MCZ · MCP SERVERS

Git MCP manual

Git MCP 0.2.0 · 15 November 2026 · https://mcz.software/Mcp

Contents

1. What it does
2. Requirements
3. Install and uninstall
4. First use
5. Tools
6. Several sessions at once
7. Settings
8. Safety
9. Licence
10. Troubleshooting
11. Logs

Git MCP 0.2.0 gives Claude the git command line through 46 typed tools. Every tool takes the repository folder, so several Claude sessions can work in different repositories at the same time, and two sessions working in the same repository take turns instead of clashing. New to the MCP servers? Read Getting started first.

What it does

- Status, diff, log, show and blame, with the repository path on every call.
- Stage, commit, restore, reset, move, remove and clean files.
- Branches, switch and checkout; merge, rebase, cherry-pick and revert, with continue, abort and skip.
- Stash, tags, remotes, fetch, pull and push (including force-with-lease), and git config.
- `git_run` for any git command without a dedicated tool.
- Never waits for input: git is run so that it never prompts, opens an editor or a pager. Long output is shortened, with a note saying so.

Requirements

- Windows, 64-bit.
- **git** installed and on the PATH (Git for Windows). Another git can be set with the `GitPath` setting.
- Nothing else: the server is self-contained (no .NET install needed).
- No administrator rights.

Install and uninstall

Run `GitMcpSetup-0.2.0.exe`. It installs for your Windows account only, into `%APPDATA%\GitMcp`:

- `server\GitMcp.Server.exe`, the server,
- `server\appsettings.json`, the settings (kept as you left them when you upgrade),
- `README.md`.

With **Add the git MCP server to Claude Desktop** ticked (the default), setup registers the server as `git`:

```
"git": {
  "command": "C:\\Users\\YourName\\AppData\\Roaming\\GitMcp\\server\\GitMcp.Server.exe",
  "args": ["--stdio"]
}
```

Setup also installs the MCZ MCP Manager if needed. Restart Claude Desktop afterwards (quit it from the notification area first).

Silent install: `GitMcpSetup-0.2.0.exe /VERYSILENT /TASKS="claude"`.

Uninstall from **Settings > Apps**: it removes the `git` entry from Claude's configuration (keeping a backup) and the program files. Licence keys stay in `%LOCALAPPDATA%\MCZ\products`.

To register by hand: `GitMcp.Server.exe --register [--name git]`, and `--unregister [--name git]` to remove it. Both print which configuration files were updated and exit with 0 when everything worked.

For Claude Code:

```
claude mcp add --scope user git -- "%APPDATA%\GitMcp\server\GitMcp.Server.exe" --stdio
```

HTTP instead of stdio

Claude Desktop starts the server over stdio (`--stdio`). Run without `--stdio`, the server is an HTTP MCP server on `http://127.0.0.1:5180/mcp` instead (change the address with `--urls http://127.0.0.1:6000`). Clients that prefer HTTP can use that, for example `claude mcp add --transport http git http://127.0.0.1:5180/mcp`. The HTTP server does not start by itself: run `GitMcp.Server.exe` yourself (or from a scheduled task) and leave it running.

First use

Ask Claude in plain words and name the repository folder:

- "*What is the git status of C:\Projects\MyApp?*"
- "*Show the last 10 commits in C:\Projects\MyApp with their authors.*"
- "*Create a branch fix/login from main in C:\Projects\MyApp, commit the changed files with the message 'Fix login timeout', and push it with upstream.*"
- "*Rebase my branch onto origin/main and tell me if there are conflicts.*"

Paths must be full paths (`C:\...`). The 30-day trial starts when you install the server on a computer.

Tools

46 tools: 44 for git and 2 for the licence.

