



# Let's be smart, doing AI with WildFly

ehugonne@redhat.com, jdenise@redhat.com  
Feedback [form](#)



# Presentation topics



- WildFly MCP project
  - MCP protocol
  - Interact with running WildFly servers in natural language
- JakartaEE meets AI
  - Develop AI applications with WildFly
  - WildFly AI feature-pack

The background of the slide is a stylized, layered mountain range. The mountains are composed of several overlapping geometric shapes in shades of dark blue, medium blue, and light grey. On the left side, a rainbow with multiple bands of color (red, orange, yellow, green, blue, purple) is partially visible, appearing to rise behind the mountain peaks. The sky is a solid, dark teal color.

# WildFly MCP

# MCP (Model Context Protocol)



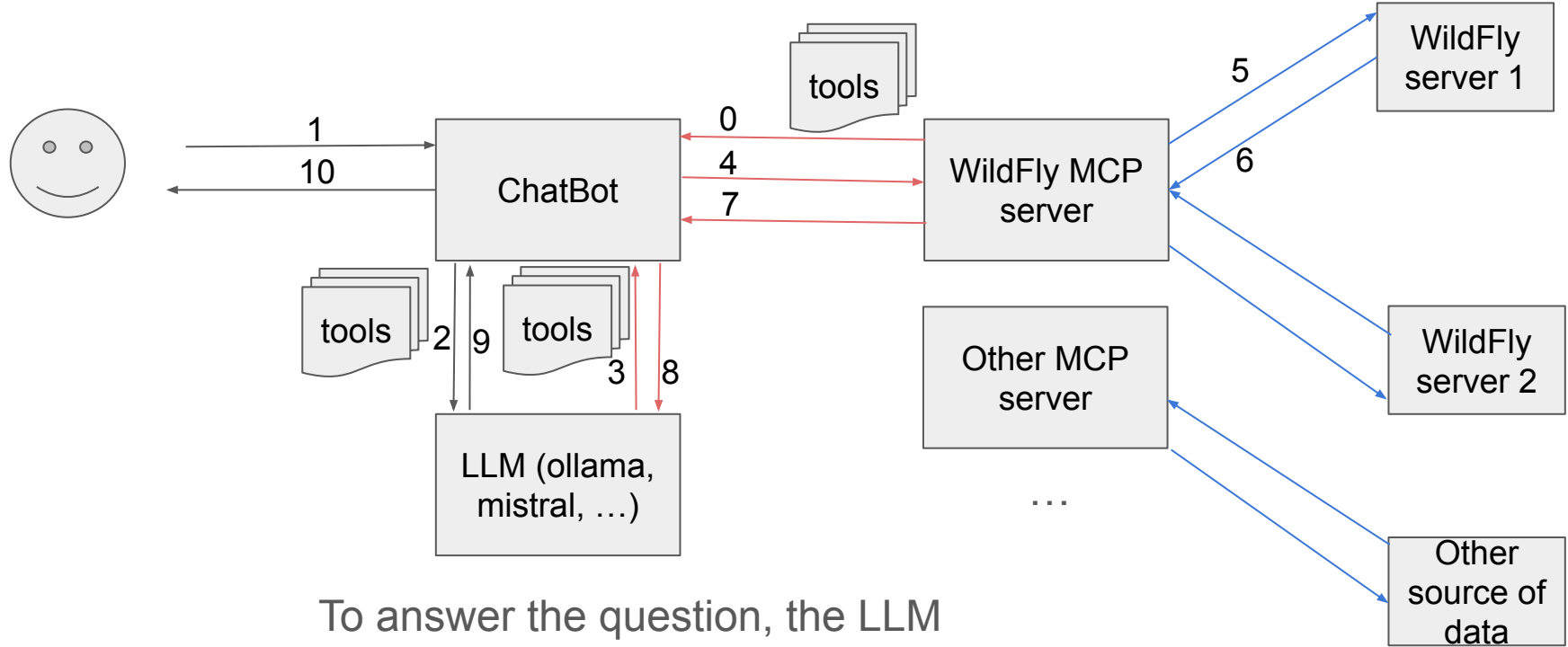
- [MCP specification](#) specifies integration of external sources of data inside the LLM (Large Language Model) context.
- Enrich LLM capabilities with access to live data (DB, running servers, graphic generation, forecasts, code, file system/web access, git repos,...).
- Although a new spec (2024-11-05), the set of [existing MCP servers](#) is quite impressive.
- 3 main features
  - Tools
  - Prompts
  - Resources

