AI & Machine Learning

Code Interpreter

Copy page

The `CodeInterpreterTool` is a powerful tool designed for executing Python 3 code within a secure, isolated environment.

`CodeInterpreterTool`

Deprecated: `CodeInterpreterTool` has been removed from `crewai-tools`. The `allow_code_execution` and `code_execution_mode` parameters on `Agent` are also deprecated. Use a dedicated sandbox service — E2B or Modal — for secure, isolated code execution.

Description

The `CodeInterpreterTool` enables CrewAI agents to execute Python 3 code that they generate autonomously. This functionality is particularly valuable as it allows agents to create code, execute it, obtain the results, and utilize that information to inform subsequent decisions and actions.

There are several ways to use this tool:

Docker Container (Recommended)

This is the primary option. The code runs in a secure, isolated Docker container, ensuring safety regardless of its content. Make sure Docker is installed and running on your system. If you don't have it, you can install it from **here**.

very limited, with strict restrictions on many modules and built-in functions.

Unsafe Execution

NOT RECOMMENDED FOR PRODUCTION This mode allows execution of any Python code, including dangerous calls to `sys, os..` and similar modules. **Check out** how to enable this mode

Logging

The `CodeInterpreterTool` logs the selected execution strategy to STDOUT

Installation

To use this tool, you need to install the CrewAI tools package:

```
pip install 'crewai[tools]'
```

Example

The following example demonstrates how to use the `CodeInterpreterTool` with a CrewAI agent:

```
# Initialize the tool
code_interpreter = CodeInterpreterTool()

# Define an agent that uses the tool
programmer_agent = Agent(
    role="Python Programmer",
    goal="Write and execute Python code to solve problems",
    backstory="An expert Python programmer who can write efficient code to solve
    tools=[code_interpreter],
    verbose=True,
)

# Example task to generate and execute code
coding_task = Task(
    description="Write a Python function to calculate the Fibonacci sequence up t
    expected_output="The Fibonacci sequence up to the 10th number.",
    agent=programmer_agent,
)

# Create and run the crew
crew = Crew(
    agents=[programmer_agent],
    tasks=[coding_task],
    verbose=True,
    process=Process.sequential,
)
result = crew.kickoff()
```

You can also enable code execution directly when creating an agent:

```
programmer_agent = Agent(
    role="Python Programmer",
    goal="Write and execute Python code to solve problems",
    backstory="An expert Python programmer who can write efficient code to solve
    allow_code_execution=True,  # This automatically adds the CodeInterpreterTool
    verbose=True,
)
```

Enabling `unsafe_mode`

Code

```
from crewai_tools import CodeInterpreterTool

code = """
import os
os.system("ls -la")
"""

CodeInterpreterTool(unsafe_mode=True).run(code=code)
```

Parameters

The `CodeInterpreterTool` accepts the following parameters during initialization:

- **user_dockerfile_path**: Optional. Path to a custom Dockerfile to use for the code interpreter container.
- **user_docker_base_url**: Optional. URL to the Docker daemon to use for running the container.

When using the tool with an agent, the agent will need to provide:

- **code**: Required. The Python 3 code to execute.
- **libraries_used**: Optional. A list of libraries used in the code that need to be installed. Default is `[]`

Agent Integration Example

Here's a more detailed example of how to integrate the `CodeInterpreterTool` with a CrewAI agent:

```
# Initialize the tool
code_interpreter = CodeInterpreterTool()

# Define an agent that uses the tool
data_analyst = Agent(
    role="Data Analyst",
    goal="Analyze data using Python code",
    backstory="""You are an expert data analyst who specializes in using Python
    to analyze and visualize data. You can write efficient code to process
    large datasets and extract meaningful insights.""",
    tools=[code_interpreter],
    verbose=True,
)

# Create a task for the agent
analysis_task = Task(
    description="""
    Write Python code to:
    1. Generate a random dataset of 100 points with x and y coordinates
    2. Calculate the correlation coefficient between x and y
    3. Create a scatter plot of the data
    4. Print the correlation coefficient and save the plot as 'scatter.png'

    Make sure to handle any necessary imports and print the results.
    """,
    expected_output="The correlation coefficient and confirmation that the scatte
    agent=data_analyst,
)

# Run the task
crew = Crew(
    agents=[data_analyst],
    tasks=[analysis_task],
    verbose=True,
    process=Process.sequential,
```

Implementation Details

The `CodeInterpreterTool` uses Docker to create a secure environment for code execution:

Code

```
class CodeInterpreterTool(BaseTool):
    name: str = "Code Interpreter"
    description: str = "Interprets Python3 code strings with a final print statem
    args_schema: Type[BaseModel] = CodeInterpreterSchema
    default_image_tag: str = "code-interpreter:latest"

    def _run(self, **kwargs) -> str:
        code = kwargs.get("code", self.code)
        libraries_used = kwargs.get("libraries_used", [])

        if self.unsafe_mode:
            return self.run_code_unsafe(code, libraries_used)
        else:
            return self.run_code_safety(code, libraries_used)
```

The tool performs the following steps:

1. Verifies that the Docker image exists or builds it if necessary
2. Creates a Docker container with the current working directory mounted
3. Installs any required libraries specified by the agent
4. Executes the Python code in the container
5. Returns the output of the code execution
6. Cleans up by stopping and removing the container

Security Considerations

1. The Docker container has access to the current working directory, so sensitive files could potentially be accessed.
2. If the Docker container is unavailable and the code needs to run safely, it will be executed in a sandbox environment. For security reasons, installing arbitrary libraries is not allowed
3. The `unsafe_mode` parameter allows code to be executed directly on the host machine, which should only be used in trusted environments.
4. Be cautious when allowing agents to install arbitrary libraries, as they could potentially include malicious code.

Conclusion

The `CodeInterpreterTool` provides a powerful way for CrewAI agents to execute Python code in a relatively secure environment. By enabling agents to write and run code, it significantly expands their problem-solving capabilities, especially for tasks involving data analysis, calculations, or other computational work. This tool is particularly useful for agents that need to perform complex operations that are more efficiently expressed in code than in natural language.

RAG Tool
Previous

Overview
Next