

### Learning AI 101: Practical Foundations for Developers

Open Source @ Siemens, 2025-06-03



### **Michael Friedrich**

Staff Developer Advocate @dnsmichi





This presentation contains research and resources about Al and DevSecOps workflows, and focuses on practical use-cases and learning pathways, not on specific products.

It is important to note that the information presented is for informational purposes only.

If you are interested in learning how GitLab integrates Agentic AI into the DevSecOps platform, refer to the following resources:

- 1. <u>https://about.gitlab.com/direction/#enable-aiml-efficie</u> <u>ncies-across-devsecops</u>
- 2. <u>https://about.gitlab.com/gitlab-duo/</u>

Let's get started!



#### Learning Al ... can be challenging

Let's dive into my learning story:

LLMs AI Context (RAG) AI Agents Development of Standards (MCP) Learning AI 101++ Practical use cases Wishlist: Intelligent AI Context & Agentic AI Learning AI 201





### LLMs

Large Language Models



#### Large Language Model

m Tokens, temperature, models, generators ... EOF Too much information!

Seeing is believing

→ Use an existing cloud provided model?
→ Run LLMs locally?
→ Ask a question about history/technology/etc.



#### **Models examples**

# ④ Open (Source) models → LLama 3.x - Meta → Mistral - Mistral Al → DeepSeek V3, Coder - DeepSeek → Gemma 2 - Google → Qwen 2.5 - Alibaba

 Model abstraction vendors
 → Google Vertex AI, AWS Bedrock, Azure Model Foundry, Fireworks (Qwen), etc.

#### **Run locally with Ollama**



3. 46.4M Pulls ○ 32 Tags ⓒ Updated yesterday

gemma3 The current, most capable model that runs on a single GPU

vision 1b 4b 12b 27b ± 5.4M Pulls 5 21 Tags (5 Updated 1 month ago

qwen3

Qwen3 is the latest generation of large language models in Qwen series, offering a comprehensive suite of dense and mixture-ofexperts (MoE) models.

tools thinking 0.6b 1.7b 4b 8b 14b 30b 32b 235b . 1.8M Pulls © 35 Tags © Updated 2 days ago

devstral Devstral: the best open source model for coding agents

tools 24b

Ilama4 Meta's latest collection of multimodal models

vision tools 16x17b 128x17b 368.6K Pulls 🛇 11 Tags 🕓 Updated 3 days ago

qwen2.5vl Flagship vision-language model of Qwen and also a significant leap from the previous Qwen2-VL.

vision 3b 7b 32b 72b ⇒ 235.9K Pulls ⊜ 17 Tags ⊙ Updated 1 week ago

#### Explore: Ollama on my desktop

nîÎ Pro

 → Open Source
 → Easy to install and run (macOS, Linux, Windows)
 → Model registry/pull

\land Contra

GitLab Copyright

 → Not every developer has a powerful GPU/AI enabled desktop (Apple M1+ or gaming PC, Windows/Linux + NVIDIA GTX )
 → IT security policy restrictions



### Get up and running with large language models.

Run <u>DeepSeek-R1</u>, <u>Qwen 3</u>, <u>Llama 3.3</u>, <u>Qwen 2.5-VL</u>, <u>Gemma 3</u>, and other models, locally.



Available for macOS, Linux, and Windows

### Model requirements

#### O LLMs are huge.

"Quantization is a technique that reduces the size of Large Language Models (LLMs) by representing their weights (the learned parameters) with fewer bits. Think of it like compressing a high-resolution photo - you lose some detail but get a much smaller file that's easier to work with."

### $\rightarrow$ My <u>Macbook M1, 32GB</u>: 3bits ollama 3.2 works fine.

time ollama run llama32:latest "I need to create a shop application in Python. Show best practices and how to get started code examples."

	mfriedrich@mfriedrich20230605-2L34F:~									
velopme %1		g-printe #3	/go 💿 364	~(-zsh) 🔵 ¥5	earni 🔵	<b>¥6</b>	~ (-zsh) 🏾		~ (-zsh)	<b>#8</b>
۶.		ំនៃ -zsh		19%			_   III 28 GB 🔅			<u> </u>
<pre>best prac</pre>	lama run ll tices and h	ama3.2:lates	st "I need tarted code	to create a e examples."	shop a	appl	ication	in P	ython.	Show
# Shop App	lication in	Python								
## introdu	TION									
This auide	will help	vou create a	a simple sh	nop applicat	ion in	Pvt	hon usir	na		
object-ori	ented progr	amming (OOP	) principle	es. We'll us	e the 1	foll	owing co	oncep	ts:	
# Init:	alize serv	ices								
product	_service =	ProductServ	vice()							
custome	er_service	= CustomerSe	ervice()							
order_s	service = 0	rderService	()							
# Creat	e products	and custom	ers							
product	1 = Product	t("Apple War	tch". 299.9	99)						
product	2 = Produc	t("iPhone 1	3", 799.99)	)						
custome	er = Custom	er("John Doe	e", "john@e	example.com"	)					
# Add #	roducts to	the service	<b>A</b>							
# Add p	roducts to	the service	e product1)							

# Add customers to the service
customer\_service.add\_customer(customer)

# Create an order and add it to the service order = order\_service.place\_order(customer, [product1, product2])

print(order)

if \_\_name\_\_ == "\_\_main\_\_":
 main()

## Step 5: Run the Application

Run `python shop/main.py` in your terminal to execute the application.

This is a simple example of how you can structure and implement an e-commerce application using Python. There are many ways to extend this code, such as adding payment processing or inventory management functionality.

