Design, Develop, and Deploy Multi-Agent Systems with CrewAI

Working with AI Agents

DeepLearning.AI

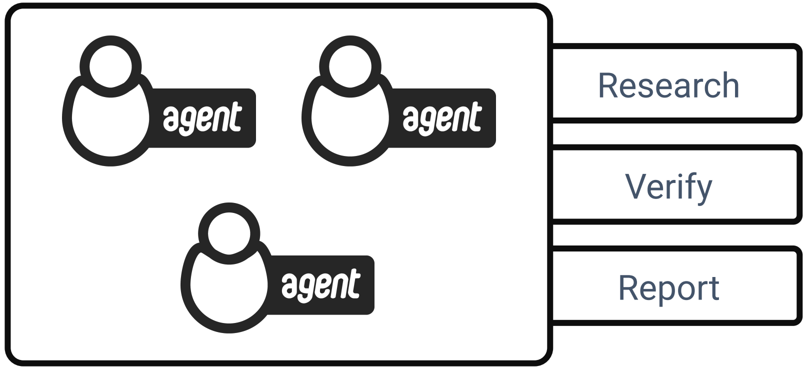
Working with AI Agents

Understanding AI Agent Workflows

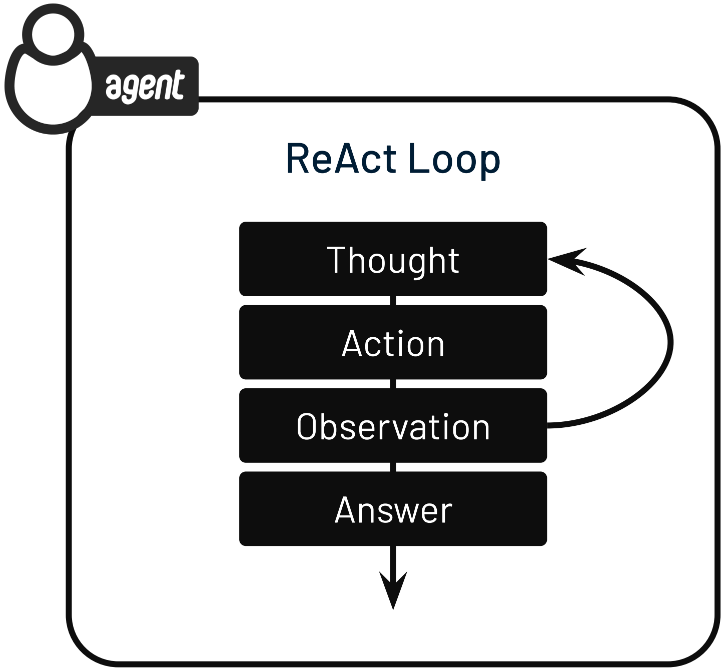
crewai

DeepLearning.AI

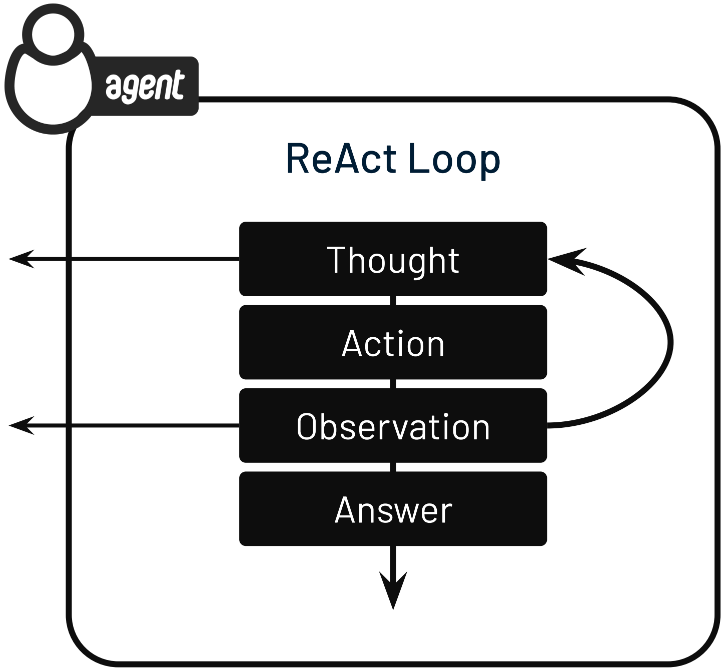
Agent Workflows



Agent Workflows



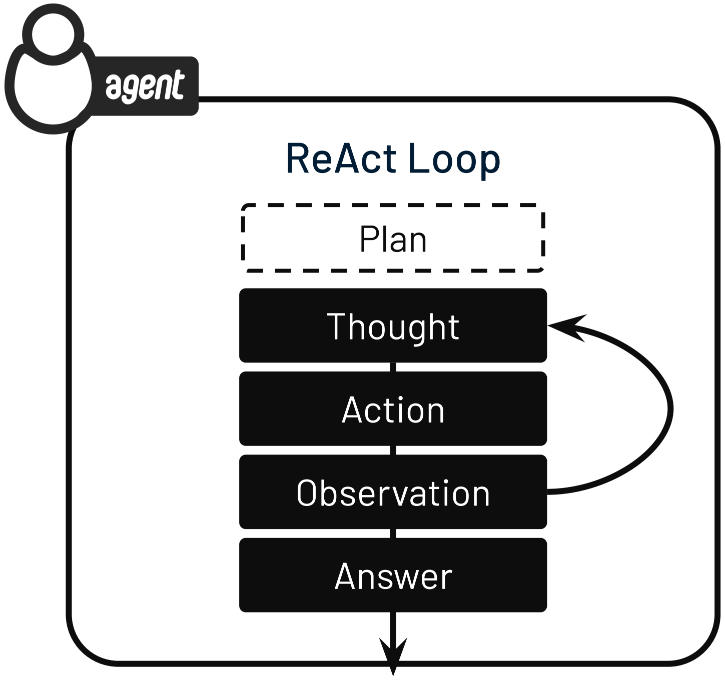
Agent Workflows



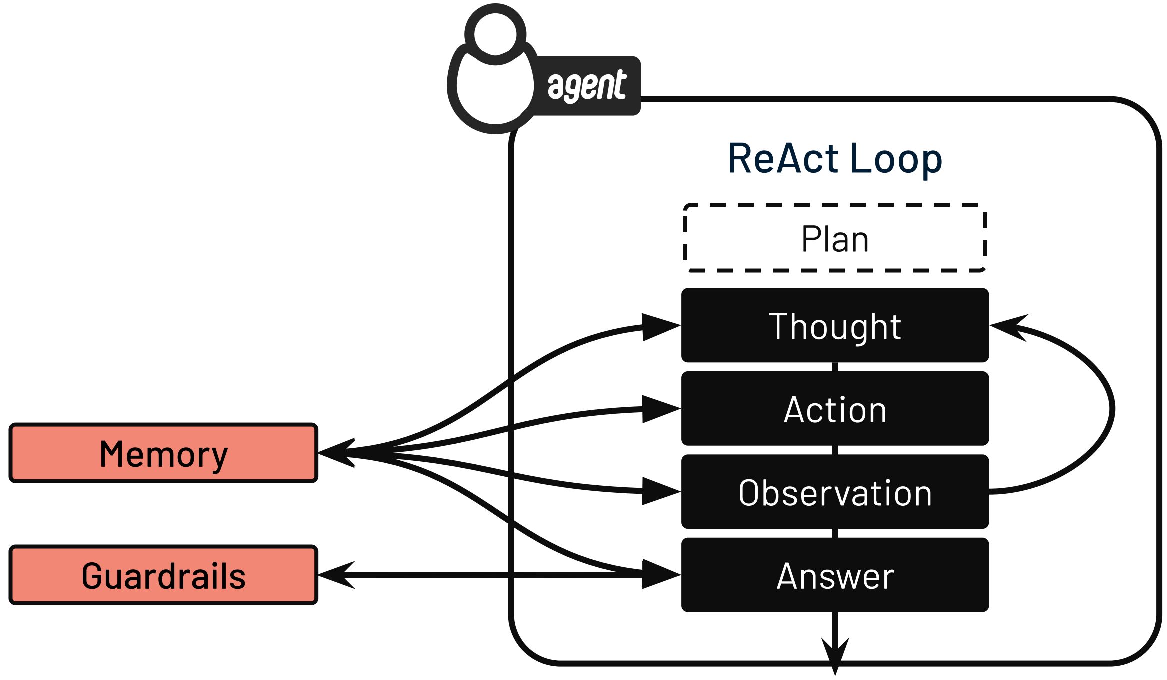
"In order to this I'll use the following tool"

"I'll need more information, let me dig more"

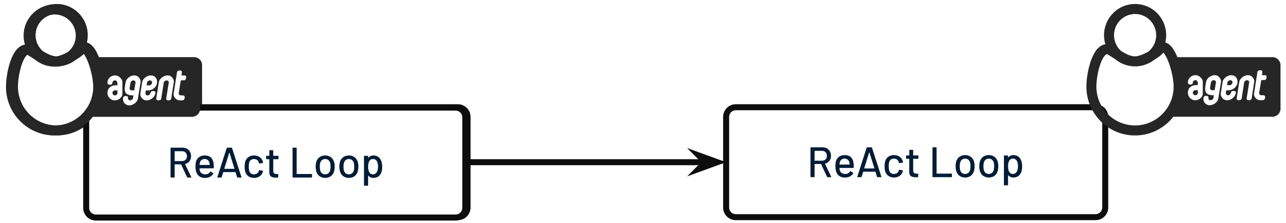
Agent Workflows



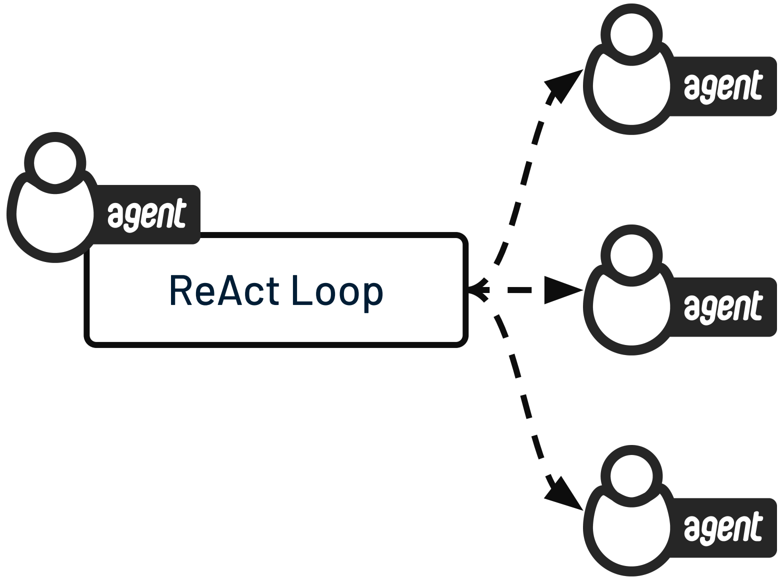
Agent Workflows



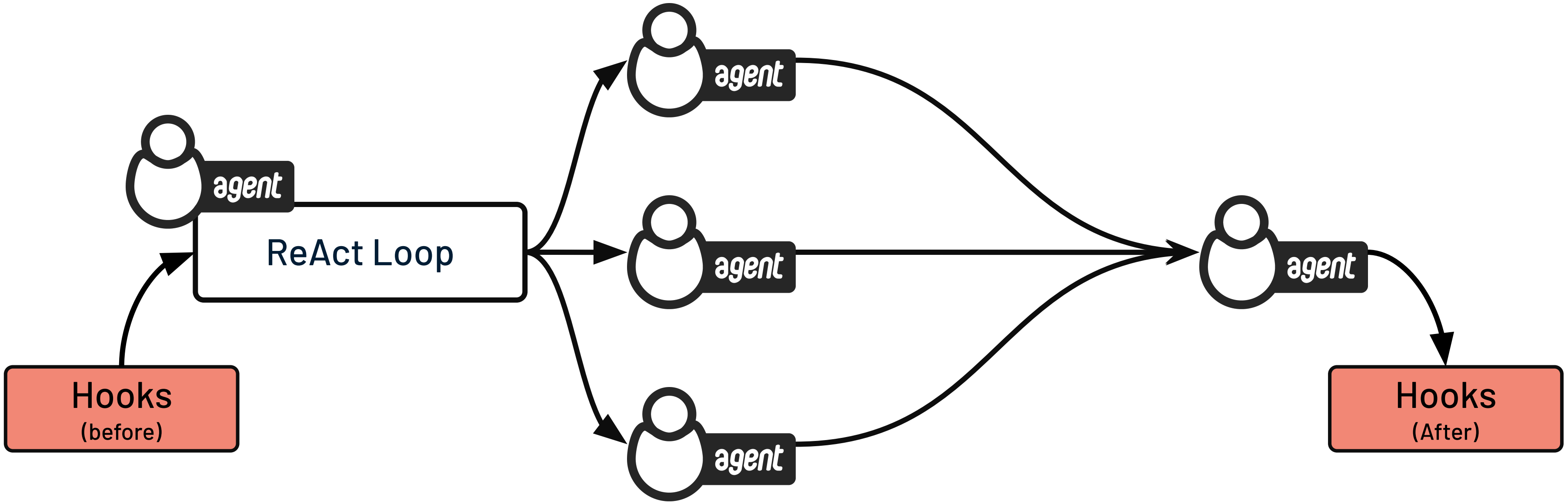
Agent Workflows



Agent Workflows



Agent Workflows



Controlling Agents

How to provide deterministic controls on probabilistic systems?

Memory	Guardrails	Hooks
Dynamically update context to help agents learn and get better over time	Adding either deterministic or probabilistic (LLM as judge) checks on output	Execute deterministic code either before or after Agents

Working with AI Agents

Incorporating Memory and Knowledge

crewai

DeepLearning.AI

Agentic Memory

How to provide agents the ability to **remember** information?

Short Term	Long Term	Entity
Stores data from past executions to add context that gets shared among agents	Reflects on differences between expect outputs and actual outputs on tasks to improve agents through feedback	Collects facts about recognizable people, companies, locations, products, etc

Entity Memory

Pretend news article (by ChatGPT)

Renewable energy corporations such as Tesla, led by CEO Elon Musk, and Orsted, based in Denmark, are at the forefront of the transition to sustainable energy. Meanwhile, tech giants like Google, under the leadership of Sundar Pichai, and Microsoft, with CEO Satya Nadella at the helm, are investing heavily in renewable energy projects and carbon offset programs to achieve carbon neutrality. Notably, countries like Germany, with its ambitious Energiewende policy, and China, the world's largest emitter of greenhouse gases, are implementing aggressive strategies to reduce emissions and accelerate the adoption of renewable energy. Additionally, startups like Beyond Meat, founded by Ethan Brown, and Impossible Foods, led by CEO Pat Brown, are disrupting the food industry with plant-based alternatives, significantly reducing the environmental impact of food production.

