lean-lsp-mcp

Tools for agents to interact with Lean

A toolbox for Agents

LLMs generate code without feedback -> Agents

Users: InfoView, syntax highlighting, explore documentation

lean-lsp-mcp provides agents with the same tools

A toolbox for Agents

Diagnostics & Goals
- diagnostic_messages
- goal
- term_goal

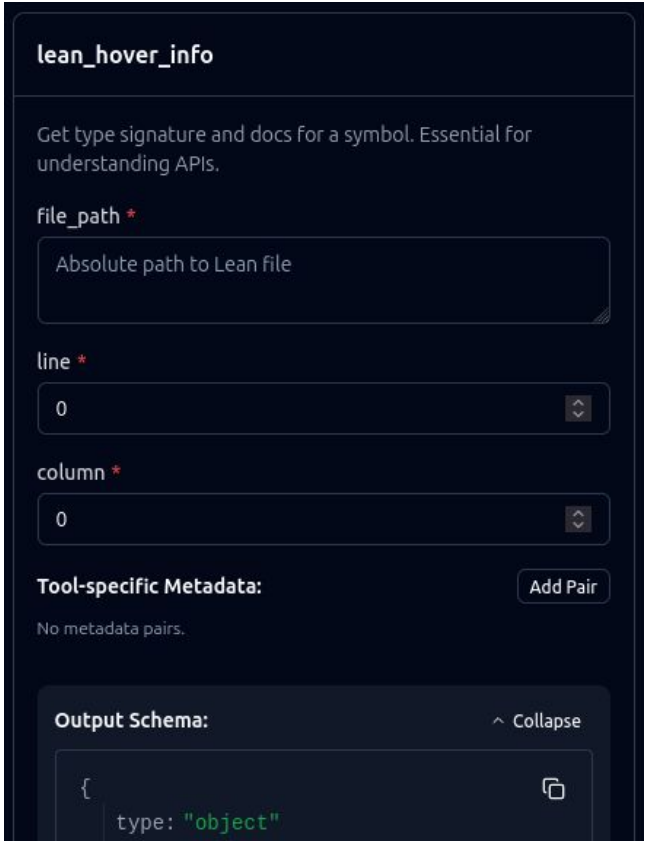
Navigation & Info
- hover_info
- completions
- declaration_file
- file_outline

Code Execution & Verification
- build
- run_code
- verify
- multi_attempt

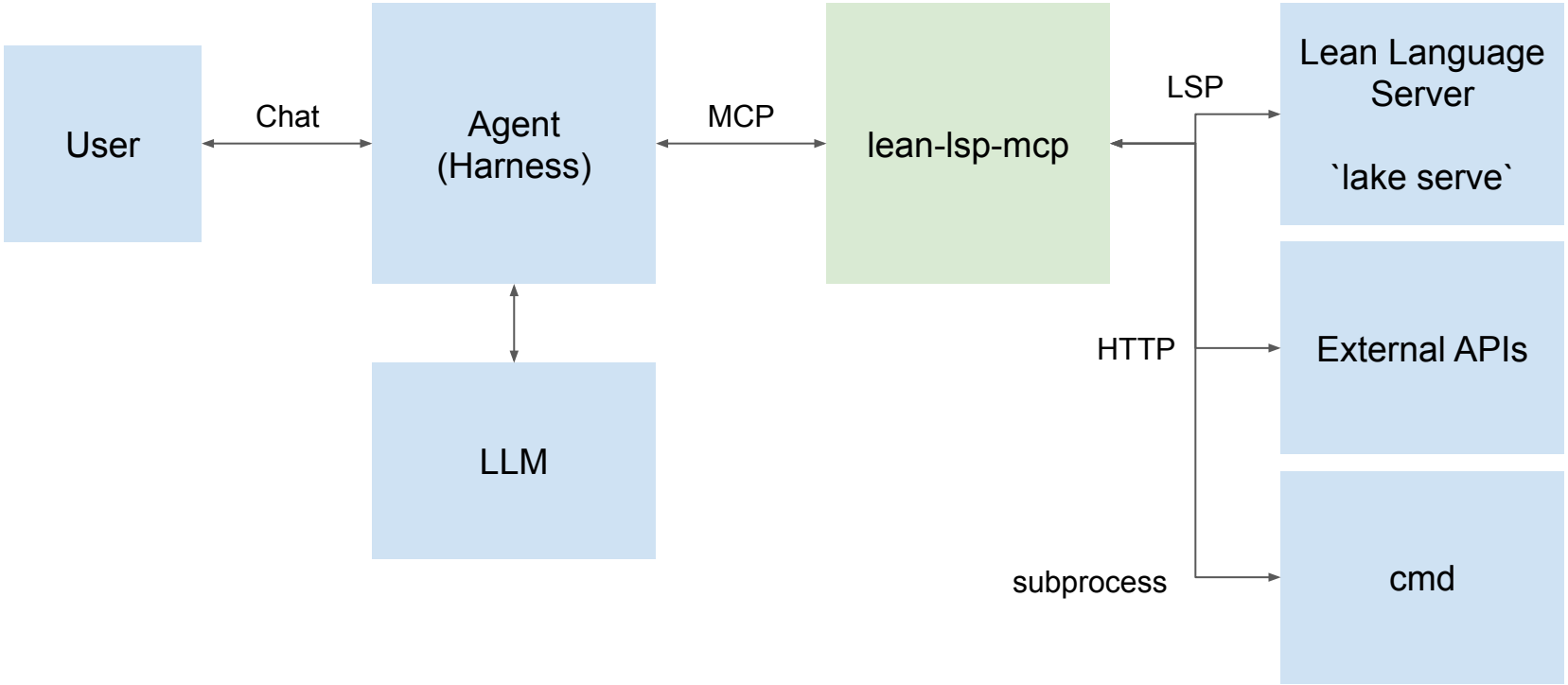
Search
- local_search
- leansearch
- loogle
- leanfinder
- state_search
- hammer_premise