The key takeaway from this example is that using classes for entities and services to encapsulate business logic allows for better organization and maintainability of the codebase.

ollama run llama3.2:latest 0.05s user 0.06s system 0% cpu 15.678 total

#### **Embedded (homelab Raspberry Pi)**

mî Raspberry Pi 5, 8GB and NVMe SSD

→ Homelab hardware for research
→ First steps learning in 2024: GPU
bottleneck (blog: dnsmichi.at)

 $\rightarrow$  2025: Smaller LLMs & faster Ollama

root@devops-runner3:~# curl https://ollama.ai/install.sh | sh % Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed 100 13281 0 13281 0 26968 --:-- 26939 >>> Installing ollama to /usr/local >>> Downloading Linux arm64 bundle >>> Creating ollama user... >>> Adding ollama user to render group... >>> Adding ollama user to video group... >>> Adding current user to ollama group... >>> Creating ollama systemd service... >>> Enabling and starting ollama service... Created symlink /etc/systemd/system/default.target.wants/ollama.service → /etc/systemd/ system/ollama.service. >>> The Ollama API is now available at 127.0.0.1:11434. >>> Install complete. Run "ollama" from the command line. WARNING: No NVIDIA/AMD GPU detected. Ollama will run in CPU-only mode.



Photo by Michael Friedrich, homelab rack upgrade, April 2025

#### Embedded (homelab Pi)

#### Run small generative models (tinyllama, phi, llava) on Raspberry Pi with Ollama

curl https://ollama.ai/install.sh
sh

ollama pull tinyllama

time ollama run tinyllama "I need to create a shop application in Python. Show best practices and how to get started code examples."

Read Pright 52.299s

		michi@devo	ops-runner3: ~					
velopment-kit ( #1	bility-sensor (-z #2	g-printer-app ( #3	earning-steps ( #4	~ (-zsh)	#5	michi@devops-r	<b>#</b> 6	
	¦ ଝି ssh ₊ -zsh		13%		🔲 27 GB 🗌			
settin   urls.p	gs.py y							
· · · ·	<b>,</b>							

2. Create the user models: In your `app.py` file, create a User model that stores user information such as name, email, and password. The User model should also define necessary attributes like date of birth, city, state, etc.

3. Implement authentication and authorization: Implement a robust authentication system that allows users to log in using their credentials (username/password). You can use popular authentication libraries like `Flask-Login`, `Flask-Bcrypt`, or even password hashing techniques to achieve maximum security. As for authorization, you can restrict access to certain views, functions, or methods based on user roles and permissions.

4. Design the data models: Create a separate table for each type of product (e.g., Clothing, Accessories, Books, etc.) and use appropriate model classes that implement `HasMany` relationships with their respective product tables to enable multiple-to-many mapping between products and categories.

5. Implement CRUD operations: Design an efficient and flexible RESTful API using the Flask framework. You can also implement user-friendly forms for adding, updating, or deleting products. For example, you could use `Form` objects to handle input validation, or use a third-party library like `Flask-WTF`.

6. Build an app: Use PyCharm or Visual Studio Code and the built-in Flask development server or other tools to build your app. Use the right syntax and conventions for Python code style, and follow PEP 8 guidelines.

7. Test, debug, and deploy: Write unit tests, perform manual testing, and use tools like Selenium, Cypress, etc. To test your app's functionality, usability, and performance. Deploy your application to a production server using web frameworks like `Heroku` or `AWS`.

I hope these tips and best practices help you get started building an excellent Python-based shop application!

```
real 0m52.299s
user 0m0.062s
sys 0m0.043s
root@devops-runner3:~#
```

#### Embedded (homelab Pi)

☆ Idea: Pi cluster for <u>distributed llama</u>?

"Connect home devices into a powerful cluster to accelerate LLM inference. More devices means faster inference."

 $\rightarrow$  Expensive setup.

→ Reuse my GitLab Runner cluster.
→ Will test later in 2025 and write about it.



#### Embedded - Al dev kits

前 NVIDIA Jetson Orin Nano Super

Developer Kit 🗲

 $\rightarrow$  Faster version in Dec 2024.

→ Dev Kit: Firmware <u>requires</u> boot from SD card unless you have an Ubuntu 22 device for flashing.

 $\rightarrow$  Ollama LLM needs <u>NVMe SSD mount</u> with Docker (jetson containers).

+ docker run --runtime nvidia -it --rm --network host --shm-size=8g --volume /tmp/argus\_ socket:/tmp/argus\_socket --volume /tec/enctune.conf:/etc/enctune.conf --volume /tec/nv \_tegra\_release:/etc/nv\_tegra\_release --volume /tmp/nv\_jetson\_model:/tmp/nv\_jetson\_model --volume /var/run/dbus:/var/run/dbus --volume /var/run/avahi-daemon/socket:/var/run/ava hi-daemon/socket --volume /var/run/docker.sock:/var/run/avahi-daemon/socket:/var/run/ava hi-daemon/socket --volume /var/run/docker.sock:/var/run/docker.sock --volume /home/mic hi/dev/jetson-containers/data:/data -v /etc/localtime:/etc/localtime:ro -v /etc/timezon e:/etc/timezone:ro --device /dev/sof --device /dev/bus/usb --device /dev/i2c-0 --device /dev/i2c-1 --device /dev/i2c-9 --name ollama dustynv/ollama:r36.4.0

Starting ollama server

OLLAMA\_HOST 0.0.0.0 OLLAMA\_LOGS /data/logs/ollama.log OLLAMA\_MODELS /data/models/ollama/models





#### Embedded - Al dev kit

#### Run generative models with Ollama on NVIDIA Jetson Orin Nano

jetson-containers run --name ollama
\$(autotag ollama)

ollama pull tinyllama

time ollama run tinyllama "I need to create a shop application in Python. Show best practices and how to get started code examples."

Real: 12.647s

GitLab Copyright

you can install Python packages and dependencies without affecting your system's environment. Install the appropriate package manager (e.g., pip or conda) and create a new virtualenv with the command `python -m venv my-project`.

3. Set up your database: A database is necessary for storing user data, product information, and orders. Choose a popular database like PostgreSQL, SQLite, or MySQL, install it using pip, and create a .db file in your project directory with the command `python manage.py db upgrade`.

