A RESOURCE FOR DESIGN SYSTEM AND ENGINEERING TEAMS

Claude Desktop MCP starter config.

Four servers you can defend, and nothing you cannot.

Pinned. Scoped. No secrets in the file.

23	02	05
rules, each with a check	configs: Claude Desktop and Claude Code	server entries launched from the kit's configs

Start small, pin everything, keep secrets out.

Most MCP starter configs have three problems. They run `npx -y` without a version, so every restart can pull new code. They put tokens in the file. And they name packages that no longer exist or no longer get fixes: `@modelcontextprotocol/server-git`, recommended in the essay this guide belongs to, is not on npm at all, because the git reference server is a Python package.

Here are two configs that avoid all three. Claude Desktop gets filesystem, git, read-only GitHub with OAuth, and the team's token server from Field Guide 27. Claude Code already has file and git tools, so its `.mcp.json` carries GitHub, the token server and a fenced browser. Every pin was checked against its registry, and every server that needs no account was launched from these exact files.

`validate-mcp-config.mjs` reads either file and flags missing pins, inline secrets, home-directory access, shell chaining, deprecated packages and plain-http remotes: 9 errors and 6 warnings on the violations example, none on the starters.

Version 1.0 / Sources checked 24 September 2026
Field Guide 25 of the Design × AI series. Pairs with Field Guides 24 (evaluation), 26 (security review) and 27 (integration patterns). Verified 24 September 2026: @modelcontextprotocol/server-filesystem 2026.8.31, mcp-server-git 2026.8.18 (PyPI), ghcr.io/github/github-mcp-server 1.12.2, @playwright/mcp 0.0.82, MCP Inspector 2.8.0.

Practical guidance, not vendor documentation; client behaviour was checked in the docs on the date above. The reference filesystem and git servers are, in their maintainers' words, not production-ready solutions, which is why they are scoped tightly here. Prepared with AI assistance and edited by hand.

START HERE

Know where each file lives, then pick a route.

Two clients, two files, two different rules for secrets. Get this table right and the rest of the guide is details.

	Claude Desktop	Claude Code
File	`claude_desktop_config.json`	`.mcp.json` in the project root (project scope)
Location	macOS `~/Library/Application Support/Claude/`	Committed; local and user scopes live in `~/.claude.json`
Remote servers	Settings, Connectors (not this file)	`"type": "http"` entries in the same file
`${VAR}` expansion	Not documented; reports show the literal text	Documented for command, args, env, url and headers
Secrets	OAuth, or a Desktop extension with keychain storage	Environment variables with names of your own

Sources: S01, S02, S03, S04.

First config on a new machine

Copy the starter, replace the example path, run the validator, then launch each server with the Inspector before restarting the client.

Cleaning up an inherited config

Run the validator with `--registry`. Deprecated servers and unpinned lines show up in seconds; section 03 has replacements.

Sharing a config with the team

Use Claude Code's project `.mcp.json`: committed, variable-expanding, approved per person. Section 02 covers the names.

Adding a server later

Evaluate it with Field Guide 24, add it to Field Guide 26's inventory, then add one line here.

Label	Meaning
`DESKTOP` / `CODE`	Applies to one client only. No label: both.
`CONFIG` / `SCRIPT`	Visible in the file / checked by `validate-mcp-config.mjs` (rule id in the evidence line).
`CLIENT` / `SPEC` / `RESEARCH` / `PRACTICE`	Client docs / MCP specification / published research / a working habit.

SECTION 01

Choose the essential set.

The set is small on purpose. Each server answers a task the team has every week, and each one has a narrowing flag you actually use.

Suggested owners: Design-system lead + engineering lead

- [ ] **Start from what the client already does**

  01.01 CLIENT

  Claude Code reads and edits files and runs git through its own tools, so its config has no filesystem or git server. A plain Claude Desktop chat has neither, so there the two reference servers earn their place.

  Evidence: Claude Code's `.mcp.json` has no filesystem or git entry.

  S02. Field Guide 24, check A.02.

- [ ] **Filesystem: one project folder, pinned**

  01.02 DESKTOP CONFIG SCRIPT

  `@modelcontextprotocol/server-filesystem@2026.8.31` with the design system repository as its only directory. It exposes 14 tools; `write_file`, `edit_file` and `move_file` are annotated destructive, so keep their approval prompts.

