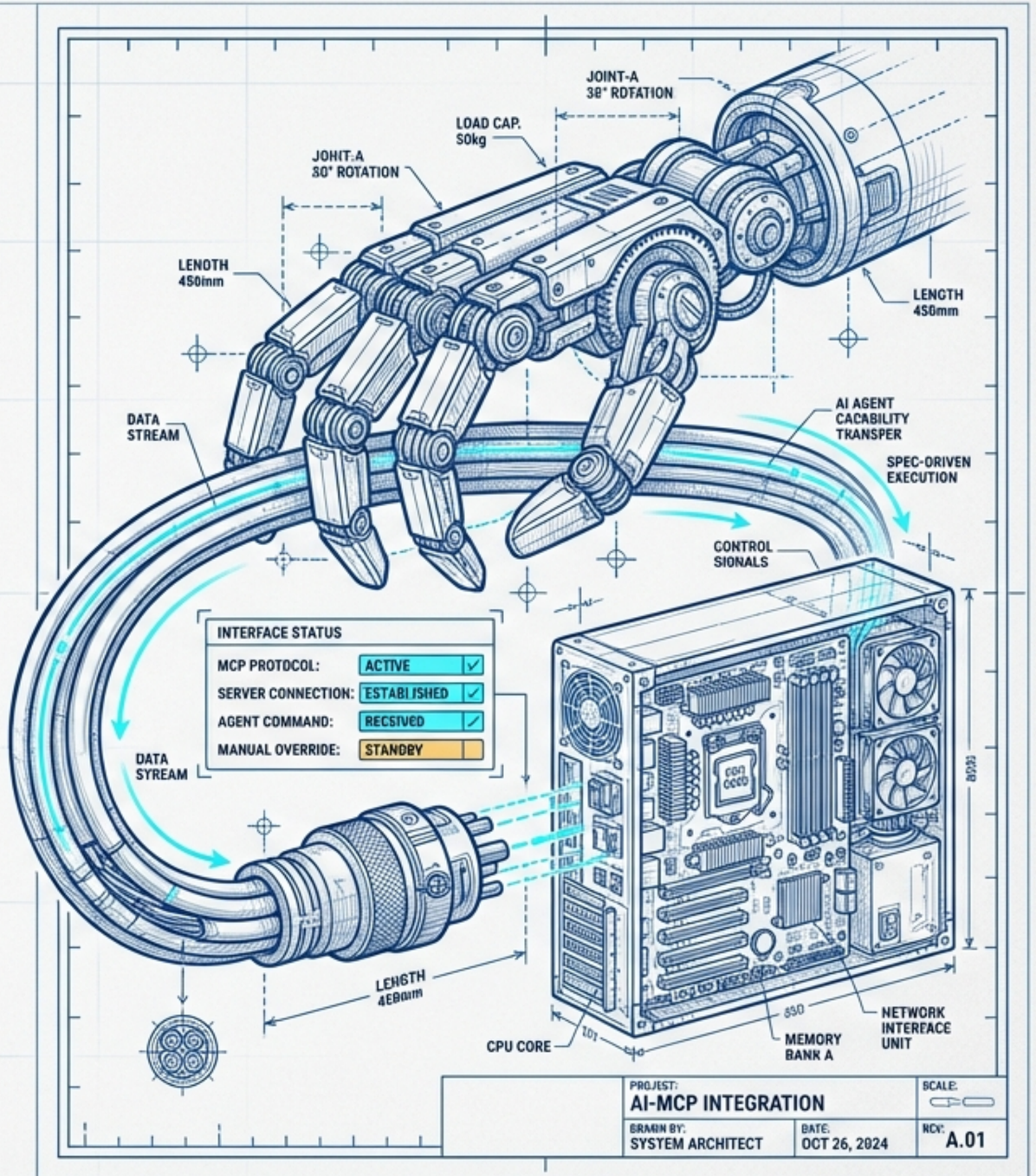
# Extend Your Claw with MCP

Giving your AI Agent hands through spec-driven server development.

