# **Swarms External Agent Integration**

Integrating external agents from other frameworks like **Langchain**, **Griptape**, and more is straightforward using **Swarms**. Below are step-by-step guides on how to bring these agents into Swarms by creating a new class, implementing the required methods, and ensuring compatibility.

## **Quick Overview**

- **Step 1**: Create a new class that inherits the `Agent` class from Swarms.
- **Step 2**: Override the `.run(task: str) -> str` method that will execute the agent and return a string response.
- **Step 3**: Optionally, add methods to save outputs to other formats like JSON, logs, or databases.

### **Agent Class**

The primary structure you'll need to integrate any external agent is the `Agent` class from **Swarms**. Heres a template for how your new agent class should be structured:

```python
from swarms import Agent
```

```
class ExternalAgent(Agent):
    def run(self, task: str) -> str:
        # Implement logic to run external agent
        pass

    def save_to_json(self, output: str, filepath: str):
        # Optionally save the result to a JSON file
        with open(filepath, "w") as file:
            json.dump({"response": output}, file)
```

## **Griptape Agent Integration Example**

In this example, we will create a **Griptape** agent by inheriting from the Swarms `Agent` class and implementing the `run` method.

### **Griptape Integration Steps**:

1. **Inherit from Swarms Agent**: Inherit from the `SwarmsAgent` class.
2. **Create Griptape Agent**: Initialize the **Griptape** agent inside your class and provide it with the necessary tools.
3. **Override the `run()` method**: Implement logic to process a task string and execute the Griptape agent.

## **Griptape Example Code**:

```python
from swarms import (
    Agent as SwarmsAgent,
)  # Import the base Agent class from Swarms
from griptape.structures import Agent as GriptapeAgent
from griptape.tools import (
    WebScraperTool,
    FileManagerTool,
    PromptSummaryTool,
)


# Create a custom agent class that inherits from SwarmsAgent
class GriptapeSwarmsAgent(SwarmsAgent):
    def __init__(self, *args, **kwargs):
        # Initialize the Griptape agent with its tools
        self.agent = GriptapeAgent(
            input="Load {{ args[0] }}, summarize it, and store it in a file called {{ args[1] }}.",
            tools=[
                WebScraperTool(off_prompt=True),
                PromptSummaryTool(off_prompt=True),
                FileManagerTool(),
            ],
            *args,
            **kwargs,
        )


    # Override the run method to take a task and execute it using the Griptape agent
    def run(self, task: str) -> str:
        # Extract URL and filename from task
        url, filename = task.split(",")  # Example task string: "https://example.com, output.txt"
        # Execute the Griptape agent
        result = self.agent.run(url.strip(), filename.strip())
        # Return the final result as a string
        return str(result)



# Example usage:
griptape_swarms_agent = GriptapeSwarmsAgent()
output = griptape_swarms_agent.run("https://griptape.ai, griptape.txt")
print(output)
```

### **Explanation**:

1. **GriptapeSwarmsAgent**: The custom class that integrates **Griptape** into **Swarms**.
2. **run(task: str)**: This method extracts inputs from the task string and runs the agent using **Griptape** tools.
3. **Tools**: The **Griptape** agent is equipped with web scraping, summarization, and file management tools.

## **Additional Features**:

You can enhance your external agents with additional features such as:

- **Saving outputs** to JSON, databases, or logs.

- **Handling errors** and retry mechanisms for robustness.

- **Custom logging** with tools like **Loguru** for extensive debugging.

## **Langchain Agent Integration Example**

Next, we demonstrate how to integrate a **Langchain** agent with **Swarms** by following similar steps.

### **Langchain Integration Steps**:

1. **Inherit from Swarms Agent**: Inherit from the `SwarmsAgent` class.
2. **Create Langchain Agent**: Initialize a Langchain agent with the necessary components (like language models or memory modules).
3. **Override the `run()` method**: Pass tasks to the Langchain agent and return the response.

