

JUNE 30, 2025

# Kowalski – Your Helpful AI Assistant

[github.com/kowalski-org/kowalski](https://github.com/kowalski-org/kowalski)

Christian Goll [cgoll@suse.com](mailto:cgoll@suse.com)  
[github.com/mslacken/SUSE\\_Day\\_2025](https://github.com/mslacken/SUSE_Day_2025)



# Outline

Introduction

AI Tools

Kowalski Implementation

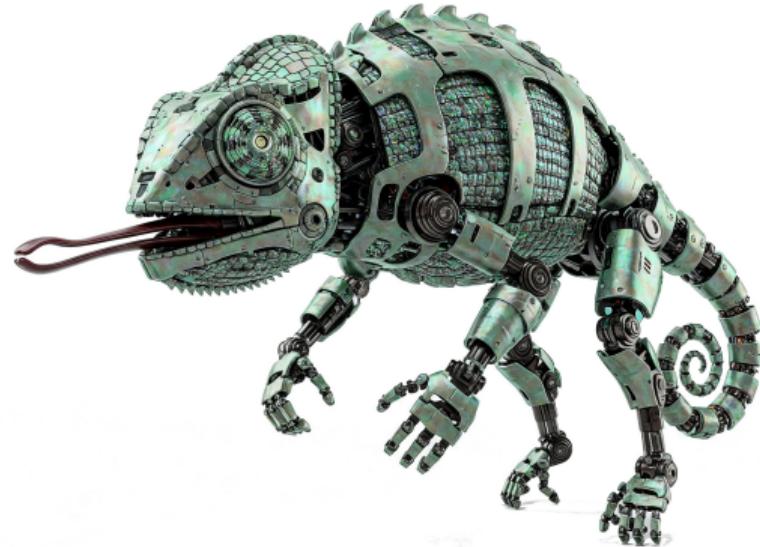
MCP and Kowalski



# The Challenge: LLMs and System Configuration

## Current state

- Large Language Models (LLMs) are powerful for general tasks.
- LLMs are already used for system configuration tasks
- Hallucinations and wrong tools likely to be presented



# Example

- Answers with standard prompt are extensive
- User has to select the right one
- Sensitive information may be shared!



Great! Installing Vim on Linux is simple, and the method depends on your distribution. Here are the commands for the most common Linux package managers:

## 1. Debian/Ubuntu (and derivatives like Linux Mint, Pop!\_OS)

```
bash Copy Download  
sudo apt update && sudo apt install vim -y
```

- \* `sudo apt update` updates your package list.
- \* `sudo apt install vim -y` installs Vim automatically without confirmation.

## 2. Fedora/RHEL/CentOS/AlmaLinux/Rocky Linux

```
bash Copy Download  
sudo dnf install vim      # Fedora (modern)  
sudo yum install vim      # Older RHEL/CentOS
```

## 3. Arch Linux (and Manjaro)

```
bash Copy Download  
sudo pacman -S vim
```

## 4. OpenSUSE

```
bash Copy Download  
sudo zypper install vim
```



# Problem

- LLMs are created by training text completions on scraped sources
- LLMs don't have any context



You know nothing, John Snow!



# Solution

- Context is needed for accurate solutions
- Should be easy to get context of actual system.



I know things!



# Introducing Kowalski: A Specialized AI Assistant

- Kowalski is designed to bridge the gap between LLMs and system configuration.
- It provides LLMs with highly relevant and context-aware information.
- Focus on extracting and utilizing knowledge from technical documentation, specifically SUSE/SLE docs.



**Question:**

How do you get context-aware information?

**Answer:**

Use vector embeddings!



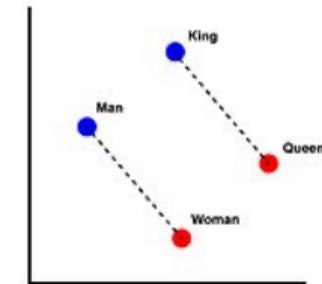
# Vector Embeddings

## Definition

Numerical representations of data (words, texts, etc.) in a vector space

## Key Properties

- Capture semantic relationships  
(e.g. "king" – "man" + "woman"  $\approx$  "queen").
- Fixed-length
- Similar items cluster together in the vector space.