**From Consumer to Producer.**



# Knowledge has a ceiling. Ability requires access.

## THE CONSULTANT

**Training Data (Knowledge)**
The agent knows timezone rules, routing algorithms, and API structures. It can discuss any topic, but relies entirely on static memory.

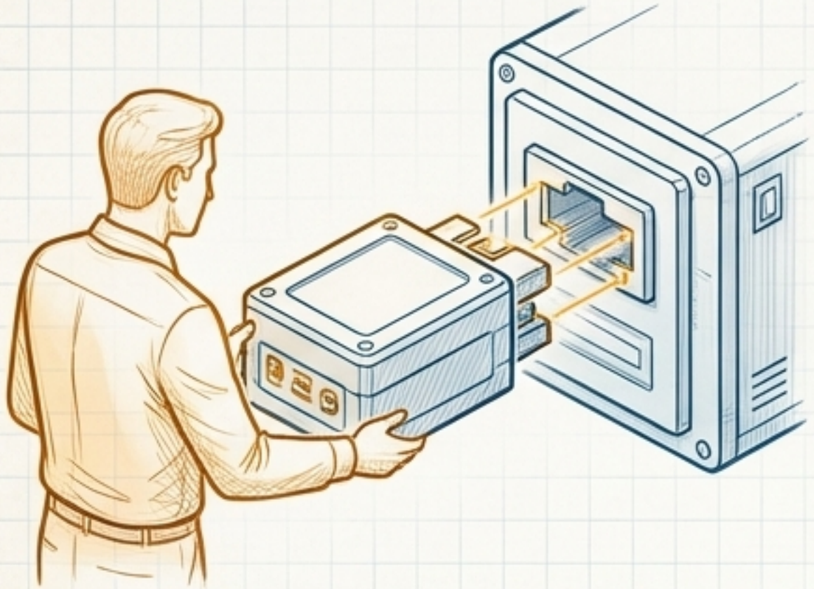
## THE WAREHOUSE WORKER

**Live Tools (Ability)**
The agent checks the exact time *now*, queries the live database, and pings API health endpoints. It acts upon the real world.

An agent with training data can discuss a topic.
An agent with a tool can act on it.

# The wire looks identical. What changes is your responsibility.

## Chapter 56: Consumer



**Key Responsibilities:**

- [x] JSON Configuration
- [x] Gateway Restarts
- [x] Verifying visibility

## The MCP Protocol

MCP does not care which side you sit on. You are moving from plugging in someone else's hands to building your own.