Code Actions & Widgets
- code_actions
- get_widgets
- get_widget_source

Profiling
- profile_proof

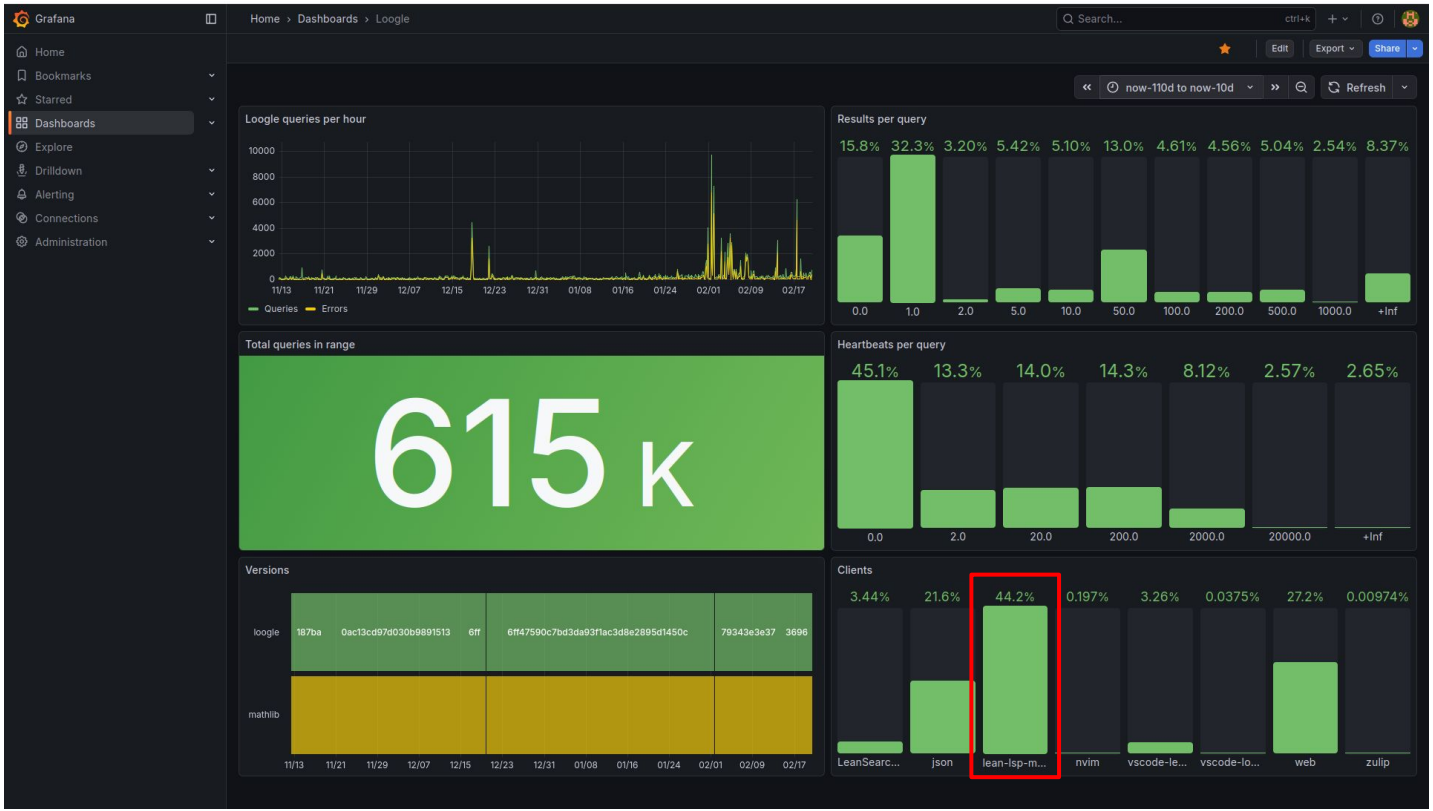


Typical Flow



MCP: Model Context Protocol https://modelcontextprotocol.io
LSP: Language Server Protocol https://microsoft.github.io/language-server-protocol/

Usage Statistics: Loogle



Last 100 days