  Evidence: Inspector launch from the kit config: 14 tools. Validator rules C04 and C05 pass.

  S01, S06. Kit capture.

- [ ] **Git: the Python server, through uvx, one repository**

  01.03 DESKTOP CONFIG RESEARCH

  `uvx mcp-server-git==2026.8.18 --repository <path>`. There is no `@modelcontextprotocol/server-git` on npm; configs that use it fail at start-up. The server has no push tool, and only `git_reset` is annotated destructive.

  Evidence: PyPI lists 2026.8.18; npm returns 404 for the npm name. Inspector launch: 12 tools.

  S05, S07. Kit capture.

- [ ] **GitHub: GitHub's own server, read-only**

  01.04 CONFIG RESEARCH

  The old reference GitHub server is deprecated on npm and archived. Use GitHub's server: in Desktop, the Docker image `ghcr.io/github/github-mcp-server:1.12.2` with its built-in OAuth login and `GITHUB_READ_ONLY=1`; in Code, the remote endpoint `https://api.githubcopilot.com/mcp/readonly`.

  Evidence: Image tag and digest verified on ghcr.io; not executed, because it needs a GitHub account.

  S05, S08, S09, S14.

- [ ] **Your design system's own server in both clients**

  01.05 CONFIG

  The token server from Field Guide 27 gives agents semantic tokens by name instead of guesses. It is read-only, closed-world and local, so it is the cheapest server in the set to trust.

  Evidence: Inspector launch from both configs: 2 tools.

  Field Guide 27. Field Guide 24 scores it APPROVE.

- [ ] **A browser only where visual checks happen, and fenced**

  01.06 CODE CONFIG SCRIPT

  `@playwright/mcp@0.0.82 --headless --isolated --allowed-origins http://localhost:6006` for checking Storybook. Its own help says the origin list is not a security boundary, and it ships `browser_run_code_unsafe`: keep every navigation and code tool behind a prompt.

  Evidence: Inspector launch from `.mcp.json`: 25 tools. Field Guide 24's scanner: 4 warnings.

  S10. Field Guide 24, check E.04.

- [ ] **Five servers or fewer, and prune monthly**

  01.07 SCRIPT PRACTICE

  Add task-specific servers when the task starts and remove them when it ends. Remove anything unused for a month.

  Evidence: Validator rule C09 warns above five.

  The essay's under-five rule and monthly pruning.

Every server in the file should have a task and a narrowing flag.

SECTION 02

Keep secrets out of the file.

The two clients handle variables differently, so the same habit needs two answers.

Suggested owners: Engineering lead + security partner

- [ ] **Claude Code: reference variables with names of your own**

  02.01 CODE CLIENT SCRIPT

  Write `"Authorization": "Bearer ${ACME_GITHUB_MCP_PAT}"`. Claude Code reads its own and your cloud provider's credential variables, and names such as `NPM_TOKEN`, as empty toward remote servers, so a copied `${ANTHROPIC_API_KEY}` arrives blank.

  Evidence: Validator rules C01 and C03 pass.

  S02.

- [ ] **Claude Desktop: no secrets in the JSON at all**

  02.02 DESKTOP CLIENT SCRIPT

  Variable expansion is not documented for `claude_desktop_config.json`, and reports show `${APPDATA}` reaching the server as literal text. Choose servers that sign in with OAuth, or install them as Desktop extensions, which store fields marked sensitive in the OS keychain.

  Evidence: Validator rule C02 warns on `${VAR}` in a Desktop config; C01 errors on literal secrets.

  S03, S04.

- [ ] **No credentials in URLs or arguments**

  02.03 SCRIPT

  A connection string such as `postgresql://admin:password@host/db` in `args` is a secret in plain sight, and it shows in process lists too.

  Evidence: Validator rule C01 (URL credential).

  Field Guide 24, check D.02.

- [ ] **Docker: pass variables by name**

  02.04 CONFIG

  Use `-e NAME` in `args` and give the value in `env`, or through the OAuth flow. `-e NAME=value` puts the value in the command line.

  Evidence: No `-e NAME=` pairs in `args`.

  GitHub's install guide uses the same pattern. S08.