Entities

- persons,
- organizations,
- locations

DeepLearning.AI

crewai

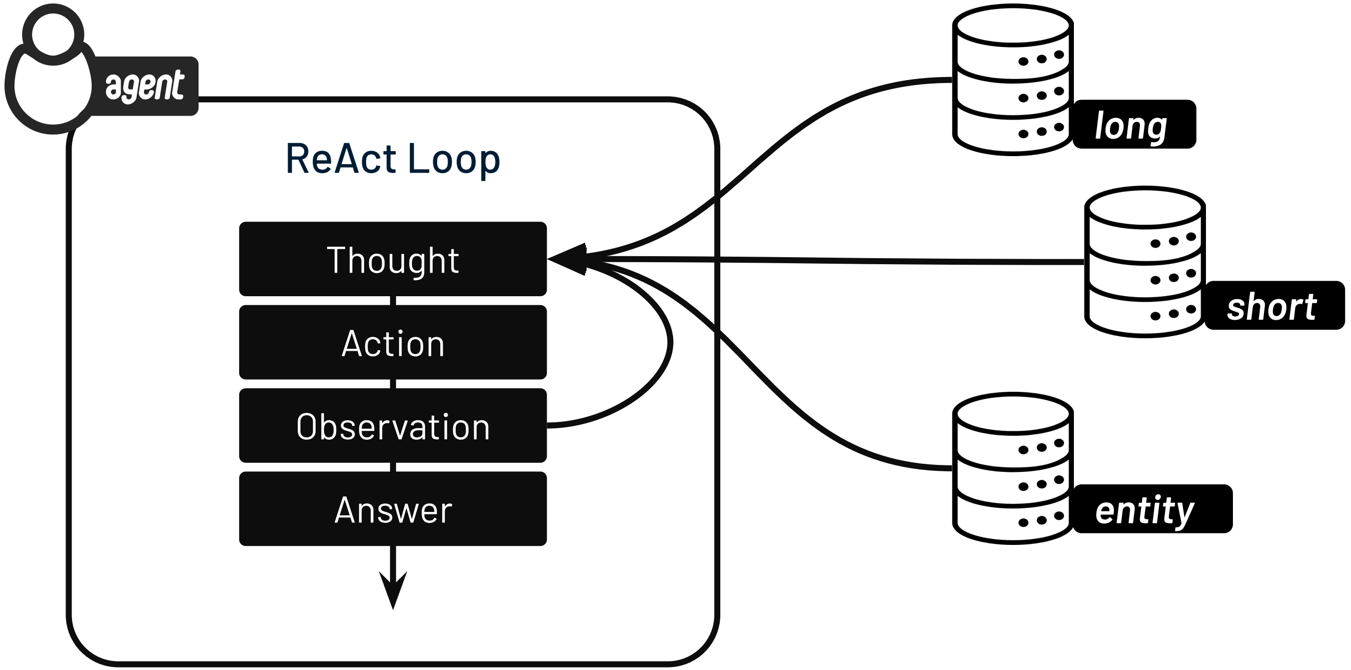
Adding Memory to Crews

```
planning_crew = Crew(
    agents=[code_explorer, documentation_planner],
    tasks=[analyze_codebase, create_documentation_plan],
    memory=True,
    verbose=False
)

planning_crew.kickoff()
```

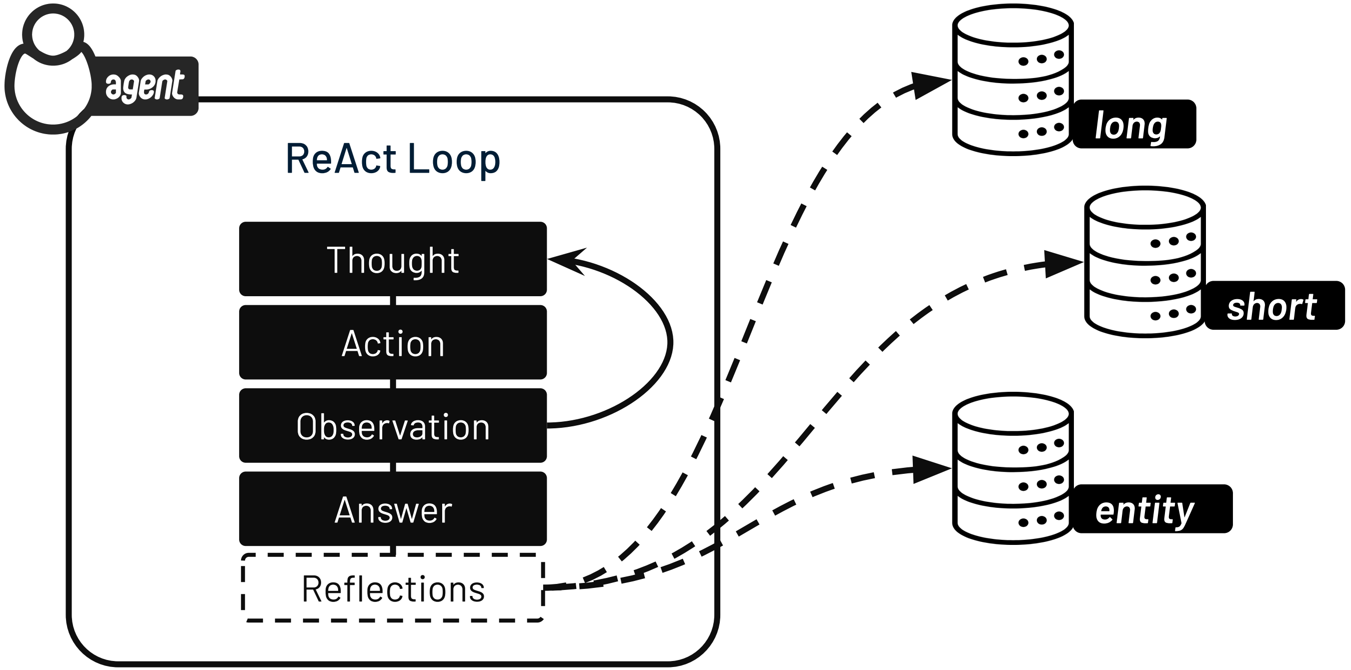
DeepLearning.AI

crewai

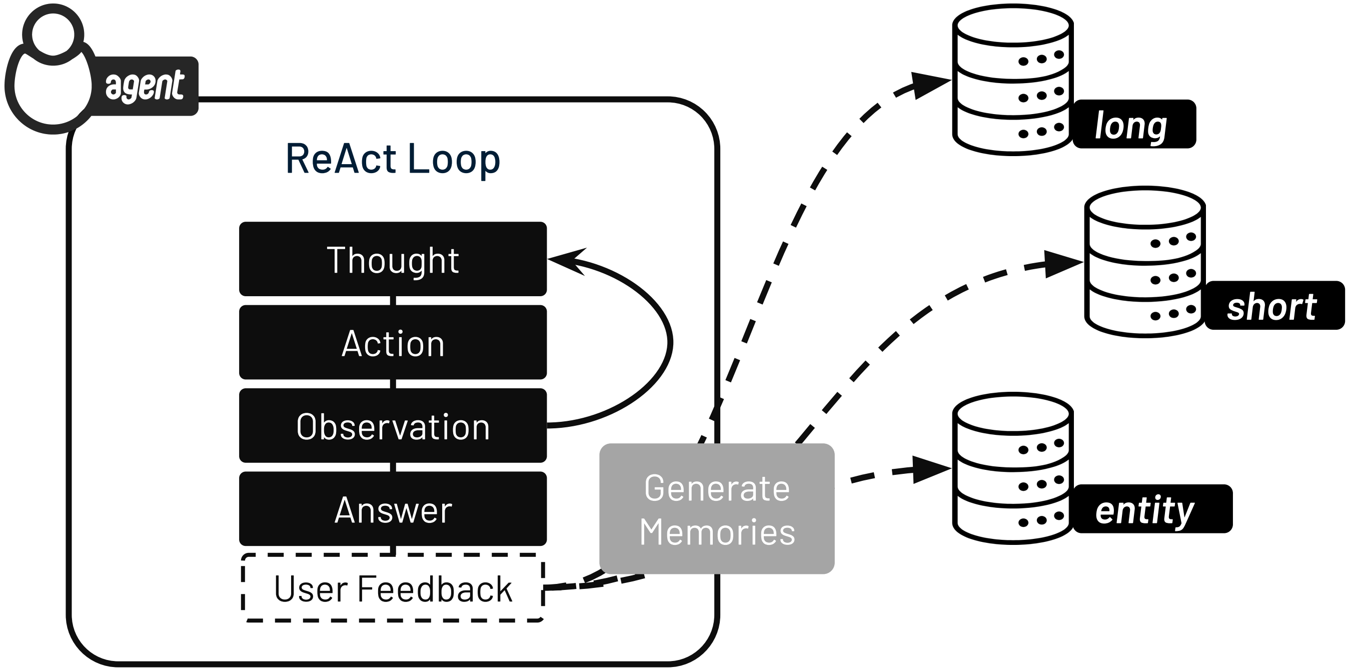
Agentic Memory



Agentic Memory



Agentic Memory



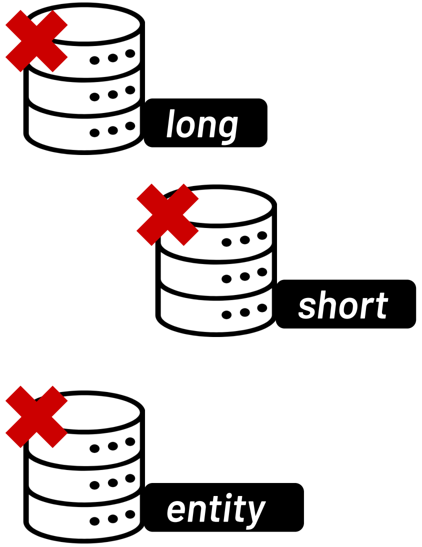
Agentic Memory

Generate Memories

Training with Feedback

Removing Data from Memory

```
C:>
crewai reset-memories
```



Removing Data from Memory

```python
from crewai import Crew


crew = Crew(agents=[...], tasks=[...], memory=True)


# Reset specific memory types
crew.reset_memories(command_type='short')     # Short-term memory
crew.reset_memories(command_type='long')      # Long-term memory
crew.reset_memories(command_type='entity')    # Entity memory
```

Agentic Memory

Whether you generate memories or train with feedback, the additional context can measurably impact agent performance

Helpful for controlling...

- Behavior
- Format
- Style

Agentic Memory vs Agentic Knowledge

Memory

- Internal information adapted from previous executions
- Selectively added to agent's context during current execution
- Updated from feedback by human users or LLM-as-a-Judge

Knowledge

- External information retrieved from different sources
- Selectively added to agent's context from flat files or vector databases
- Not updated from feedback. Pre-filled at run-time.

Agentic Knowledge

Text Sources

- Raw Text
- Text Files
- PDF Documents

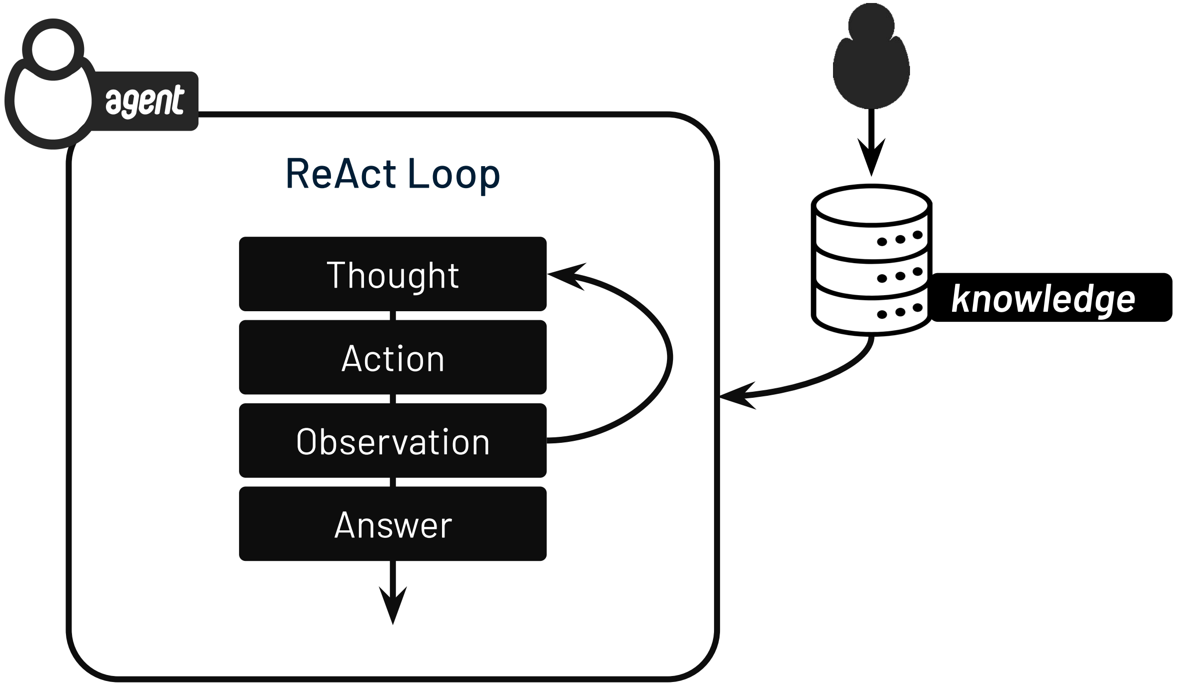
Structured Data

- CSV Files
- Excel Spreadsheets
- JSON Documents

```python
from crewai import Agent, Task
from crewai.knowledge.source.string_knowledge_source import StringKnowledgeSource

content = "User’s name is John. He is 30 years old and lives in San Francisco."
string_source = StringKnowledgeSource(content=content)

agent = Agent(
   role="About User",
   goal="You know everything about the user.",
   backstory="You are a master at understanding people and their preferences.",
   knowledge_sources=[string_source],
)
```

```python
from crewai import Agent, Task
from crewai.knowledge.source.string_knowledge_source import StringKnowledgeSource

content = "User’s name is John. He is 30 years old and lives in San Francisco."
string_source = StringKnowledgeSource(content=content)

agent = Agent(
    role="About User",
    goal="You know everything about the user.",
    backstory="You are a master at understanding people and their preferences.",
    knowledge_sources=[string_source],)
```