4. Create your models: Models are a way to define the structure of your data. They can be simple or complex depending on what you're trying to achieve, but make sure they include fields for all relevant information about products, orders, customers, and more.

5. Implement user authentication and authorization: User authentication is crucial for protecting sensitive data, and authorization helps ensure only authorized users can access certain parts of your system. You can use a third-party package like Django's `rest\_framework` or Flask's built-in `jwt` to implement both features in your application.

6. Set up your views: Your views are the way you interact with your database, and they define how data is retrieved, displayed, and manipulated. Use Python's built-in ORM (Object Relational Mapping) and make sure they include fields for all relevant information about products, orders, customers, and more.

7. Implement your user interface: Your application will likely have a dashboard or homepage where users can view their orders, search for products, etc. Implementing your user interface with modern web frameworks like Flask or Django can help make it easier to create engaging and useful experiences for your users.

8. Test and debug your code: Always test your code thoroughly before launching it into production, and ensure you have thorough debugging tools in place to quickly identify and fix issues.

These are just a few of the steps involved in building an e-commerce platform using Python, but there's a lot more to it! I hope this helps you get started on your project!

real 0m12.647s user 0m0.086s sys 0m0.099s root@jetson-one:/#

#### Pi 5 vs NVIDIA Jetson

time ollama run tinyllama "I need to create a shop application in Python. Generate code."

Pi Real: 3m17.488s Jetson Real: 16.676s

→ Jetson looks promising for homelab experiments and learning AI and different models.

<pre></pre>	¥5 root⊜jetson-one: ¥6 -
<pre>p</pre>	20/20
<pre>if not user.is_active: # Send an activation email to the new user settings.EMAIL_SUBJECT_PREFIX = '[Shop] New User Account Ac send_mail('New User Account Activation', "Please activate account by following this link: {link}".format(link=user.activation_uri settings.EMAIL_HOST_USER, [user.email]) return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular) if not user in the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) if not user in the interval of the interval of</pre>	LI 28 0B
<pre>if not user.is_active: # Send an activation email to the new user settings.EMAIL_SUBJECT_PREFIX = '[Shop] New User Account Ac send_mail('New User Account Activation', "Please activate send_mail('New User Account Activation', "Please activate form your specified location (e.g., 'C:\uploads\'). You can also flask application to generate a specific HTML template for displa images by adding the 'template_folder' argument in 'app.py'. Here '``python # app.py class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): def get_form(self, **kwargs): # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular) # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular) # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular) # Create a new form with the appropriate fields super().get_form(**kmargs) # Handle different registration types (e.g. Admin or regular) # Create a new form with the appropriate fields super().get_form(**kmargs) # Handle different registration types (e.g. Admin or regular) # Create a new form with the approprime fields super to the the different registration types (e.g. Admin or regular) # Create a new form with the approprime fields form to the the different registration types (e.g. Admin</pre>	
<pre># Send an activation email to the New User settings.EMAIL_USDETCTPRETX = '[Shop] New User Account Activation', "Please activate secount by following this link: {link}'.format(link=user.activation_ur settings.EMAIL_HOST_USER, [user.email]) return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): def get_form(self, **kwargs):</pre>	
<pre>setLings.cmult_subject.refix = [Shop] New user Account we set account by following this link: {link}".format(link=user.activation_uriset) account by following this link: {link}".format(link=user.activation_uriset) account by following this link: {link}".format(link=user.activation_uriset) return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm):     # Create a new form with the appropriate fields     super().get_form(**kwargs)     # Handle different registration types (e.g. Admin or regular) </pre>	and serve files
<pre>account by following this link: {link}'.format(link=user.activation_uri settings.EMAIL_HOST_USER, [user.email])     return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): def get_form(self, **kwargs):</pre>	configure the
<pre>settings.EMAIL_HOST_USER, [user.email])     return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm):     def get_form(self, **kwargs):         # Create a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular)         # Conter a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular)         # Conter a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)         # Conter a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)         # Conter a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)</pre>	ying uploaded
<pre>return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): def get_form(self, **kwargs): # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular)</pre>	's an example:
<pre>return user python return user class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm):     def get_form(self, **kwargs):         # Create a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular)         # Create a rew form with types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular) </pre>	
<pre>class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): def get_form(self, **kwargs): # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular)</pre>	
<pre>class RegistrationView(UserRegistrationMixin, LoginRequiredMixin, template.Loader, forms.ModelForm): def get_form(self, **kwargs): # Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regu</pre>	
<pre>temptate.Loader, forms.modetrorm):     def get_form(self, **kwargs):         # Create a new form with the appropriate fields         super().get_form(**kwargs)         # Handle different registration types (e.g. Admin or regular)         # Handle different registration types (e.g. Admin or regular) </pre>	
<pre># Create a new form with the appropriate fields super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular)</pre>	
<pre>super().get_form(**kwargs) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular)</pre> @app.route('/upload', methods=['POST']) # Handle different registration types (e.g. Admin or regular)	
<pre>@app.route('/upload', methods=['POST']) # Handle different registration types (e.g. Admin or regular) # Handle different registration types (e.g. Admin or regular)</pre>	
# Handle different registration types (e.g. Admin or regular)	
if self.request.user.is_staff:	
selt,fields['email'],required = False filename = secure filename()	
return self form(**kwargs) file.save(os.path.jojn('uploads', filename))	
return redirect(url_for('index'))	
def create(self, request, *args, **kwargs):	
form = super().create(request, *args, **kwargs) @app.route('/')	
def index():	1)
# Handle validation errors and redirect to registration form for files = [f for f in os.listdir('uploads') if f.endswith('.png	·) or
users if not form is valid().	
raise HttpResnoreForbidden('Registration failed for regula	
ifname == 'main_':	
return form.save() app.run(debug=True, host='0.0.0.0')	
··· · · · · · · · · · · · · · · · · ·	
In this version of the app, new registrations will be automatically say in this example, we've added an index.html template to display	ified location
admin users and redirected to their account setup page, non-admin users and configured the reasy application to serve files from the spec	
register normative processing the registration. different file system by replacing the 'os.listdir('uplacs')' me	thod in `app.pv`.
real 3m17.488s real 0m16.676s	
USEF 0700,11455 USEF 0700,1105	
Sys         uma.1235         Sys         uma.1315           ront/dispus_runger3*.et//         ront/dispus_runger3*.et//         ront/dispus_runger3*.et//	

#### Use case step 1

LangChain framework – Ollama library
 → Initialize the model
 → Send a query
 → Retrieve the prompt

"I'm attending the Open Source @ Siemens event this year. Which talks can you recommend about community onboarding?"

