My MCP Server Code Works, but the Agent Fails

The Case for MCP-Specific Evaluation

Calum Murray
Software Engineer

Wesley Chun
Technical Program Manager - AI

Red Hat

About the speakers

Calum Murray, Software Engineer

CNCF Ambassador, Kubernetes MCP Server and Knative Maintainer, Creator of mcpchecker

Wesley Chun, AI Technical Program Mgr.

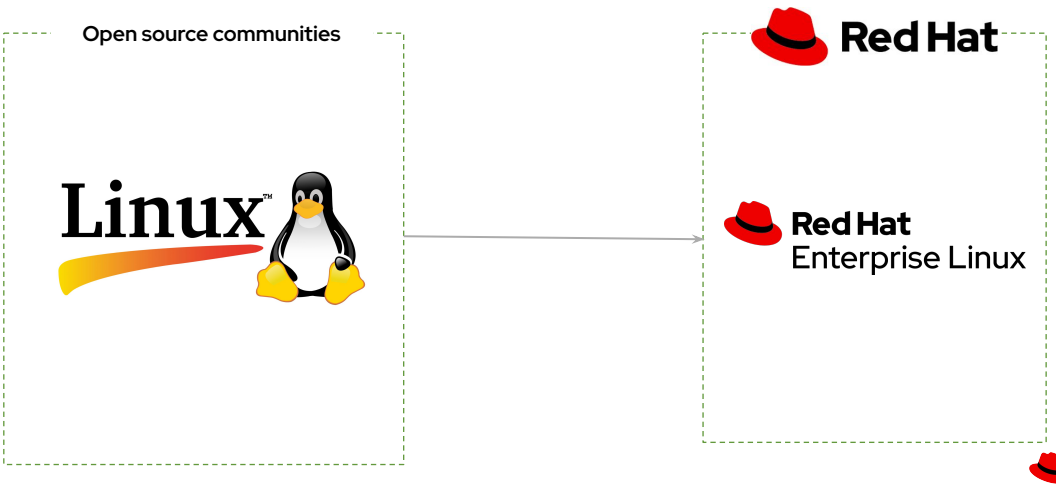
Author of "Core Python" books, Google Developer Expert; original Yahoo!Mail engineer, Fellow of the Python Software Foundation; connect w/me @wescpy (T/BS)

What we'll discuss today

- Why Red Hat and Red Hat AI?
- Why do we need MCP-specific evals?
- Mcpchecker: what it is & how it works
- Demo
- General findings from using mcpchecker on the Kubernetes MCP Server
- Try mcpchecker & help us improve it!

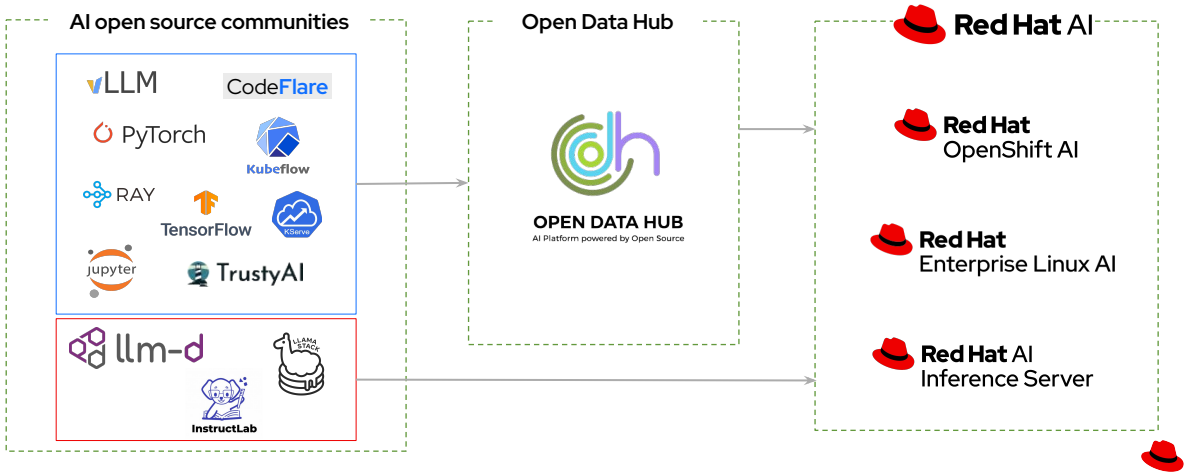
(Years ago) The future of Unix is open

Red Hat's open source community engagement is a catalyst for powerful collaboration