Agentic Knowledge



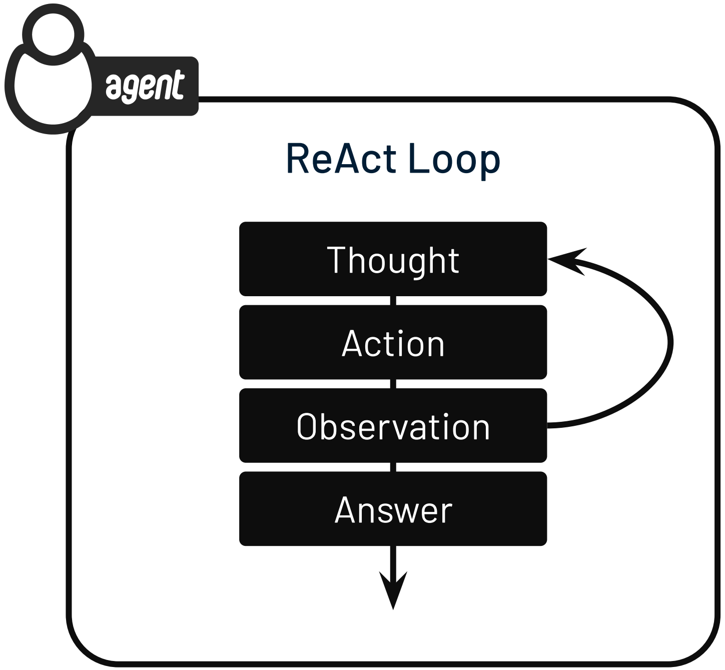
Working with AI Agents

Controlling Agents with Guardrails

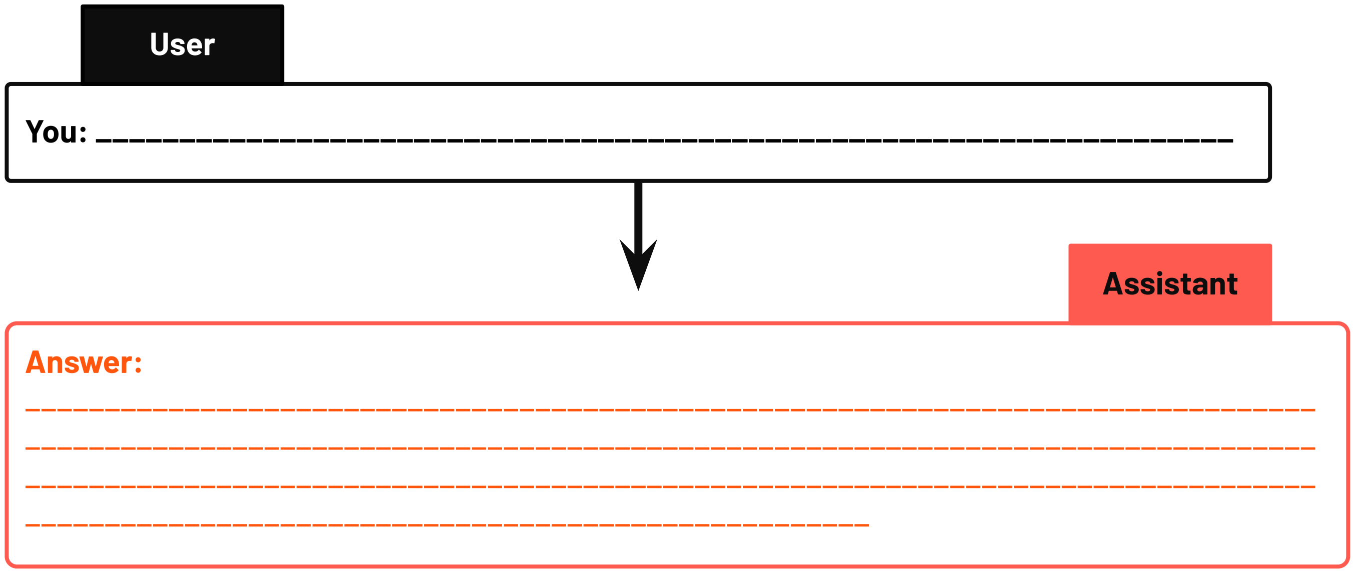
crewai

DeepLearning.AI

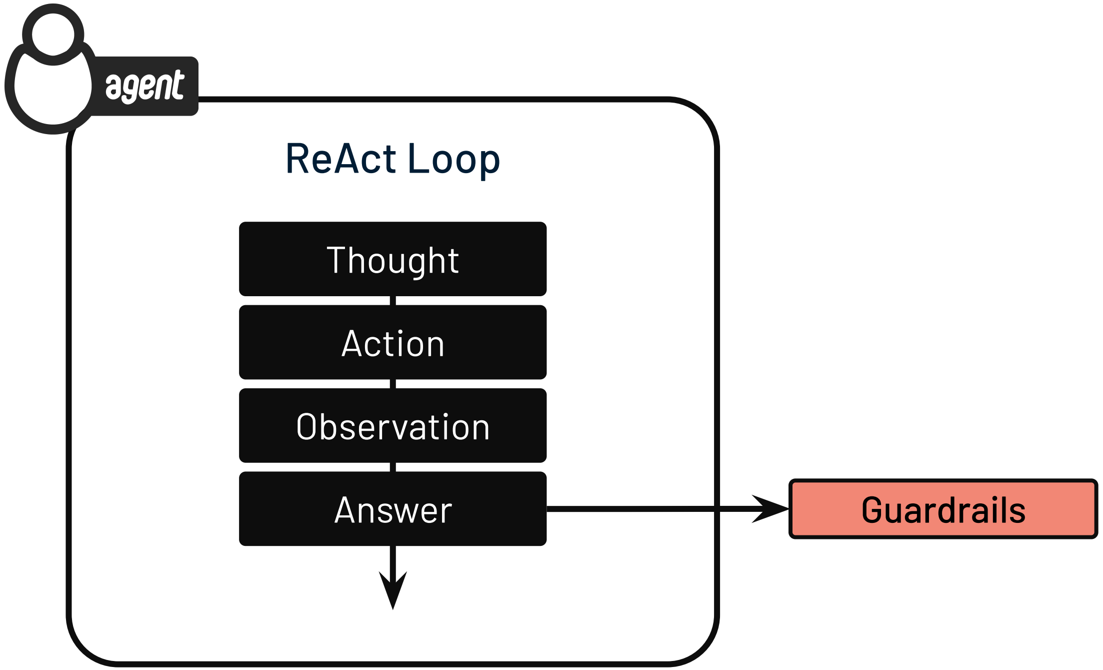
Agent Guardrails



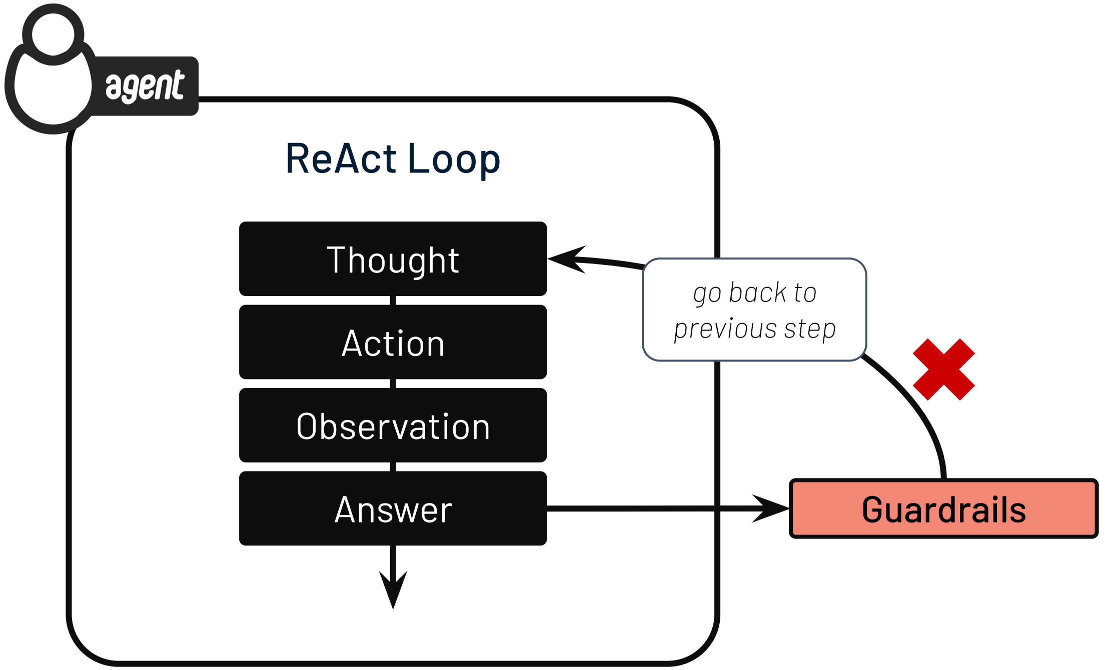
Agent Guardrails



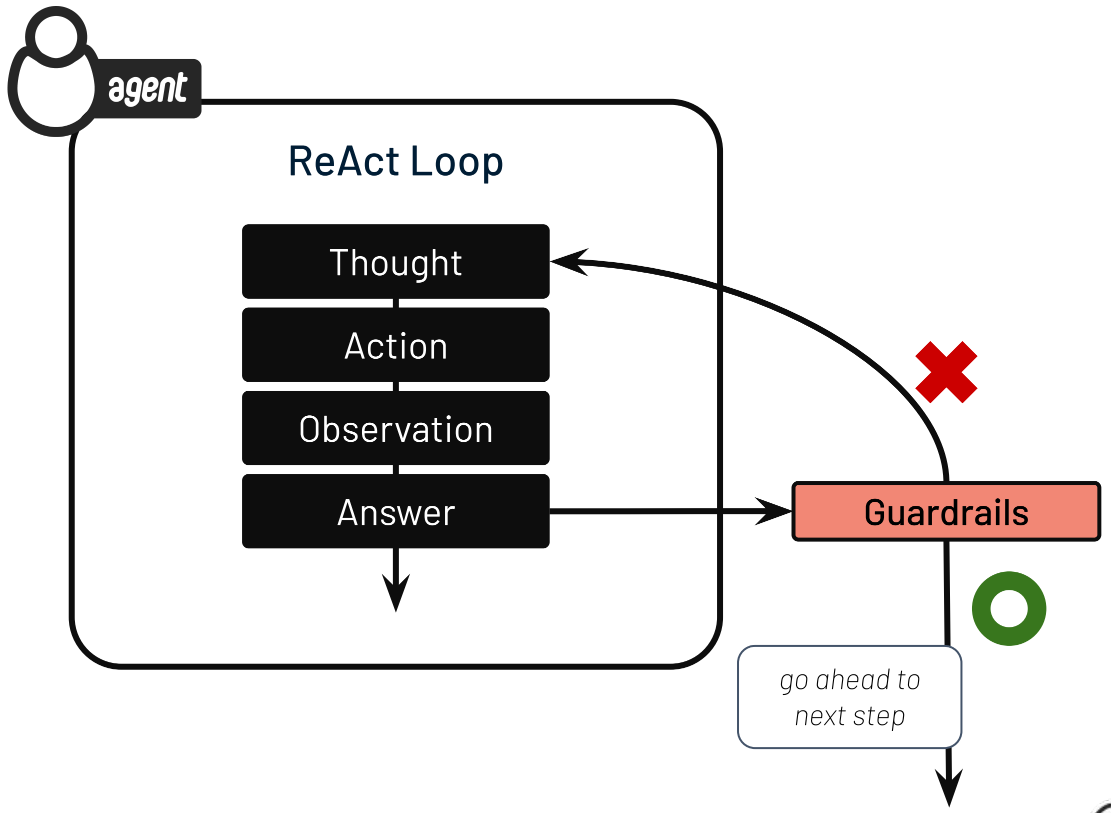
Agent Guardrails



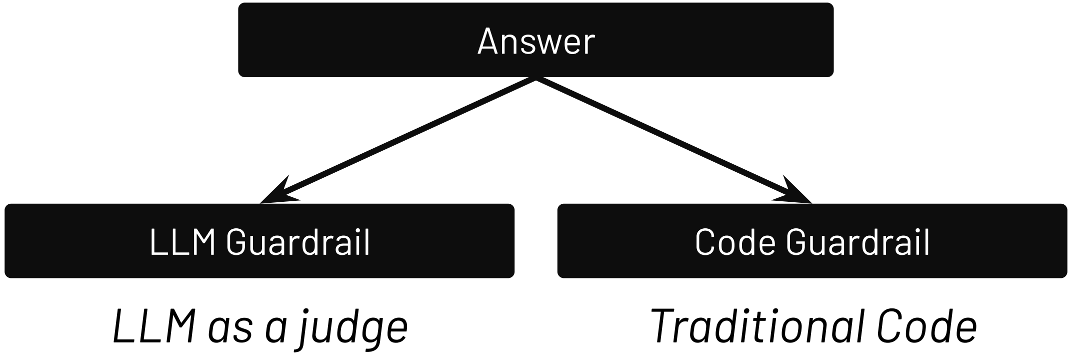
Agent Guardrails



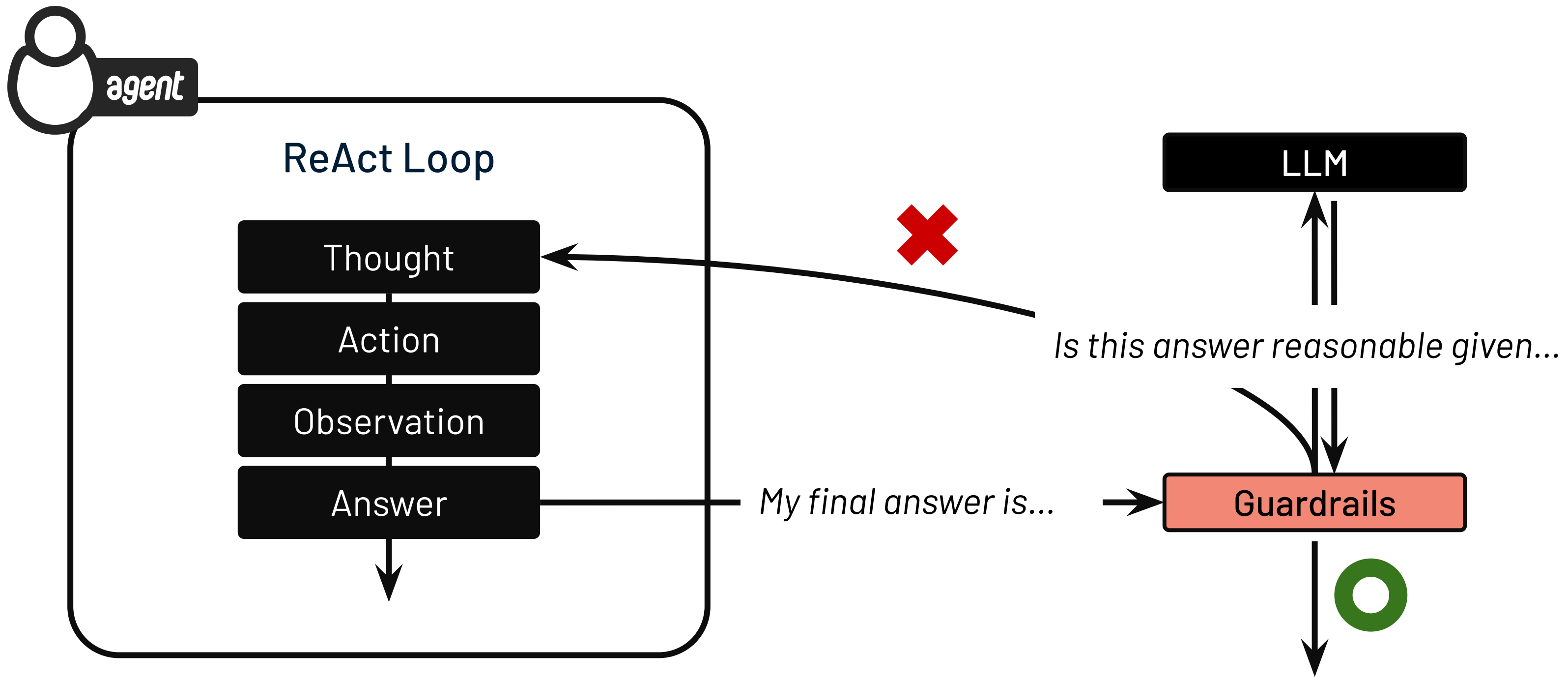
Agent Guardrails



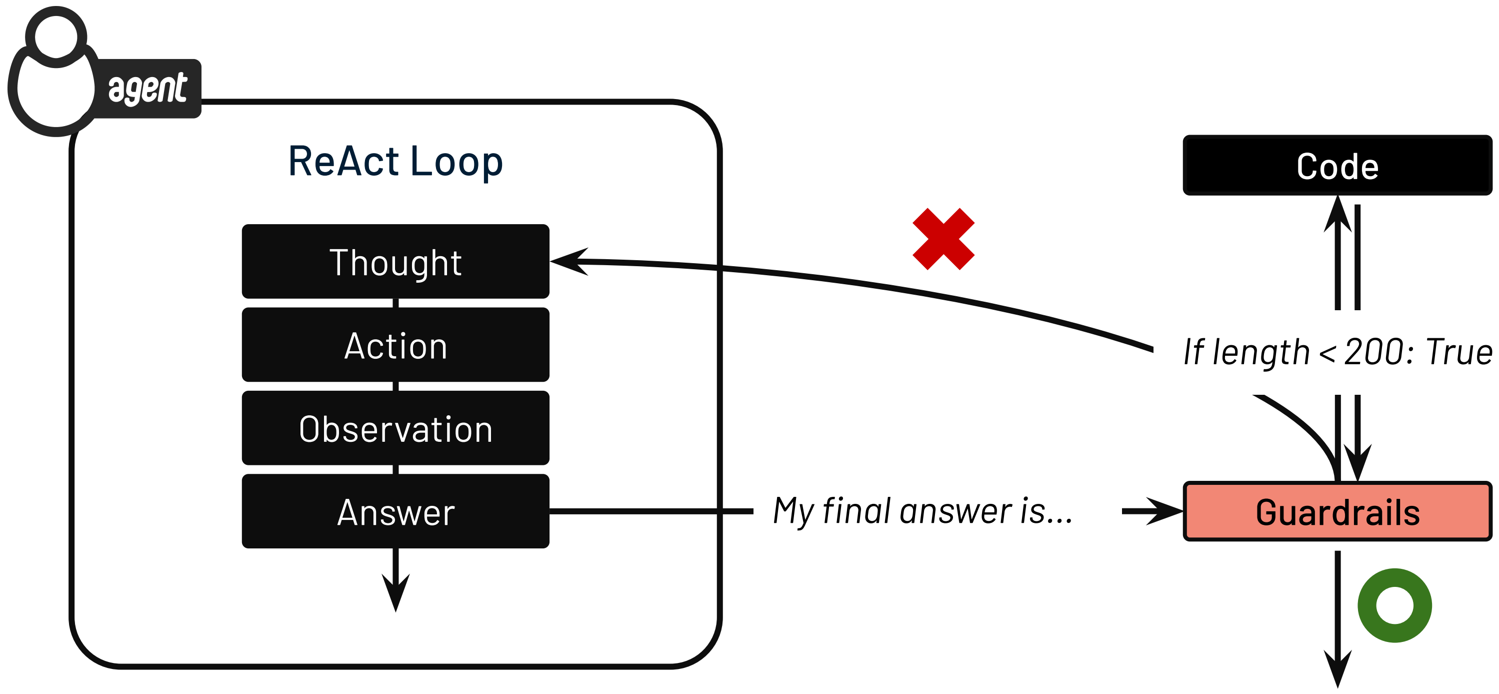
Deterministic Guardrails vs Probabilistic Guardrails



LLM Guardrail

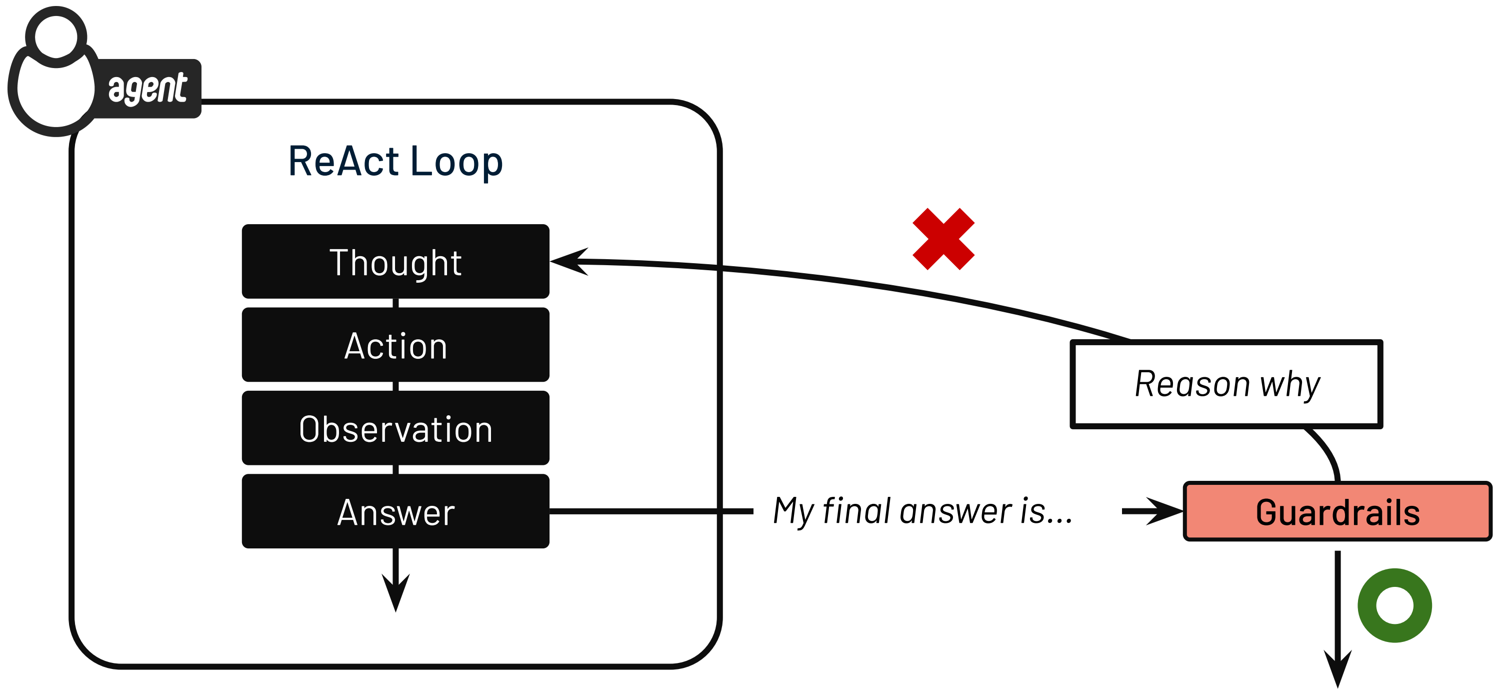


Code Guardrail



DeepLearning.AI

crewai


agent
ReAct Loop
Thought
Action
Observation
Answer
Reason why
My final answer is...
Guardrails

```python
from typing import Tuple, Any

def word_count_guardrail(output: str) -> Tuple[bool, Any]:
    """check word count of string"""

    word_count = len(output.split())
    if (word_count < 200):
         return (True, output)
    else:
         return (False, "Word count exceeds limit of 200")
```

```python
from crewai import Agent, Task

copywriter = Agent(role='Copywriter',
                  goal='Create persuasive marketing copy',
                  backstory='An expert in writing engaging promotional content')


copywriting_task = Task(description="Write marketing copy for the new product",
                       expected_output="A brief paragraph with marketing copy",
                       agent=copywriter,
                       guardrail=word_count_guardrail)
```

```python
from crewai import Agent, Task

copywriter = Agent(role='Copywriter',
                   goal='Create persuasive marketing copy',
                   backstory='An expert in writing engaging promotional content')


