

# Model Driven Agents:

## How AWS Moved Beyond Orchestration with Strands SDK

Nicholas Clegg  
Senior Software Engineer - AWS  
Strands Agents SDK

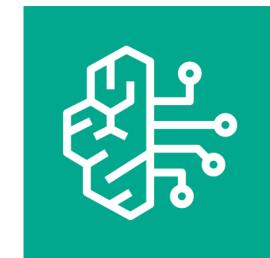




Kiro IDE



Q Developer



AWS Bedrock





## Agenda:

- When Orchestration Breaks Down
- The Model-Driven Approach
- Your First Strands Agent
- Multi and Meta Agents
- Takeaways

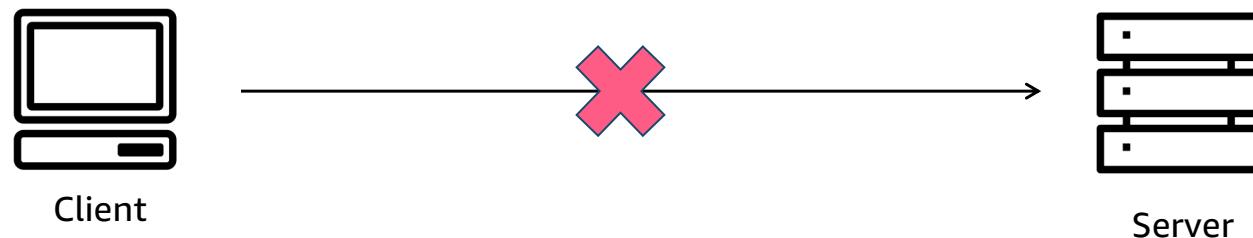
# When Orchestration Breaks Down



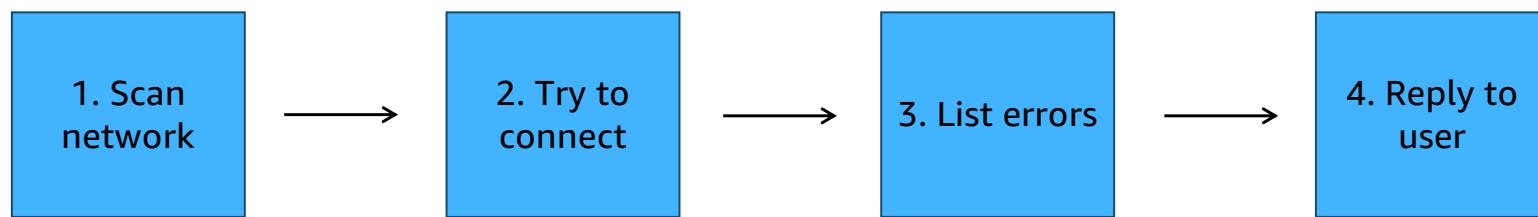
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

## The Problem: Analyzing a Network

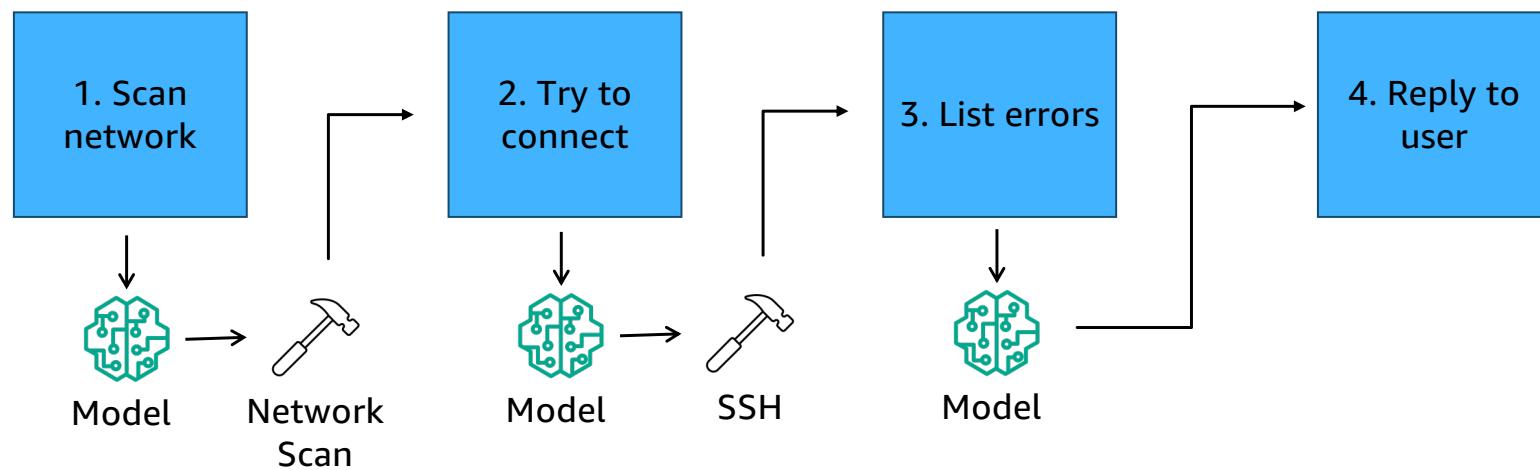
- Customers had a hard time debugging network connections
- Build an Agent that can analyze network issues



# Network Analyzer Design - Workflow



# Network Analyzer Design - Workflow





User

Why can't I connect?



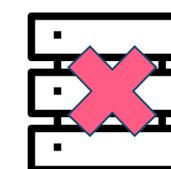
Amazon Q

Let me scan the network...

I didn't find anything wrong.



Client



Server

I still can't connect...