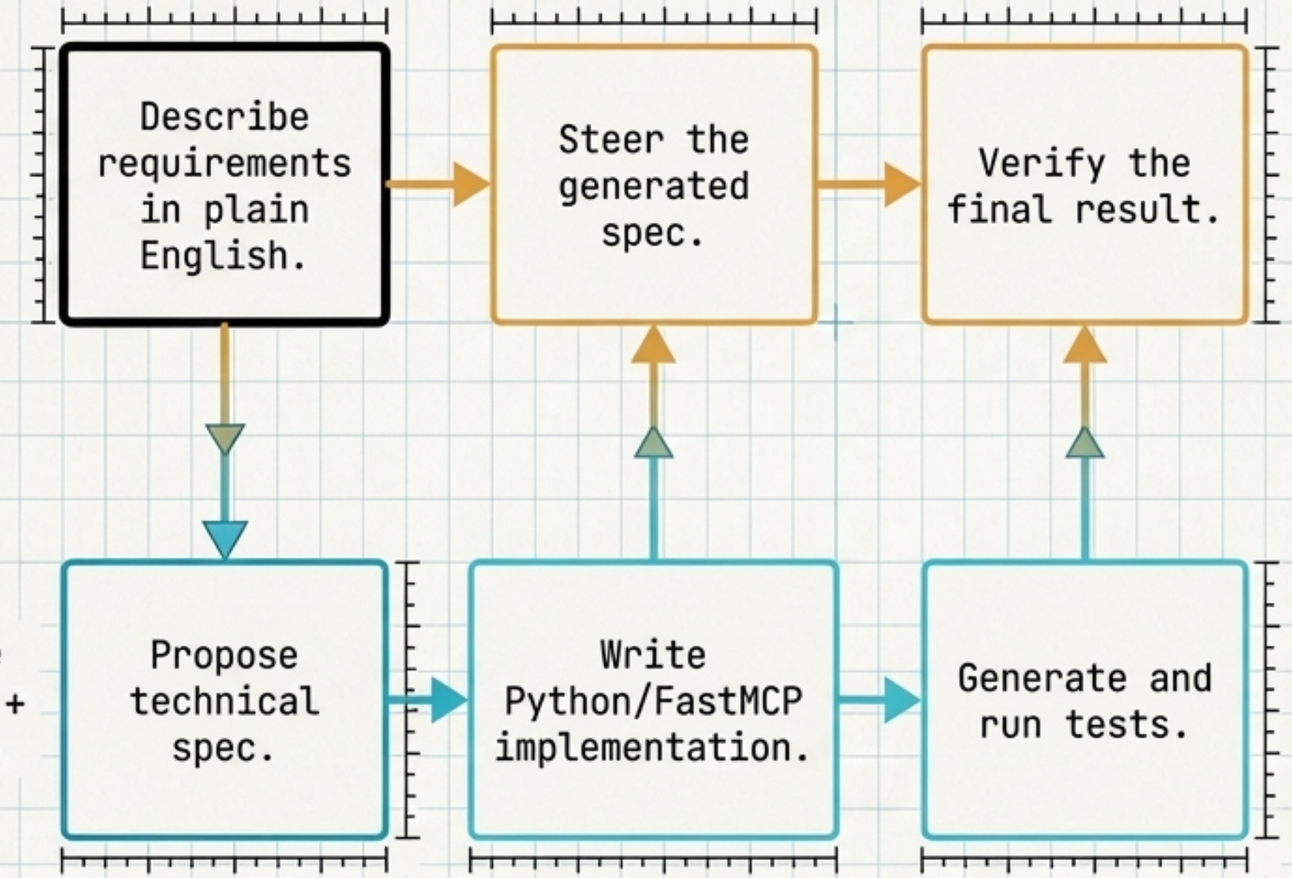
## Chapter 57: Producer

**Key Responsibilities:**

- [x] Designing the spec
- [x] Writing tool descriptions
- [x] Steering the AI builder

