# Serving GenAI in an Enterprise World

## Red Hat User Group

Brian Ball, PhD
Associate Principal AI Specialist
Solutions Architect

Red Hat

# Agenda

1. Models as a Service (MaaS) Demo
2. Agentic Systems



Red Hat

# Models as a Service Demo



Red Hat

# Agentic Systems



Red Hat

# Fundamentals

- **Large Language Model**: a model that, given input words, predicts the next “word”
  - **Multimodal**: Can handle pictures/video/audio as well as written language
- Software design principles still matter



Red Hat

# Agents and Tools

- **Agent**: The LLM can (or must) “decide” when to reach out and to whom
  - Depending on the framework, you can either specify the flow or let the LLM handle it
  - Output format includes tags to call a registered system with specific inputs
- **Tool**: A system that can be reached out to

## Example:

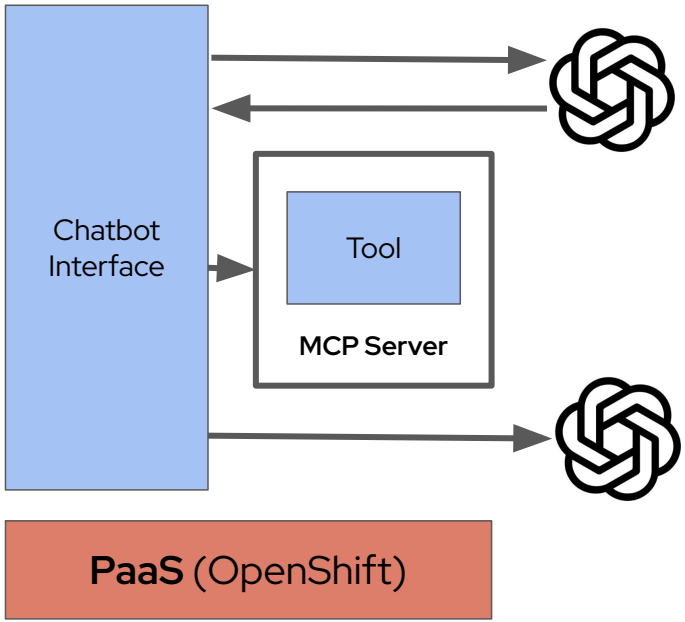
What is today’s weather?

Today it is get_temperature(Michigan) degrees in Michigan.



Red Hat

# Model Context Protocol




## What?

- Open standard for remote tool calling created by Anthropic

## Why?

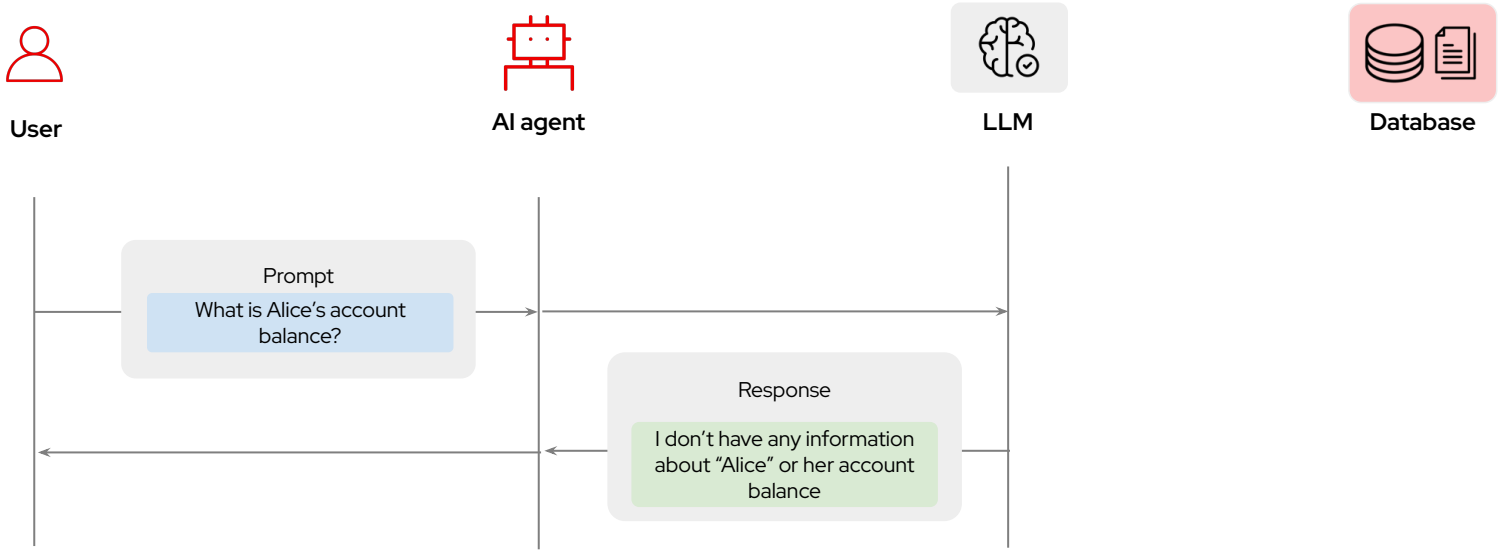
- Existing tools can be redeployed into their own containers.
- Tools can have independent lifecycles from the application and each other.
- Access to the services can be strictly restricted
- Scaling of the containers can be serverless and/or auto scale.
- Dedicated health checks can be developed as appropriate for each tool.