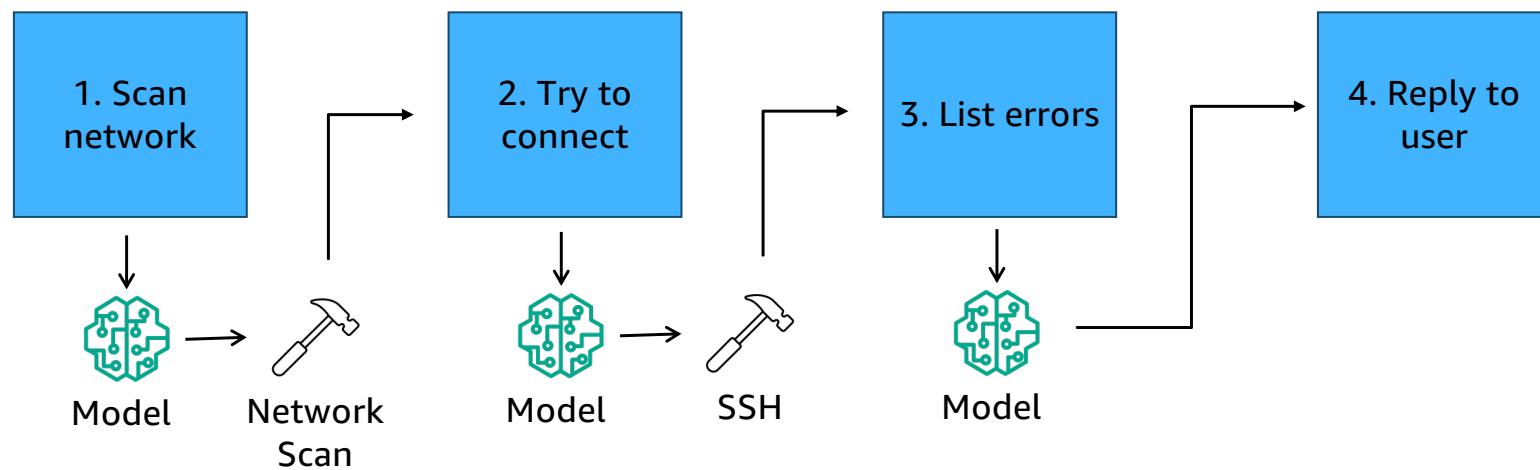
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Network Analyzer Issues

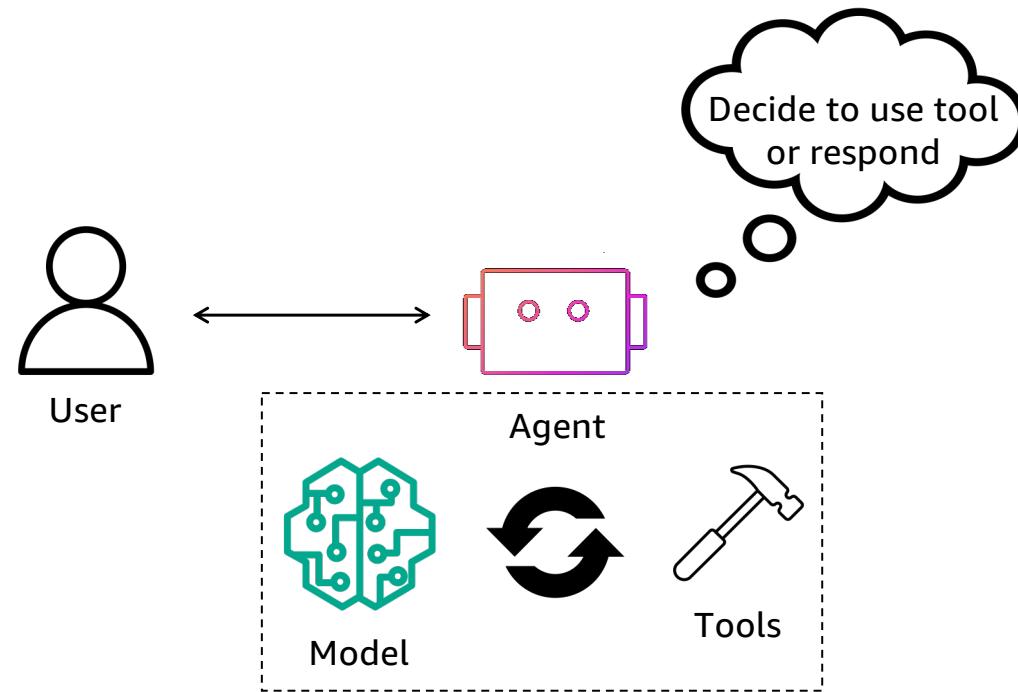
- Small number of “success” cases
- Complex state machines and error handling
- Frustrating error messages



# Network Analyzer Design - Workflow



# Network Analyzer Design – Model Driven



# The Model-Driven Approach



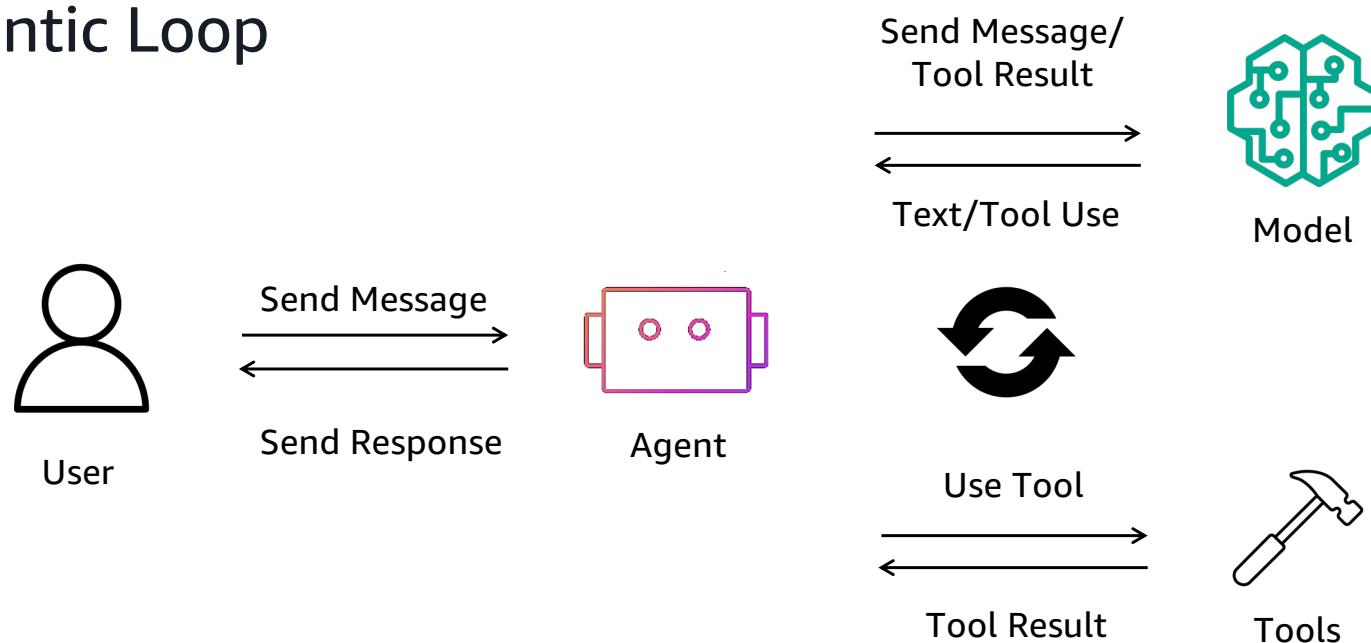
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# The Model-Driven Approach