# You are the general contractor. Claude Code is the licensed electrician.

## Responsibility Matrix



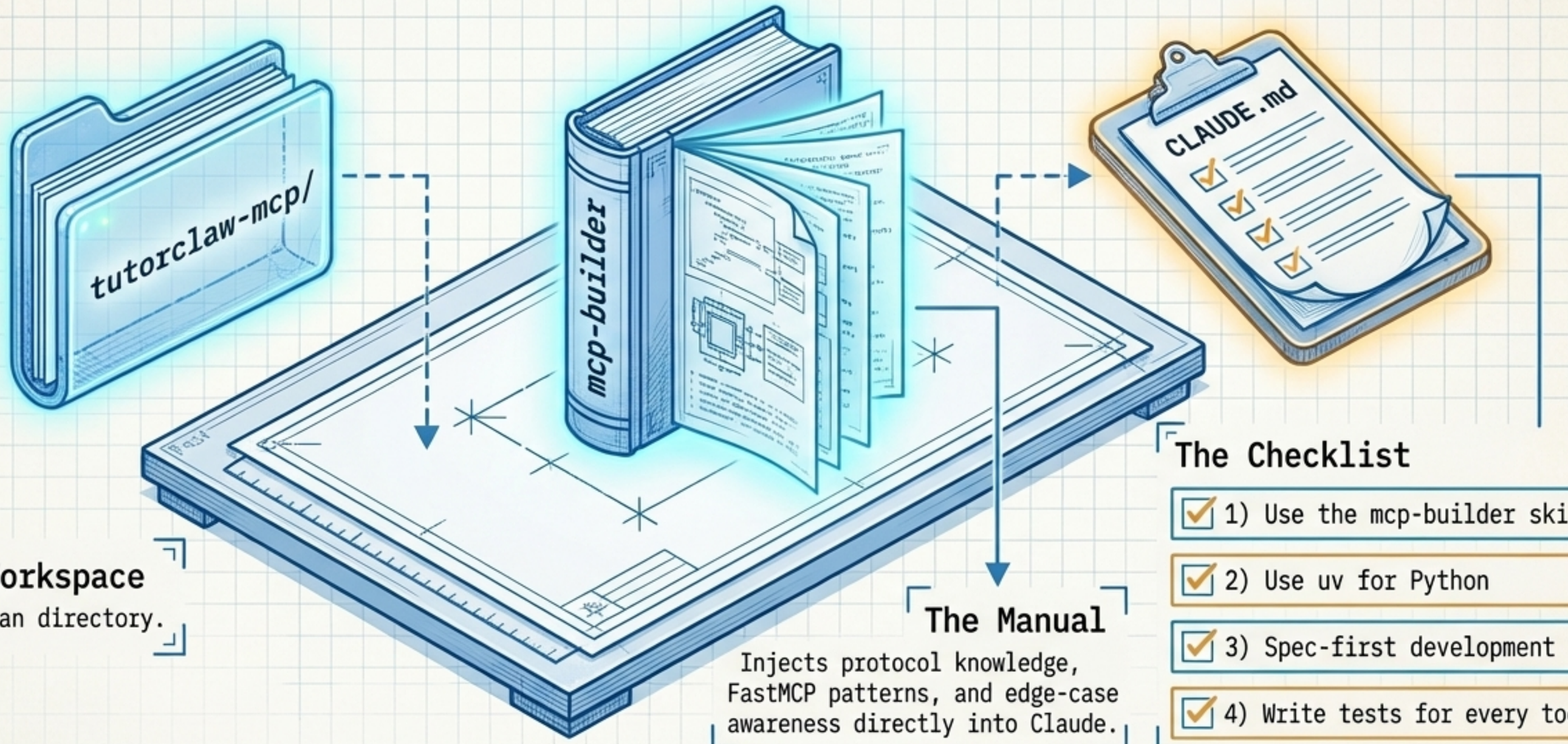

## Insight Bubble

You do not hand-write the server.

A skill gives Claude Code the knowledge to build it.

Your job is design and verification.

# The workshop before the work.




## The Workspace

The clean directory.

## The Manual

Injects protocol knowledge, FastMCP patterns, and edge-case awareness directly into Claude.