## **Langchain Example Code**:

```python
```

```
from swarms import Agent as SwarmsAgent
from langchain import LLMChain
from langchain.llms import OpenAI
from langchain.prompts import PromptTemplate


# Create a custom agent class that inherits from SwarmsAgent
class LangchainSwarmsAgent(SwarmsAgent):
    def __init__(self, *args, **kwargs):
        # Initialize the Langchain agent with LLM and prompt
        prompt_template = PromptTemplate(template="Answer the question: {question}")
        llm = OpenAI(model="gpt-3.5-turbo")
        self.chain = LLMChain(llm=llm, prompt=prompt_template)
        super().__init__(*args, **kwargs)


    # Override the run method to take a task and execute it using the Langchain agent
    def run(self, task: str) -> str:
        # Pass the task to the Langchain agent
        result = self.chain.run({"question": task})
        # Return the final result as a string
        return result


# Example usage:
langchain_swarms_agent = LangchainSwarmsAgent()
output = langchain_swarms_agent.run("What is the capital of France?")
print(output)
```

### **Explanation**:

1. **LangchainSwarmsAgent**: The custom class integrates **Langchain** into **Swarms**.

2. **run(task: str)**: The task is passed to a language model via Langchain and returns a result.

### Additional Examples from other providers

### 1. **OpenAI Function Calling Agents**

- **Description**: OpenAI models like GPT-4 can now call functions programmatically. This makes it possible to create agents that execute external functions, APIs, or code snippets.

## Example Integration:

```python
from swarms import Agent as SwarmsAgent
import openai

# Custom OpenAI Function Calling Agent
class OpenAIFunctionAgent(SwarmsAgent):
    def __init__(self, *args, **kwargs):
        # Initialize OpenAI API credentials and settings
        self.api_key = "your_openai_api_key"
        super().__init__(*args, **kwargs)

    def run(self, task: str) -> str:
        # Example task: "summarize, 'Provide a short summary of this text...'"
        command, input_text = task.split(", ")
        response = openai.Completion.create(
            model="gpt-4",
            prompt=f"{command}: {input_text}",
            temperature=0.5,
            max_tokens=100,
        )
        return response.choices[0].text.strip()

# Example usage:
openai_agent = OpenAIFunctionAgent()
output = openai_agent.run("summarize, Provide a short summary of this text...")
print(output)
```

### 2. **Rasa Agents**

- **Description**: **Rasa** is a popular open-source framework for building conversational AI agents. You can integrate **Rasa** to build dialogue-based agents with **Swarms**.

## Example Integration:

```python
from swarms import Agent as SwarmsAgent
from rasa.core.agent import Agent as RasaAgent
from rasa.core.interpreter import RasaNLUInterpreter
```

```
# Custom Rasa Swarms Agent
class RasaSwarmsAgent(SwarmsAgent):
    def __init__(self, model_path: str, *args, **kwargs):
        # Initialize the Rasa agent with a pre-trained model
        self.agent = RasaAgent.load(model_path)
        super().__init__(*args, **kwargs)

    def run(self, task: str) -> str:
        # Pass user input to the Rasa agent
        result = self.agent.handle_text(task)
        # Return the final response from the agent
        return result[0]["text"] if result else "No response."

# Example usage:
rasa_swarms_agent = RasaSwarmsAgent("path/to/rasa_model")
output = rasa_swarms_agent.run("Hello, how can I get a refund?")
print(output)
```

### 3. **Hugging Face Transformers**

- **Description**: **Hugging Face** offers a variety of pre-trained models, including transformers for NLP tasks. These can be easily integrated into **Swarms** for various tasks like text generation, question answering, and more.

## Example Integration:

```python
```

```
from swarms import Agent as SwarmsAgent
from transformers import pipeline


# Custom Hugging Face Agent
class HuggingFaceSwarmsAgent(SwarmsAgent):
    def __init__(self, model_name: str, *args, **kwargs):
        # Initialize a pre-trained pipeline from Hugging Face
        self.pipeline = pipeline("text-generation", model=model_name)
        super().__init__(*args, **kwargs)


    def run(self, task: str) -> str:
        # Generate text based on the task input
        result = self.pipeline(task, max_length=50)
        return result[0]["generated_text"]


# Example usage:
hf_swarms_agent = HuggingFaceSwarmsAgent("gpt2")
output = hf_swarms_agent.run("Once upon a time in a land far, far away...")
print(output)
```

### 4. **AutoGPT or BabyAGI**

- **Description**: **AutoGPT** and **BabyAGI** are agent frameworks designed to be autonomous, where agents can recursively execute tasks and create new tasks based on previous outputs.

