

Multi LLM LangGraph Agent with Function Calling, Context from Images, and Image Generation using Automatic1111



Multi LLM LangGraph Agent by Karim Lalani

Overview

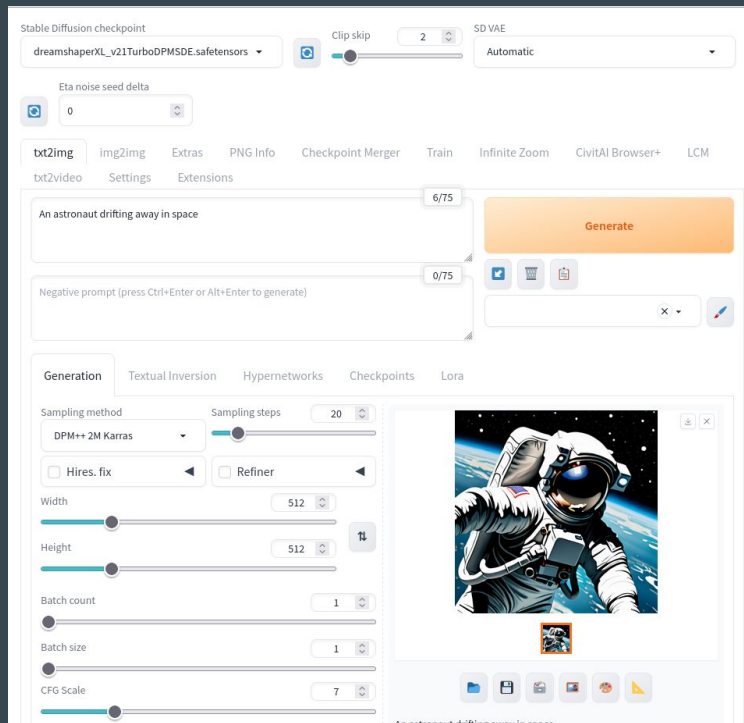
1. Local LLMs using Ollama
2. Multiple LLMs
 - a. Image description using Bakllava
 - b. Image generation using Automatic 1111
 - c. Question Answering using Llama2
3. Function calling
 - a. With custom finetuned LLM
4. Custom tools
 - a. Image generation
 - b. Image description (LLM as a tool)

Stable Diffusion and Automatic 1111

Stable Diffusion is a generative artificial intelligence (generative AI) model that produces unique photorealistic images from text and image prompts.

Automatic 1111 is a web interface for Stable Diffusion, implemented using Gradio library.

<https://github.com/AUTOMATIC1111/stable-diffusion-webui>



Function Calling using Local LLM

#####

user_prompt

[INST]SYSTEM: You are a helpful assistant with access to the following functions. Use them if required -

```
{
  "name": "calculate_tip",
  "description": "Calculate the tip amount",
  "parameters": {
    "type": "object",
    "properties": {
      "bill_amount": {
        "type": "number",
        "description": "The total bill amount"
      },
      "tip_percentage": {
        "type": "number",
        "description": "The percentage of tip to be given"
      }
    }
  },
  "required": [
    "bill_amount",
    "tip_percentage"
  ]
}
```

USER: Hi, I need help with calculating a tip. My bill amount is \$50 and I want to give a 15% tip.[/INST]

Function Calling

Expectation...

```
FUNCTION: {"name": "calculate_tip", "arguments": '{"bill_amount": 50, "tip_percentage": 15}'}  
FUNCTION RESPONSE: {"tip_amount": 7.5}  
ASSISTANT: The tip amount for a bill of $50 with a 15% tip rate is $7.5.  
USER: Great, thanks for the help!  
ASSISTANT: You're welcome! If you have any other questions, feel free to ask.  
</s>
```

Reality !!!

To calculate the tip amount using the provided function, follow these steps:

1. Ensure the function "calculate_tip" is available in your current environment.
2. Create an object with required keys "bill_amount" and "tip_percentage" with suitable values.
3. Call the "calculate_tip" function, passing the object as the argument.