## The Checklist

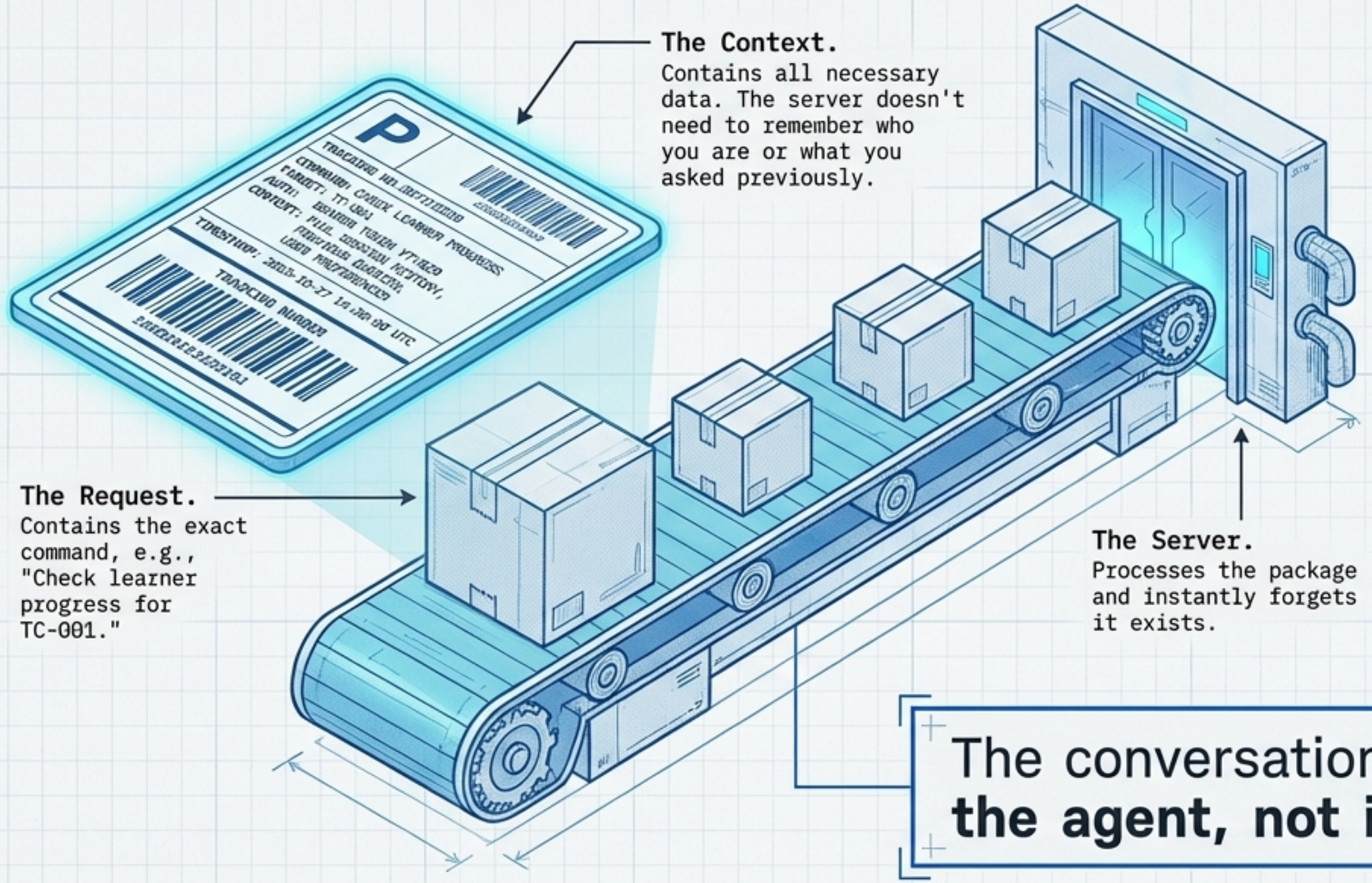
- [x] 1) Use the mcp-builder skill
- [x] 2) Use uv for Python
- [x] 3) Spec-first development
- [x] 4) Write tests for every tool

**Setup is not overhead. Setup is the work.**

# Evaluating your transport options.

| Transport Type | Key Characteristic | Verdict |
| --- | --- | --- |
| SSE | Status: **Deprecated** | Do not use for new projects.  |
| `stdio` | Location: **Local Only** | Great for **same-machine** prototyping, limits external scaling. |
| `Streamable HTTP (Stateful)` | Memory: **Tracks Sessions** | **High complexity.** Prone to session **leaks.** Use only when multi-step memory is strictly required. |
|  `Streamable HTTP (Stateless)` | Memory: **None** | **RECOMMENDED**. Clean, scalable, self-contained.** |

# Why stateless streamable HTTP wins for agents.



**The Context.**
Contains all necessary data. The server doesn't need to remember who you are or what you asked previously.