copywriting_task = Task(description="Write marketing copy for the new product",
                        expected_output="A brief paragraph with marketing copy",
                        agent=copywriter,
                        guardrails=[word_count_guardrail])
```

```python
from crewai import Agent, Task
from crewai.tasks.llm_guardrail import LLMGuardrail
copywriter = Agent(role='Copywriter',
                  goal='Create persuasive marketing copy',
                  backstory='An expert in writing engaging promotional content')

tone_guardrail = LLMGuardrail(description="Ensure writing has positive tone")
copywriting_task = Task(description="Write marketing copy for the new product",
                       expected_output="A brief paragraph with marketing copy",
                       agent=copywriter,
                       guardrails=[word_count_guardrail, tone_guardrail])
```

DeepLearning.AI

crewai

Enforcing structured output

Getting results in the format you expect

DeepLearning.AI

crewai

Structured Output

- Include a formatter agent
  - If format is important, make it a specialized role. Easier to test
- Use Pydantic models

```
from pydantic import BaseModel
from typing import Optional, List


class UserQuery(BaseModel):
  text: str
  intent: Optional[str] = None
  entities: List[str] = []
  confidence: Optional[float] = None
```

Structured Output

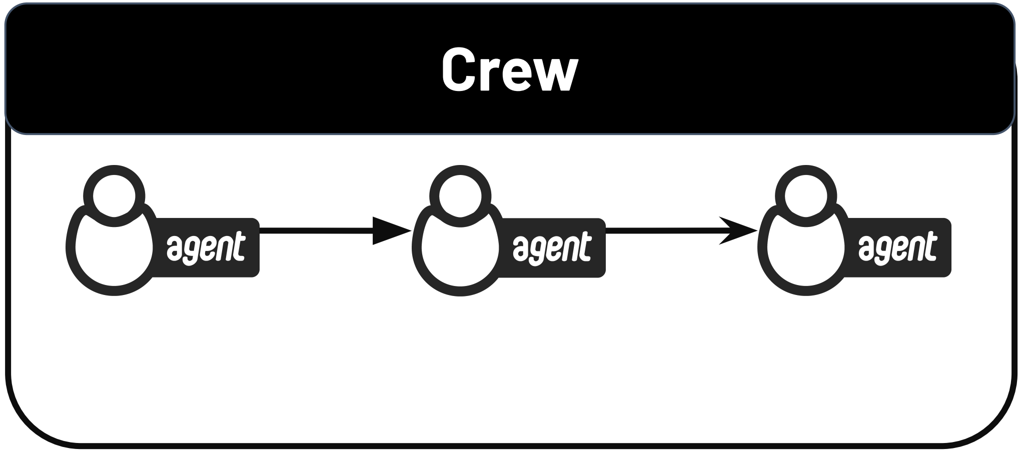
- Include a formatter agent
  - If format is important, make it a specialized role. Easier to test
- Use Pydantic models

```
task = Task(
   description=(
            "Given a message, produce: text, intent, entities and"
            "confidence (0–1). Keep it concise."
   ),
   expected_output="JSON: text, intent, entities, confidence.",
   agent=query_agent,
   output_pydantic=UserQuery # alternative to output_json=UserQuery
)
```