- [ ] **Commit the project file, never the personal one**

  02.05 CODE PRACTICE

  Commit `.mcp.json`; keep `~/.claude.json`, `.env` files and Desktop configs out of version control. A committed file with only `${VAR}` references is safe to share.

  Evidence: `git ls-files` shows `.mcp.json` and no `.env`.

  S02.

A config file you cannot paste into a ticket is a config file with a secret in it.

SECTION 03

Pin every version, then prove it runs.

A restart should never change what runs. Pin, verify the pins exist, and launch each server once outside the client.

Suggested owners: Engineering lead

- [ ] **Exact versions for npx, uvx and images**

  03.01 CONFIG SCRIPT RESEARCH

  `pkg@1.2.3`, `pkg==1.2.3`, `image:1.2.3` or a digest. A package that impersonated Postmark shipped 15 clean releases before the 16th added a BCC; an unpinned `npx -y` would have pulled it on the next restart.

  Evidence: Validator rule C04 passes.

  S12. Field Guide 24, check B.05.

- [ ] **Every pinned package exists and is current**

  03.02 SCRIPT

  Run the validator with `--registry` to ask npm and PyPI for each pin. It catches typos, missing names and deprecations before the client does.

  Evidence: `--registry` shows each pin resolving, with no DEPRECATED line.

  Kit script.

- [ ] **Replace deprecated reference servers**

  03.03 SCRIPT RESEARCH

  GitHub, GitLab, Postgres, Slack, Puppeteer, Brave Search, Google Maps, Redis, Google Drive and others moved to an archive and carry an npm deprecation. The kit's `known-packages.json` lists them with replacements where one exists.

  Evidence: Validator rule C07 passes.

  S05. npm, 24 September 2026.

- [ ] **Upgrade bridges with known vulnerabilities**

  03.04 SCRIPT RESEARCH

  `mcp-remote` before 0.1.16 lets a malicious remote server run commands on your machine through its authorization endpoint (CVE-2025-6514).

  Evidence: Validator rule C07 (vulnerableBelow).

  S13.

- [ ] **Launch each server once with the Inspector**

  03.05 SCRIPT CLIENT

  `npx @modelcontextprotocol/inspector --cli --config <file> --server <name> --method tools/list` runs the exact entry from your config. A server that fails here would fail silently inside the client.

  Evidence: All five no-account entries in the kit's configs launched and listed their tools.

  S11.

If the Inspector cannot start it, neither can the client.

SECTION 04

Scope it, watch it, review it.

A working config drifts. These rules keep it narrow and make failures visible.

Suggested owners: Engineering lead + design-system lead

- [ ] **Absolute paths to named folders**

  04.01 DESKTOP SCRIPT

  Claude Desktop needs absolute paths. Name the project folder; never `~`, `/Users/you` or `/`. A local server can do anything your account can do inside the paths it is given.

  Evidence: Validator rule C05 passes.

  S01.

- [ ] **No shell wrappers in launch commands**

  04.02 SCRIPT SPEC

  Call the server binary with arguments. `sh -c "npx x && curl ... | sh"` hides a second program behind the one you approved, which is the local-compromise pattern the spec warns about.

  Evidence: Validator rule C06 passes.

  S12.

- [ ] **Local ports bind to 127.0.0.1**

  04.03 SCRIPT SPEC

  The GitHub image's OAuth callback publishes a port; `-p 127.0.0.1:8085:8085` keeps it off the network. Remote URLs use https, except loopback.

  Evidence: Validator rules C08 and C10 pass.

  S15, S08.

- [ ] **Keep approval on writes and on project servers**

  04.04 CLIENT SPEC

  Claude Code asks each person to approve the servers in a project `.mcp.json` before first use, and `claude -p` runs load them without asking. Keep write, navigation and code tools off any allow list.

  Evidence: `claude mcp list` shows project servers approved by name; no allow rule covers destructive tools.

  S02. MCP tools: a human who can deny calls.

- [ ] **Read the logs before you change the config**

  04.05 CLIENT

  Desktop writes `mcp.log` and one `mcp-server-<name>.log` per server. Code shows status and the failure detail in `claude mcp get <name>` and `/mcp`. If a Desktop log shows `spawn ... ENOENT`, give the command's absolute path.

  Evidence: The failing server's log line quoted in the fix.