Example code:

```
```javascript  
const tip = {
 bill_amount: 50,
 tip_percentage: 0.15
}
```

# Custom Tools - txt2image

● ● ● labs/LangChain\_105/ollama-llava-auto1111/graph.py

```
1 # txt2image tool
2 @tool("txt2image", args_schema=Txt2ImageInput)
3 def txt2image(prompt: str, **kwargs) -> Dict:
4 (
5 "An image generation tool that takes in a prompt as string "
6 "and returns a json response with images encoded in base64 string. "
7 "The prompt is transformed from simple English "
8 "to a comma separate MidJourney image generation prompt."
9)
10
11 config = Config(prompt=prompt, **kwargs)
12 if config.seed is None:
13 config.seed = int(random.normal(scale=2**32))
14 response = requests.post(a_1111_base_url + "/sdapi/v1/txt2img",
15 json=config.dict()).json()
16 return response
```

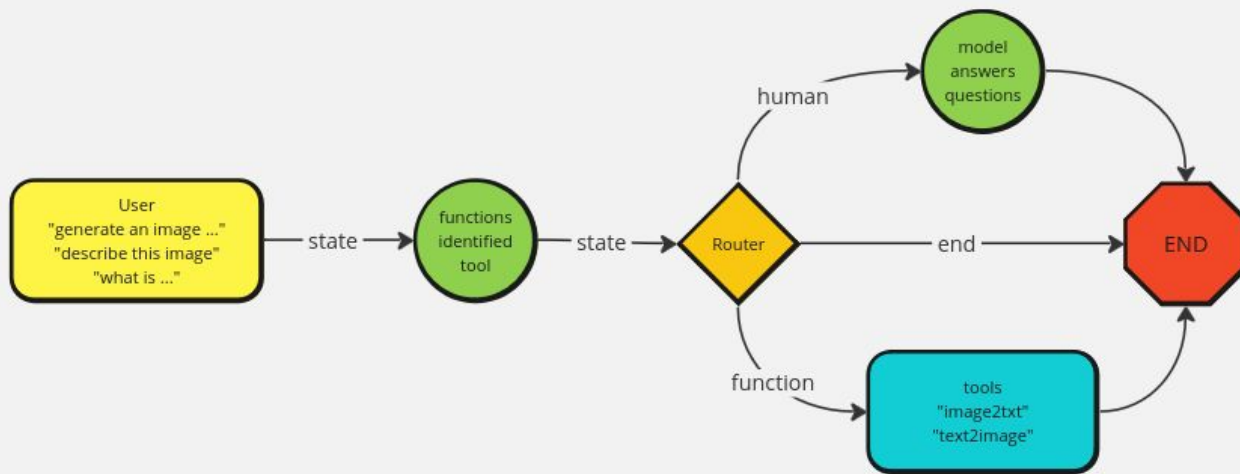
# Custom Tools - image2txt