# MCP tools



- Some LLM have the capability to execute tools (functions) to compute a reply.
- MCP specifies a way to describe and invoke tools.
- MCP tools are served by an MCP server
- MCP tools are consumed by an MCP client (AI application, e.g.: chatbot)

# MCP tools workflow



To answer the question, the LLM identifies a need to invoke 0 to N tools

# JSON Message sent to the LLM



```
"messages" : [ {  
    "role" : "user",  
    "content" : "What is the status of my wildfly server?"  
} ],  
  
"tools" : [ {  
    "function" : {  
        "name" : "getWildFlyServerConfiguration",  
        "description" : "Gets the server configuration and the deployed applications in JSON  
format",  
    }, {  
        "function" : {  
            "name" : "getWildFlyHealth",  
            "description" : "Get the WildFly server Health.",  
        }, ...
```

# LLM reply, chatbot call the MCP servers



```
{"role": "assistant",  
  
  "tool_calls": [  
  
    {"id": "5Q4LZOT8w",  
  
      "function": {"name": "getWildFlyHealth", "arguments": "{}"}  
  
    }  
  
  ]  
  
}
```



# Chatbot then re-send to the LLM



```
"messages" : [ {  
    "role" : "user",  
    "content" : "What is the status of my wildfly server?"  
},  
  
{ "role" : "assistant",  
  "tool_calls" : "getWildFlyHealth"  
},  
  
{ "role" : "tool",  
  
  "content" : [{ "name" : "deployments-status", "outcome" : true, "data" : [{  
    "servlet-security.war" : "OK" }]}, {"name" : "suspend-state", "outcome" : true, "data" : [{  
    "value" : "RUNNING" }]}], {"name" : "boot-errors", "outcome" : true}, {"name" :  
    "server-state", "outcome" : true, "data" : [{ "value" : "running" }]}], {"name" :  
    "live-server", "outcome" : true}, {"name" : "started-server", "outcome" : true}, {" "outcome"  
    : true }]  
  
} ]
```

# LLM final reply targeted to the user



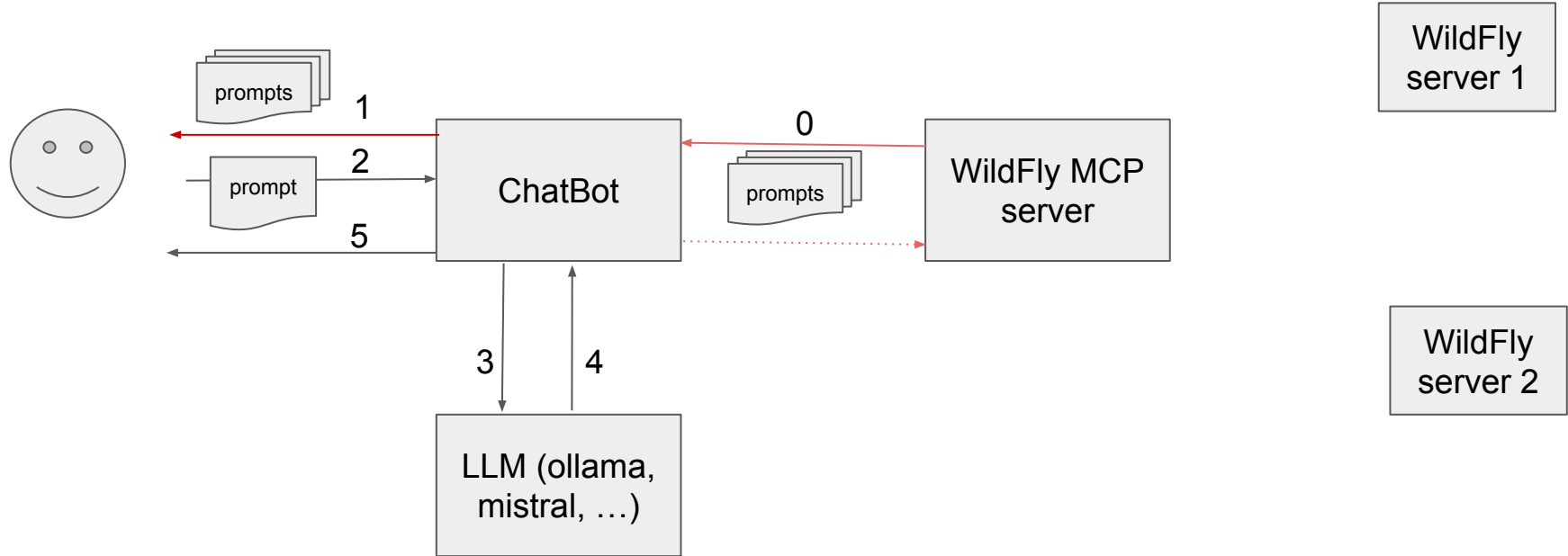
```
"messages" : [ {  
    "role" : "assistant",  
    "content" : "The WildFly server is running, all deployments are up and running."  
} ]
```