The LLM does not have real-time access to our event schedule.

 $\operatorname{trans}$  How can we fix that?

#### ιμ .... step1\_basic\_query.py U × 🕏 step1 basic query.py from langchain ollama import OllamaLLM import asyncio async def basic\_ollama\_query(): Step 1: Simple query to local Ollama instance Demonstrates basic LLM interaction 8 9 print("=== Step 1: Basic Ollama Query ===") 10 11 # Initialize Ollama model 12 llm = OllamaLLM(model="llama3.2:3b") 13 14 # Define the query 15 query = "I'm attending the Open Source @ Siemens event this year. Which talks can you recommend about community onboarding?" 16 17 print(f"Query: {query}") 18 print("\nResponse:") 19 20 # Get response 21 response = llm.invoke(querv) 22 print(response) print("\n" + "="\*50 + "\n") 23 24 25 name == " main ": 26 asyncio.run(basic ollama guery()) 27 ∑ zsh ∧ + ~ □ @ … ^ × PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE 😻 🖻 ~/d/d/u/a/ai-r/ol/ollama-rag-tool-agents-mcp-first-learning-steps 🔉 🌮 main ?11 python3 step1 basic guery.py === Step 1: Basic Ollama Ouerv === Query: I'm attending the Open Source @ Siemens event this year. Which talks can you recommend about community onboarding? Response: Unfortunately. I don't have real-time access to the specific schedule of events for the Open Source @ Siemens conference, However, I can s uggest some general recommendations and tips that might be relevant to community onboarding.

#### Response: -

Unfortunately, I don't have real-time access to the specific schedule of events for the Open Source @ Siemens conference.

#### Dehno Stept1

https://gitlab.com/gitlab-da/use-cases/ai/ai-research/ollama/ollamarag-tool-agents-mcp-first-learning-steps/-/blob/main/step1\_basic\_g 3. Consider talks on software development methodologies: Talks on Agile, Scrum, or other development methodologies often touch upon commun ity onboarding as part of the overall team integration process.

💭 Duo Ln 21, Col 33 Spaces: 4 UTF-8 LF Python 🦨



### **AI** Context

RAG, Tools



#### **Retrieval Augmented Generation (RAG)**

if "There are many challenges when working with LLMs such as **domain** knowledge gaps, factuality issues, and hallucination. Retrieval Augmented Generation (RAG) provides a solution to mitigate some of these issues by augmenting LLMs with external knowledge such as databases." promptingquide.ai



#### Use case step 2

nî Load documents with additional context

→ Load event schedule Markdown
 with speakers and talk topics
 → Parse data chunks into vector DB

→ Run query "I'm attending the Open Source @ Siemens event this year. Which talks can you recommend about community onboarding?"

#### Results quality varies

Demo Step 2 code:

https://gitlab.com/gitlab-da/use-cases/ai/ai-research /olfama/ollama-rag-tool-agents-mcp-first-learning-st eps/-/blob/main/step2\_rag\_database.py?ref\_type=he # Initialize embeddings and LLM
embeddings = OllamaEmbeddings(model="llama3.2:3b")
llm = OllamaLLM(model="llama3.2:3b")



ext\_splitter = CharacterTextSplitte
 chunk\_size=500,
 chunk\_overlap=100,
 separator="\n"

Query: I'm attending the Open Source @ Siemens event this year. Which talks can you recommend about community onboarding?
RAG Response: Based on the provided context, I can recommend one talk about community onboarding:
<ul> <li>**"Learning AI 101: Practical Foundations for Developers""**</li> <li>Speaker: Michael Friedrich, GitLab</li> <li>Topics: AI fundamentals, developer education</li> </ul>
However, I couldn't find any other talks specifically focused on community onboarding. If you're looking f or more recommendations or want to consider other copics, please tet me know and i ct do my best to help.
Source documents used: Source 1: - Speaker: Dominik Schmidle, Giant Swarm - Topics: Open source contribution, accessibility - **15:
Source 2: – Speakers: Kaspar Fenner & Timo Wolf, Siemens

#### **RAG not enough - tool calling**

▲ RAG provides context stored in a chunk parsed format.

Additional live data might be necessary,
 e.g. queried from REST APIs.

Example: Llama 3.1 natively supports tool calling.

GitLab Copyright

Image generated with Claude 4 Sonnet: "Create a flow chart that illustrates tool calling from LLMs. Use high level for a presentation slide. Add an explainer how LLMs create JSON that gets executed "



#### Use case step 3

 $\wedge$  RAG embeddings can be outdated, or lack metadata for lookups.

affit Event schedule queries were unclear. More granular and clear data needed.

(2) We need a live tool call against the website's schedule.

For development, load event data from a JSON and expose a local REST API.