The future of AI is open

Red Hat's open source community engagement is a catalyst for powerful AI collaboration



Red Hat AI OSS key contributor

Provides resources & support, building enterprise versions for customers

PyTorch

Organizations leaderboard

Organizations ranked by the number of contribution activities performed by contributors on their behalf during the selected time period. Learn more

Include collaborations

All activities

Organization	Total contributions
Meta Platforms, Inc.	85,816 - 54%
Intel Corporation	9,853 - 6%
NVIDIA Corporation	8,585 - 5%
Advanced Micro Devices, Inc.	7,682 - 5%
Quansight LLC	6,747 - 4%
Codeacademy	3,528 - 2%
Red Hat	3,291 - 2%
DeepSeek AI	2,762 - 2%

vLLM

Organizations leaderboard

Organizations ranked by the number of contribution activities performed by contributors on their behalf during the selected time period. Learn more

Include collaborations

All activities

Organization	Total contributions
Red Hat	24,481 - 11%
Huawei	18,891 - 8%
IBM	18,841 - 8%
Python Discord	15,823 - 7%
Intel Corporation	12,542 - 6%
Meta Platforms, Inc.	7,285 - 3%
Hugging Face Inc	7,174 - 3%
Neural Magic	5,453 - 2%

llm-d

Organizations leaderboard

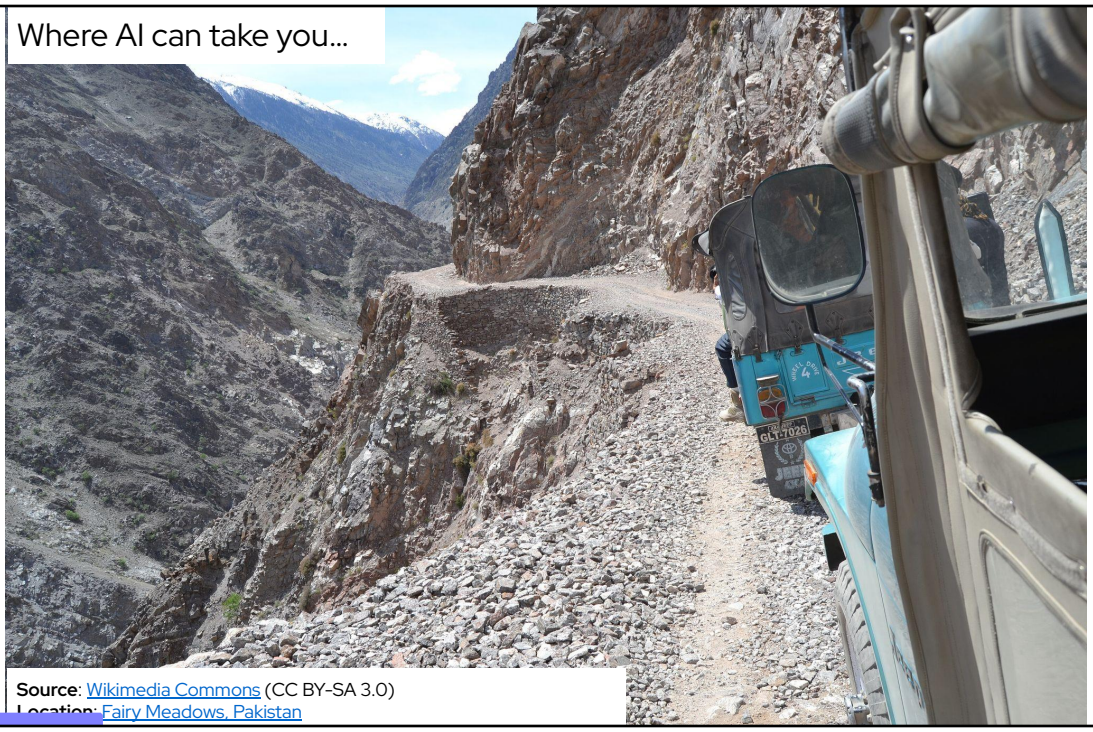
Organizations ranked by the number of contribution activities performed by contributors on their behalf during the selected time period. Learn more