  S01, S02.

- [ ] **Check the whole config for the trifecta**

  04.06 CODE RESEARCH

  The Claude Code set holds private data (GitHub), untrusted content (GitHub issues, web pages) and a way out (the browser). Field Guide 26's inventory check flags it; the written break is read-only GitHub and a prompt on every navigation.

  Evidence: `check-inventory.mjs` reports the trifecta as accepted with a written break.

  Field Guide 26.

Narrow flags, visible logs, a monthly look. That is the whole maintenance plan.

APPENDIX A

The Claude Desktop config.

Replace `/Users/you/Projects/acme-design-system` with your path. Nothing in this file is secret.

claude_desktop_config.json

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem@2026.8.31",
               "/Users/you/Projects/acme-design-system"]
    },
    "git": {
      "command": "uvx",
      "args": ["mcp-server-git==2026.8.18",
               "--repository", "/Users/you/Projects/acme-design-system"]
    },
    "github": {
      "command": "docker",
      "args": ["run", "-i", "--rm", "-p", "127.0.0.1:8085:8085",
               "-e", "GITHUB_OAUTH_CALLBACK_PORT", "-e", "GITHUB_READ_ONLY",
               "-e", "GITHUB_TOOLSETS", "ghcr.io/github/github-mcp-server:1.12.2"],
      "env": { "GITHUB_OAUTH_CALLBACK_PORT": "8085", "GITHUB_READ_ONLY": "1",
               "GITHUB_TOOLSETS": "repos,issues,pull_requests" }
    },
    "acme-ds-tokens": {
      "command": "node",
      "args": ["/Users/you/Projects/acme-design-system/tools/ds-tokens-mcp/src/server.mjs"]
    }
  }
}
```

Arrays are wrapped for print; the kit file has one value per line.

Entry	Pin verified	Launched with the Inspector	Tools
filesystem	npm 2026.8.31	Yes, from this file	14
git	PyPI 2026.8.18	Yes, from this file	12
github	ghcr.io 1.12.2, digest `sha256:508a0857...`	No: needs a GitHub account	not captured
acme-ds-tokens	Field Guide 27 kit	Yes, from this file	2

Paths were substituted with a scratch repository for the launch. Every other character of the file ran as printed.

APPENDIX B

The Claude Code config, and the validator.

The project file is committed. Each person sets `ACME_GITHUB_MCP_PAT` to a fine-grained, read-only token in their own environment.

.mcp.json

```
{
  "mcpServers": {
    "acme-ds-tokens": { "type": "stdio", "command": "node",
      "args": ["${CLAUDE_PROJECT_DIR:-.}/tools/ds-tokens-mcp/src/server.mjs"] },
    "github": { "type": "http", "url": "https://api.githubcopilot.com/mcp/readonly",
      "headers": { "Authorization": "Bearer ${ACME_GITHUB_MCP_PAT}" } },
    "playwright": { "type": "stdio", "command": "npx",
      "args": ["-y", "@playwright/mcp@0.0.82", "--headless", "--isolated",
               "--allowed-origins", "http://localhost:6006"] }
  }
}
```

Compacted for print. `CLAUDE_PROJECT_DIR` is set in the server's environment, not Claude Code's, so the docs ask for the `:-.` default in a project file. S02.

node scripts/validate-mcp-config.mjs examples/violations.claude_desktop_config.json (excerpt)

