Exercise 7 - Drive Studio From Your Own AI Tool

Exercises · https://docs.functionize.com/hc/en-us/articles/43767027489431-Exercise-7-Drive-Studio-From-Your-Own-AI-Tool

Lab handout — Lab 7
A printable worksheet for this exercise: the tasks, the exact prompts, a checkpoint and stretch challenges. Hand it round when you are onboarding a team.
Download the PDF (71 KB)

Everything so far happened in the browser. This exercise connects Functionize to the AI tool you already work in, so you can create and check tests without leaving your editor. About fifteen minutes, most of it waiting for a sign-in.

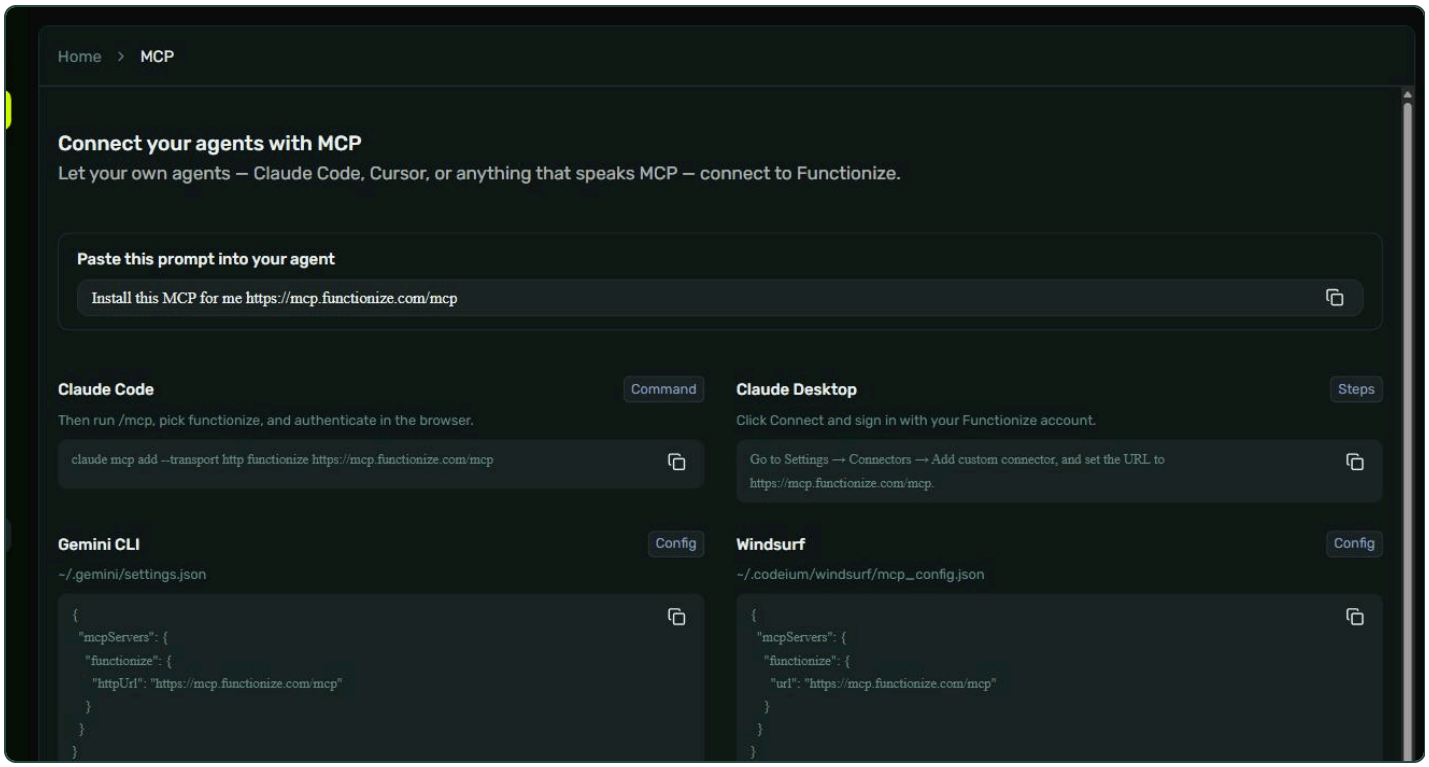
You need a tool that speaks MCP — Claude Code, Claude Desktop, Cursor, VS Code with GitHub Copilot, Gemini CLI, Codex CLI, Windsurf and others all do.