Include collaborations

All activities

Organization	Total contributions
Red Hat	2,540 - 36%
IBM	1,620 - 23%
Google LLC	1,241 - 18%
Intel Corporation	676 - 10%
Neural Magic	218 - 3%
Flame	121 - 2%
DJV Imaging	112 - 2%
BHOI	84 - 1%

Where AI can take you...



Source: Wikimedia Commons (CC BY-SA 3.0)
Location: Fairy Meadows, Pakistan

Guardrails help but...

Source: Gemini Flash 3.1 Image (Nano Banana 2)

... but also need evals

Source: Gemini Flash 3.1 Image (Nano Banana 2)

Why do we need MCP-specific Evals?

Why not traditional tests or agent evals?

The illusion of perfection

With time and caring (and the help of agentic coding tools!) it is possible to create MCP servers that meet the traditional definition of “perfect”/really good software

- High test coverage
- Clean code
- Typed Schemas

The illusion of perfection

However despite the seemingly high quality of software, when an agent uses an MCP server, it can easily run into problems:

- Agent doesn’t understand when to use different tools
- Agent doesn’t call certain tools
- Tool definitions explode the context of the agent
- Agent can’t understand tool outputs
- Agent struggles to correctly set inputs for tools

The probabilistic gap

This problem is what we call the probabilistic gap: the server itself is deterministic, but the agent using it is stochastic

- Traditional software testing can not give a full picture of quality/effectiveness of MCP servers
- Need to test these MCP servers with agents to understand how well they can understand and use them

Agent evals to the rescue?

As agents have exploded over the past few years, there have been agent evaluation tools created to help evaluate and compare agents across many metrics

- Quality/tone/accuracy of response
- Correctness of actions taken
- Focus primarily on the outcome and what the agent did, not on the MCP server(s) connected to the agent

Agent evals to the rescue?

Using an agent eval to test your MCP server *may* make sense if

- MCP Server only meant to be used by the specific agent
  - Sort of defeats the purpose of having a MCP server instead of agent specific tools
- You care more about agent level metrics than about the MCP server specifically

Types of Traditional Software Testing

Unit Tests
Examine individual components or functions of the system in isolation.

Integration Tests
Verify different modules or services interact with each other correctly.

System Tests
Check that the entire system works correctly end-to-end

Acceptance Tests
Formal tests to confirm that the system meets business requirements and customer needs

Evaluations/Benchmarks are the new Tests

As we build newer agentic software systems we are going to need new types of benchmarks/evaluations to properly test the probabilistic gap posed by agents

- LLM benchmarks published by researchers/model providers are like the new unit tests
- Agent evaluations are like the new system tests
- MCP specific evaluations should be the new integration tests

What is mcpchecker and how does it work?

mcpchecker

An open source MCP evaluation framework created originally for the Kubernetes MCP Server

- Focuses on testing the agent/MCP server semantic interface
- Compatible with (almost) any agent
- Compatible with any MCP server or combo of MCP servers

mcpchecker

Project GitHub

Key Design Considerations

Several key design considerations while building mcpchecker:

- Framework needs to be able to capture all MCP interactions
- Framework should work with any agent without requiring code changes from the agent
- Eval scenarios should be easy to review and understand

Capture all MCP interactions

In order to capture all MCP interactions, mcpchecker creates an MCP proxy server that sits between the real MCP server and the agent

- Captures and records all communications between the agent and the MCP server
- Allows for seeing what the agent does on the MCP server without instrumenting the MCP server *or* the agent

Work with any agent

Mcpchecker supports several ways to “work with any agent:”