**The Request.**
Contains the exact command, e.g., "Check learner progress for TC-001."

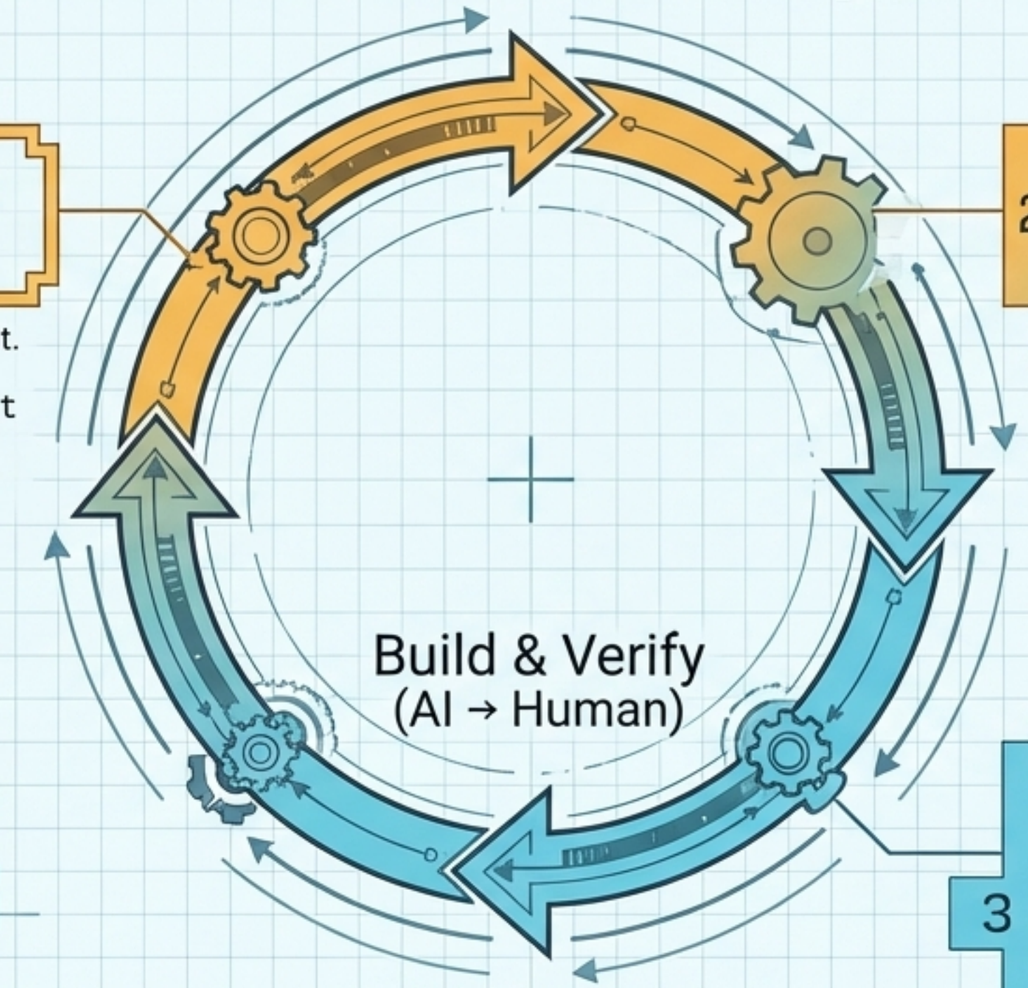
**The Server.**
Processes the package and instantly forgets it exists.

## Key Architecture Benefits

- ☑ No session cleanup timers.
- ☑ No sticky routing.
- ☑ Scales naturally across multiple instances.

The conversation memory **lives in the agent, not in your server.**

# The Describe-Steer-Verify build cycle.



## 1 Describe (Human)

Plain English requirement.

"I need a tool that registers a new learner by name."

## 2 Spec & Steer (AI ↔ Human)

Claude proposes a spec (names, parameters, descriptions). You push back if it's vague. Do not let it write code until the spec is perfect.

## 3 Build & Verify (AI → Human)

Claude implements via FastMCP and writes unit tests. You run `uv run pytest`. If tests fail, loop back to Claude to fix.

Build & Verify (AI → Human)

# The tool description acts as a routing mechanism.

## The Vague Spec

```
{
  "description": "Registers stuff."
}
```

Agent guesses, hallucinates, or **ignores** the tool **entirely** because it lacks selection criteria.