```
violations.claude_desktop_config.json (desktop): 7 servers, 9 errors, 6 warnings: FAIL
  error C05 filesystem: Filesystem access to /Users/you. Grant named project folders,
        never a whole home directory or root.
  error C07 git: @modelcontextprotocol/server-git: Not on npm. The git reference
        server is Python: uvx mcp-server-git.
  error C01 github: env.GITHUB_PERSONAL_ACCESS_TOKEN contains what looks like a live
        credential.
  error C01 postgres: args[2] embeds a credential in a URL.
  error C06 notes: Launches through a shell with -c. Call the server binary directly
        so the command you approve is the command that runs.
  error C07 docs: mcp-remote@0.1.15 is below 0.1.16: CVE-2025-6514 (CVSS 9.6): OS
        command injection through a malicious authorization_endpoint, versions 0.0.5
        to 0.1.15.
```

Six of the 15 findings, wrapped for print. The other nine: four unpinned packages, the server count, two more deprecations, a `${VAR}` in a Desktop file and a plain-http URL. The starter configs report 0 errors and 0 warnings; `examples/violations.mcp.json` reports 2 errors and 2 warnings.

KEEP WITH THE CONFIG

Leave a config change record.

One record per change to the file. It makes the monthly prune a five-minute job.

Client and file (Desktop / Code, path)	Server added, changed or removed
Task it serves / owner	Exact version or image digest
Validator run (errors / warnings)	Inspector launch (tool count)
Credential: kind, scope, where it lives	Evaluation record (Field Guide 24)
Inventory entry updated (Field Guide 26)	Remove-by date if unused

> **The file is the inventory's front door.**
>
> A server that is in the file but not in the permissions inventory is a server nobody reviewed. Fix the inventory, or remove the line.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Package versions were read from npm, PyPI and ghcr.io on that date; the launches were run with MCP Inspector 2.8.0 against the kit's own files.

S01 / MCP docs, Connect to local MCP servers
Claude Desktop config location, absolute paths, log files, and the warning that servers run with your permissions.
https://modelcontextprotocol.io/docs/2026-07-28/develop/connect-local-servers

S02 / Claude Code, Connect Claude Code to tools via MCP
Scopes, project approval, ${VAR} expansion, credential variables that read as empty, CLAUDE_PROJECT_DIR.
https://code.claude.com/docs/en/mcp

S03 / Claude Help Center, local MCP servers on Claude Desktop
Desktop extensions; fields marked sensitive are stored in the OS keychain.
https://support.claude.com/en/articles/10949351-getting-started-with-local-mcp-servers-on-claude-desktop

S04 / modelcontextprotocol/servers issue 1039
Claude Desktop passed ${APPDATA} through as literal text (closed as not planned).
https://github.com/modelcontextprotocol/servers/issues/1039

S05 / modelcontextprotocol/servers
Active and archived reference servers; not production-ready solutions.
https://github.com/modelcontextprotocol/servers

S06 / npm, @modelcontextprotocol/server-filesystem
Version 2026.8.31, verified with npm view.
https://www.npmjs.com/package/@modelcontextprotocol/server-filesystem

S07 / PyPI, mcp-server-git
Version 2026.8.18; run with uvx.
https://pypi.org/project/mcp-server-git/

S08 / GitHub MCP server, install guide for Claude
Docker image with built-in OAuth, PAT alternative, Claude Code add-json.
https://github.com/github/github-mcp-server/blob/main/docs/installation-guides/install-claude.md

S09 / GitHub MCP server, remote server
Toolset URLs and /readonly endpoints.
https://github.com/github/github-mcp-server/blob/main/docs/remote-server.md

S10 / Playwright MCP
Version 0.0.82; --headless, --isolated and --allowed-origins, which its help says is not a security boundary.
https://github.com/microsoft/playwright-mcp

S11 / MCP Inspector, CLI client
--config and --server run an entry from a config file; exit codes.
https://modelcontextprotocol.io/docs/2026-07-28/tools/inspector/cli

S12 / MCP, Security Best Practices
Local MCP server compromise: malicious startup commands, sandboxing, showing the exact command.
https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/security_best_practices

S13 / JFrog, CVE-2025-6514 in mcp-remote
Versions 0.0.5 to 0.1.15; fixed in 0.1.16.
https://jfrog.com/blog/2025-6514-critical-mcp-remote-rce-vulnerability/

S14 / Official MCP Registry API, github-mcp-server
io.github.github/github-mcp-server lists ghcr.io/github/github-mcp-server:1.12.2.
https://registry.modelcontextprotocol.io/v0/servers?search=github-mcp-server&version=latest

S15 / MCP specification 2026-07-28, Streamable HTTP
Bind local servers to 127.0.0.1 and validate Origin.
https://modelcontextprotocol.io/specification/2026-07-28/basic/transports/streamable-http

Maintenance: run `node scripts/validate-mcp-config.mjs <file> --registry` monthly, refresh `known-packages.json` when a reference server is archived or a bridge gets an advisory, and re-launch every entry with the Inspector after each version bump. Update the PDF, HTML, Markdown, JSON and both configs together.