## Example Integration:

```python
from swarms import Agent as SwarmsAgent
from autogpt import AutoGPT

# Custom AutoGPT Agent
class AutoGPTSwarmsAgent(SwarmsAgent):
    def __init__(self, config, *args, **kwargs):
        # Initialize AutoGPT with configuration
        self.agent = AutoGPT(config)
        super().__init__(*args, **kwargs)

    def run(self, task: str) -> str:
        # Execute task recursively using AutoGPT
        result = self.agent.run(task)
        return result

# Example usage:
autogpt_swarms_agent = AutoGPTSwarmsAgent({"goal": "Solve world hunger"})
output = autogpt_swarms_agent.run("Develop a plan to solve world hunger.")
print(output)
```

### 5. **DialogFlow Agents**

- **Description**: **DialogFlow** by Google is used to build conversational agents. These agents can process user intents and deliver responses based on predefined conversation flows.

## Example Integration:

```python
from swarms import Agent as SwarmsAgent
from google.cloud import dialogflow


# Custom DialogFlow Agent
class DialogFlowSwarmsAgent(SwarmsAgent):
    def __init__(self, project_id: str, session_id: str, *args, **kwargs):
        # Initialize DialogFlow session client
        self.session_client = dialogflow.SessionsClient()
        self.project_id = project_id
        self.session_id = session_id
        super().__init__(*args, **kwargs)


    def run(self, task: str) -> str:
        session = self.session_client.session_path(self.project_id, self.session_id)
        text_input = dialogflow.TextInput(text=task, language_code="en-US")
        query_input = dialogflow.QueryInput(text=text_input)
        response = self.session_client.detect_intent(
            request={"session": session, "query_input": query_input}
        )
        return response.query_result.fulfillment_text


# Example usage:
dialogflow_swarms_agent = DialogFlowSwarmsAgent("your_project_id", "your_session_id")
output = dialogflow_swarms_agent.run("Book me a flight to Paris.")
```

```
print(output)
```

### 6. **ChatterBot Agents**

- **Description**: **ChatterBot** is a Python-based machine-learning conversational agent. It learns from previous conversations to generate intelligent responses.

## Example Integration:

```python
from swarms import Agent as SwarmsAgent
from chatterbot import ChatBot

# Custom ChatterBot Agent
class ChatterBotSwarmsAgent(SwarmsAgent):
    def __init__(self, name: str, *args, **kwargs):
        # Initialize ChatterBot
        self.agent = ChatBot(name)
        super().__init__(*args, **kwargs)

    def run(self, task: str) -> str:
        # Get a response from ChatterBot based on user input
        response = self.agent.get_response(task)
        return str(response)

# Example usage:
chatterbot_swarms_agent = ChatterBotSwarmsAgent("Assistant")
```

  output = chatterbot_swarms_agent.run("What is the capital of Italy?")
  print(output)

### 7. **Custom APIs as Agents**

- **Description**: You can create agents that integrate with any REST or GraphQL API by defining them as a task runner within Swarms. This allows for interaction with third-party services.

  ## Example Integration:
  ```python
  from swarms import Agent as SwarmsAgent
  import requests

  # Custom API Agent
  class APIAgent(SwarmsAgent):
      def run(self, task: str) -> str:
          # Parse task for API endpoint and parameters
          endpoint, params = task.split(", ")
          response = requests.get(endpoint, params={"q": params})
          return response.text

  # Example usage:
  api_swarms_agent = APIAgent()
  output = api_swarms_agent.run("https://api.example.com/search, python")
  print(output)
  ```

### **Summary of Integrations**:

- **Griptape**: Integrate with tools for web scraping, summarization, etc.

- **Langchain**: Use powerful language model orchestration.

- **OpenAI Function Calling**: Directly run OpenAI API-based agents.

- **Rasa**: Build and integrate conversational agents.

- **Hugging Face**: Leverage transformer models.

- **AutoGPT/BabyAGI**: Recursive, autonomous task execution.

- **DialogFlow**: Integrate conversational flows for voice/chat-based systems.

- **ChatterBot**: Machine-learning conversational agents.

- **Custom APIs**: Leverage external APIs as agents for custom workflows.

## **Conclusion**:

By following the steps outlined above, you can seamlessly integrate external agent frameworks like **Griptape** and **Langchain** into **Swarms**. This makes Swarms a highly versatile platform for orchestrating various agentic workflows and leveraging the unique capabilities of different frameworks.

For more examples and use cases, please refer to the official Swarms documentation site.