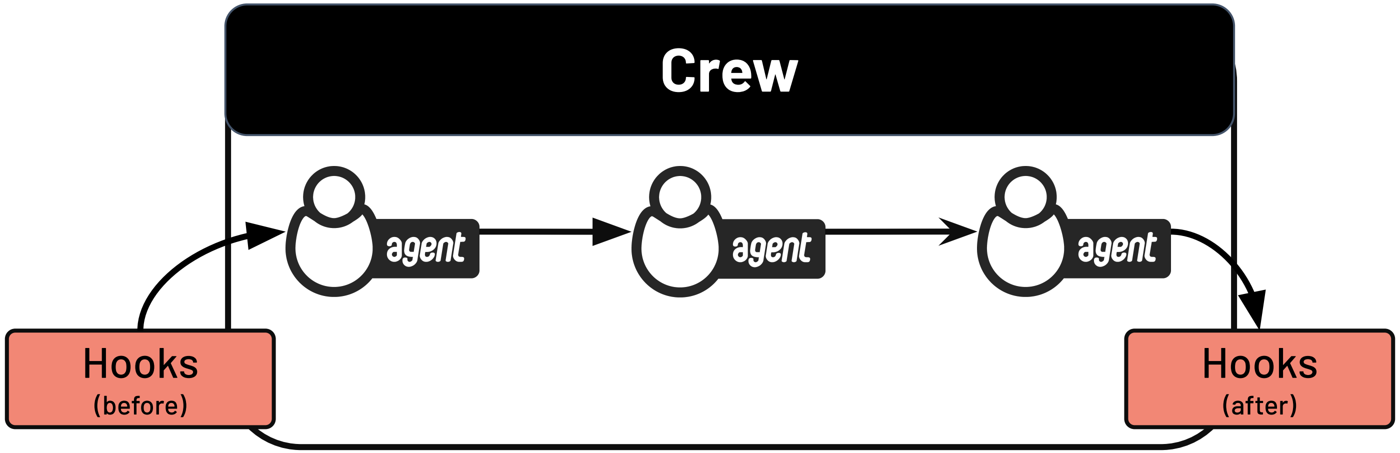
Working with AI Agents

Controlling Agents with Execution Hooks

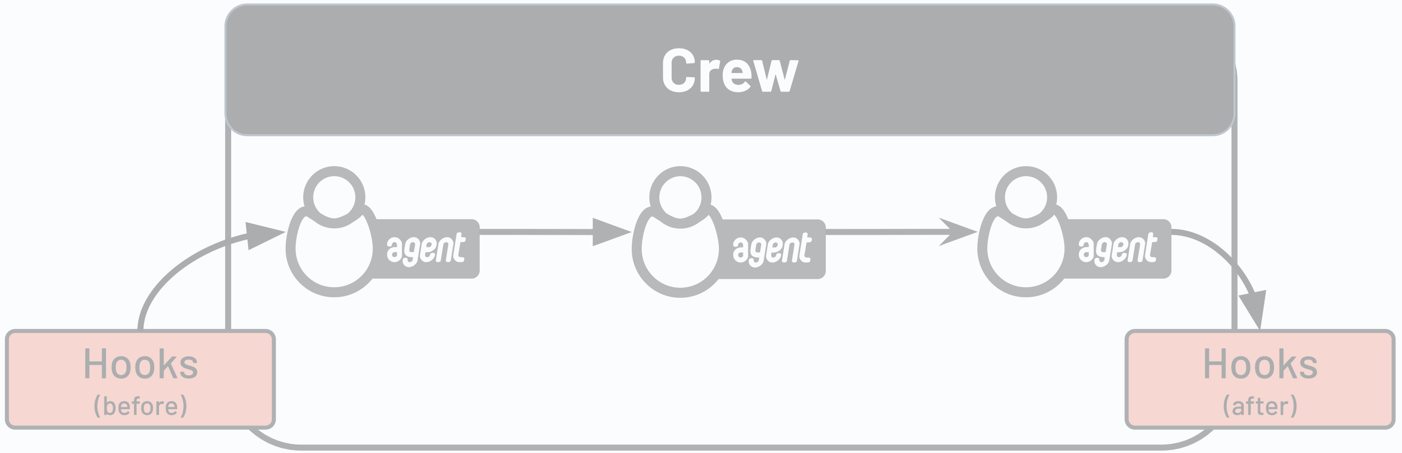
Execution Hooks



Execution Hooks

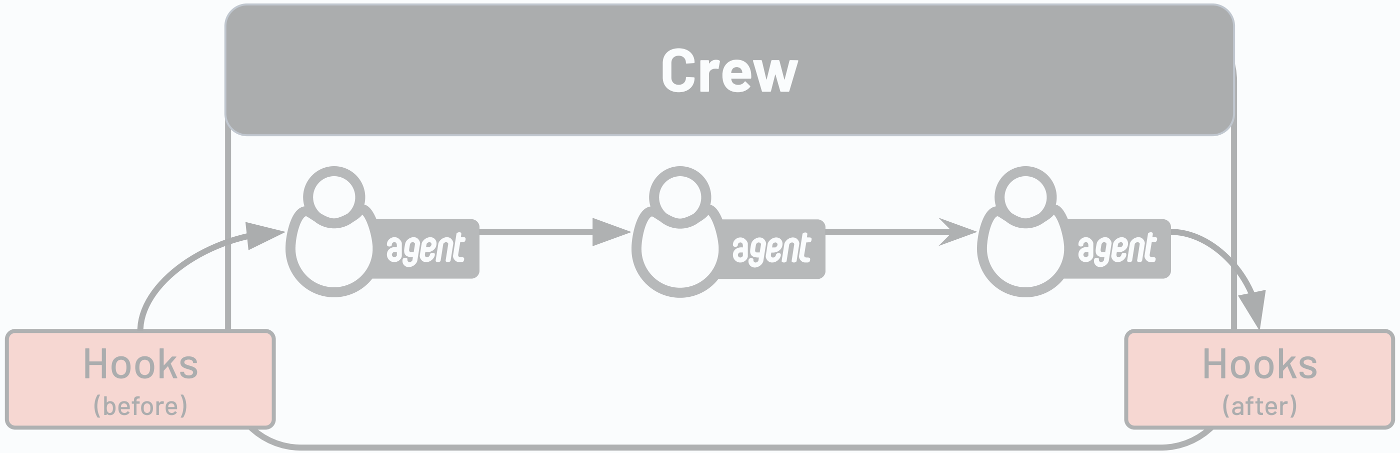


Execution Hooks



- *Fetch input data*
- *Clean input data*
- *Check inputs for PII*
- ...

Execution Hooks



- *Validate outputs*
- *Moderate output content*
- *Log outputs*
- ...

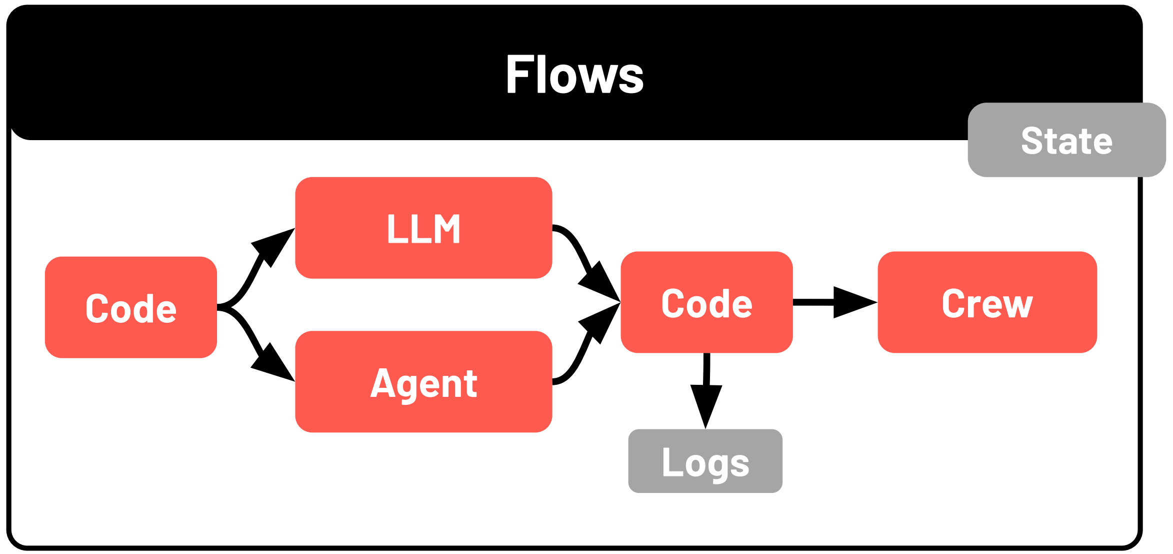
```python
# pre-hook function
def format_phone_pre_hook(inputs):
  phone = inputs.get("phone_number", "")
  digits = ''.join(filter(str.isdigit, phone))
  formatted = f"({digits[:3]}) {digits[3:6]}-{digits[6:]}"
  inputs["phone_number"] = formatted
  return inputs

# post-hook function
def confirm_phone_post_hook(outputs):
  phone = outputs.get("phone_number", "")
  safe_display = "(XXX) XXX-" + phone[-4:]    # Obfuscate first 6 digits
  confirmation = f"Phone number {safe_display} successfully processed"
  return confirmation
```

```python
# pre-hook function
def format_phone_pre_hook(inputs):
  phone = inputs.get("phone_number", "")
  digits = ''.join(filter(str.isdigit, phone))
  formatted = f"({digits[:3]}) {digits[3:6]}-{digits[6:]}"
  inputs["phone_number"] = formatted
  return inputs


# post-hook function
def confirm_phone_post_hook(outputs):
  phone = outputs.get("phone_number", "")
  safe_display = "(XXX) XXX-" + phone[-4:]    # Obfuscate first 6 digits
  confirmation = f"Phone number {safe_display} successfully processed"
  return confirmation
```

```
# define task with hooks
validate_task = Task(
  description="Collect, process, and store contact information from users.",
  expected_output="A confirmation message about the valid and invalid fields
with PII masked.",
  agent=validator_agent,
  pre_hooks=[format_phone_pre_hook],
  post_hooks=[confirm_phone_post_hook],
)
```

It’s about systems that are reliable

Flows



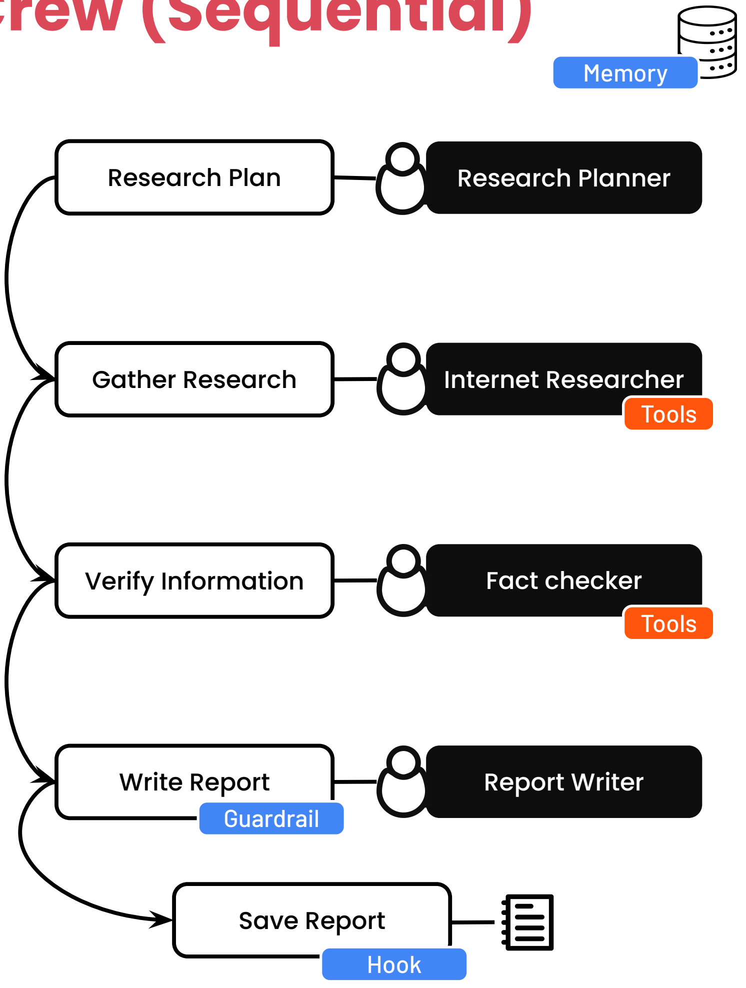
Working with AI Agents

Building a Deep Research Crew

crewai

DeepLearning.AI

Deep Research Crew (Sequential)