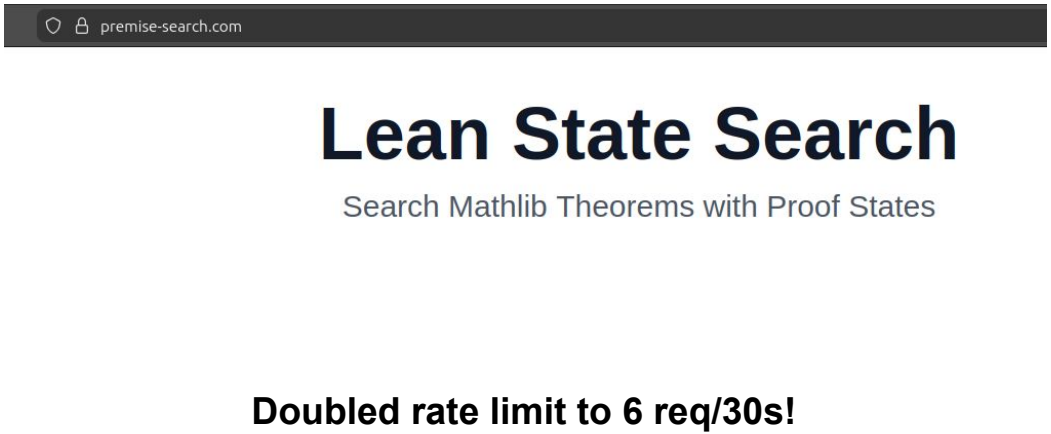
BUT
1M uncategorized requests within a few days!

https://github.com/nomeata/loogle

Usage Statistics: Lean State Search

API Calls Per Month (total)

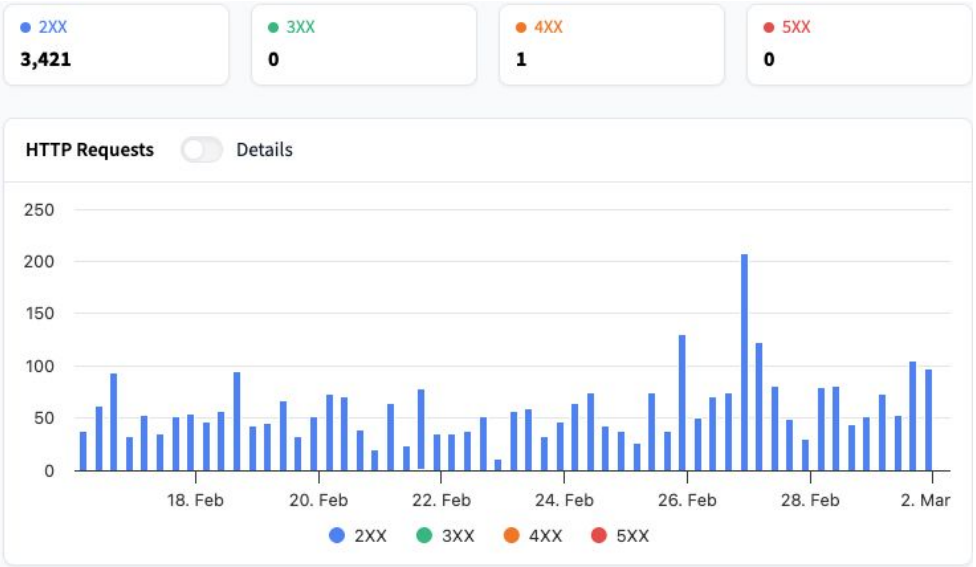
2025-08: 30
2025-09: 161
2025-10: 547
2025-11: 5864
2025-12: 1347
2026-01: 21108
2026-02: 2140
2026-03: 93 (until 3.3)



