AgentCore 1P Tool - AgentCore Code Interpreter

Advanced Data Analysis with Code Interpreter

Introduction

Amazon Bedrock AgentCore Code Interpreter is a secure, managed tool that enables AI agents to execute code in isolated sandbox environments. This capability allows agents to perform advanced data analysis, generate visualizations, and process files while maintaining strict security boundaries. The Code Interpreter tool is frame work agnostic and integrates seamlessly with AI frameworks like Strands and LangChain, enabling you to build sophisticated agents that combine natural language reasoning with computational capabilities.

Why is this Important?

In modern AI applications, agents often need to perform complex data analysis and computations that go beyond simple text generation. Without secure code execution capabilities:

Security risks arise when agents execute arbitrary code in production environments **Limited analytical capabilities** restrict agents to basic text-based responses **Infrastructure complexity** requires building and maintaining custom sandboxed environments **Data processing limitations** prevent agents from working with files, datasets, and visualizations **Compliance concerns** emerge when code execution lacks proper isolation and audit trails

AgentCore Code Interpreter solves these challenges by providing a fully managed, secure execution environment with built-in managed code execution sandbox capabilities. The Code Interpreter supports various programming languages including Python, JavaScript, and TypeScript, making it versatile for different use cases.

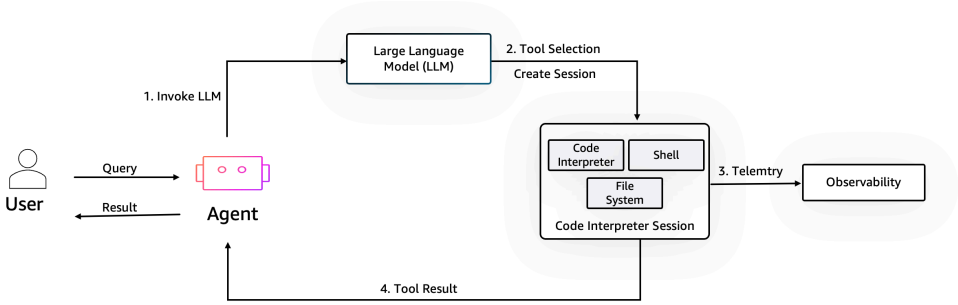
What You Will Learn/Build

In this lab, you will:

- **Create a sandbox environment** and upload data to perform analysis
- **Build an agent** that will perform advanced data analysis by generating code based on the user query
- **Execute code** securely in the sandbox environment using Code Interpreter
- **Process results** and handle file operations

Lab Architecture

This lab demonstrates how to create an AI agent that safely processes user queries by executing generated code within an isolated sandbox environment. The Code Interpreter provides a secure execution environment with Python capabilities, file system access, and visualization tools.



Prerequisites

To execute this lab you will need:

- Python 3.10+
- AWS credentials with Bedrock AgentCore permissions
- Access to Claude 3.7 Sonnet model in US Oregon (us-west-2) region
- Required Python packages: bedrock-agentcore, langchain
- IAM permissions for Code Interpreter operations

Required IAM Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "bedrock-agentcore:CreateCodeInterpreter",
        "bedrock-agentcore:StartCodeInterpreterSession",
        "bedrock-agentcore:InvokeCodeInterpreter",
        "bedrock-agentcore:StopCodeInterpreterSession",
        "bedrock-agentcore:DeleteCodeInterpreter",
        "bedrock-agentcore:ListCodeInterpreters",
        "bedrock-agentcore:GetCodeInterpreter"
      ],
      "Resource": "*"
    }
  ]
}
```

How Code Interpreter Works

The Code Interpreter creates a secure sandbox environment where AI agents can:

- **Generate Code**: LLM analyzes user queries and generates appropriate Python code
- **Execute Safely**: Code runs in isolated environment with controlled resources
- **Access Data**: Read and process uploaded files within the sandbox
- **Create Visualizations**: Generate charts, graphs, and visual analysis
- **Maintain State**: Preserve variables and context across multiple executions

Setting Up the Code Interpreter

> ***Note:*** *You can get started with the AgentCore Code Interpreter using either the AgentCore SDK or boto3. In this lab we will use the AgentCore SDK.*

To begin using AgentCore Code Interpeter, initialize the Code Interpreter client and start a session, we will extended the timeout duration since we will be performing complex analysis:

```python
from bedrock_agentcore.tools.code_interpreter_client import CodeInterpreter

# Initialize Code Interpreter within a supported AWS region
code_client = CodeInterpreter('us-west-2')

# Start Code Interpreter sandbox session with extended timeout (20 minutes)
session_id = code_client.start(session_timeout_seconds=1200)
print(f"Session started: {session_id}")
```

Working with Data Files

In this lab, we'll analyze a sample dataset with information about names, cities, animals, and things:

Our goal is to analyze this file using an agent to understand distributions and outliers, but first we will prepare and upload the data to the sandbox environment, the following functions show how to do so:

```python
# Prepare data files for sandbox environment
files_to_create = [{
    "path": "data.csv",
    "text": data_file_content
}]

# Write files to the sandbox
response = code_client.invoke("writeFiles", {"content": files_to_create})
print("Files uploaded successfully")

# Verify files in sandbox
listing_files = code_client.invoke("listFiles", {"path": ""})
print("Available files:", files_list)
```

Creating the Data Analysis Agent

Here we define a function as a tool that will be used by the agent to executer code in the Code Interpreter sandbox environment:

> ***Note:*** *In the language parameter we have specified python as the coding language. However, Code Interpreter supports other languages such as Typescript and Javascript.*

```python
# Define code execution tool using the @tool decorator to create a custom tool for the Age
@tool
def execute_python(code: str, description: str = "") -> str:
    """Execute Python code in the sandbox."""

    if description:
        code = f"# {description}\n{code}"

    # Print generated Code to be executed
    print(f"\n Generated Code: {code}")

    # Call the Invoke method and execute the generated code within the initialized code interpret
    response = code_client.invoke("executeCode", {
        "code": code,
        "language": "python",
        "clearContext": False
    })
    for event in response["stream"]:
        return json.dumps(event["result"])
```

Exploratory Data Analysis

Now we execute our agent with with Code Interpreter tool to perform EDA on our csv file.

```
query = "Perform exploratory data analysis on 'data.csv'. Show distributions and identify c
response = agent_executor.invoke({"input": query})
print(response['output'])
```

Cleanup

Finally, we'll clean up by stopping the Code Interpreter session. Once finished using a session, the session should be stopped to release resources and avoid unnecessary charges.

```
# Stop session when finished
code_client.stop()
print("Code Interpreter session stopped successfully!")
```

Security and Best Practices

- **Isolated Execution**: All code runs in secure, isolated environments
- **Session Timeouts**: Configure appropriate timeouts to prevent errors, the Code Interpreter tool execution time can be extended for up to 8 hours
- **Data Privacy**: Files and variables are contained within session boundaries
- **Resource Management**: Stop sessions when finished to release compute resources
- **Error Handling**: Implement proper exception handling for code execution failures

Congratulations!

You have successfully learned how to use Amazon Bedrock AgentCore Code Interpreter to create AI agents that perform advanced data analysis through secure code execution. This powerful combination enables safe, reliable data processing workflows with AI-generated code.

Discover what this template is all about and the core concepts behind it.

Try it out

At an AWS Event

If you are following the workshop via workshop studio, now go to JupyterLab in SageMaker Studio. In the JupyterLab UI navigate to `05-AgentCore-tools/01-Agent-Core-code-interpreter 03-advanced-data-analysis-with-agent-using-code-interpreter.`

Self-paced

Here's a notebook that lets you try out the above: Advanced Data Analysis using Amazon AgentCore Bedrock Code Interpreter.