endpoint, scopes and transport in one table
- Troubleshooting the Functionize MCP Connection — when it does not connect
- Exercise 7 - Drive Studio From Your Own AI Tool — ten minutes, against the demo store

- The Independent Tester Principle: Never Build and Test in the Same Session — why this belongs in a session of its own