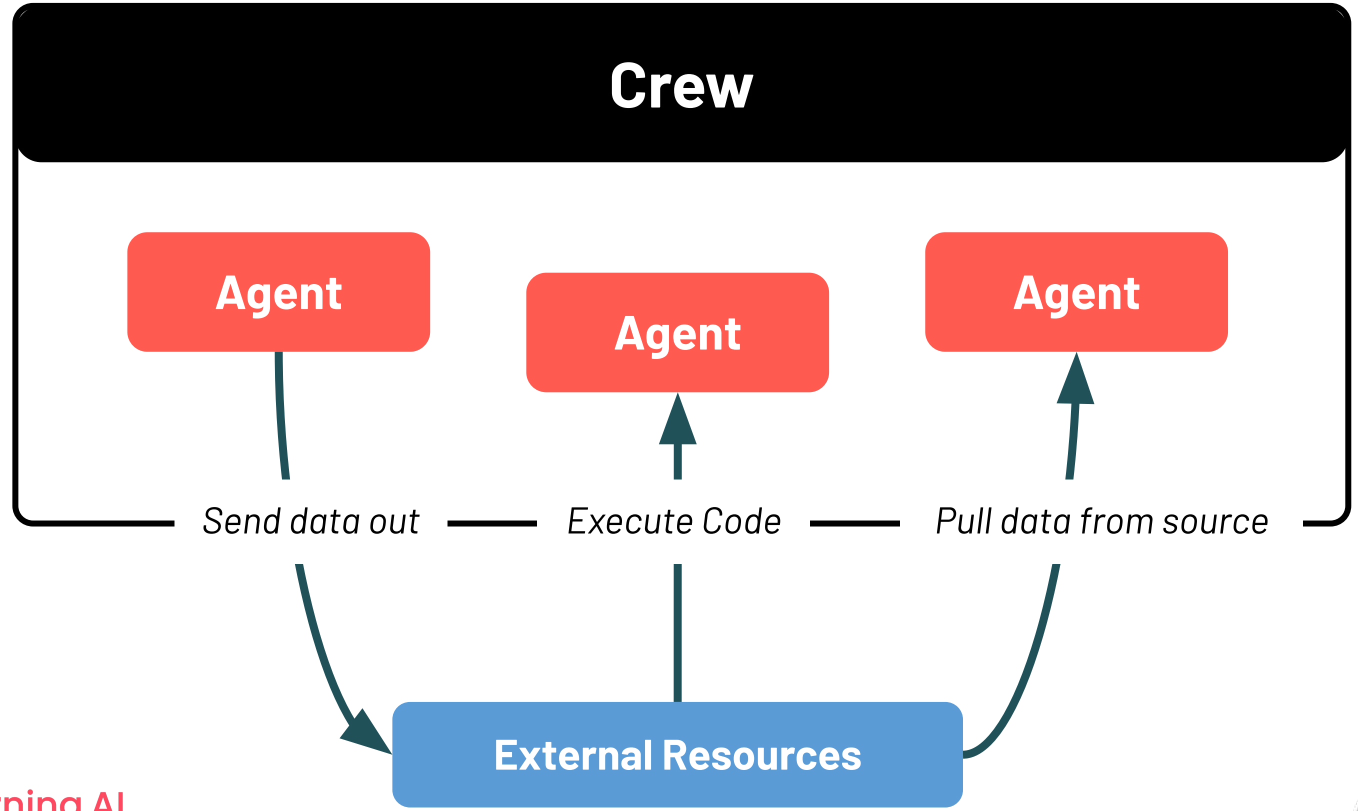
Working with AI Agents

Using Tools in Agents

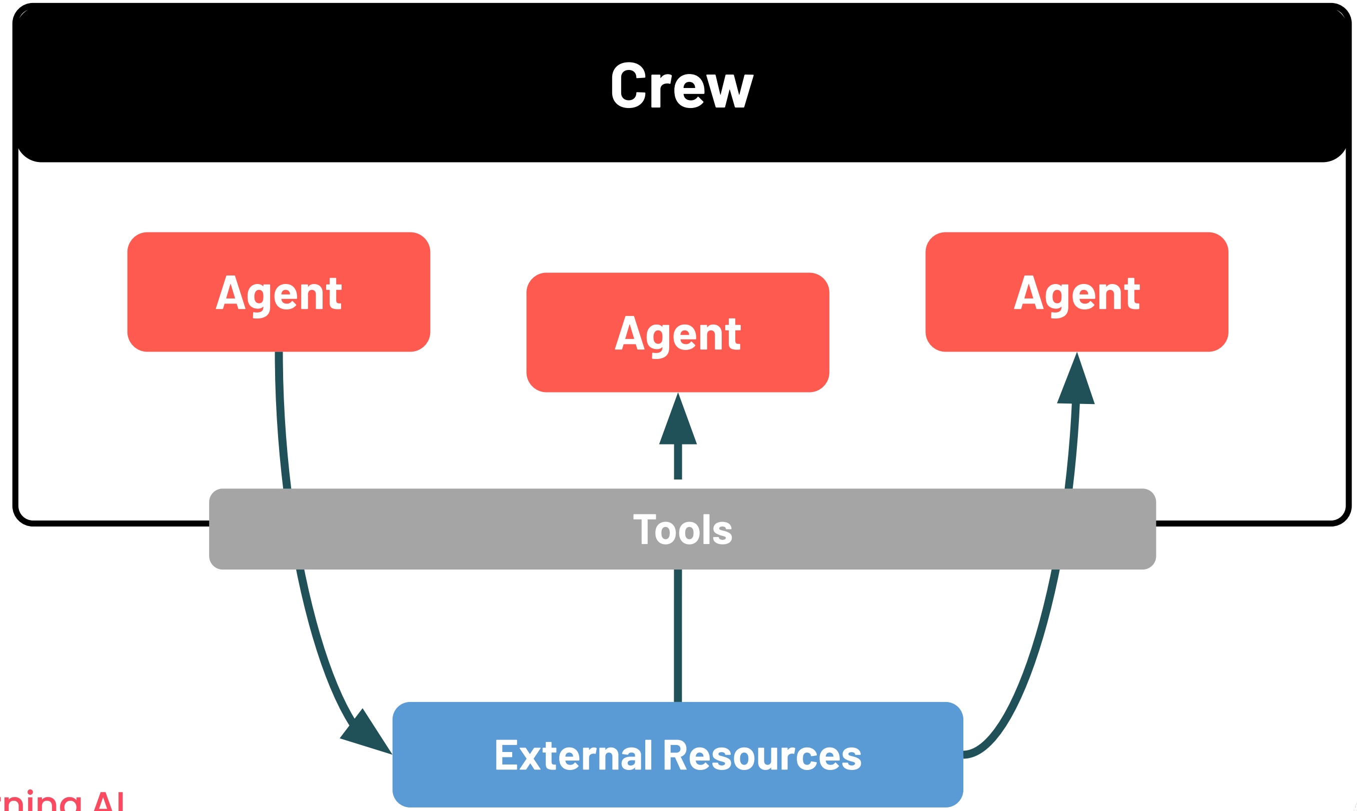
crewai

DeepLearning.AI

Accessing External Resources

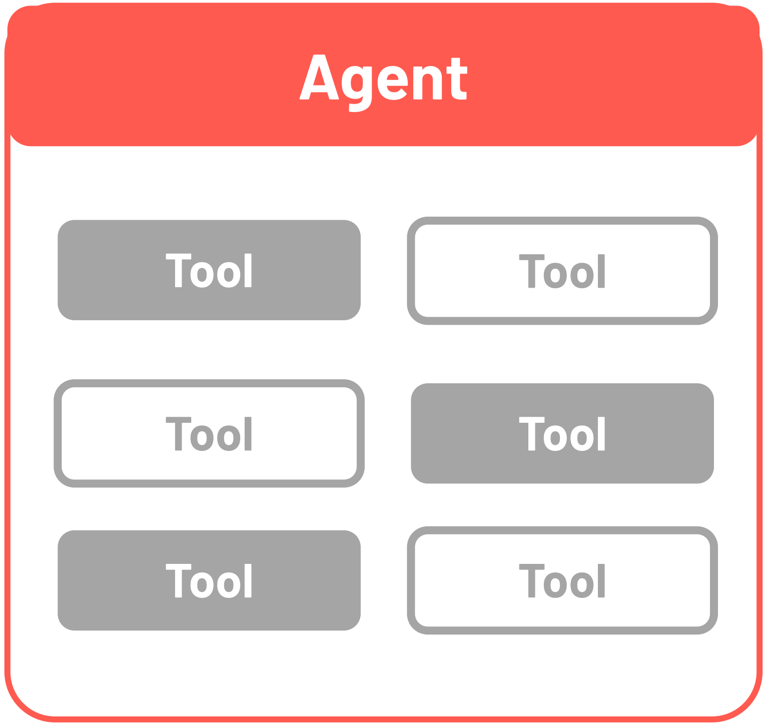


Accessing External Resources



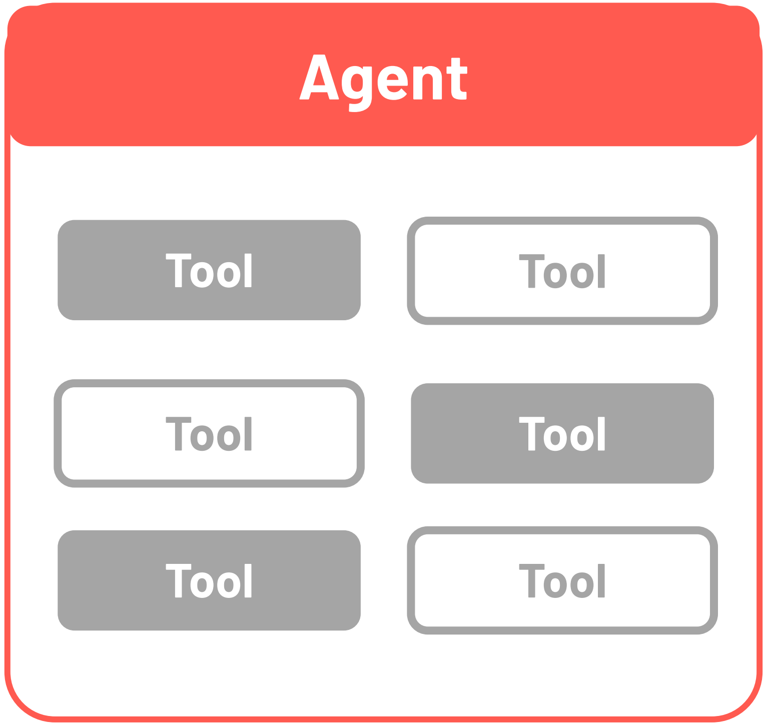
Tool Calling

- Tools provides capabilities to agents
- Access to too many tools can degrade performance
- Common tools include
  - Files & Folder
  - Web Scraping & Browsing
  - Database Connections
  - Code Execution



Importance of Tool Calling

- **Utility:** extends agent capabilities by integrating tools into their react loop
- **Flexibility:** users can make custom tools to meet their specific needs

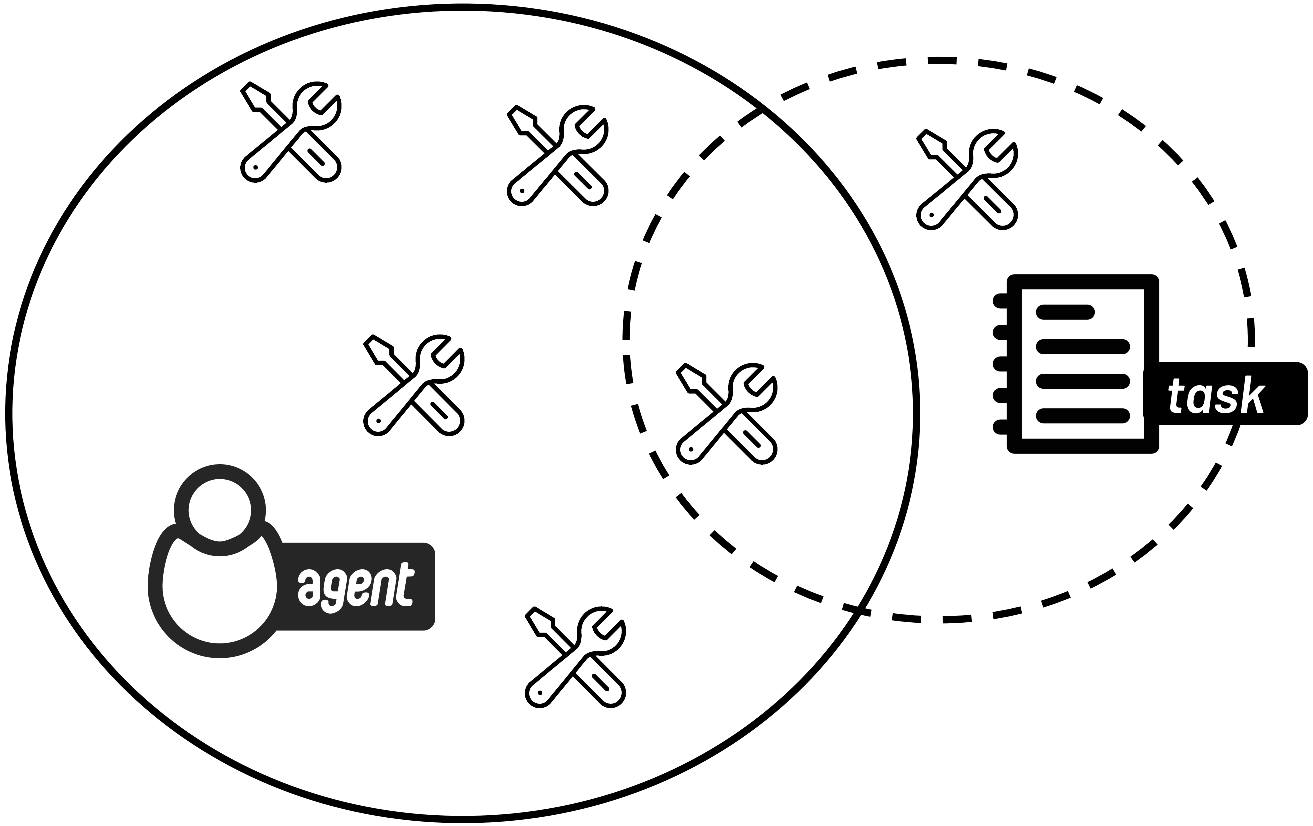


Considerations for Tool Calling

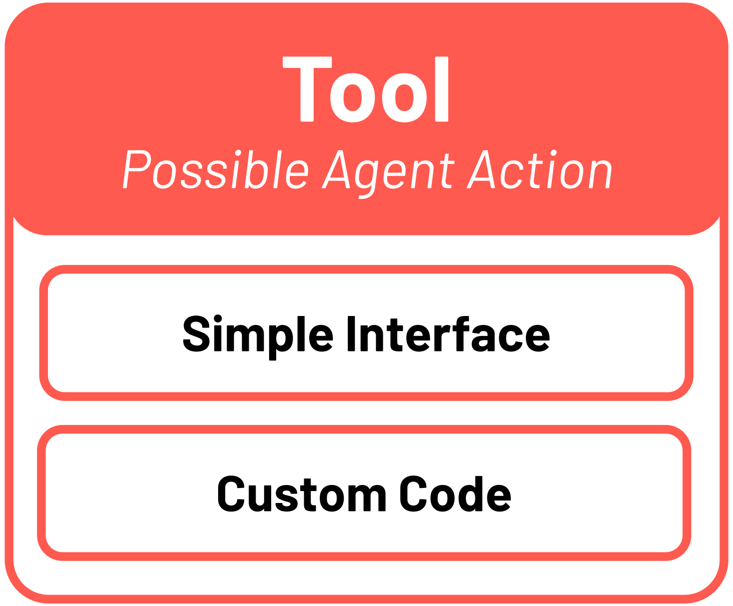
- **Error Handling:** robust exception handling to ensure graceful failures
- **Caching:** cross-agent caching to optimize performance and reduce redundant operations
- **Asynchronous Support**: handles both synchronous and asynchronous tools, enabling non-blocking operations

Agent
Tool
Tool
Tool
Tool
Tool
Tool

Precedence in Tool Calling



Components of Tools



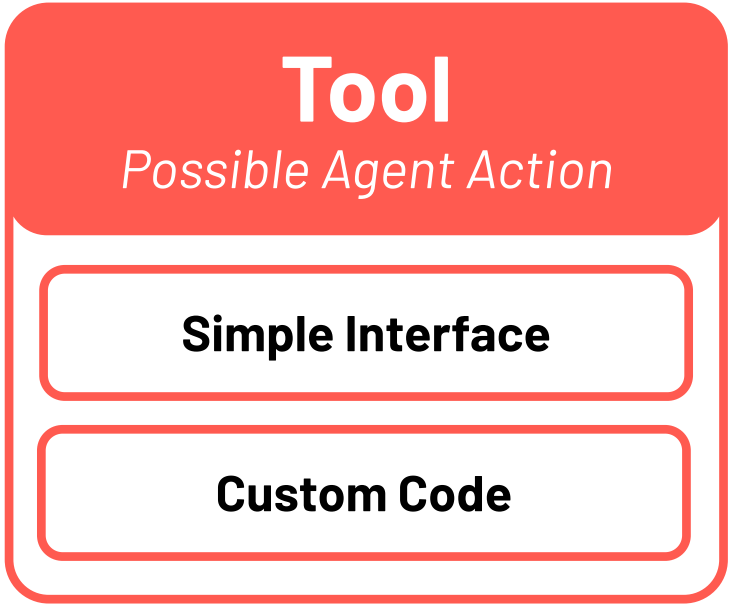
Simple Interface

- Name
- Description
- Inputs

Custom Code

- Authentication
- API Calling
- Service Connection

Components of Tools



Scrape Website Tool

Name: *Read website content*
Description: *A tool that can be used to read a website content.*
Inputs: { *website_url: str* }

Custom Code
Simple scraping logic using http requests

DeepLearning.AI

crewai

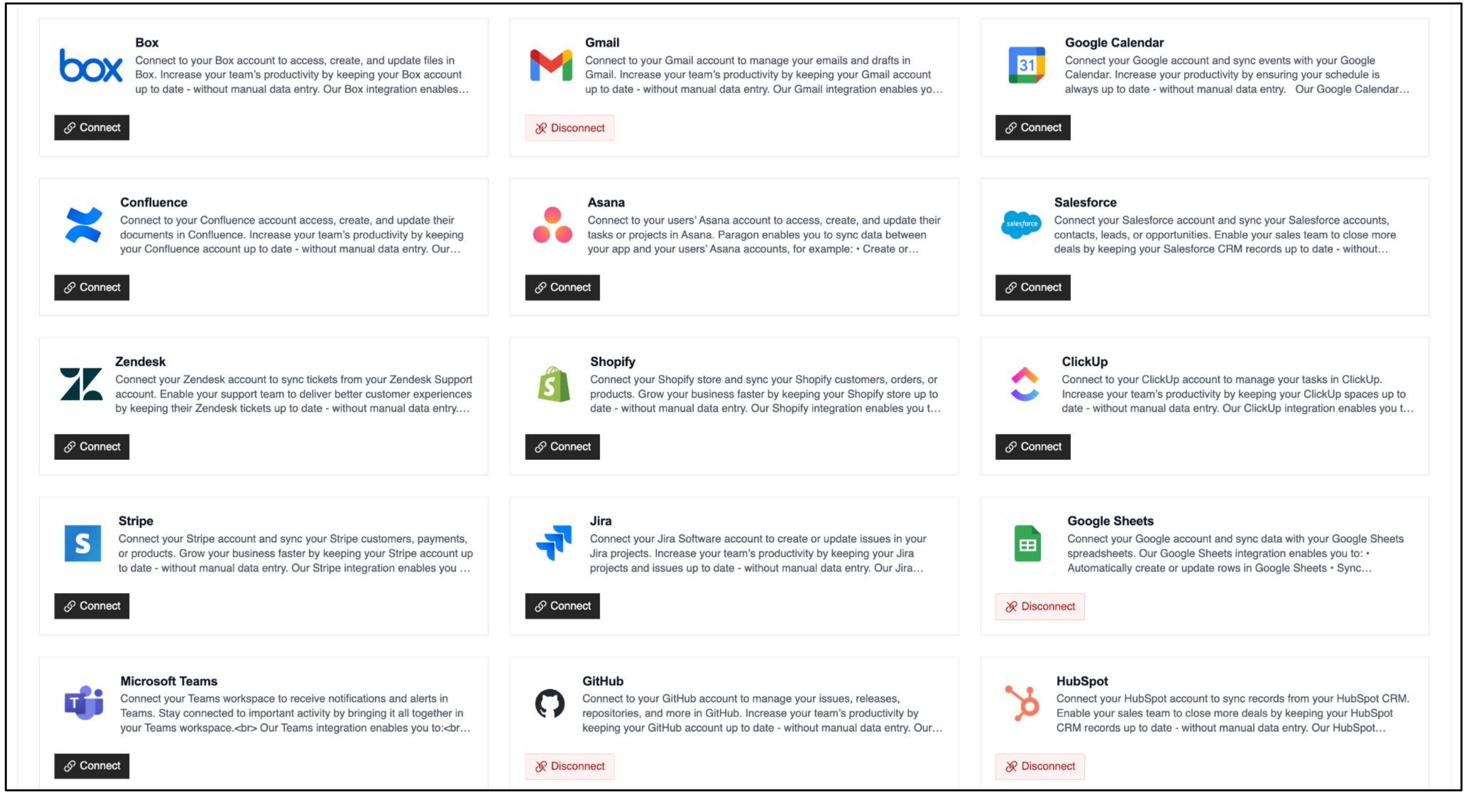
```python
import requests
from crewai_tools import tool


@tool("Read website content")
def read_website_content(website_url: str) -> str:
    """A tool that can be used to read website content."""
    try:
        response = requests.get(website_url, timeout=5)
        response.raise_for_status()
        return response.text
    except Exception as e:
        return f"Failed to read website: {str(e)}"
```

```
from crewai import Agent

agent = Agent(
    role='Web Reader',
    goal='Understand and summarize content from websites',
    backstory='You specialize in reading and summarizing website content.',
    tools=[read_website_content],
    verbose=True,
    memory=True
)
```