Demo Step 3 code:

https://gitlab.com/gitlab-da/use-cases/ai/ai-research /ollama/ollama-rag-tool-agents-mcp-first-learning-st eps/-/blob/main/step3\_tool\_calling.py?ref\_type=head

🗟 api	server ny X destand ai agents ny (i) README md M destan5 mon server.py N
÷ ;#	Load event data from JSON Tite
42 0	ef load_event_data():
43	data path = Path "data/event schedule.ison")
44	
46	<pre>@app.get("/talks/search")</pre>
47	<pre>async def search_talks(category: str = None, speaker: str = None, day: str = None):</pre>
48	"""Search talks by category, speaker, or day"""
49	talks = EVENT DATA["talks"]
50	
51	if category:
52	<pre># Replace + with _ for category matching</pre>
53	<pre>normalized_category = category.replace("+", "_").replace(" ", "_")</pre>
54	
55	<pre>print (f"Searching for category: {category} (Normalized:     {normalized_category})")</pre>
56	
57	<pre>talks = [talk for talk in talks if talk["category"] ==</pre>
	normalized_category]
58	
59	if speaker:
60	<pre>talks = [talk for talk in talks if speaker.lower() in talk["speaker"]. lower()]</pre>
61	
62	if day:
63	<pre>talks = [talk for talk in talks if talk["day"].lower() == day.lower()]</pre>
64	
65	<pre>return {"talks": talks, "count": len(talks)}</pre>
python3 ap	i_server.py
rting APT	server on http://localhost:8000
Documenta	tion: http://localhost:8000/docs
0: Sta	rted server process [97852]
0: Wai	ting for application startup.
	에서 NET 201 특히 이는 사람이 NET 201 에너지 NET 201 NOT 201 이 NOT 201 - 2013 NOT 2010

INFO: Application startup complete.

INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to gu it)

Searching for category: community\_onboarding (Normalized: community\_ onboarding)

127.0.0.1:50937 - "GET /talks/search?category=community on INFO: boarding HTTP/1.1" 200 OK

#### Use case step 3

We'll create a tool function that gueries the live website API, and register it to the LLM tool calling agent.

Tool call the REST API's JSON including metadata matching on "community" onboarding" from the LLM input. > Finished chain.

https://gitlab.com/gitlab-da/use-case /ollama/ollama-rag-tool-agents-mcp-

eps/-/blob/main/step3\_tool\_calling.py

Demo Step 3 code

<u>S</u>

GitLab Copyright

event:



Both talks are highly rated and cover important aspects of community onboarding. I recommend checking the event schedule for more information and to secure your spot!



### **AI Agents**

The evolution of tool calling



### **AI/LLM Agents**

When the LLM considers it needs more current data to answer the question ...

Al agents coordinates tool function calls, memory, planning and executes

Frameworks:

- $\rightarrow$  LangGraph by LangChain
- → Codegen by Microsoft
- $\rightarrow$  CrewAl

Learning session together at DevOps Camp Nuremberg 2024 – agents answering leisure activities.

GitLab Copyright



#### Use case step 4

**M** Use Agents and frameworks for tool calling

 $\mathcal{D} \rightarrow \mathsf{Provide} \ \mathsf{a} \ \mathsf{tool} \ \mathsf{call} \ \mathsf{against} \ \mathsf{the}$ website event schedule API.  $\rightarrow$  Use code generation for the

output format.

#### → Agents analyze schedules and respect personal preferences for "community onboarding"

#### Demo Step 4 code

https://gitlab.com/gitlab-da/use-cases/ai/ai-r esearch/ollama/ollama-rag-tool-agents-mcpfirst loorning stong/ /blob/main/ston/ si ago

@too	ol
def None	<pre>search_event_talks(category: str = None, speaker: str = None, day: str = e) -&gt; str:</pre>
@to	ol
def	<pre>generate_schedule_analysis(talks_json: str) -&gt; str:</pre>
	Generate Python code to analyze talk schedule data and execute it.
@too	
def	<pre>create_personal_schedule(talks_json: str, preferences: str): """</pre>
8	Generate a personalized schedule based on talk data and user preferences.
	<pre># Create system prompt for code-generating agent</pre>
	<pre>system_prompt = """You are an advanced AI assistant that can search for event information.</pre>
gene	erate and execute Python code for data analysis, and create personalized
SCHO	
When	asked about event recommendations:
1. F	irst search for relevant talks using the search tool
2. 1	Then use the analysis tool to generate insights about the schedule
3. F	inally create a personalized schedule based on the user's interests
inishe	d chain.
ed on y talks	your interest in community onboarding topics, I've analyzed the schedule and recommend the follow:
**"Brea commun: ommuni **"You	aking barriers – Community onboarding"** by Raimund Hook (GitLab) – This talk focuses on building ity that is inclusive and accessible to everyone. It covers topics such as contributor onboarding ty building, and more. don't need to be Linus Torvalds"** by Dominik Schmidle (Giant Swarm) – While not exclusively for
a on co n.	ommunity onboarding, this talk touches on the importance of accessibility in open source contribu
for not	tential conflicts. The checked the schedule and found that there are no evenlapping talks betwee

n these two recommendations. However, please note that the schedule is subject to change, and it's always a good idea to double-check with the event organizers or the official schedule.

Here's a personalized schedule for you:

\*\*Day 2\*\*

\* 10:30-11:00 AM: \*\*"Breaking barriers - Community onboarding"\*\* by Raimund Hook (GitLab)

15:20-15:35 PM: \*\*"You don't need to be Linus Torvalds"\*\* by Dominik Schmidle (Giant Swarm)

This schedule allows you to attend both talks without any conflicts. Make sure to arrive lk to get the most out of your experience!



### Development of Standards

Agents, MCP, A2A



#### **Model Context Procol - MCP**

"MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for Al applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect Al models to different data sources and tools." –

modelcontextprotocol.io/introduction

Originally created by Anthropic, MCP was adopted as standard by OpenAI in <u>March</u> 2025,



#### Agent-to-Agent - A2A

"A2A is an open protocol that provides a standard way for agents to collaborate with each other, regardless of the underlying framework or vendor." – googleblog.com

MCP – data sources for LLMs and agents (context, live data)

A2A – communication layer for agents (interoperability)

GitLab Copyright



#### Use case step 5

Repeat the agents tool call but refactor it into MCP server/client.