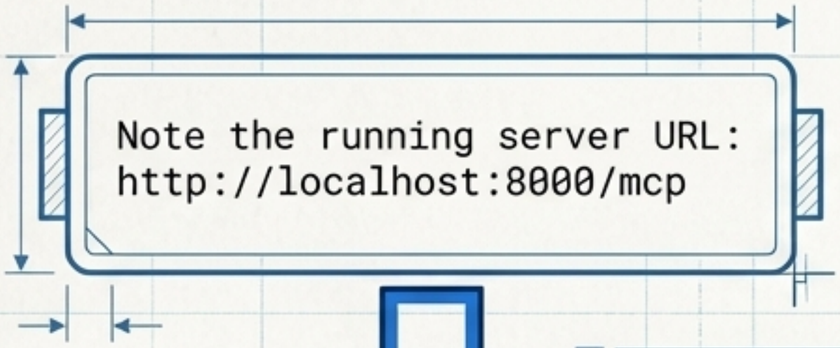
## The Specific Spec

```
{
  "description": "Register a new learner by
  name. Call this when a user wants to sign
   up or create a new learning profile."
}
```

Exact routing. The agent knows precisely when to fire the `register_learner` tool.

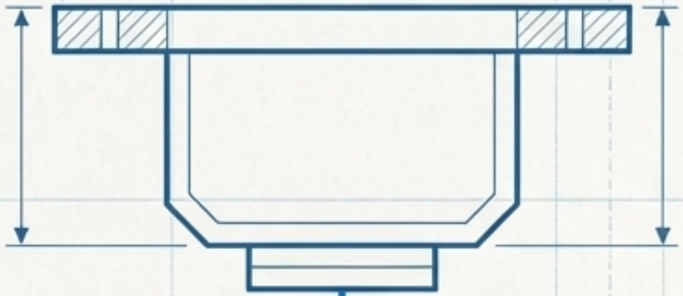
**The description matters more to the agent than the code inside the tool.**

# Wiring the custom server to the agent.




Step 1 - The Plug

```
openclaw mcp set tutorclaw http://localhost:8000/mcp
 --transport streamable-http
```



Step 2 - The Connection

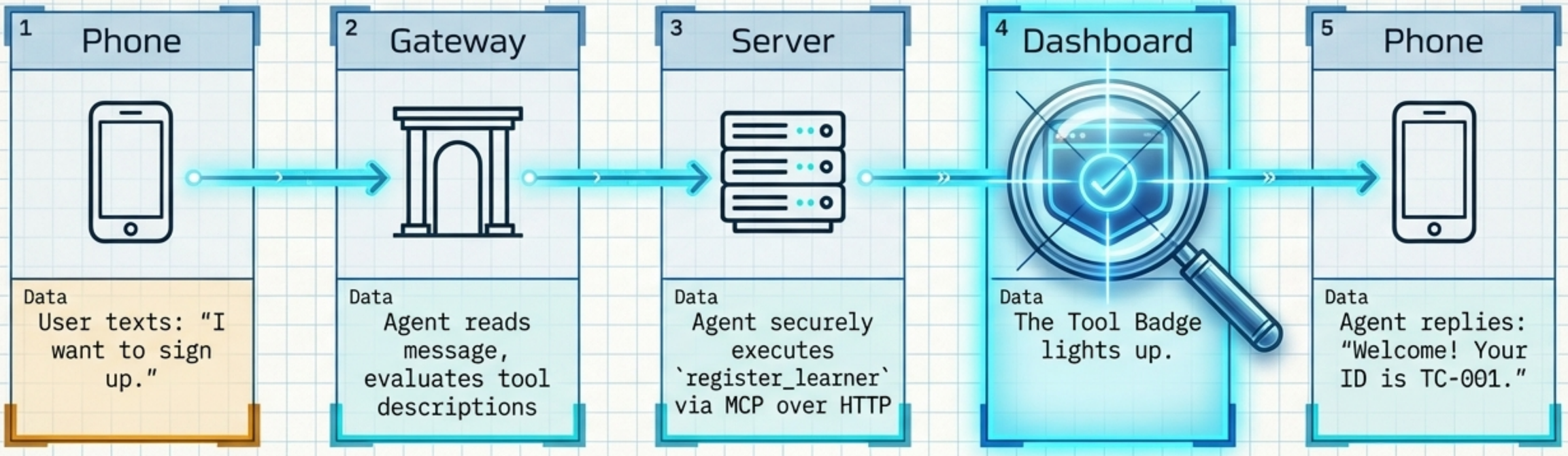
Step 3 - The Activation

```
openclaw gateway restart
```

⚠ **CRUCIAL:** Config is read on startup, not live.