- Let the Agent decide
- Provide flexible tools
- Informative error messages



# Agentic Loop





User

Why can't I connect?



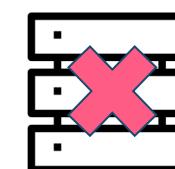
Amazon Q

NetworkScan: []



Client

I checked the network, and  
there were no servers.  
Did you turn it on?



Server



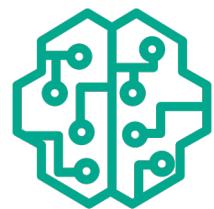
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Your First Strands Agent



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

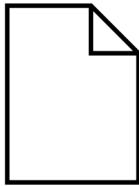
# Building Blocks of an Agent



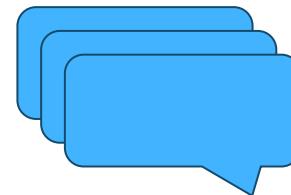
Model



Tools



System Prompt



Context

# Writing a Strands Agent

```
from strands import Agent
from strands_tools import calculator

agent = Agent(
    model="global.anthropic.claude-sonnet-4-5-20250929-v1:0",
    tools=[calculator],
    system_prompt="You are a helpful assistant who talks like a pirate!"
)

agent("what is the sqrt of 1764?")
```



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Write Your Own Tools

```
from strands import Agent, tool

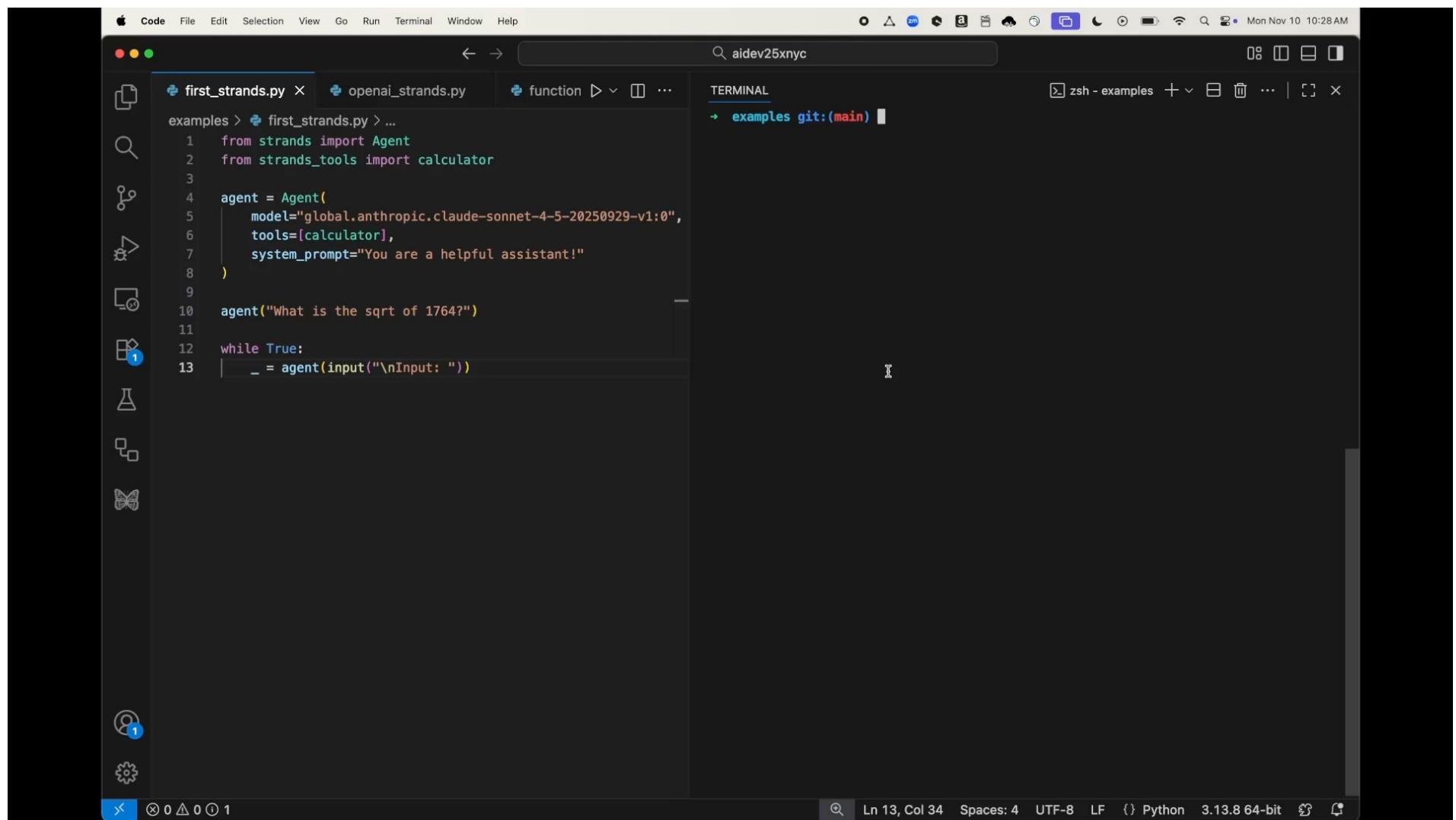
@tool
def multiply(x: int, y: int) -> int:
    return x * y

agent = Agent(
    tools=[multiply],
)

agent("what is 123 * 456")
```



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.



