FREE RESOURCE

MCP Server Vetting in 10 Minutes

Six checks before you add any MCP server to your own setup.

No compliance team, no approval meeting. Just you, a config file, and a server you found twenty minutes ago. Run these six checks in order. Any single **walk away** condition means skip this server or sandbox it hard. Deploying for a team or production? Use the full *MCP Stack Readiness Checklist* instead.

STEP 01 Check who published it ~2 MIN

- The repo, package, and publisher match. The same org name appears on the registry listing, the source repo, and the vendor's official docs.
- You copied the install command from the vendor's own page, not a random blog or a chat answer.
- Stars, forks, and recent commits look like a living project. A commit or release within about 90 days is a good sign.

WALK AWAY IF The package name is one character off a popular server, the repo is an unexplained fork, or the publisher can't be traced to the company whose API it wraps. Typosquatted MCP servers are a real, active attack.

STEP 02 Read the permissions before you paste the config ~2 MIN

- List what the config actually grants: which directories, which API scopes, which token.
- Use a scoped, minimal token created for this server. Do not use the admin key you already had lying around.
- Scope filesystem servers to one project directory. Never your home directory or drive root.

WALK AWAY IF The quickstart tells you to paste a full-access token into a plaintext JSON file and move on. You can do better in ninety extra seconds.

STEP 03 Know what it can do, not just read ~2 MIN

- Skim the tool list: which tools *write, delete, send, or execute*? Those are the ones that can hurt you.
- If the server both reads untrusted content (email, tickets, web pages) and can send or write things, treat that combination as dangerous. That is the prompt-injection exfiltration pattern.
- Keep destructive tools behind your client's per-call confirmation. Do not blanket-approve on day one.

WALK AWAY IF You can't tell from the docs what the write-capable tools touch, or the server wants execute/shell access it can't justify.

STEP 04 Measure the token tax ~1 MIN

- Every enabled server loads its tool definitions into context in *every* session. You pay whether you use it or not.
- Rough guide: a lean server adds about 1 to 5K tokens of definitions. Check yours. Most clients can show connected-tool details.
- Disable tools you will not use. Keep total enabled tools across all servers to a few dozen. Selection accuracy drops as the list grows.

WALK AWAY IF One server injects 20K+ tokens of definitions, or you are about to run 10+ servers "just in case."

STEP 05 Break it once on purpose ~2 MIN

- Make one real call and read the raw result. Is the output sane and reasonably sized?
- Kill the server (or cut the network) mid-session. Does your client tell you clearly, or does the model shrug and claim success?
- Note how re-auth works when a token expires. You will hit this within a month either way.

WALK AWAY IF Failures are silent. A server that fails quietly will eventually "succeed" quietly at something that never happened.

STEP 06 Pin it and write down the exit ~1 MIN

- Pin the version in your config instead of "latest." Otherwise every upstream release ships straight into your setup, unreviewed.
- Save your working config (minus secrets) somewhere you will find it again.
- Know your removal move: delete the config entry, revoke the token. Two minutes now, zero drama later.

VERDICT

- Use it
- Use it sandboxed / read-only
- Skip it

SERVER . . . . . .

VERSION PINNED . . . . . .

TOKEN SCOPE . . . . . .

Server reviews, client limits, and setup guides at mcpverdict.com