If you skip the gateway **restart,** your agent remains blind to the new tool.

# End-to-End Verification




A text response is not proof. The dashboard tool badge is the absolute source of truth that execution occurred.

# The diagnostic checklist for silent connections.

**Decision 1:**
Is the server process still running in your first terminal?

→ If No → Restart it.

**Decision 2:**
Is the URL a perfect match? (Watch for missing `/mcp` or wrong port).

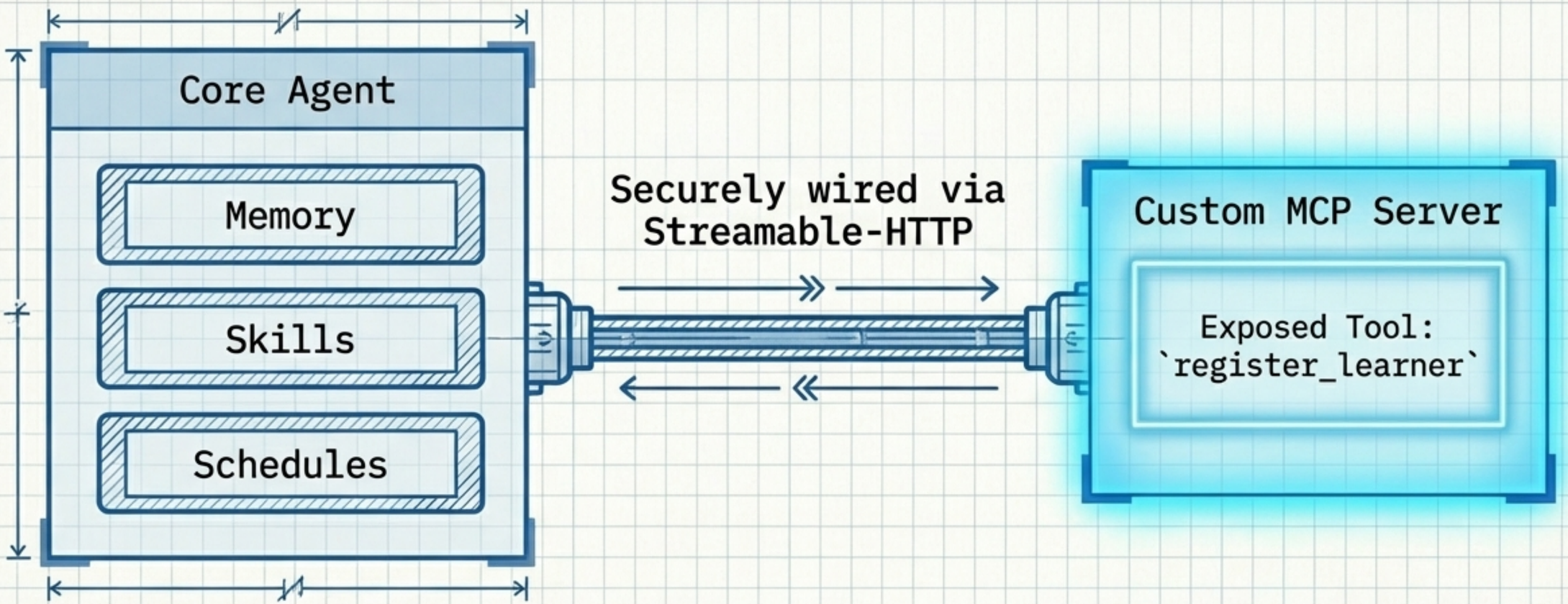
→ If No → Re-run `mcp set`.

**Decision 3:**
Did you restart the gateway after setting the config?

→ If No → Run `gateway restart`.

**Final Check: Check the Gateway Logs. This is the ultimate source of truth for connection refused or timeout errors.**

# The fully equipped AI employee.




You now have an agent that not only knows your business domain, but has the custom-built hands to act on it. Every new tool you need follows this exact cycle.