A screenshot of a dark-themed code editor, likely VS Code, showing a Python script and a terminal window.

**Code Editor (Left Panel):**

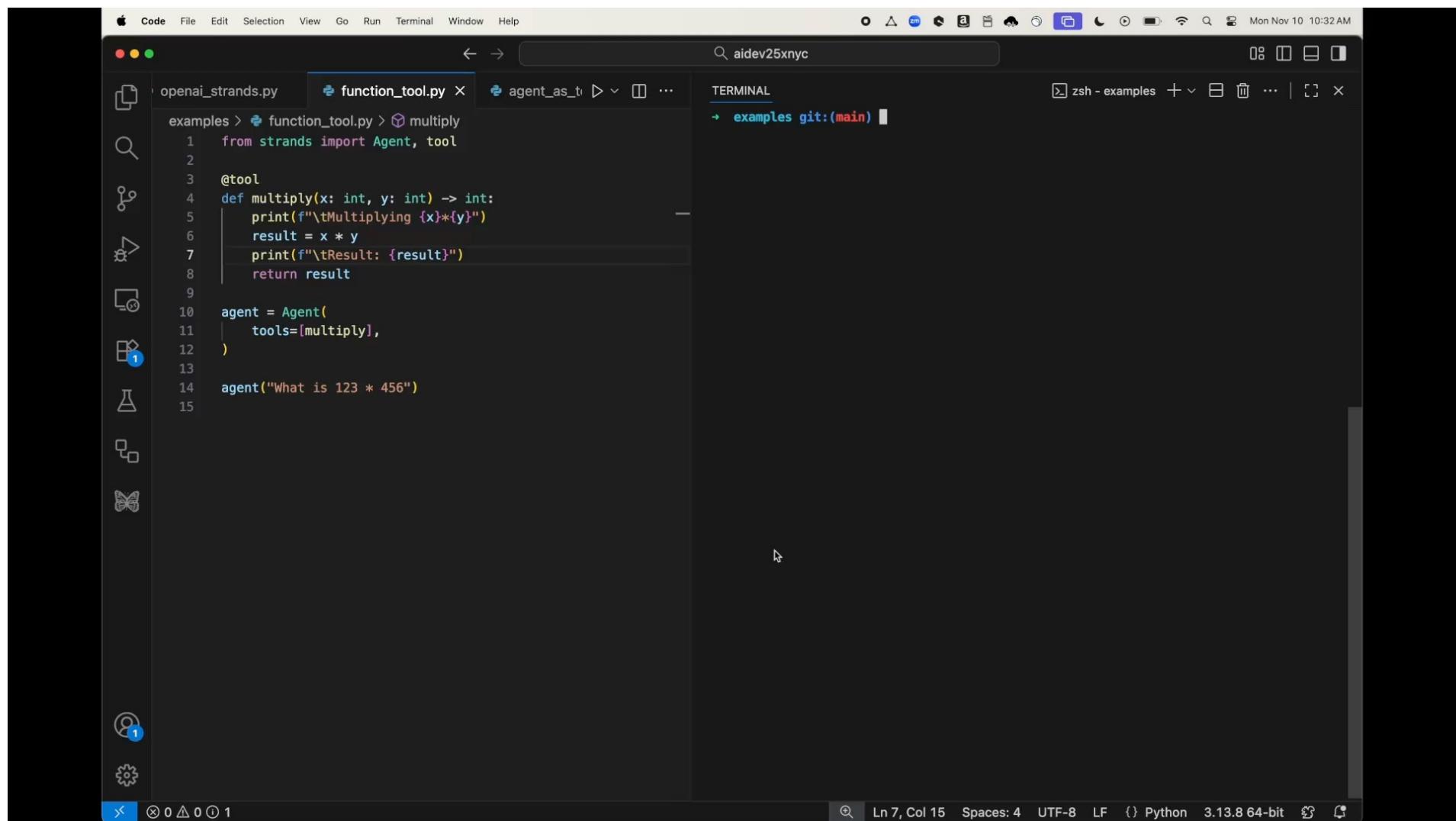
- File: `first_strands.py`
- File: `openai_strands.py`
- File: `function`

```
examples > first_strands.py > ...
1  from strands import Agent
2  from strands_tools import calculator
3
4  agent = Agent(
5      model="global.anthropic.claude-sonnet-4-5-20250929-v1:0",
6      tools=[calculator],
7      system_prompt="You are a helpful assistant!"
8  )
9
10 agent("What is the sqrt of 1764?")
11
12 while True:
13     _ = agent(input("\nInput: "))
```

**Terminal (Right Panel):**

```
TERMINAL
→ examples git:(main) █
```

Bottom status bar: `Ln 13, Col 34`, `Spaces: 4`, `UTF-8`, `LF`, `{ } Python 3.13.8 64-bit`



A screenshot of the Visual Studio Code (VS Code) interface on a Mac. The window title is "Code" and the search bar at the top right contains the text "aidev25xnyc". The main area shows a code editor with a dark theme, displaying Python code. The code defines a function `multiply` that prints the multiplication of two numbers and returns the result. It then creates an `Agent` with the `multiply` tool and asks it a question. The terminal tab is active, showing a terminal window with the command `examples git:(main)`. The sidebar on the left contains various icons for file operations like copy, paste, and search, as well as a status bar at the bottom with icons for file status, search, and code analysis.

```
from strands import Agent, tool

@tool
def multiply(x: int, y: int) -> int:
    print(f"\tMultiplying {x}*{y}")
    result = x * y
    print(f"\tResult: {result}")
    return result

agent = Agent(
    tools=[multiply],
)

agent("What is 123 * 456")
```