DeepLearning.AI

crewai

Enterprise Connectors



Box
Connect to your Box account to access, create, and update files in Box. Increase your team's productivity by keeping your Box account up to date - without manual data entry. Our Box integration enables…
Connect

Gmail
Connect to your Gmail account to manage your emails and drafts in Gmail. Increase your team's productivity by keeping your Gmail account up to date - without manual data entry. Our Gmail integration enables yo…
Disconnect

Google Calendar
Connect your Google account and sync events with your Google Calendar. Increase your productivity by ensuring your schedule is always up to date - without manual data entry. Our Google Calendar…
Connect

Confluence
Connect to your Confluence account access, create, and update their documents in Confluence. Increase your team's productivity by keeping your Confluence account up to date - without manual data entry. Our…
Connect

Asana
Connect to your users' Asana account to access, create, and update their tasks or projects in Asana. Paragon enables you to sync data between your app and your users' Asana accounts, for example: • Create or…
Connect

Salesforce
Connect your Salesforce account and sync your Salesforce accounts, contacts, leads, or opportunities. Enable your sales team to close more deals by keeping your Salesforce CRM records up to date - without…
Connect

Zendesk
Connect your Zendesk account to sync tickets from your Zendesk Support account. Enable your support team to deliver better customer experiences by keeping their Zendesk tickets up to date - without manual data entry.…
Connect

Shopify
Connect your Shopify store and sync your Shopify customers, orders, or products. Grow your business faster by keeping your Shopify store up to date - without manual data entry. Our Shopify integration enables you t…
Connect

ClickUp
Connect to your ClickUp account to manage your tasks in ClickUp. Increase your team's productivity by keeping your ClickUp spaces up to date - without manual data entry. Our ClickUp integration enables you t…
Connect

Stripe
Connect your Stripe account and sync your Stripe customers, payments, or products. Grow your business faster by keeping your Stripe account up to date - without manual data entry. Our Stripe integration enables you …
Connect

Jira
Connect your Jira Software account to create or update issues in your Jira projects. Increase your team's productivity by keeping your Jira projects and issues up to date - without manual data entry. Our Jira…
Connect

Google Sheets
Connect your Google account and sync data with your Google Sheets spreadsheets. Our Google Sheets integration enables you to: • Automatically create or update rows in Google Sheets • Sync…
Disconnect

Microsoft Teams
Connect your Teams workspace to receive notifications and alerts in Teams. Stay connected to important activity by bringing it all together in your Teams workspace.
 Our Teams integration enables you to:<br…
Connect

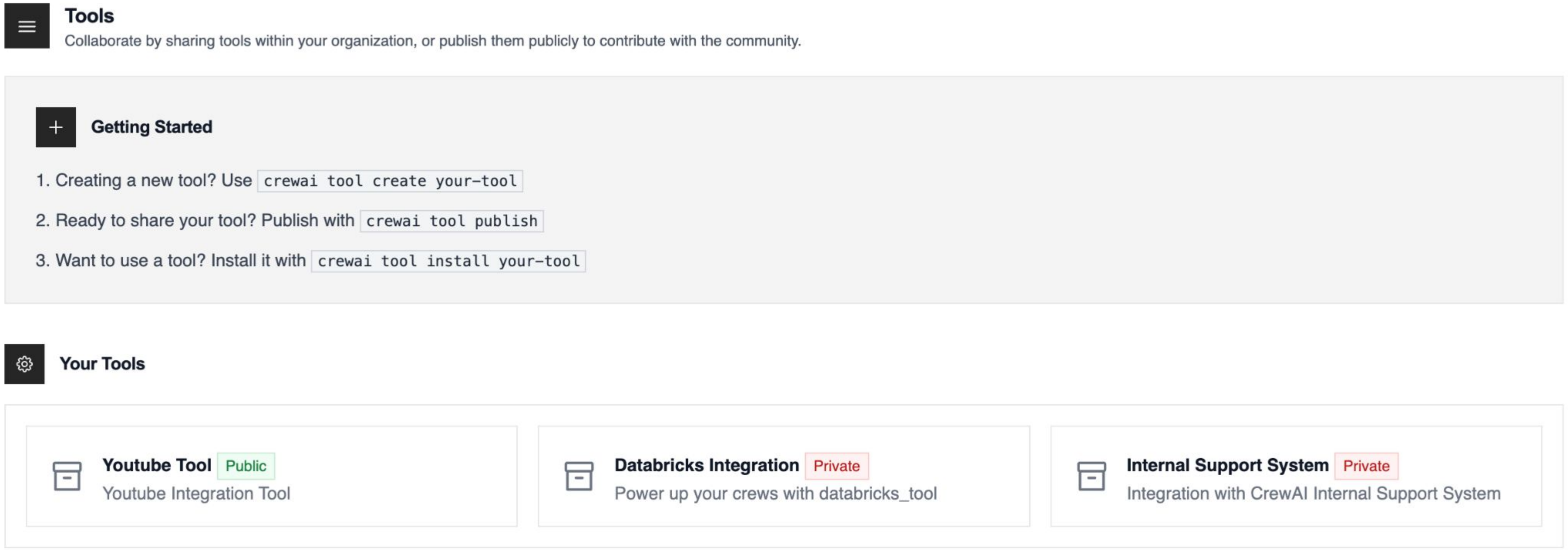
GitHub
Connect to your GitHub account to manage your issues, releases, repositories, and more in GitHub. Increase your team's productivity by keeping your GitHub account up to date - without manual data entry. Our…
Disconnect

HubSpot
Connect your HubSpot account to sync records from your HubSpot CRM. Enable your sales team to close more deals by keeping your HubSpot CRM records up to date - without manual data entry. Our HubSpot…
Disconnect

```python
from crewai import Agent
from crewai_tools import CrewaiEnterpriseTools

# Get enterprise tools (Gmail tool will be included)
enterprise_tools = CrewaiEnterpriseTools(enterprise_token="your_enterprise_token")

# Create an agent with Gmail capabilities
email_agent = Agent(role="Email Manager",
                    goal="Manage and organize email communications",
                    backstory="An AI assistant specialized in email",
                    tools=[enterprise_tools])
```

```python
from crewai import Agent
from crewai_tools import CrewaiEnterpriseTools

# Get enterprise tools (Gmail tool will be included)
enterprise_tools = CrewaiEnterpriseTools(enterprise_token="your_enterprise_token")

# Create an agent with Gmail capabilities
email_agent = Agent(role="Email Manager",
                    goal="Manage and organize email communications",
                    backstory="An AI assistant specialized in email",
                    tools=[enterprise_tools])
```

Tool Repository



Tools

Collaborate by sharing tools within your organization, or publish them publicly to contribute with the community.

Getting Started

1. Creating a new tool? Use `crewai tool create your-tool`
2. Ready to share your tool? Publish with `crewai tool publish`
3. Want to use a tool? Install it with `crewai tool install your-tool`

Your Tools

Youtube Tool Public
Youtube Integration Tool

Databricks Integration Private
Power up your crews with databricks_tool

Internal Support System Private
Integration with CrewAI Internal Support System

Tool Reliability

How to configure tools for reliable run-time behavior?

Force Return

Directly return the output of tool by specifying `return_direct=True` in agent

Rate Limits

Use retry logic and a max usage limit to help agents recover from temporary failures while preventing infinite loops

Tool Repository

Promotes reuse and sharing of tools across multiple agents and tasks

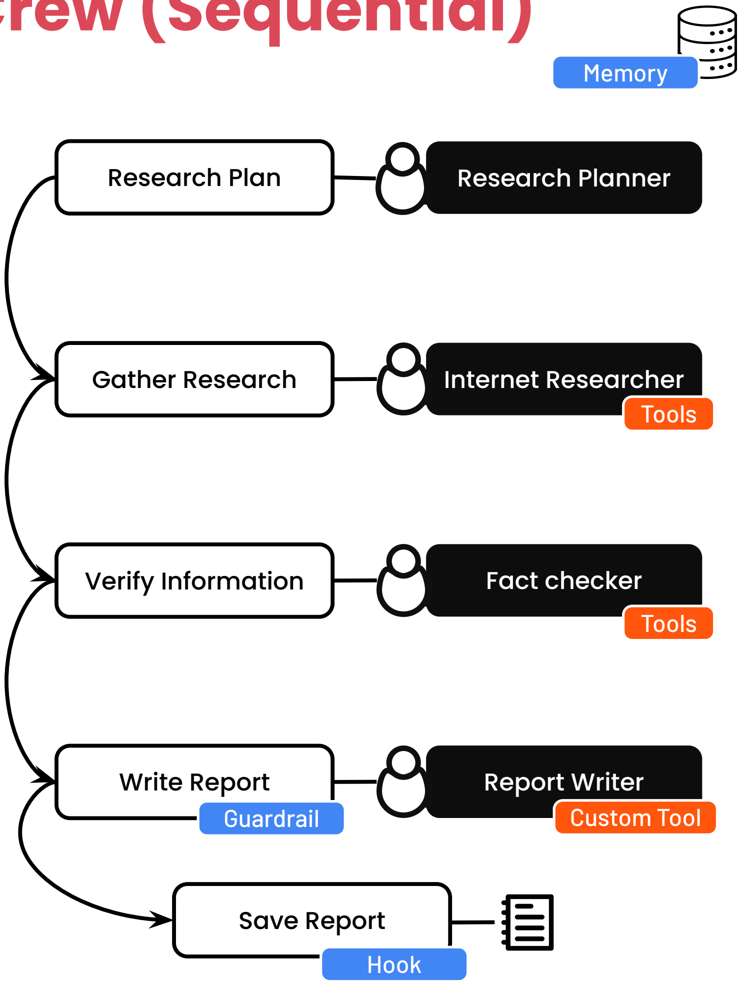
Working with AI Agents

Adding Tools to the Deep Research Crew

crewai

DeepLearning.AI

Deep Research Crew (Sequential)



Working with AI Agents

Adopting Model Context Protocol

crewai

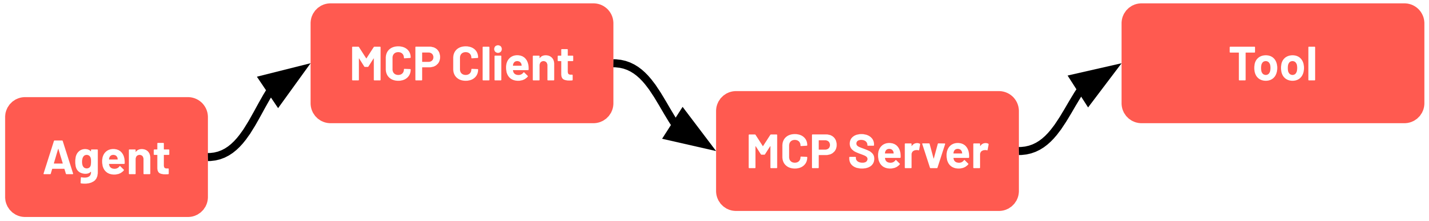
DeepLearning.AI

Why have protocols for AI?

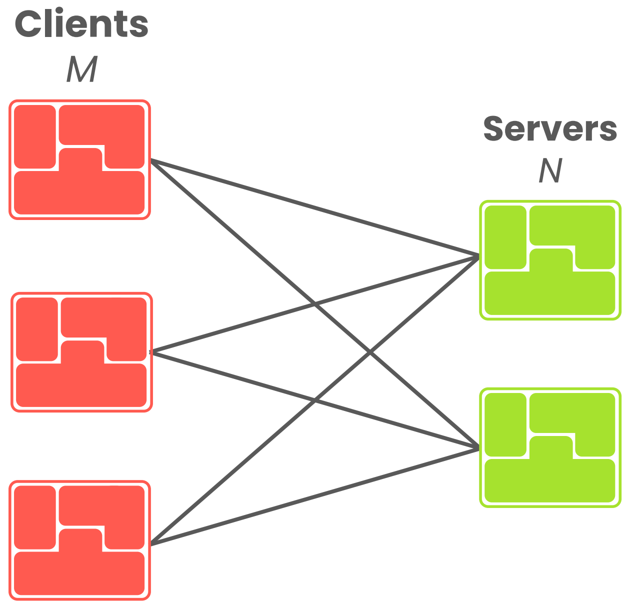
- Everyone exposes tools differently
- Integrations require custom adapters
- Difficult to share tools among multiple agents
- Entire companies built around integrations

What is Model Context Protocol?

- Open protocol for tools, resources, prompts.
- JSON-RPC 2.0 under the hood.
- Client ↔ Server architecture.