# MCP prompts



- Pre-defined questions (user prompts) that MCP server exposes
- Chatbot discovers and exposes the prompts to the user
- User can configure and execute a given prompt
- The LLM receives the selected prompt like any other question

# MCP prompts workflow



LLM receives the user selected prompt as a normal question, it can initiate a tool workflow if needed.

# MCP resources



- Data exposed by an MCP server (documentation, log, ...)
- Chatbot discovers the resources and expose them to the user.
- User can select resources
- Usage of resources is unspecified and chatbot specific. For example:
  - RAG
  - Search,
  - Injected in LLM context, ...

# WildFly MCP



- A new project: [wildfly-extras/wildfly-mcp](https://github.com/wildfly-extras/wildfly-mcp)
- An MCP server, [WildFly MCP server](#)
  - A bridge between LLM and WildFly running instances
  - Can be used in any MCP client (claude-desktop, ...)
- An MCP client example, [WildFly Chat Bot](#)
- All those technologies are currently in Alpha grade and evolve quickly!

# WildFly MCP server



- A [quarkus](#) fat jar application.
- Allows to interact with multiple WildFly server running instances.
- Exposes JVM and WildFly servers source of data:
  - JVM configuration, version, ...
  - Log files, server configuration, Prometheus Metrics, Health status
  - Execute WildFly CLI operation
- Expose Deployment content
  - Deployed applications, binary content (XML descriptors, ...)
- Allows to execute WildFly CLI operations (baby steps... ;-) )
- Defines some predefined user prompts one can discover and use.
- Can be built or run as a [container image](#).

# WildFly Chat Bot



- Compliant MCP client chat bot:
  - Support for MCP tools and prompts.
  - Configured to use the WildFly MCP server
  - Supports also other MCP stdio and SSE mcp servers.
- Developed with WildFly:
  - Slimmed server provisioned with the [WildFly AI Galleon feature-pack](#)
  - WildFly Bootable JAR
  - Lanchain4j API
  - Web socket + javascript
- Can connect to:
  - [Ollama](#) (models run locally)
  - [Grog](#)
  - [Mistral](#)
  - [Github models](#)
- Can be built or run as a [container image](#).



# WildFly Chat Bot, a UI for developers



- You can use it to interact with:
  - WildFly server (WildFly MCP server), monitor and troubleshoot WildFly.
  - Developed MCP servers using the WildFly API ([AI feature-pack](#))
- Allows to test:
  - Various LLM
  - How your own MCP server (eg: [car-booking](#) application) is behaving.
    - You can extend the chatbot system prompt with your own system prompt. For example:  
  
"You are a car booking fraud detection AI for Miles of Smiles.You have to detect customer fraud in bookings."
    - Direct testing of your MCP tools and prompts.

# On WildFly Youtube channel



- Monitoring WildFly with claude-desktop [demo](#)
  - Search for security attacks
  - Display metrics in nice charts
- Monitoring and troubleshooting WildFly with the WildFly chatbot [demo](#)
  - General monitoring
  - Attempt to identify and fix a configuration issue

# Today demos



- Simple monitoring of WildFly server and JVM.
- Monitoring resource consumption and configuration issues
  - Is my application running properly?
- Chasing deployment configuration issues
  - Why do I get an NPE?
- Those demos use [Mistral Small](#) LLM



# Limitations

- Strongly bound to the capabilities of your LLM
- WildFly CLI operations and WildFly management model not well understood by all LLM.
- Hallucinations and wrong replies are still in the picture...
- Token limits with large logs, server configuration.