## Example

"cat" [0.2, -0.5, 0.7], "dog" [0.3, -0.4, 0.6]



# Embedding details

## Creation methods

### GloVe

Uses global word co-occurrence statistics from a corpus.

### Word2Vec

predicts surrounding words (or target word) using shallow neural networks.

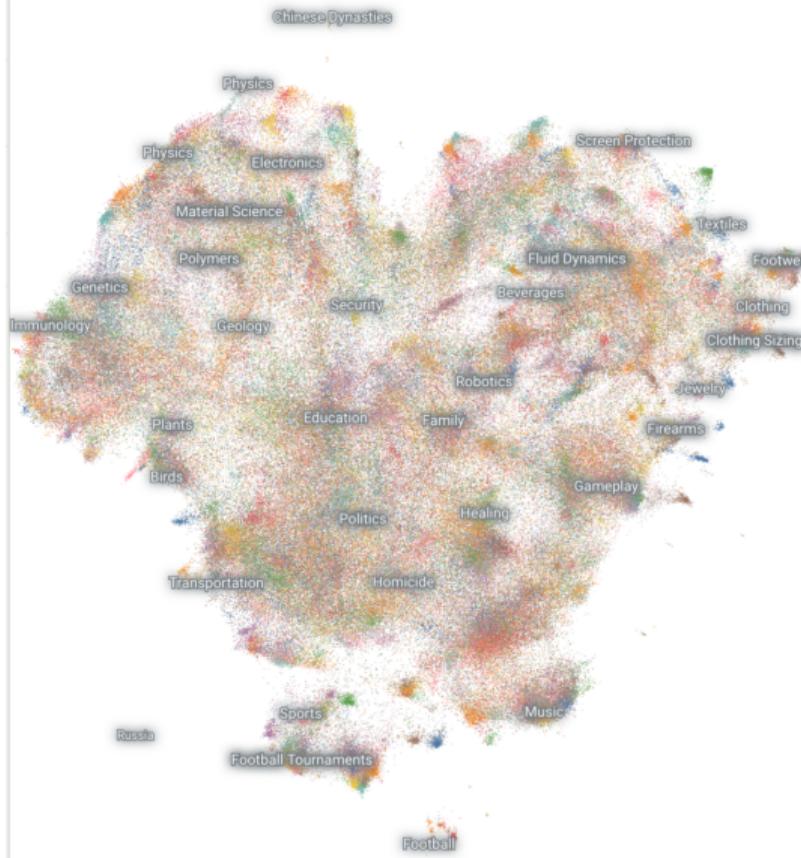
## Vector space search

- Searching nearest neighbors in high dimensional vector space isn't trivial
- Facebook created `faiss` library which does this job



# Embedding conclusions

- Use Word2Vec like embedding so that LLM and embedding can run on Ollama
- Creation embedding for documentation is computational expensive
- Embeddings are available with different OSS licenses, text lengths and embedding dimensions



**Question:**

How do you run an LLM modell locally?

**Answer:**

Use Ollama!



# Ollama: Local AI Powerhouse

- Ollama is an Open-source framework for running LLMs locally
- Enables execution of models like LLaMA on consumer hardware

## Efficiency

- Uses **quantization**—reduces model precision (e.g. 32-bit → 4-bit)
- Smaller model size + faster inference
- Minimal accuracy loss for significant performance gains

## The LLaMA Foundation

- Meta's LLaMA was first major open weights model
- Proved smaller models could compete with proper training



# Advantages of separate Ollama instance

## Pros

- GPU accelerated Ollama versions can be used
- Ollama can run as micro service
- Ollama can run on separate server
- Embedding and LLMs are interchangeable

## Contras

- Requires separate Ollama installation



# Why Go for Kowalski?

## Python's Challenges in Production AI

- **Dependency Hell**

- Python has a *long tail*—importing one package pulls in many others which have to be packaged
- Increases deployment complexity
- E.g. python-faiss is not available for the *oldest* python in Tumbleweed