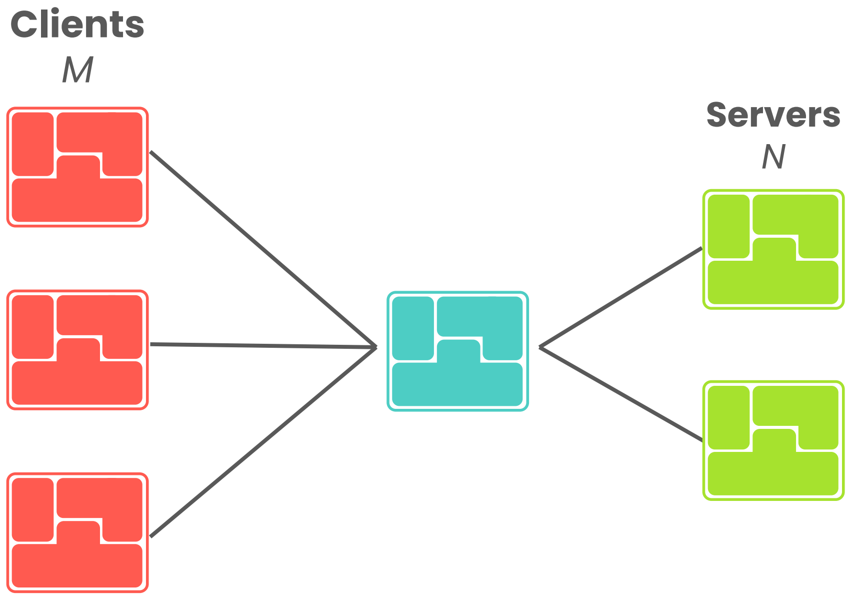


What is Model Context Protocol?



No MCP

M x N Connections



MCP

M + N Connections

Components of MCP

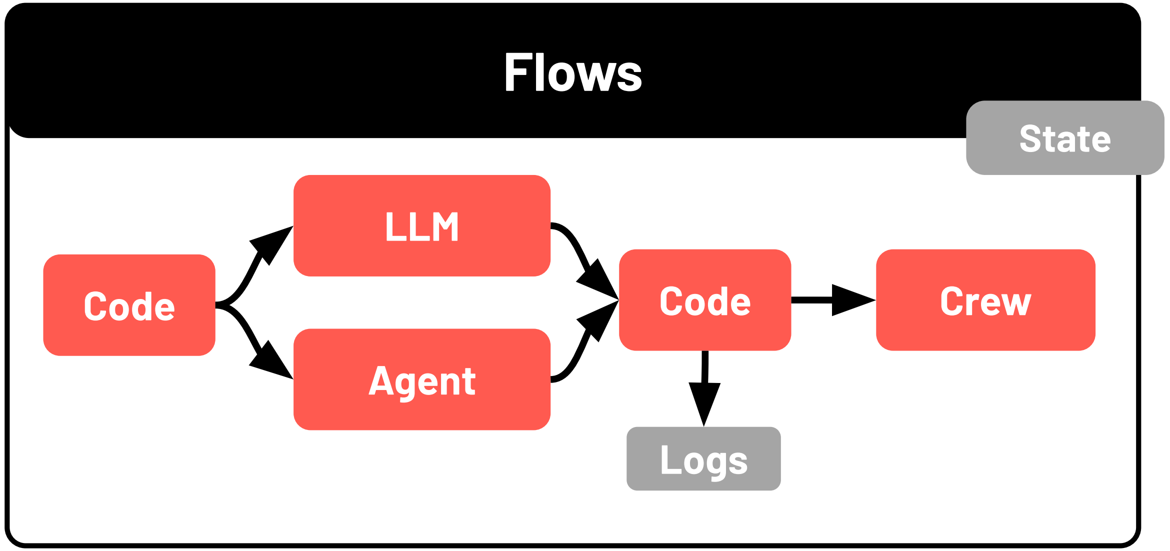
Primitives

- Tools
- Prompts
- Resources

Transports

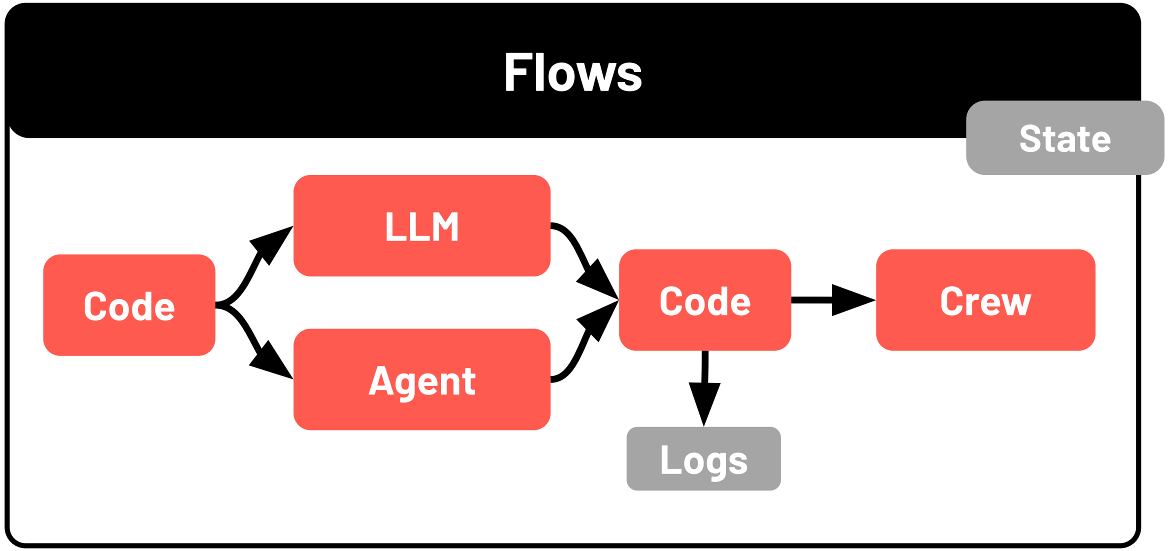
- Stdio (local)
- SSE (streaming)
- Streamable HTTP (remote)

CrewAI and MCP



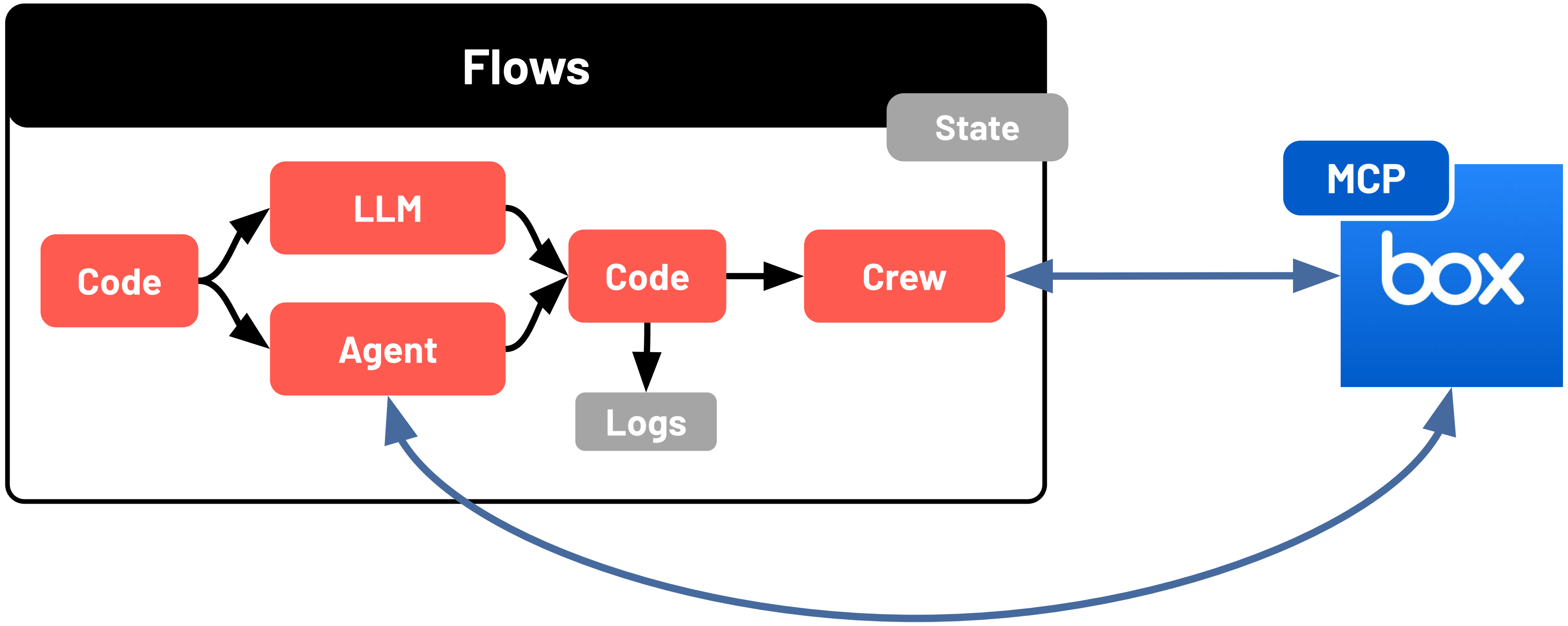
- Agents both inside crews and outside of crews can use MCP Servers
- Allowing them to filter and aggregate tools

CrewAI and MCP



- Crews themselves can become MCP Servers
- Allowing them to be used by others agents

CrewAI and MCP



```
from crewai import Agent
from crewai_tools import MCPServerAdapter


server_params = {"url": "http://localhost:8000/sse",
                 "transport": "sse”}
```

DeepLearning.AI

```python
from crewai import Agent
from crewai_tools import MCPServerAdapter

server_params = {"url": "http://localhost:8000/sse",
                 "transport": "sse”}
```

```
server_params = {"url": "http://localhost:8000/sse",
                 "transport": "sse”}

with MCPServerAdapter(server_params) as mcp_tools:
   print(f"Available tools: {[tool.name for tool in mcp_tools]}")

   my_agent = Agent(
       role="MCP Tool User",
       goal="Utilize tools from an MCP server.",
       backstory="I can connect to MCP servers and use their tools.",
       tools=[mcp_tools])
```

```
server_params = {"url": "http://localhost:8000/sse",
                 "transport": "sse”}

with MCPServerAdapter(server_params) as mcp_tools:
   print(f"Available tools: {[tool.name for tool in mcp_tools]}")

   my_agent = Agent(
       role="MCP Tool User",
       goal="Utilize tools from an MCP server.",
       backstory="I can connect to MCP servers and use their tools.",
       tools=[mcp_tools])
```

DeepLearning.AI

crewai

Safeguards for MCP

Before using an MCP server, you must trust it!

- SSE transports can be vulnerable if not properly secured.
- Always validate Origin headers on incoming SSE connections
- Avoid binding servers to all interfaces locally - bind only to localhost instead
- Implement proper authentication for all SSE connections

Safeguards for MCP

Before using an MCP server, you must trust it!

SSE transports can be vulnerable if not properly secured.

Without these protections, attackers could use DNS rebinding to interact with local MCP servers from remote websites.

- Avoid binding servers to all interfaces locally - bind only to localhost instead
- Implement proper authentication for all SSE connections

Outlook for MCP

- Reuse tools across frameworks/agents
- Standardize integrations inside large orgs
- Expose a crew as a service (future)
- Overhead for small/local scripts

Working with AI Agents

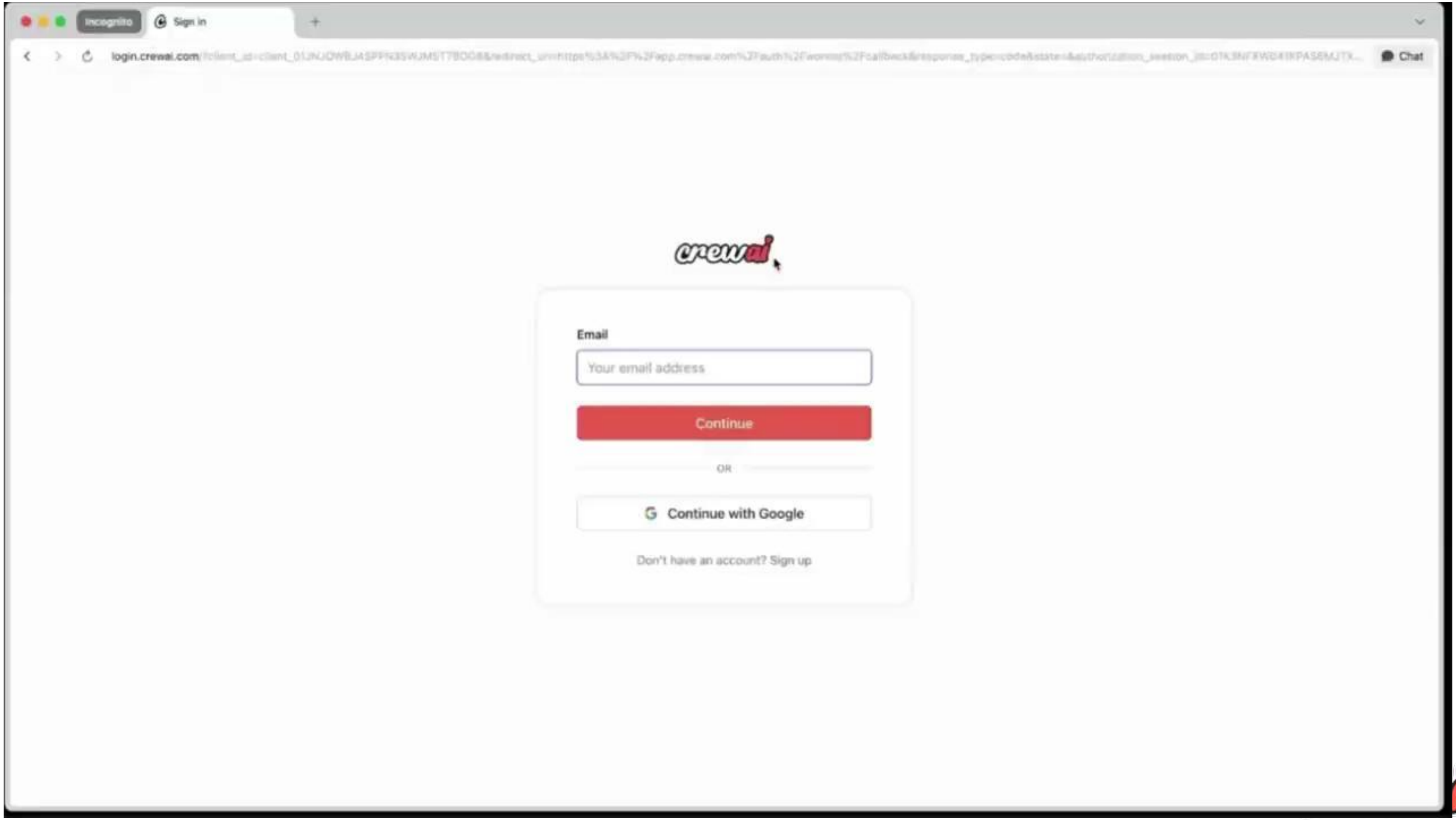
Building a Meeting Preparation Co-Pilot

crewai

DeepLearning.AI

Let’s Build it Live

app.crewai.com


Incognito
Sign in
login.crewai.com
Chat
crewai
Email
Your email address
Continue
OR
Continue with Google
Don't have an account? Sign up