Code File Edit Selection View Go Run Terminal Window Help

aidev25xnyc

first\_strands.py function\_tool.py openai\_strands.py TERMINAL zsh - examples

```
examples > openai_strands.py > ...
1 import os
2 from strands import Agent
3 from strands.models.openai import OpenAIModel
4 from strands_tools import calculator
5
6 openai_model = OpenAIModel(
7     model_id="gpt-4",
8     client_args={
9         "api_key": os.environ.get("OPENAI_API_KEY")
10    }
11 )
12
13 agent = Agent(
14     model=openai_model,
15     tools=[calculator],
16     system_prompt="You are a helpful assistant."
17 )
18
19 agent("What is the sqrt of 1764?")
20
21 while True:
22     _ = agent(input("\nInput: "))
```

TERMINAL

examples git:(main)

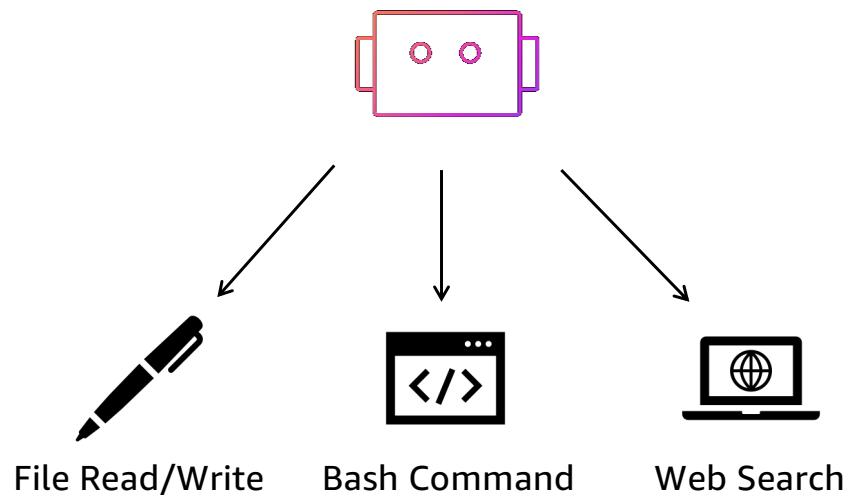
Ln 11, Col 2 Spaces: 4 UTF-8 LF {} Python 3.13.8 64-bit

# Multi and Meta Agents

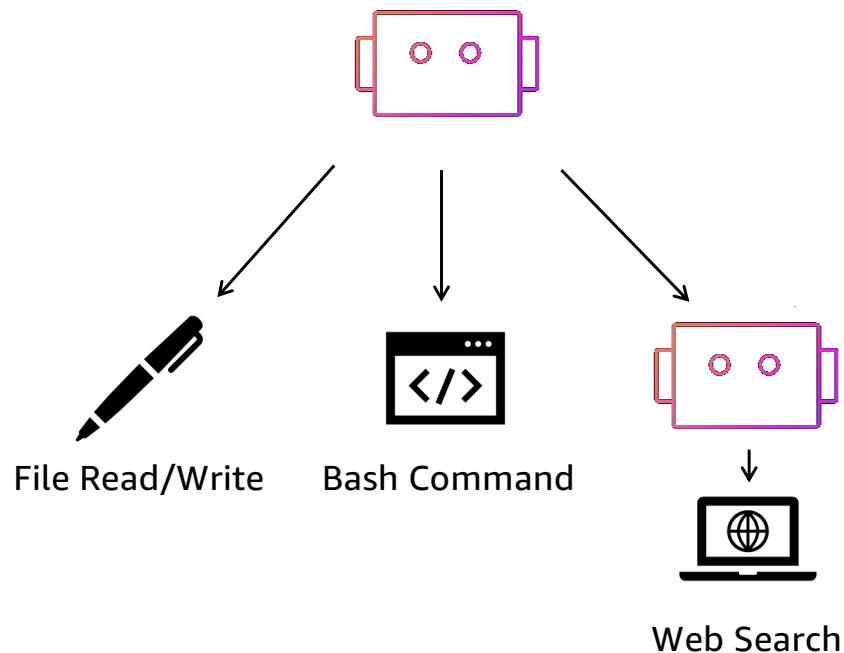


© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Basic Agent



# Agents as Tools



Code File Edit Selection View Go Run Terminal Window Help

ai25xny

agent\_as\_tool.py

```
examples > agent_as_tool.py > ...
1  from strands import Agent, tool
2  from strands_tools import tavity
3  import json
4
5  web_researcher_agent = Agent(
6      system_prompt=
7          "You research a query and give a one sentence summary",
8      tools=[tavity]
9  )
10
11 @tool
12 def web_researcher(query: str) -> str:
13     summary = str(web_researcher_agent(query))
14     return summary
15
16 agent = Agent(
17     tools=[web_researcher],
18 )
19
20 agent("Tell me about Pine Trees.")
21
22
23 print(f"\n\nMain Agent context: " \
24     + f"{len(json.dumps(agent.messages))}")
25 print(f"Web Agent context: " \
26     + f"{len(json.dumps(web_researcher_agent.messages))}")
27
28 while True:
29     _ = agent(input("\nInput: "))
```