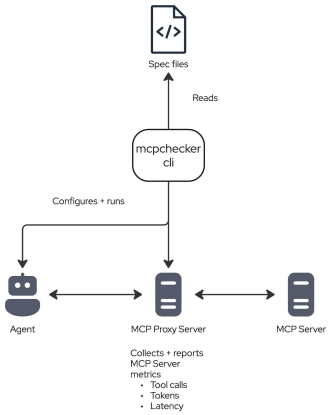
- Agent Client Protocol (ACP), with emerging support & popularity, allows for agents like Claude code, OpenCode, etc.
- A built-in generic LLM agent allows testing directly against providers like OpenAI/Anthropic/Google/local OpenAI-compatible models
- Using a shell wrapper to any arbitrary agent you provide

Easy to understand Evals

To enable eval scenarios to be easily understandable and portable across different developer ecosystems, each eval scenario should be understandable on its own and not tied to a single dev language

- Eval tasks are defined in yaml
- Declarative, easy to tell what is being done in each task and how the agent is being prompted
- Doesn’t place dependency on any one language ecosystem (e.g. python) into all projects using the framework

Capture all MCP interactions



Demo

Follow along!

Mcpchecker quickstarts repo, scenario 3

Our Scenario

We have a simple MCP server that assists with text processing and provides four tools: that

- Convert to uppercase
- Convert to lowercase
- Convert to title case
- Capitalize only the first letter

Our Scenario

Initially this MCP has been created in a way where agents will struggle to understand/use these simple tools

- Misleading generic names for tools (process/transform/convert/format_text)
- Vague tool descriptions
- No type hints

Our Scenario

Our goal is to first check if the agent can use these tools successfully to complete tasks (hint: it won't).

If not, improve our server based on what we find to make the agent use our tools successfully.

Recap

mcpchecker allows us to:

- Identify whether the agent is using specific tools (and how it's using them)
- See how the agent thinks about these tasks and how to approach them
- Fix our MCP server and see our evaluations pass, confirming the agent is understanding and correctly using our MCP server

General findings from using mcpchecker on the Kubernetes MCP Server

Why the Kubernetes MCP Server results?

The Kubernetes MCP server is a popular, widely used MCP server and is the project mcpchecker was originally developed for.

- Has a well maintained set of evaluation tasks
- Can show some more realistic examples of what integrations tests for your MCP server can find

Does MCP “Code Mode” work?

In the Kubernetes MCP Server, we experimented with & measured the success of a “Code Mode” version, which had a single “execute_ script” tool allowing the server to run JS that calls Kubernetes client code

- When JS mapped to MCP tools and not official client library (which had many examples online), the agent was only successful 41.67% of tasks vs. 87.5% of tasks
- With an added tool to search available APIs, the agent was able to get to 91.67% accuracy, but at a 4x token cost to the baseline
- When the agent had both code mode and normal tools available, it only called code mode about 6% of the time

Tokens and Agents Giving Up

One interesting finding: as tool definitions expand/tool inputs and outputs grow agents seem to "give up" faster

- With fewer tokens in the context, agents will retry and often figure out how to solve issues
- With more tokens used, the agent often "gives up" and tells the user it can't complete the task or explains how to continue without actually taking the actions itself

Deleting to fix incorrect updates

There are certain scenarios in Kubernetes where an update request for a resource is rejected:

- When agents encounter this, the default seems to be to delete the resource and then recreate it with whatever updates applied
- Very dangerous behaviour, as this can cause outages, loss of data, etc.
- Would not be caught if only asserting on the final output/result!

Help us improve mcpchecker!

Get involved, try it out!

Join the mcpchecker community

Recent open source project; we welcome new collaborators!

- Try it on your MCP server & give us feedback on how we can improve
- Several "quickstarts" to kick off your journey!
  - github.com/mcpchecker/quickstarts
- Contribute new features, bug fixes, extensions, etc.
- Help us shape the future of integration testing for MCP servers
- Hop on the GitHub Discussions to chat with us -- we are all there
- Also see us at the Red Hat booth in the expo hall

Help us improve mcpchecker

mcpchecker

Project GitHub

Progress bars
goo.gl/69EJVw

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/@redhat

facebook.com/RedHat

x.com/RedHat

Red Hat