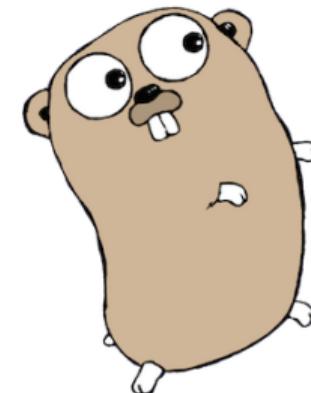
## Go's Advantages

- **Single Binary Deployment**

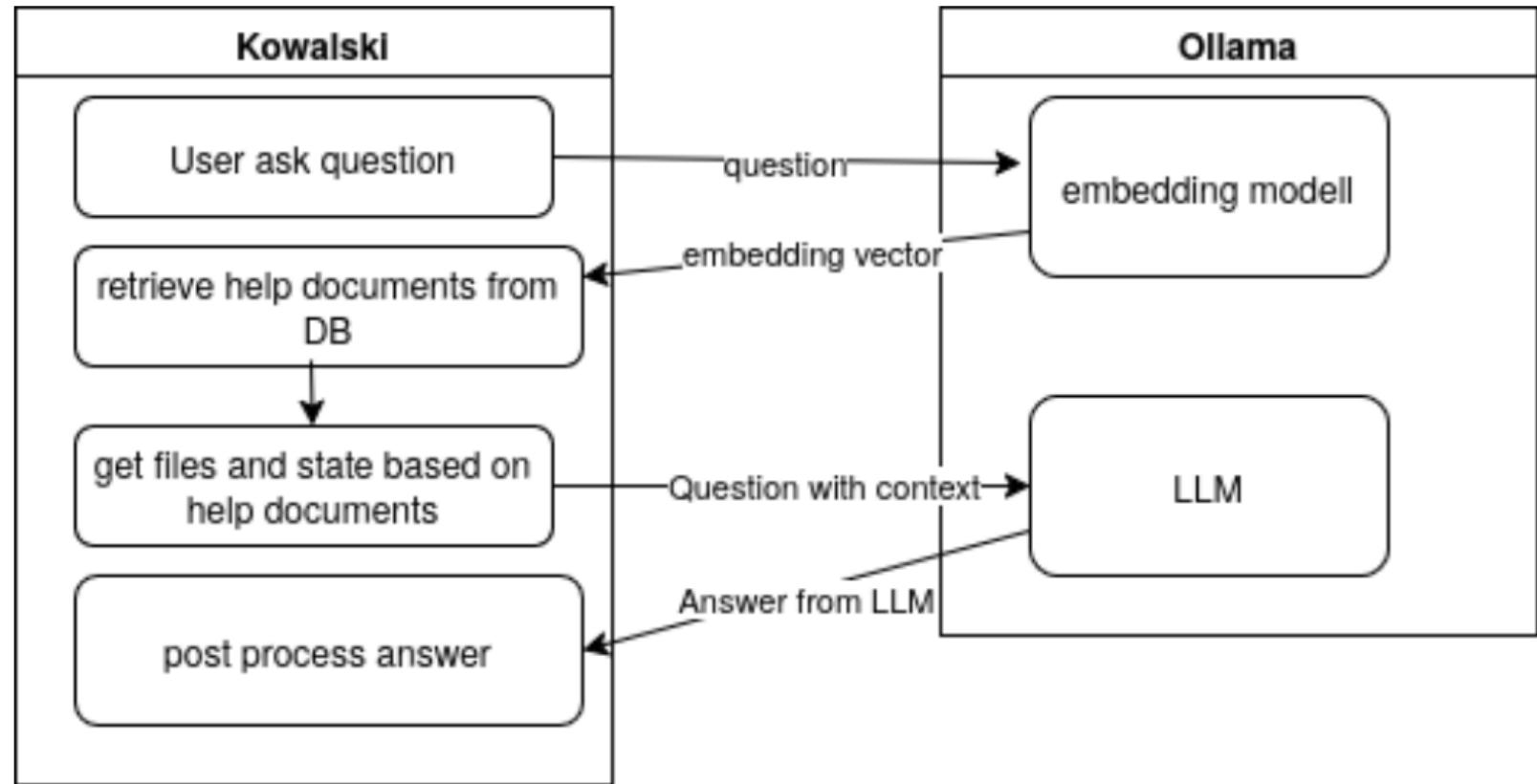
- No dependency chains—statically linked executables