What MCP gives you

MCP is a standard way for an AI tool to reach an external system. Connected, your tool can start Functionize agent sessions, send them instructions and read back what happened — so "write me a test for the checkout flow and tell me if it passes" becomes something you say in the editor where the code lives.

Step 1 — Find the connection details

Studio has the configuration built in. Open **MCP** under **Integrations** in the left sidebar:



The page gives you a copy-paste snippet for each tool, and the endpoint they all share:

```
https://mcp.functionize.com/mcp
```

Step 2 — Let your agent do it

The quickest route is the prompt at the top of that page. Paste it into your AI tool:

```
Install this MCP for me https://mcp.functionize.com/mcp
```

Most tools work out the rest themselves.

If you would rather configure by hand, the same page has the exact file and key names per tool. The key names genuinely differ — Gemini CLI wants `httpUrl`, VS Code needs `"type": "http"`, Goose uses `uri` — so copy from the tool's own block rather than adapting another one. Our per-tool articles cover the details and the traps.

Step 3 — Sign in

The first time your tool connects, it opens a browser to sign in. Use **the same account you use for Functionize Studio** — there is no separate MCP account. If your organization uses SSO, choose the SSO option rather than typing a password.

The sign-in page is at `functionize.us.auth0.com`. If you are ever asked for Functionize credentials on a different domain, stop and tell us.

Step 4 — Check it worked

```
List my Functionize agent sessions.
```

If your team's sessions come back, you are connected. The list is team-wide, so on a shared team you will see colleagues' sessions too — that is expected, not a leak.

Step 5 — Do the exercises again, from your editor

```
Using Functionize, create a test that opens the Electronics > Laptops category on
https://ecom.functionizeapp.com/testsites/anaqa/index.php?route=common/home and verifies it lists
exactly 2 products. Put it in my Studio Exercises project, run it, and tell me whether it passed.
```

Then, after making a change to your own application:

```
Run the Guest smoke orchestration in Functionize and summarize any failures.
```

That is the point of the integration. The tests live next to the work.

If it does not connect

Check the key name matches the file. Copying a snippet from a different tool's documentation is the most common cause of a connection that silently does nothing.

Check whether your organization allowlists MCP servers. On some enterprise plans, once any server is allowlisted every server not on the list is blocked. Ask an admin to allow `https://mcp.functionize.com/mcp`.

Force a fresh sign-in by clearing the tool's cached MCP token. The per-tool articles say where each one keeps it.

See Troubleshooting the Functionize MCP Connection for the rest.

Checkpoint

You are done when your AI tool can list your Functionize sessions, and you have created or run at least one test without opening Studio.

You have finished the series

You can now describe a test and have it built, write assertions that prove something, read a run and its evidence, change a test by asking, diagnose a failure and tell drift from disagreement, group tests into a suite, and drive all of it from your own tooling.

The obvious next step is to point this at your own application. Two things make that go better:

- Set up your environments and target URLs first — ask the agent, it can do it.
- Ask us to build a **skill** for your application, so the agent starts out knowing your domain instead of discovering it. It is complimentary; see Complimentary Skill Building for Your System Under Test.

Where to go next

- Study Kit: Getting a Team Up to Speed in a Week — run all of this with your team
- The Independent Tester Principle: Never Build and Test in the Same Session — why the testing session should not be the building one
- Setting Up an Adversarial Test Loop With Claude and the Functionize MCP Server — the full two-session pattern
- Exercise 8 - Tag a Suite and Give It a Schedule — the last one: build a suite from a tag rather than from memory.