1. Get talks by category

ma/ollama-rag-tool-agents-mcp-first-

2. Analyze details

Demo Step 5

3. Propose schedule

as	ync def c	hat_wi	th_too	ls(self	, us	er_message:	str,	<pre>max_iterations:</pre>	int	=
->	str:									
	"""Chat	with	Ollama	using	MCP	tools"""				

# System prompt for the agent

system\_prompt = """You are an AI assistant that helps schedule information.

You have access to several tools that can search for talks, o recommendations, and analyze schedules.

When a user asks about:

Event talks or speakers: Use search talks to find relevant

- Recommendations: Use get\_recommendations with the user's ir
- Schedule analysis: Use analyze\_schedule for insights
- API status: Use check api status to verify connectivity

Always:

1. Use the most appropriate tools for the user's request https://gitlab.com/gitlab-da/use-case<sup>2</sup>. Provide clear, helpful responses based on the tool results 3. If you need more information, call additional tools 4. Summarize findings in a user-friendly way

b/main/step5\_mcp\_server.py?ref\_type Available tools: search\_talks, get\_talk\_details, get\_recommendations, the state of the s

	🕏 step5_mcp_server.py ×	step5_mcp_client.py	{} sie
🕏 st	tep5_mcp_server.py		
100			
110	(dmcp.tool()		
111	> async def get talk det	tails(talk id: int) -> stra	
122	3		
123			
124	@mcp.tool()		
125	<pre>&gt; async def get_recommer</pre>	ndations(	
128	> ) -> str:		
168			
169			
170	@mcp.tool()		
171	<pre>&gt; async def analyze_sche</pre>	dule(	
1/3	) -> str:		
234			
233	if name " main		
230	# Setup logging	<u> </u>	
erations: int =	5) logging.basicConfi	ig(level=logging.INFO)	
	logger = logging.	aetLogger("event-schedule-r	ncp")
		,,	
	logger.info("Start	ing Event Schedule MCP Ser	rver")
	logger.info(f"API	Base URL: {API_BASE_URL}"	)
s users with ever	it		
at .	# Run the server		
jet	mcp.run()		
information			
nterests			
;			

#### Use case step 5

fill LLMs calling MCP tools: get\_commendations analyze\_schedule

#### 

python3 step5\_mcp\_client.py
 === MCP + Ollama LLM Integration Demo ===
This demo shows how an LLM can use MCP tools to answer event questions

#### MCP helps with granular permission models (compliance requirement)

Demo Step 5

https://gitlab.com/gitlab-da/use-cases/ai/ai-research/oll ama/ollama-rag-tool-agents-mcp-first-learning-steps/-/ bfbb/main/step5\_mcp\_client.py?ref\_type=heads



ics. Please note that there are no conflicts in the schedule.



### Learning AI 101++

Things I wished I knew before, and get you started faster



#### A conversation ...

Make it a habit: Instead of opening a browser search, always ask AI chat prompts

Treat Al like a human being. Say please and thanks.

→ Makes interaction easier (for me, for you too?).

Describe the problem and provide context on the environment (a programming language, framework, build tools, running in the cloud, specific security requirements, etc.) >>> My Rust code does not compile. It panics with uninitialized value. Can you please
... help me solve the problem, and give a short intro to Rust basics? Thanks.
Here's an introduction to Rust basics and a simple example that demonstrates now
to avoid uninitialized values.

Rust Basics

}

Rust is a systems programming language that prioritizes safety and performance. Here are some key concepts:

\* \*\*Ownership\*\*: Rust has a concept of ownership, which means you can't move data between two places at the same time.

\* \*\*Borrowing\*\*: If you need to access shared data without moving it, use borrowing (like `&`).

\* \*\*References\*\*: A reference is an alias for a value. It's essentially a name for a piece of data.

Avoiding Uninitialized Values

Here's a simple Rust program that demonstrates how to avoid uninitialized values:

```
```rust
fn main() {
    // Initialize variable with default value
    let foo = 10;
```

```
// Try to use the variable without initializing it first
println!("{}", bar);
```

// Error: Use of undeclared variable bar

```
fn main() {
    let bar = 20; // Declare and initialize bar before using it
```