Group	Tools
Repository and history	`git_init`, `git_clone`, `git_status`, `git_log`, `git_show`, `git_diff`, `git_blame`

Group	Tools
Working tree and commits	`git_add`, `git_restore`, `git_rm`, `git_mv`, `git_commit`, `git_reset`, `git_clean`
Branches	`git_branch_list`, `git_branch_create`, `git_branch_delete`, `git_branch_rename`, `git_checkout`, `git_switch`
Merge and rebase	`git_merge`, `git_rebase`, `git_cherry_pick`, `git_revert`
Stash	`git_stash_push`, `git_stash_pop`, `git_stash_apply`, `git_stash_list`, `git_stash_show`, `git_stash_drop`
Tags	`git_tag_list`, `git_tag_create`, `git_tag_delete`
Remotes	`git_fetch`, `git_pull`, `git_push`, `git_remote_list`, `git_remote_add`, `git_remote_remove`, `git_remote_set_url`
Config	`git_config_get`, `git_config_set`, `git_config_list`
Anything else	`git_run`
Licence	`license_status`, `license_install`

Notes:

- `git_diff` shows unstaged changes, staged changes (`staged`), the working tree against a revision, or two revisions.
- `git_clean` only lists what it would delete unless asked to delete (`dryRun=false`).
- `git_merge`, `git_rebase`, `git_cherry_pick` and `git_revert` take an action: start (the default), continue, abort, skip or quit. Rebases are never interactive.
- `git_push` can force with lease, set the upstream and delete remote branches or tags.
- `git_tag_delete` deletes local tags only.
- `git_run` runs any git arguments (but not options before the git command, such as `-C`); with `readOnly=true` it runs alongside other reads.

Several sessions at once

Every Claude session starts its own copy of the server, and they all coordinate through lock files in `%LOCALAPPDATA%\GitMcp\locks`:

- reads of a repository run in parallel and never take git's own lock,
- writes to the same repository take turns (a write waits up to 10 minutes for the one before it),
- linked worktrees share their main repository's lock.

Settings

Settings live in `server\appsettings.json` (section `GitMcp`), or as environment variables `GitMcp__<Key>`, or on the command line as `--GitMcp:<Key>=<value>`.

Setting	Default	Meaning
`GitPath`	`git`	The git program to run.

Setting	Default	Meaning
`CommandTimeoutSeconds`	300	Time limit for local commands.
`NetworkTimeoutSeconds`	1800	Time limit for clone, fetch, pull, push and writing `git_run` commands.
`LockTimeoutSeconds`	600	How long a write waits for another write to the same repository.
`MaxOutputChars`	200000	Longer output is shortened.
`AllowedRoots`	(empty)	If set, only repositories inside these folders can be used.
`LockDirectory`	`%LOCALAPPDATA%\GitMcp\locks`	Where the lock files go.

A command that runs past its time limit is stopped, with its child processes.

Safety

- Paths must be absolute (and inside `AllowedRoots` when that is set).
- Branch names, refs and URLs that start with `-` are refused, so a value can never be read as an option; file paths are always passed after `--`.
- In HTTP mode the server accepts only requests addressed to this computer (localhost) and refuses web pages from other sites.
- Interactive git (`rebase -i`, `add -p`) is not available. Credentials come from Git Credential Manager as usual; if git would need to ask for a password, the command fails instead of waiting.

Licence

30 days free from the first use on a computer, then a key. `license_status` and `license_install` always work. From a prompt: `GitMcp.Server.exe --license` shows the state (exit code 0 when the tools work, 2 when not); `GitMcp.Server.exe --license MCZ1-...` installs a key (or the path of a file holding it). `--license` must be the first argument. See Getting started and the licence terms.

Troubleshooting

- **"Is git installed and on PATH?"**: install Git for Windows, or set `GitPath` to the full path of `git.exe`.
- **A git command failed**: the reply shows the git command, its exit code and git's own message, as git printed it.
- **"timed out after N s and was stopped"**: raise `CommandTimeoutSeconds` or `NetworkTimeoutSeconds`.
- **A push or fetch hangs or fails on credentials**: sign in once with git in a terminal so Git Credential Manager stores the credentials.
- **The server is not listed in Claude**: see Troubleshooting in Getting started.

Logs

The server writes no log files. In stdio mode its messages go to standard error, which Claude Desktop keeps in its MCP log (`%APPDATA%\Claude\logs\mcp-server-git.log`). In HTTP mode they go to the console window. The installer's log is in `%TEMP%` (`Setup Log ....txt`).