## Issues

- Authentication and access control immaturity
- No integration with or awareness of core services (vector store, embeddings, guardrails, standards, identity)
- Rapidly evolving standard
- Model accuracy (especially small models) falls off quickly

Red Hat

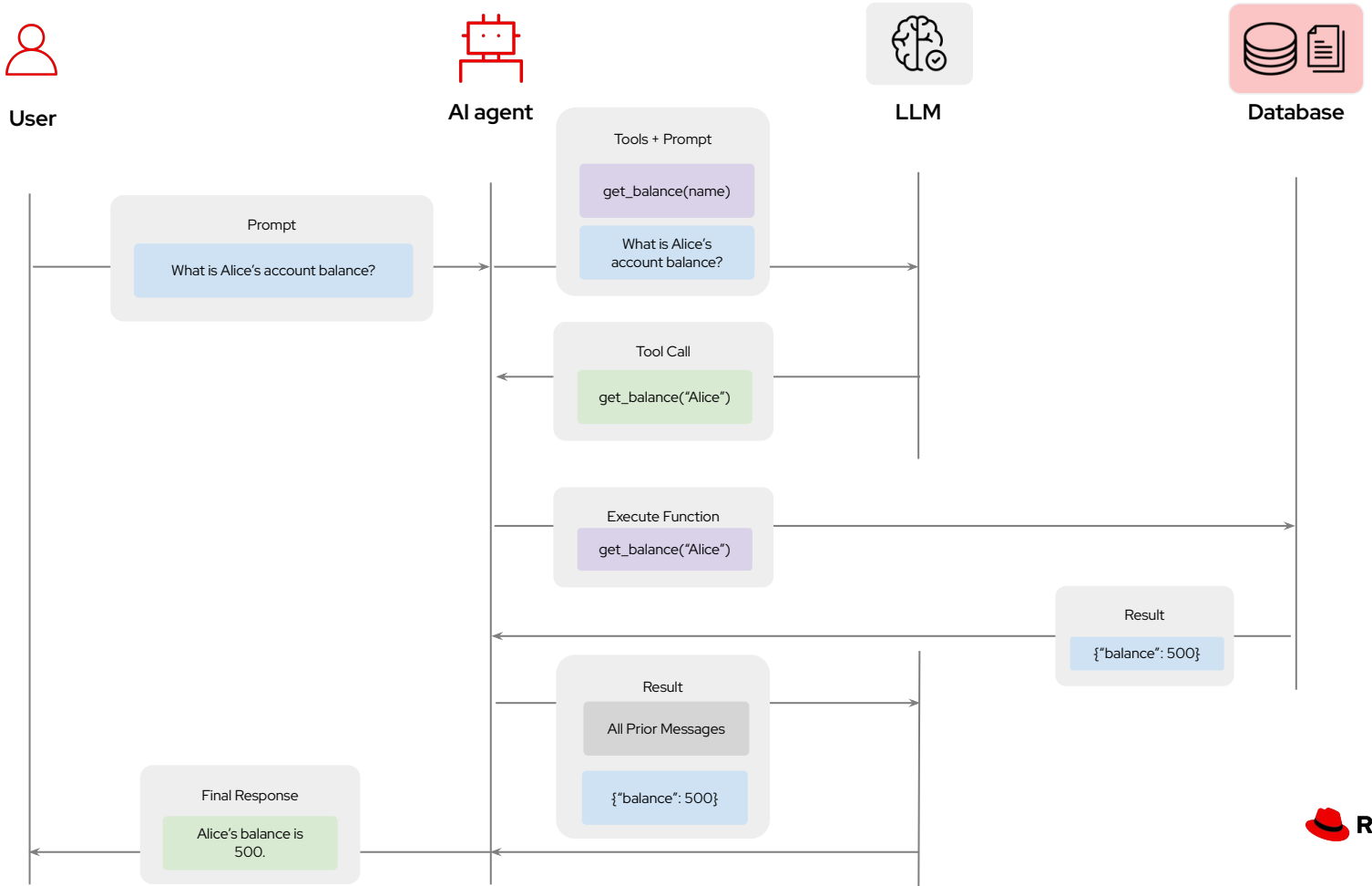
# AI Agents Need Tools






Red Hat

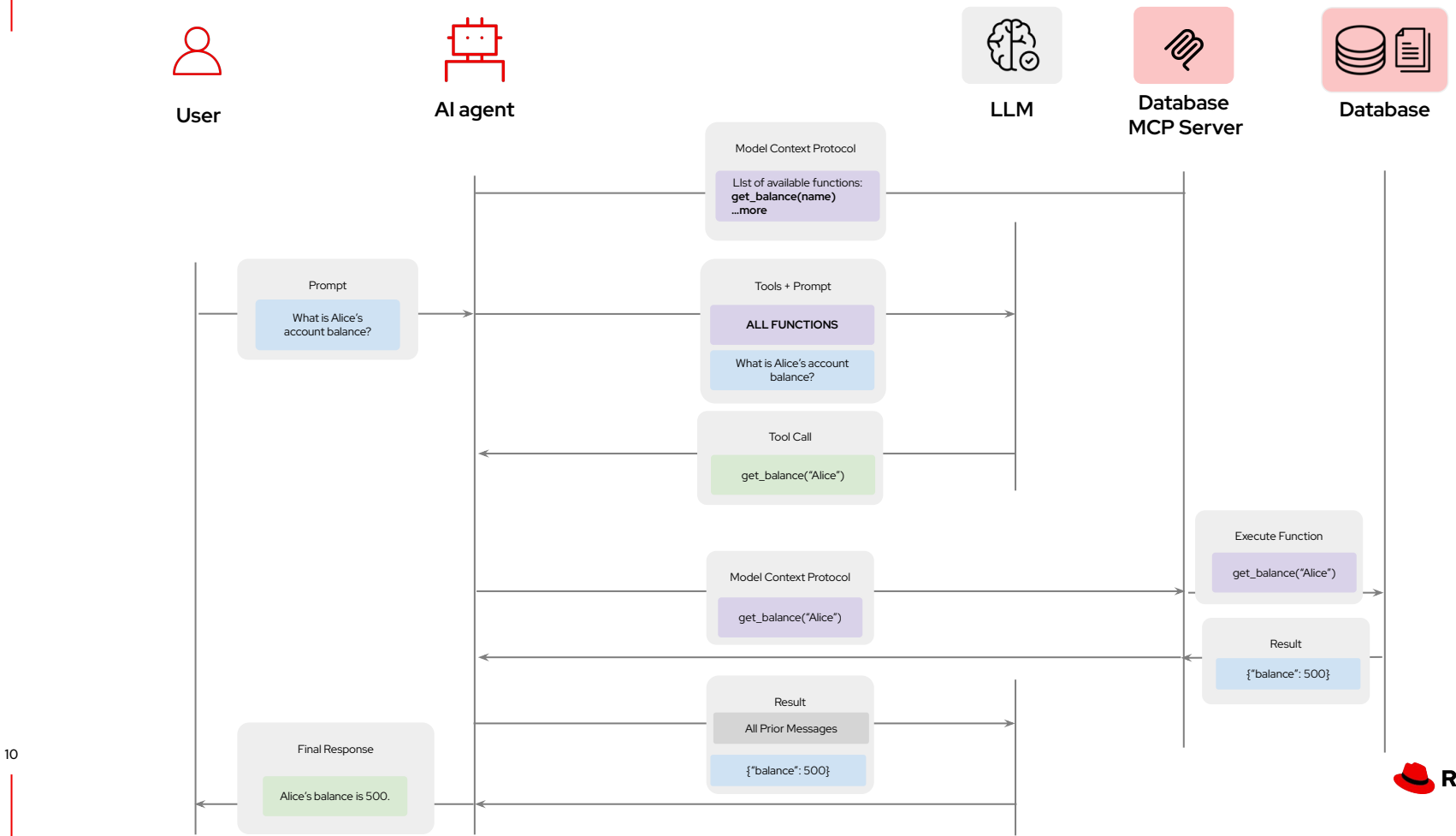
# Tool Calling






Red Hat

# Tool Calling with MCP






Red Hat

# Open Source Agentic Frameworks

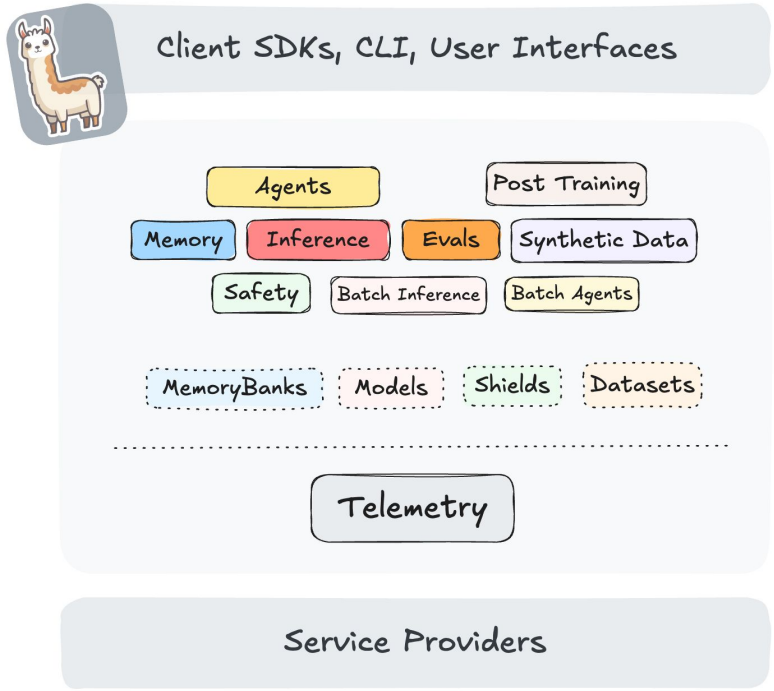
- LangChain/LangGraph
- Microsoft AutoGen
- CrewAI
- Llama Stack



Red Hat

# Enterprise Design

- Challenge with LangChain is that it's a decentralized design
  - Everyone has their own code to maintain
  - When centralizing, worst case scenario, if one component goes down, the entire system goes down