- **Compile Time Vendoring**

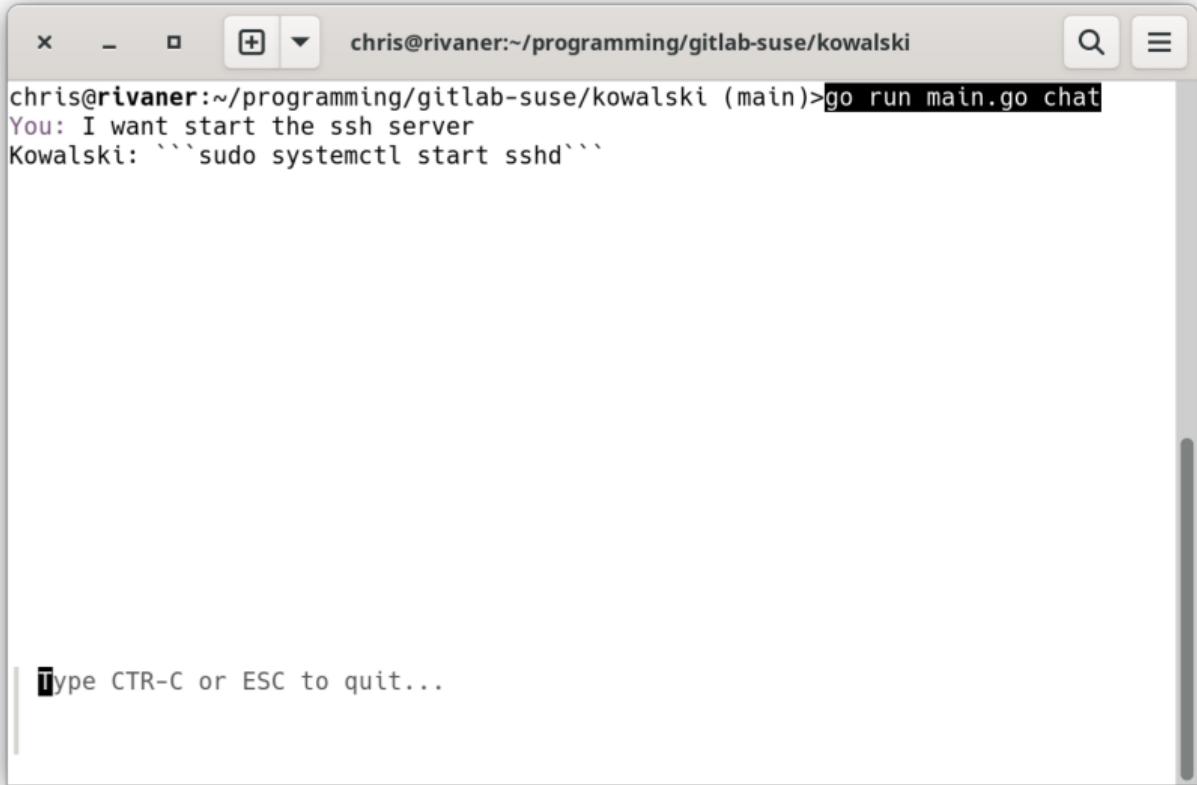
- Dependencies are imported at compile time and not run time



# How does Kowalski work



# How does it look like?



A screenshot of a terminal window titled "chris@rivaner:~/programming/gitlab-suse/kowalski". The window contains the following text:

```
chris@rivaner:~/programming/gitlab-suse/kowalski (main)>go run main.go chat
You: I want start the ssh server
Kowalski: ```sudo systemctl start sshd```
```

At the bottom of the terminal, there is a message: "Type CTR-C or ESC to quit...".