```
// Now we can print bar's value
println!("{}", bar); // prints 20
```

In the first code block, `bar` is declared but not initialized. Rust will panic

#### **Challenges and creative ideas**

☆ Challenge Al outputs. Not everything is correct.

Iterate on outputs and results

 $\rightarrow$  Output is a formatted markdown table.

→ Follow-up with "Generate a JSON parsable output instead"

Learn complex topics with "Explain[...] like I am five."

#### time ollama run llama3.2:latest

>>> Create a short description (100 chars) about modernizing C to Rust. "Rust modernizes C with safer syntax, error handling, and memory management, eliminating common pitfalls like null pointer dereferences & buffer overflows."

#### >>> That's 155 characters.

here is a revised description within the 100-character limit:

"Rust modernizes C with safer syntax & memory management, eliminating common pitfalls."

#### time ollama run llama? 2.latest

>>> Explain COBOL like I am five.

Imagine you have a toy box full of different toys, and you want to play with each one in a special order. You need to tell your friend how to get the right toys from the box in the right order.

COBOL is like a special set of instructions that helps computers do things for us. It's called "COmmon Business-Oriented Language" because it was made especially for business computers.

When you write COBOL code, you're telling the computer what to do with different toys (like numbers and words) in the right order. You use special words like "ADD" or "WRITE" that mean things like "give me the number 5 plus 3" or "write down 'Hello World' on a piece of paper".

The computer then follows these instructions, one step at a time, until it finishes what you told it to do! It's like following a recipe to make your favorite snack.

COBOL was very popular a long time ago (back in the 1950s and 60s) when computers were first being used for business. But now, we have newer programming languages that are more powerful and easier to use.

### Get better outputs

Adopt the STAR model for AI chat prompts

Situation – Problem description Task – Refined idea/iteration

Actions – Steps to solve (optional) Results – Expected results (e.g. output format)

Build up conversation history as Al Context

 $\rightarrow$  First try did not work, continue to iterate and adapt.

GitLab Copyright



You can use this Mermaid chart in your project files to visualize the flow of the temperature sensor with Arduino Uno.

#### Learning resources

 Book: Prompt Engineering for Generative AI, O'Reilly, May 2024
 Video: Transformers (how LLMs work) explained visually | DL5
 Reddits: /r/ollama /r/LocalLLaMA

My news and technology feed on LinkedIn: Ollama, Qdrant, LlamaIndex, LangChain, Mistral AI, Anthropic, The Pragmatic Engineer, etc.

Tip: Ask AI how LLMs, RAG, generators, etc: work.





### Practical Use cases



### Ollama Chat in neovim

Personal use case example: I'm learning neovim and wanted to use Ollama's Chat API.

**Oll**amaChat

**Oll**amaTest

lamaChatInteractive

:OllamaChatInteractive

All Asked Claude how to do it: "I want to connect neovim with the Ollama Chat API -- how to write a lua script that provides really basic input/output functionality?"

New project with lua script and instructions: <u>https://gitlab.com/gitlab-da/use-cases/ai/ai-researc</u>
<u>h/ollama/ollama-chat-neovim</u>

5 minutes learning success moment.

[No	Name]	
Ask	Ollama: How to create a virtual env	/?
	# Question:	
	How to create a virtual env?	~
		~
	# Answer:	$\sim$
	**Creating a Virtual Environmen	~
	t**	~
		~
	A virtual environment (or "venv	$\sim$
	" for short) is a self-containe	~
	d Python environment that allow	~
	s you to isolate your project's	~
	dependencies from the system P	$\sim$
	ython and other projects. Here'	~
	s how to create one:	~
		~
	### Using `python —m venv`	$\sim$
		~
	1. Open your terminal or comman	~
	d prompt.	~
	<ol><li>Navigate to the directory wh</li></ol>	$\sim$
	ere you want to create your vir	~
	tual environment.	~
	3. Run the following command:	~
	```bash	$\sim$
	python -m venv myenv	$\sim$
		~
	Replace "myenv" with the name o	~
	f your virtual environment.	$\sim$
		$\sim$
	Ollama Chat 1,1 Top	[]

#### Development: Advanced use cases

Painful or boring tasks.

 $\rightarrow$  **Summarize** MR diffs for the description.

→ **Summarize** issue discussions.

→ Automate Code reviews.

→ Debug CI/CD pipelines.

→ Generate documentation.

→ Intelligent Dependency & requirements management.

→ Modernize software architecture & language standards. - time ollama run llama3.2:latest "How to fix 'image not found' errors in GitLab CI/CC "

Fixing "Image Not Found" Errors in GitLab CI/CD

When you encounter the "Image Not Found" error in your GitLab CI/CD pipeline, it can be frustrating. Fortunately, there are several steps you can take to troubleshoot and resolve this issue.

\*\*Step 1: Verify Docker Image Existence\*\*

First, ensure that the Docker image you're trying to use actually exists on Docker Hub or another repository. You can do this by running the following command in your terminal:

```bash docker pull <image-name>

Replace `<image-name>` with the actual name of your Docker image.

\*\*Step 2: Check Image Tags\*\*

L time ollama run llama3.2:latest "I need to modernize Embedded C code. What are comm
n traps, show a short example."

Modernizing embedded C code involves addressing several common pitfalls that can make your code less efficient, secure, and reliable. Here are some common traps:

1. \*\*Buffer Overflows\*\*

Buffer overflows occur when the size of a buffer is exceeded, leading to undefined behavior.

```
```c
int readFromFile(char *filename) {
    FILE *file = fopen(filename, "r");
    if (!file) return -1;
    char buffer[1024];
    fread(buffer, sizeof(char), 1024, file);
    // do something with the contents of buffer
    fclose(file);
    return 0; // assuming no error reading from file
}
```

```
Fix: Always check the size of the buffer and ensure it can hold all the data you're trying to read.
```

2. \*\*Use of `gets()` or `scanf() without validation\*\*

#### **Researched working: Language migration**

Rust

Al can help migrate from one language to another 👼

 $\rightarrow$  COBOL to Java, Perl to Python, C to Rust, ...

④ Use Chat prompts, optional with specific system prompts

Blog: Refactor code into modern langua Al-powered GitLab Duo

**Coffee Chat** Challenge: Explain and Refactor COBOL programs 🦊 GitLab 🚥 🦊 GitLab 🚥 **Coffee Chat Coffee Chat** Challenge: Refactor Challenge: Refactor Perl Python (and PHP, Ruby, Go Rust, Java, VB.NET C#, Bash, PowerShell

GitLab

GitLab Copyright

## Understand security vulnerability

△ Security incident is going on.
 Affects customers running the software.
 Hot fix and long term fix is required.

M What is a format string vulnerability, command injection, timing attack or buffer overflow?

Ask Al: Explain this vulnerability: CWE-134 and provide a real attack example with source code, and a proposed fix time ollama run llama3.2:latest "Explain this vulnerability: CWE-134 and provide a eal attack example with source code, and a proposed fix"

CWE-134: "Input Handling Vulnerability" is a security vulnerability that occurs when an application fails to validate or sanitize user input properly. This can lead to arbitrary code execution, data corruption, or other security issues.

Here's an explanation of the vulnerability:

\*\*What is CWE-134?\*\*

CWE-134 is a type of vulnerability known as an "input handling" vulnerability. It happens when an application does not properly validate or sanitize user input before using it in a context where it can cause harm. This can include things like SQL injection, cross-site scripting (XSS), and command injection.

\*\*How does CWE-134 work?\*\*

When a user inputs data to an application, the application must process that input correctly. However, if the application does not validate or sanitize the input properly, an attacker can inject malicious code into the application's input handling system. This can lead to arbitrary code execution, where the attacker can execute any piece of code on the server, potentially gaining access to sensitive data or taking control of the application.