- Microservices design implies that you should build a middleware service to decouple the LLM from other components of the system

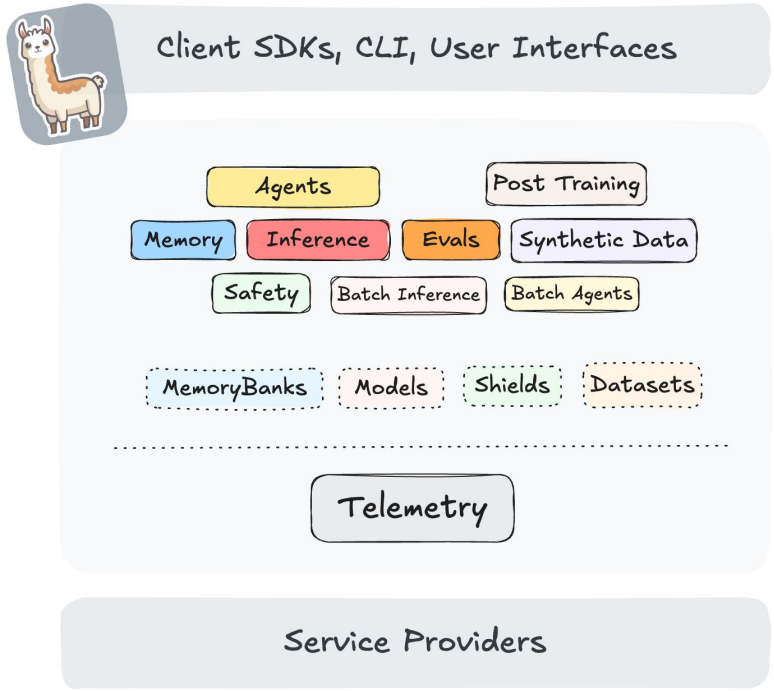





Red Hat

# Enterprise Design

- Middleware needs to exist in a location where it can talk to the other resources it needs
  - Needs to compose the components in the right way






Red Hat

# Example: Bank Agent

- Retrieve account information and relevant regulations
- Need to be aware of the context and stick to it
- Can’t reveal information in the wrong systems (e.g. auto loans, bank accounts)
- Can’t reveal someone else’s information
- Different models may be suited for different tasks
  - Some models (e.g. public SaaS) may not be allowed for some tasks



Red Hat

# Agentic Issues

- Model accuracy falls off quickly with more tools especially for small models
- Authentication and access control immaturity
- No integration with or awareness of core services (vector store, embeddings, guardrails, standards, identity)
- Rapidly evolving standards



Red Hat

# RHOAI 3.x MCP Roadmap

## Give AI Engineers the power to connect models to real-world tools – right now



**Llama Stack** - Tech Preview today (2.23), GA in 3.0

**Registry (Dev Preview 3.1)** – Central source of truth

**Catalog (3.2)** – Curated, enterprise-approved list

**Gateway (3.2)** – Unified access to external + internal MCP servers

**MCP as a Service (MCPaaS)** – Easy deployment & discovery

**Agents, Knowledge Sources, & Guardrails** – Enterprise-ready workflows (later)

*Roadmap subject to change

# Questions?



linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat



Red Hat