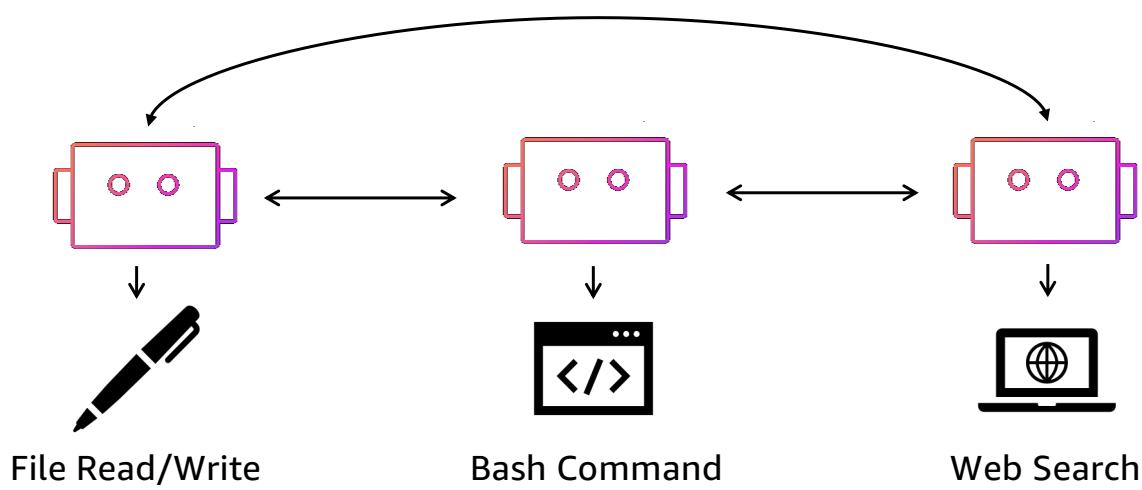
TERMINAL

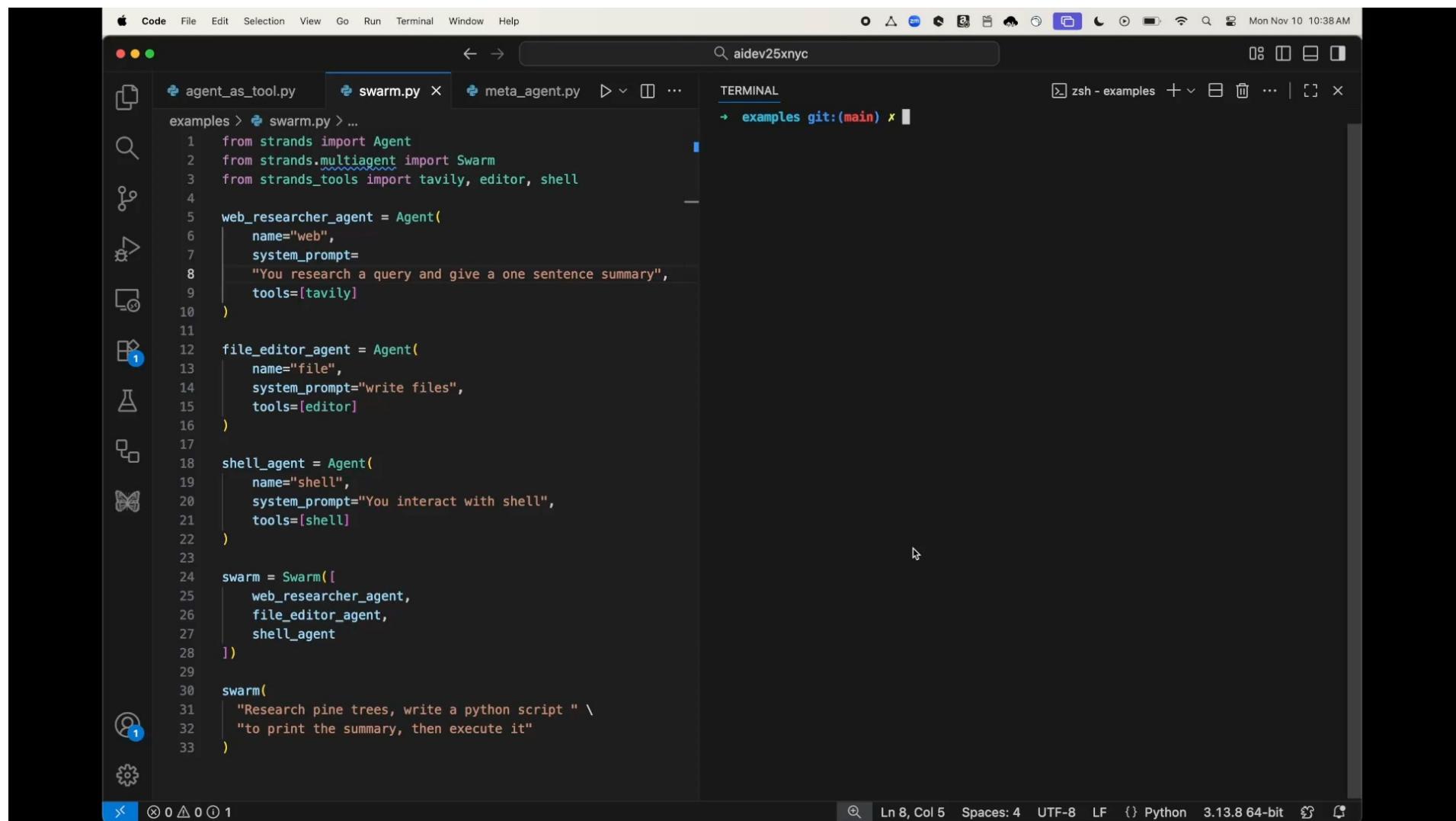
examples git:(main)