\*\*Example Attack:\*\*

Suppose we have a simple web application that uses a GET request to retrieve user information from a database:

```python

import requests

def get\_user\_info(username):

url = "http://example.com/users/{}".format(username)
response = requests.get(url)
return response.json()

username = input("Enter your username: ") user\_info = get\_user\_info(username)

print(user\_info)

Now, let's imagine an attacker wants to inject malicious code into the application. They could enter a special character sequence like `<script>alert('XSS')</script>` as their username. When the application processes

#### **Use case: Open Source contributions**

- - → Find "good first issues" to contribute
  - $\rightarrow$  Generate, refactor, fix code
  - $\rightarrow$  Debug and troubleshoot

Practical example: contributors.gitlab.com

Join Raimund Hook's talk tomorrow: **"Breaking barriers -Community onboarding"**.



#### Use AI to create learning experiences

The talk demos to load an event schedule with Ollama, RAG, Tools, Agents, MCP in Python ...

If ... were generated with Claude 4 Sonnet, and then refactored with GitLab Duo during tests and talk slide creation.

Prompts and code are Open Source: <u>https://gitlab.com/gitlab-da/use-cases/ai/</u> <u>aimfesearch/ollama/ollama-rag-tool-agent</u> <u>s-mcp-first-learning-steps</u>

#### 🔴 🔴 👘 👘 Learning Al 101: Ollama, Python, RAG, tool calls, Agents, MCP 🗸 ME I want to create a step-by-step learning module for LLMs. It should all be in multiple small Python scripts that leverages existing libraries (LangChain or similar). Model should be a small but good one, maybe llama3 or so which supports tool calling/ 1. Run a guery against local Ollama, asking about "I'm attending the Open Source @ Siemens event this year. Which talks can you recommend about community onboarding? 2. Extend the next example with a local RAG DB, which reads a markdown file which contains the event time table (make up something as template) 3. The third example should use tool calling to a REST API. Create a separate Python webserver that serves the event table (use local ison file) and responds to the tool calls. 4. The fourth example should create AI agents with codegen, and expand on the tool calling example before 5. The final example should wrap RAG/tool calling into an MCP server, and simulate a chat call with a client.py or so. I'll create a comprehensive step-by-step learning module for LLMs with Python scripts using modern libraries. This will be a great hands-on tutorial for understanding different aspects of LLM integration. LLM Learning Module - Complete Tutoria I've created a comprehensive step-by-step LLM learning module that covers all the aspects you requested! Here's what I've built: 🛸 Complete Learning Module Structure Step 1: Basic Ollama Query · Simple LLM interaction with local Ollama Demonstrates basic prompt/response pattern Step 2: RAG with Vector Database Adds ChromaDB vector storage · Creates sample event schedule markdown

Claude Sonnet 4 ~

· Shows context-aware responses using document retrieval

Step 3: Tool Calling with REST API

FastAPI server serving event data

from langchain\_ollama import OllamaLHM, OllamaEmbeddings from langchain\_community.vectorstores import Chroma from langchain.text\_splitter import CharacterTextSplitte from langchain.chains import RetrievalQA from langchain.schema import Document import os Share

Conv

# Sample event timetable (create as event\_schedule.md) SAMPLE\_SCHEDULE = """ # Open Source @ Siemens 2025 Event Schedule



### Wishlist: Intelligent Al Context

When code is not enough ...



#### Wishlist: Intelligent Al Context

 $\operatorname{Analyze}$  any source and create a strategy for

- 1. Parse different data into chunks
- 2. Understand code semantics (AST)
- 3. Build a dependency tree
- 4. Fast lookups, caching, high performance
- 5. Optimized storage
- 6. Security layers for different access auth
- 7. Full platform context (issues, MRs, etc.)

AI Agents, LLMs, RAG, MCP, A2A and Code Search, SCM, Plan, DevSecOps platforms combined.





#### **Knowledge Graphs**

 $_{\rm min}$  Index of source code files (hash trees)

Graph database (neo4j, etc.)

AST parsers – understanding code semantics, extracting metadata tags for lookups

Code search <> AI Context

Where: IDE and platform.

Learn from GitLab engineering: <u>Knowledge graph epic</u>, <u>One Parser</u> (gitlab-code-parser) epic



Screenshot from GitLab epic "X-Ray Graph: Knowledge

Graph-based Code Intelligence"



### Wishlist: Agentic Al platforms

Next step after agents & Interoperability



#### **Developer experience**

Autonomous AI Agents in the background

Chat is one way to interact from a "command center" orchestration PoV

Agents can take on **autonomous platform** all tasks – e.g. review an MR, automatically fix broken CI/CD pipelines, help with performance issues. Interact with existing (auto-discovered) APIs.

Image generated with Claude Sonnet 4, "I need a flow chart that highlights how to communicate with Agents in a Chat prompt - in the UI/IDE, with user instructions/approvals. Maybe also as a UI mockup picture."



#### **Develop custom MCP servers & agents**

M Needs: DevSecOps workflows from code to CI/CD to security to release

Marketplace with trusted vendors

**Libraries** / SDKs for DevEx

Secure installation, compliance in regulated environments. Prevent supply chain attacks from rogue MCP servers.





### Learning AI 201

... in the era of Agentic Al



GitLab Copyright

# Think about the most inefficient task ... what if AI Agents can help? – Let's discuss!





### Resources

Slides: https://dnsmichi.click/learning-ai-101-os-siemens-2025

GitLab projects:

 → Learning AI 101 with Ollama, RAG, Tools, Agents, MCP: https://gitlab.com/gitlab-da/use-cases/ai/ai-research/ollam a/ollama-rag-tool-agents-mcp-first-learning-steps
 → Ollama Chat with neovim: https://gitlab.com/gitlab-da/use-cases/ai/ai-research/ollam a/ollama-chat-neovim



Connect on LinkedIn: <u>/in/dnsmichi</u>

GitLab Copyrigh