Doubled rate limit to 6 req/30s!

AI4Math team at Renmin University of China https://premise-search.com/

Usage Statistics: Lean Finder



2 weeks - total requests

v2 coming up!

1. Compatibility across Lean versions

2. Stronger robustness to ambiguous and incomplete search queries

3. Broader coverage of third-party Lean repositories

https://arxiv.org/html/2510.15940v1
https://huggingface.co/spaces/delta-lab-ai/Lean-Finder

Case Study: M2F/ReasBook: *Analysis II* (Tao)

272132 lines of Lean code, 210M model tokens
772 tokens / LOC

Total Lean wall-time (lower bound):
94,553 s ≈ 26.3 h

Breakdown (no timeout failures):
- `lean_check`: 49,224.5 s ≈ 13.7 h
- Agent Lean calls (subtotal): 45,329.0 s ≈ 12.6 h
  - `lean-lsp.*`: 19,687.0 s over 7,765 calls ≈ 5.5 h (21 %)
  - `lake env lean`: 25,642.0 s over 4,201 calls

lean-lsp-mcp
calls: 7,881
success: 6,837
failed: 1,044
success rate: 86.75%
total time: 26,646.982s (~7.40h)

Tool	Calls	Success	Avg Time
lean_local_search	2,658	99.5%	0.3s
lean_run_code	2,356	98.9%	7.3s
lean_loogle	1,411	42.6%	1.0s
lean_leansearch	955	89.8%	0.7s
lean_leanfinder	153	100%	1.1s
lean_goal	100	88.0%	10.0s
lean_diagnostic…	88	23.9%	48.2s
lean_file_outline	63	96.8%	15.6s
lean_state_search	25	68.0%	1.0s
lean_hammer_…	19	73.7%	0.6s
lean_file_contents	16	100%	<0.1s
lean_multi_attempt	15	100%	11.5s
lean_hover_info	10	90.0%	6.3s
lean_term_goal	7	100%	2.1s
lean_declaration_file	4	0%	3.7s
lean_code_actions	1	100%	2.3s

(including timeout failures)

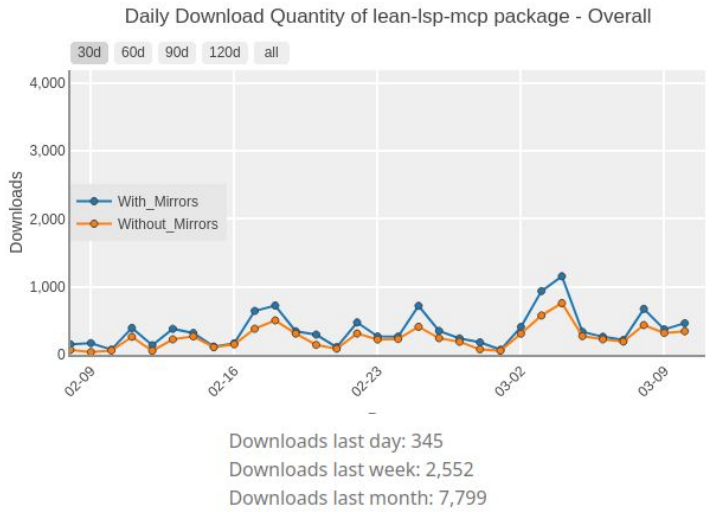
M2F: Automated Formalization of Mathematical Literature at Scale https://arxiv.org/abs/2602.17016