```
labs/LangChain_105/ollama-llava-auto1111/graph.py

1 # image2txt tool
2 @tool("image2txt", args_schema=Image2TxtInput)
3 def image2txt(prompt: str, image: str) -> str:
4 (
5 "An image description tool that takes "
6 "in a question about an image or a picture as a prompt "
7 "and returns the answer as string"
8)
9
10 no_image_error = (
11 "No image available within context. "
12 "Upload an image or generate using prompt to describe it."
13)
14
15 try:
16 if image is None or len(image) == 0:
17 return no_image_error
18 _ = base64.b64decode(image)
19 except Exception:
20 return no_image_error
21
22 bound = image_llm.bind(images=[image])
23 response: str = bound.invoke(prompt)
24 return response.strip()
```

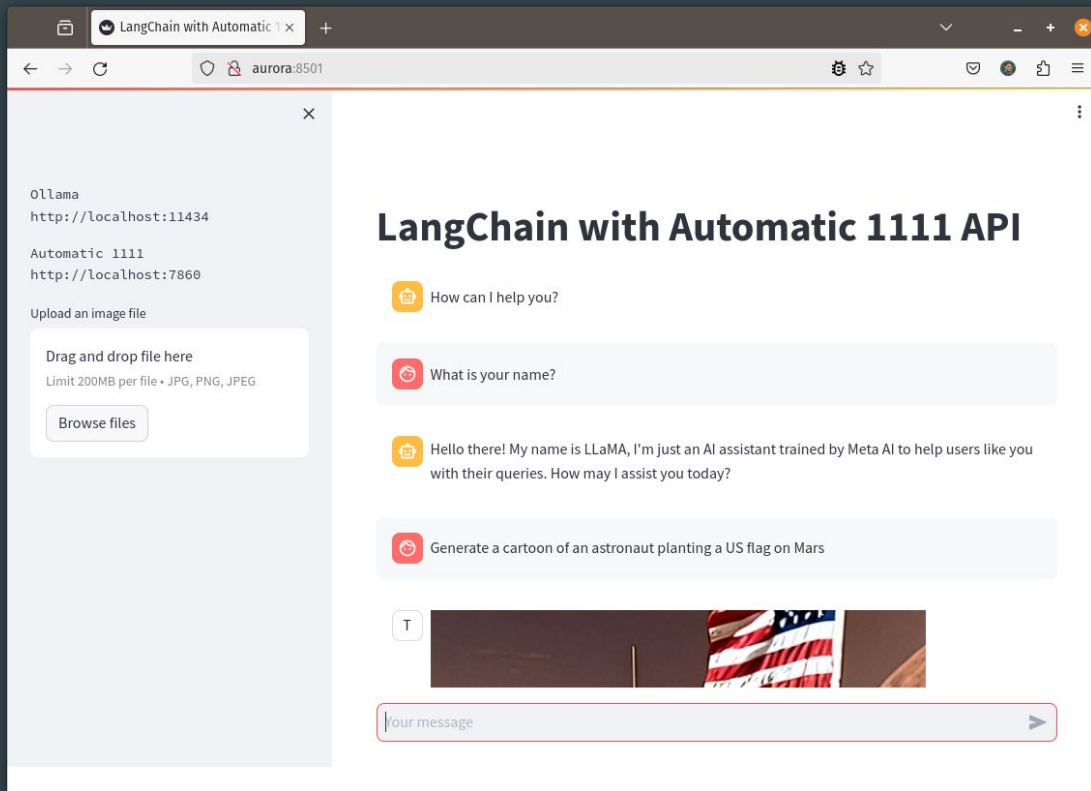
# Agent Graph

## Multi LLM LangGraph Agent

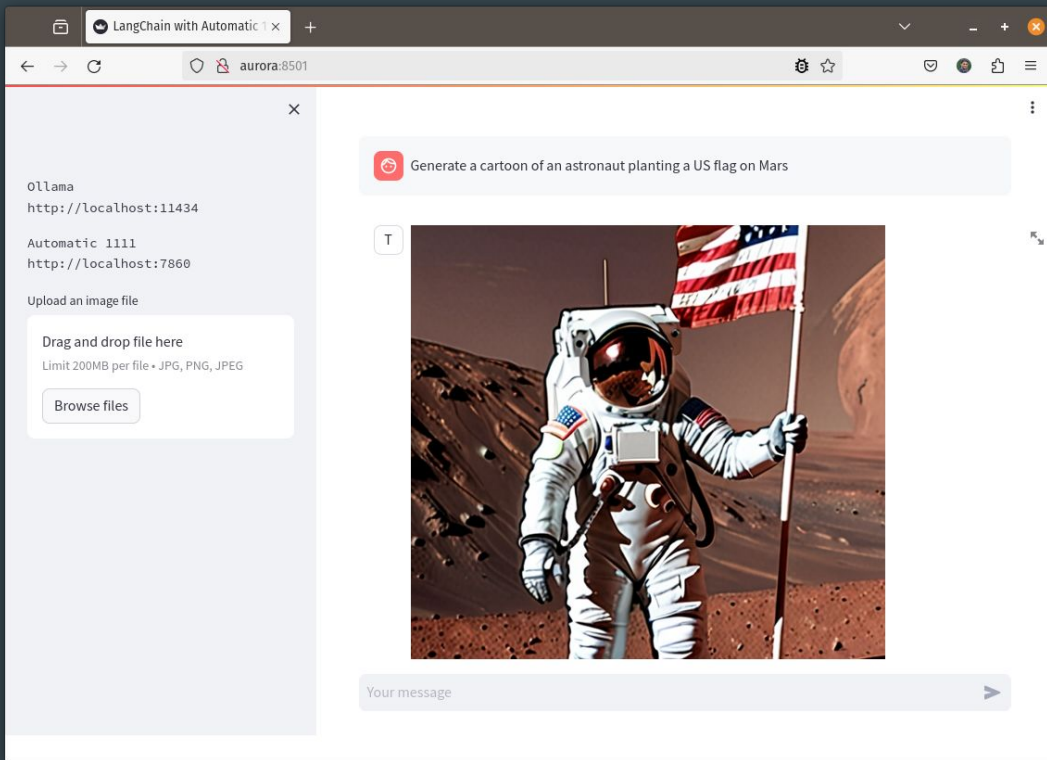




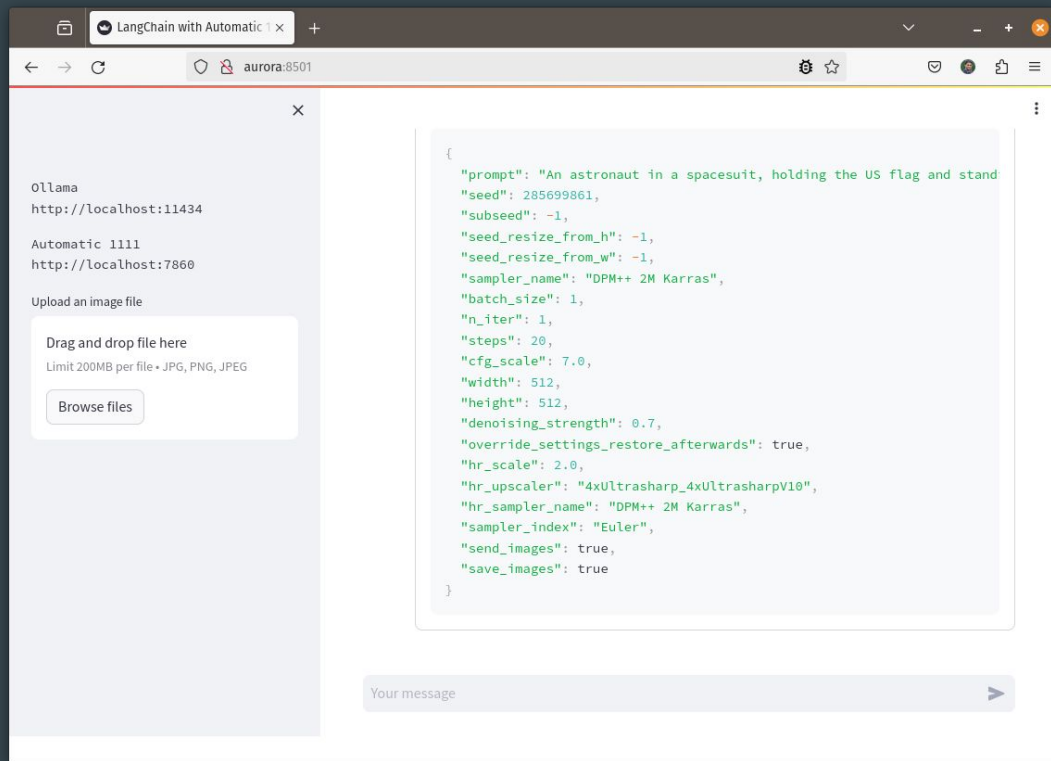
# Agent in Action



# Agent in Action



# Agent in Action



# Resources

1. Ollama - <https://ollama.ai>
2. Automatic1111 - <https://github.com/AUTOMATIC1111/stable-diffusion-webui>
3. Ollama model for Mistral 7b Finetuned for Function Calling - <https://ollama.com/klcoder/mistral-7b-functioncall>
4. Function calling Prompt Templated hosted on LangChain hub - <https://smith.langchain.com/hub/klcoder/mistral-functioncalling>