zsh - examples

Ln 26, Col 60 Spaces: 4 UTF-8 LF {} Python 3.13.8 64-bit

# Swarms

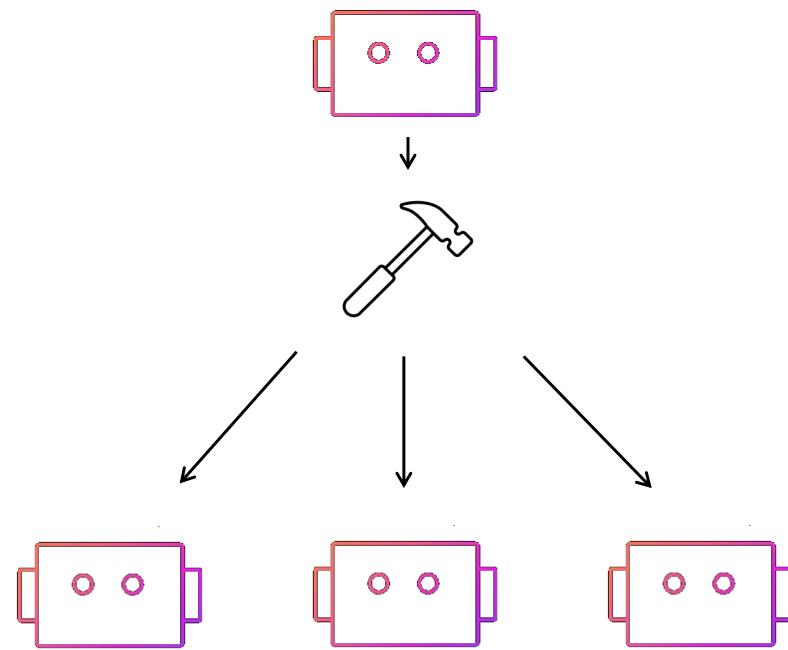




A screenshot of a Mac OS X desktop environment. The main window is a Code editor with a dark theme, showing Python code for a multiagent system. The code defines several agents: a web researcher agent, a file editor agent, and a shell agent, which are then combined into a Swarm. The terminal window to the right shows a command-line session in zsh, with the current directory being 'examples' and the command 'git:(main)' being run. The status bar at the bottom of the screen displays file statistics (0△0①1), encoding (UTF-8), and a Python version (3.13.8 64-bit).

```
examples > swarm.py > ...
1  from strands import Agent
2  from strands.multiprocess import Swarm
3  from strands_tools import tavily, editor, shell
4
5  web_researcher_agent = Agent(
6      name="web",
7      system_prompt=
8          "You research a query and give a one sentence summary",
9      tools=[tavily]
10 )
11
12 file_editor_agent = Agent(
13     name="file",
14     system_prompt="write files",
15     tools=[editor]
16 )
17
18 shell_agent = Agent(
19     name="shell",
20     system_prompt="You interact with shell",
21     tools=[shell]
22 )
23
24 swarm = Swarm([
25     web_researcher_agent,
26     file_editor_agent,
27     shell_agent
28 ])
29
30 swarm(
31     "Research pine trees, write a python script " \
32     "to print the summary, then execute it"
33 )
```

# Meta Agents



```
examples > meta_agent.py > ...
1  import json
2  import logging
3  from strands import Agent
4  from strands_tools import use_agent, file_read, file_write, shell
5
6  # Enables Strands debug log level
7  logging.getLogger("strands_tools.use_agent").setLevel(logging.DEBUG)
8
9  # Sets the logging format and streams logs to stderr
10 logging.basicConfig(
11     format="%(levelname)s | %(name)s | %(message)s",
12     handlers=[logging.StreamHandler()]
13 )
14
15 agent = Agent(
16     system_prompt=
17     "You MUST use the `use_agent` tool write or run files.",
18     tools=[use_agent, file_read, file_write, shell]
19 )
20
21 agent(
22     "Write a script to print a christmas tree, then run it"
23 )
24
25 print(f"\n\nMain Agent context: " \
26     + f"\n{len(json.dumps(agent.messages))}")
27
28 while True:
29     _ = agent(input("\nInput: "))
```

# Takeaways



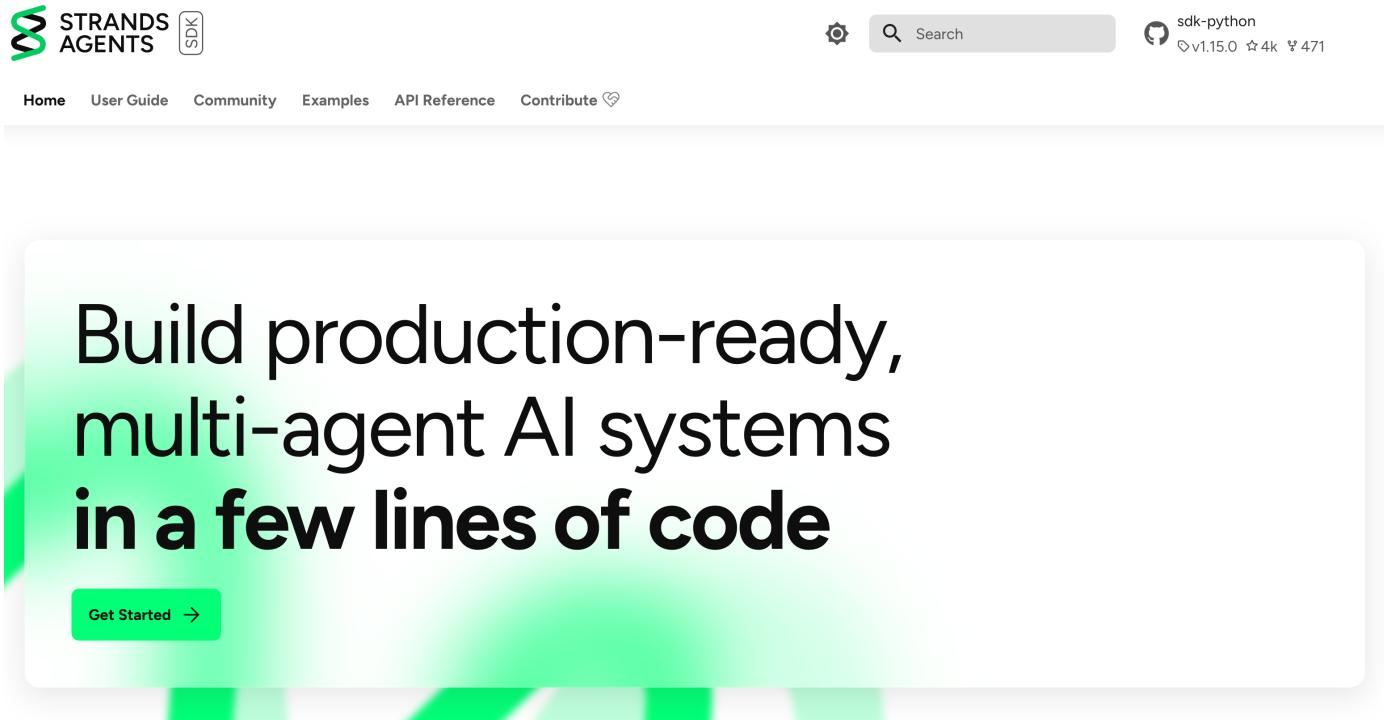
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Design Model-Driven Agents

- Put the Agent in control
- Informative Errors
- Flexible Tool
- Think about context bloat