Zichen Wang, Wanli Ma, Zhenyu Ming, Gong Zhang, Kun Yuan, Zaiwen Wen

Codex CLI with model reasoning effort="high"

Usage Statistics: lean-lsp-mcp

github



pypi

Agents

Claude Code
Cursor
VS Code
Codex
Gemini CLI

Seed2.0 Code (ByteDance - Thomas Zhu)

Table 5 Evaluation on Putnam-200 [98], Pass@8. Agent-based multi-turn setup with Lean, Python, and Lean search tools. The highest score is marked in **bold**, and the second is underlined.

Benchmark	Deepseek-Prover-V2	Seed-1.5 Prover	Gemini-3-Pro	Seed2.0 Lite	Seed2.0 Pro
Putnam-200	<4.0	26.5	26.5	30.5	**35.5**

https://x.com/hanwen_zhu/status/2022723551422484778

lean4-skills + lean-lsp-mcp

Mutually beneficial

- lean4-skills has more extensive descriptions of tools
- lean4-skills uses lean-lsp-mcp tools in workflows

Advantages

- **Explicit workflows** that agents invoke directly. E.g.
  - iterative proving
  - filling sorries
  - replacing axioms
- **Faster exploration**: rapid testing via LSP without full rebuilds; parallel tactic attempts
- **Better mathlib discovery**: find more relevant lemmas → shorter, more modular proofs
- **Parallelization**: delegate to subagents/background tasks because of explicit workflows
- **Guardrails**: non-standard axioms, detects false conjectures on decidable goals early

https://github.com/cameronfreer/lean4-skills

Local & Self-hosted Tools

Feature	Trigger	Notes
REPL mode (multi attempt)	--repl or LEAN_REPL=true	~5x faster, local subprocess
Loogle	--loogle-local or LEAN_LOOGLE_LOCAL=true	~6GB, first run builds index
Local search (ripgrep)		Requires rg installed

Service	Env Var	Default (remote)
Loogle	LOOGLE_URL	loogle.lean-lang.org
Premise Search	LEAN_STATE_SEARCH_URL	premise-search.com
Lean Hammer	LEAN_HAMMER_URL	leanpremise.net

Challenge: Context

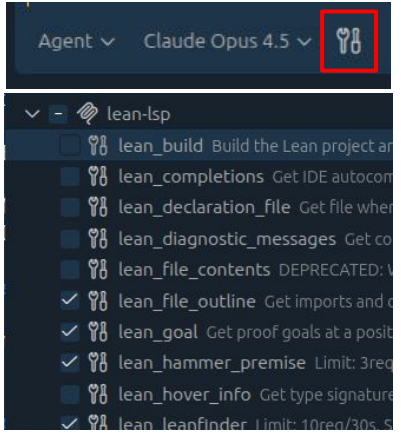
```
claude-sonnet-4-5-20250929 · 23k/200k tokens (12%)
⛁ System prompt: 2.9k tokens (1.5%)
⛁ System tools: 17.0k tokens (8.5%)
⛁ MCP tools: 3.5k tokens (1.7%)
⛁ Messages: 8 tokens (0.0%)
⛶ Free space: 132k (65.8%)
⛝ Autocompact buffer: 45.0k tokens (22.5%)
```

```
MCP tools · /mcp
└ mcp__lean-lsp__lean_build: 201 tokens
└ mcp__lean-lsp__lean_file_contents: 157 tokens
└ mcp__lean-lsp__lean_diagnostic_messages: 262 tokens
└ mcp__lean-lsp__lean_goal: 248 tokens
└ mcp__lean-lsp__lean_term_goal: 190 tokens
└ mcp__lean-lsp__lean_hover_info: 178 tokens
└ mcp__lean-lsp__lean_completions: 227 tokens
└ mcp__lean-lsp__lean_declaration_file: 143 tokens
└ mcp__lean-lsp__lean_multi_attempt: 191 tokens
└ mcp__lean-lsp__lean_run_code: 105 tokens
└ mcp__lean-lsp__lean_local_search: 189 tokens
└ mcp__lean-lsp__lean_leansearch: 222 tokens
└ mcp__lean-lsp__lean_loogle: 225 tokens
└ mcp__lean-lsp__lean_leanfinder: 200 tokens
└ mcp__lean-lsp__lean_state_search: 234 tokens
└ mcp__lean-lsp__lean_hammer_premise: 257 tokens
└ mcp__lean-lsp__lean_profile_proof: 232 tokens
```