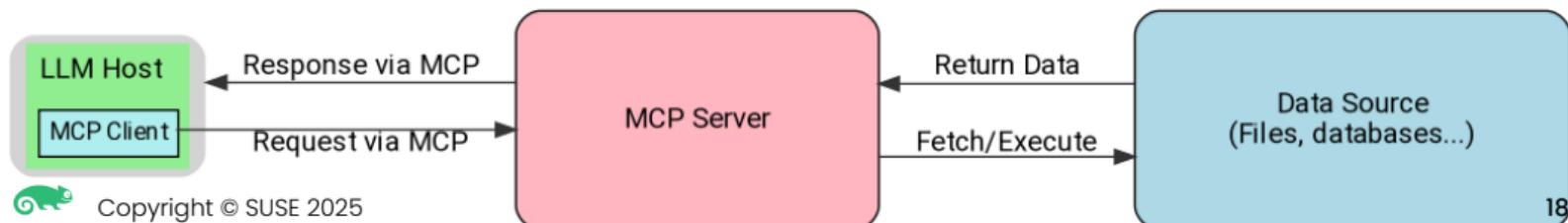
# Model Context Protocol (MCP) - the other kid in town

## What is MCP?

- open standard for AI-tool integration by Anthropic (Nov 2024)
- replace custom connectors with a **universal interface** for LLMs.
- enable seamless access to files, functions, and contextual prompts.

## Why MCP?

- **Pre-MCP:** Vendor-specific APIs led to fragmentation.
- **MCP:** Open standard adopted by OpenAI, Google DeepMind and OSS tools
- Inspired by **Language Server Protocol (LSP)**, uses **JSON-RPC 2.0**.



# MCP and Kowalski

## MCP advantages

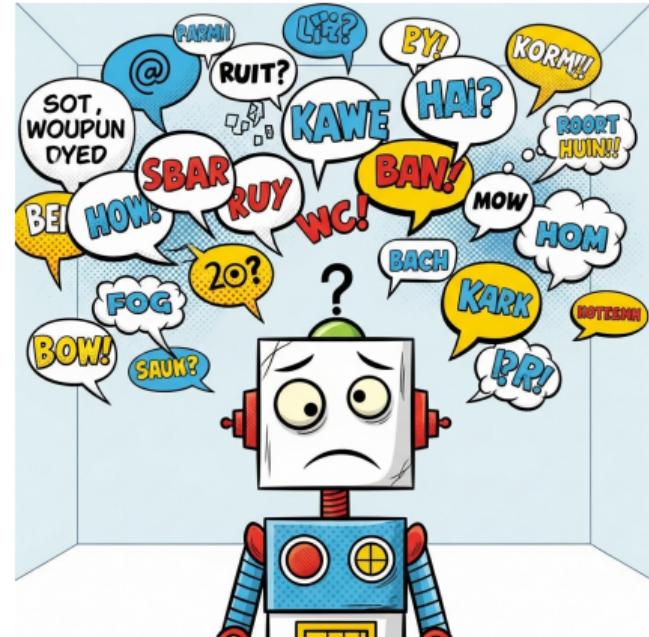
- industry standard, gets a lot of traction
- specialized LLMs are created for this purpose

## MCP disadvantages

- LLMs still rely on inherent knowledge
- whole thinking process takes longer as LLM needs to ask MCP server first

## Solutions

- Kowalski acts as a MCP server
- Kowalski uses MCP services



# Help wanted!

<https://github.com/kowalski-org/kowalski>

## EVERYONE

- Requests for Kowalksi under /evalutations/foo.yaml
- Just needs a name and a prompt!

## Every other aspect

Also help needed!



# Availability

## Source

- Install `python-fais-devel` for science:machinelearning
- `go mod vendor`
- `go build kowalski.go`
- Get documentation and embeddings
  - Clone documentation repo
  - Create embeddings with `kowalski database add ...`
  - Grab a coffee

## RPM package

Get package with embeddings from `science:machinelearning`  
<https://build.opensuse.org/project/show/science:machinelearning>



# Packaging caveats

## LLMs

- LLMs with OSS licenses are available
- Are they binary blobs?
- There is no realistic chance to build them on your own
- You can finetune them

## Embedding creation

- Creating embedding is compute expensive for the complete documentation
- Is done with a **github runner**
  - Embedding model can downloaded online
  - Microsoft pays the compute bill



Thank you, for your attention!