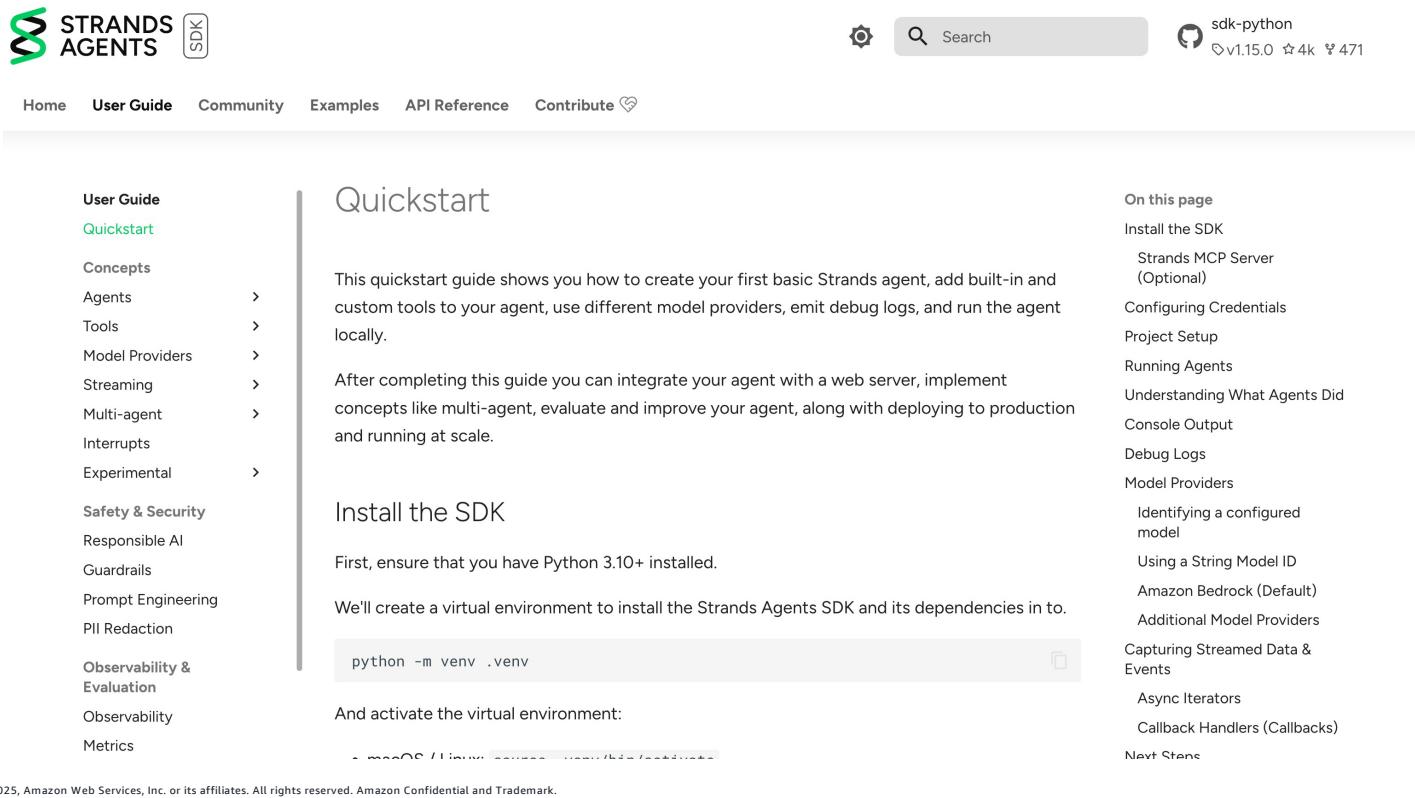


# Start building a Strands Agent! – strandsagents.com



The screenshot shows the homepage of strandsagents.com. At the top, there is a navigation bar with links for Home, User Guide, Community, Examples, API Reference, and Contribute. On the right side of the header, there is a search bar, a gear icon for settings, and a GitHub icon for the repository "sdk-python" with the version "v1.15.0", 4k stars, and 471 forks. The main content area features a large green gradient box with the text "Build production-ready, multi-agent AI systems in a few lines of code" and a "Get Started" button. At the bottom of the page, there is a small AWS logo and a copyright notice: "© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.".

# Easy Getting Started Guide



The screenshot shows the Strands Agents User Guide website. At the top, there is a navigation bar with links for Home, User Guide (which is the active page), Community, Examples, API Reference, and Contribute. The User Guide page has a sidebar on the left with a tree view of topics like Concepts, Agents, Tools, Model Providers, Streaming, Multi-agent, Interrupts, Experimental, Safety & Security, Responsible AI, Guardrails, Prompt Engineering, PII Redaction, Observability & Evaluation, Observability, and Metrics. The main content area is titled "Quickstart". It contains two sections: "Install the SDK" and "Quickstart". The "Install the SDK" section includes a code block with the command "python -m venv .venv". The "Quickstart" section contains text about creating a basic agent and integrating with a web server. On the right side, there is a "On this page" sidebar with links to various documentation topics. At the bottom, there is a copyright notice for Amazon Web Services and a "Next Steps" link.

**User Guide**

[Quickstart](#)

Concepts

Agents >

Tools >

Model Providers >

Streaming >

Multi-agent >

Interrupts >

Experimental >

Safety & Security

Responsible AI

Guardrails

Prompt Engineering

PII Redaction

Observability & Evaluation

Observability

Metrics

## Quickstart

This quickstart guide shows you how to create your first basic Strands agent, add built-in and custom tools to your agent, use different model providers, emit debug logs, and run the agent locally.

After completing this guide you can integrate your agent with a web server, implement concepts like multi-agent, evaluate and improve your agent, along with deploying to production and running at scale.

### Install the SDK

First, ensure that you have Python 3.10+ installed.

We'll create a virtual environment to install the Strands Agents SDK and its dependencies in to.

```
python -m venv .venv
```

And activate the virtual environment:

```
source .venv/bin/activate
```

On this page

Install the SDK

Strands MCP Server (Optional)

Configuring Credentials

Project Setup

Running Agents

Understanding What Agents Did

Console Output

Debug Logs

Model Providers

Identifying a configured model

Using a String Model ID

Amazon Bedrock (Default)

Additional Model Providers

Capturing Streamed Data & Events

Async Iterators

Callback Handlers (Callbacks)

Next Steps

© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

# Thank you!

Nicholas Clegg

<https://www.linkedin.com/in/nicholasclegg>