Disable useless tools

- Tool incompatible with toolchain
- Agent keeps misusing tool
- LSP support not needed

Claude Code via permissions

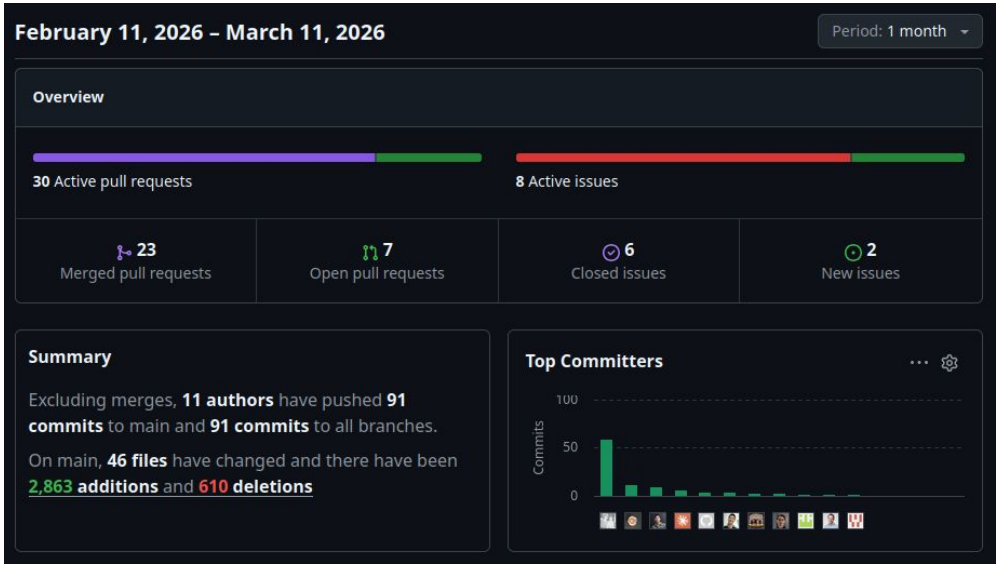


VS Code

Tool Search allows Claude Code to dynamically load tools into context when MCP tools would otherwise take up a lot of context.

https://github.com/anthropics/claude-code/issues/7336

Challenge: AI contributions



Move fast

Long-term vision and quality

Reviewing is less fun than coding!

Maintenance responsibility

Measures

- Issue before PR (enforce?)
- PR template w prompt suggestions
- AI pre-screening
- Tougher CI
- Auto-close stale PRs
- Max 1-2 open PRs per contributor

Related Projects

Interact

- leanclient https://github.com/oOo0oOo/leanclient
- REPL https://github.com/leanprover-community/repl
- PyPantograph https://github.com/stanford-centaur/PyPantograph
- LeanInteract https://github.com/augustepoiroux/LeanInteract
- Kimina Server https://github.com/project-numina/kimina-lean-server

Faster!
RL / MCTS

MCP

- LeanExplore MCP https://www.leanexplore.com/docs/mcp
- axle-mcp-server https://github.com/AxiomMath/axle-mcp-server
- project-numina/lean-lsp-mcp https://github.com/project-numina/lean-lsp-mcp

Skills

- Lean Skills https://github.com/leanprover/skills
- mathlib-quality https://github.com/CBirkbeck/mathlib-quality

Thanks!

Releases 71

Version	Release date
0.24.0	19 minutes ago
0.0.1	Mar 31, 2025

Contributors: Alok Singh, Elias Judin, Jesse Alama, Rex Wang

External APIs: leansearch, loogle, leanfinder, lean-state-search, lean-hammer

Lean FRO: Language server and an incredible ecosystem

Try it yourself

github.com/oOo0oOo/lean-lsp-mcp

VS Code

https://vscode.dev/redirect/mcp/install?name=lean-lsp&config=%7B%22command%22%3A%22uvx%22%2C%22args%22%3A%5B%22lean-lsp-mcp%22%5D%7D

Claude Code

$ claude mcp add lean-lsp uvx lean-lsp-mcp

Questions?