Creating an AWS DevOps AI Agent with the Strands Agents SDK

SPL-TF-300-MLAIST-1 - Version 1.0.0

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab overview

AnyCompany Airlines operates a complex multi-cloud infrastructure, requiring constant monitoring, maintenance, and rapid response to operational issues. Their DevOps and Site Reliability Engineering teams currently spend significant manual effort on routine tasks such as monitoring CloudWatch alarms, researching AWS best practices for new deployments, provisioning resources, and coordinating responses across multiple services. The company seeks to streamline these operations by implementing an intelligent AWS management assistant that can understand natural language requests and autonomously execute complex operational workflows across their infrastructure.

In this lab, you will build a comprehensive multi-agent AWS management system using the Strands Agents SDK and Amazon Bedrock models. You will progress from creating specialized agents for different operational domains—including coding, monitoring, research, and resource management—to building a sophisticated multi-agent orchestrator that can handle complex requests like “Deploy a web application that displays our user data from DynamoDB.” This demonstrates how AnyCompany Airlines can transform their manual, error-prone operational processes into an intelligent, conversational interface that reduces deployment time from hours to minutes.

Objectives

By the end of this lab, you will be able to do the following:

- Implement intelligent file analysis and code generation capabilities using built-in SDK tools.
- Build specialized monitoring solutions that integrate with AWS CloudWatch for real-time system health assessment.
- Construct research-enabled agents that access current AWS documentation through MCP servers.
- Develop agents capable of managing AWS services and automating resource deployment workflows.
- Design coordinated systems that translate natural language requests into complex operational workflows.

Icon key

- Caution: Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- Command: A command that you must run.
- Consider: A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
- Expected output: A sample output that you can use to verify the output of a command or edited file.
- Learn more: Where to find more information.
- Note: A hint, tip, or important guidance.
- Task complete: A conclusion or summary point in the lab.
- Warning: An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).

Start lab

1. To launch the lab, at the top of the page, choose **Start Lab**.
   Caution: You must wait for the provisioned AWS services to be ready before you can continue.
2. To open the lab, choose **Open Console** .
   You are automatically signed in to the AWS Management Console in a new web browser tab.
   Warning: Do not change the Region unless instructed.

Common sign-in errors

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the Start Lab button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

Lab environment

The following diagram shows the architecture you create in this lab:

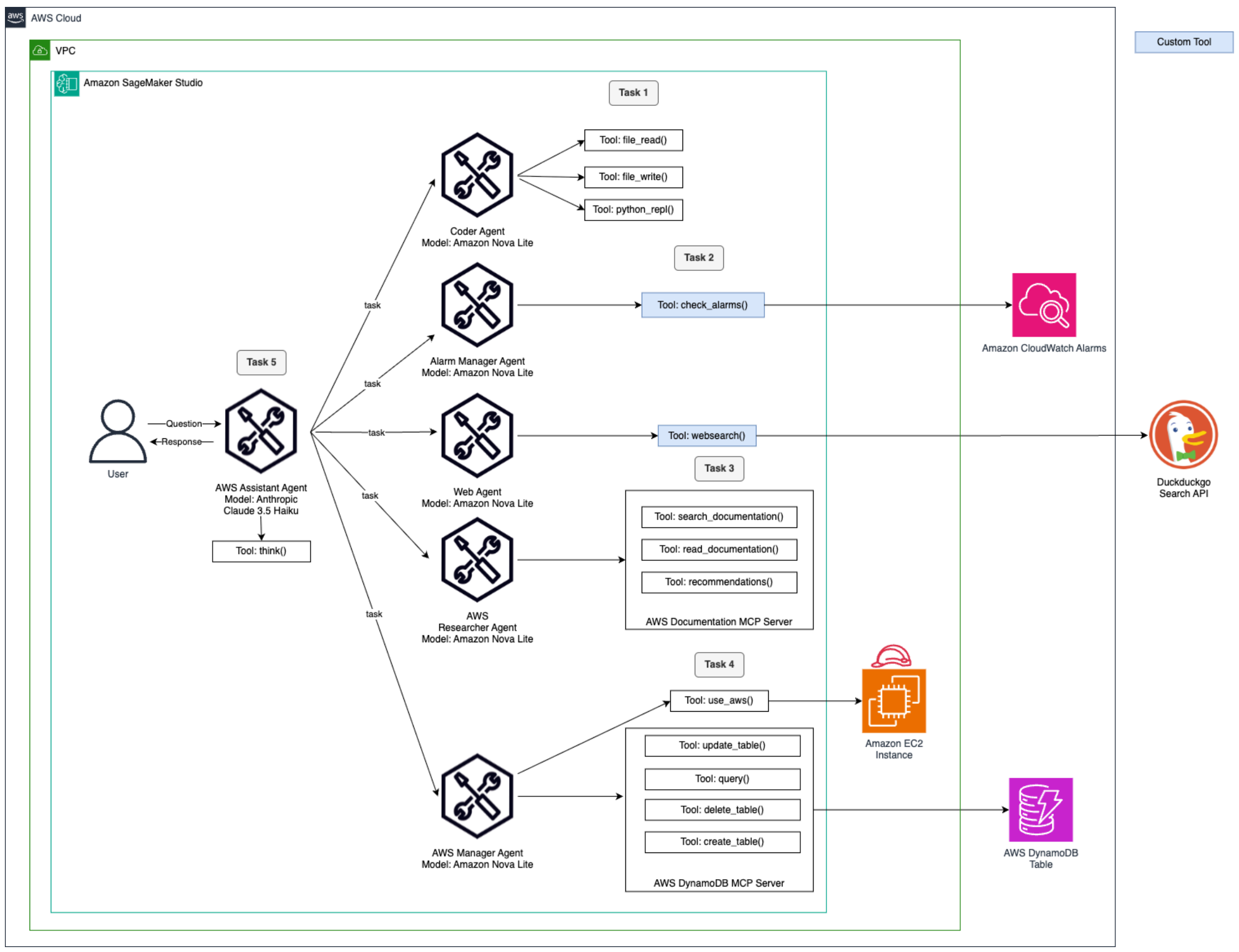

Image description: Five specialized agents (coder, alarm manager, AWS researcher, AWS manager, and orchestrator) work together to handle different aspects of AWS DevOps operations. The orchestrator agent coordinates between the specialized agents to handle complex multi-step requests.

The lab environment includes:

- Pre-configured EC2 instance with development tools
- IAM roles with necessary permissions for Bedrock and AWS services
- Access to Amazon Nova Lite, Nova Pro, and Claude Haiku 3.5 models

- Pre-installed Python environment with required packages

AWS services not used in this lab

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

Task 1: Create agents for coding, alarm monitoring, and web research

In this task, you connect to a development environment, and create agents for specialized tasks, like a coding assistant that can edit files and execute Python, an agent to detect issues in your AWS account, and one to perform web searches.

Task 1.1 Connect to Code Editor

A development environment with the needed authentication and model requests has been set up for you in the Code Editor environment. Connect to an integrated development environment (IDE) and install the necessary packages to use Strands Agents in the Kernel.

Prerequisites for using a Strands agent include:

- Requesting access to the Bedrock model your agent will use
- Using an IAM user or role that has been configured with the permission bedrock:InvokeModelWithResponseStream
- Having valid credentials for the IAM user or role configured in the executing environment or a Bedrock API key

Access to Amazon Nova Lite, Amazon Nova Pro, and Claude Haiku 3.5 has been requested for you and an instance profile with the necessary IAM permission has been attached to the lab EC2 instance.

3. To open the lab instance, copy the LabInstanceURL value that is listed to the left of these instructions. Paste the URL into a new web browser tab and press enter.
4. To open the file Explorer, from the activity bar at the left of Code Editor, choose .
5. To open the lab python notebook, choose lab.ipynb.
6. At the top right of the lab.ipynb editor, choose Select Kernel.
7. In the Select Kernel menu, choose Install/Enable suggested extensions.
8. On the Do you trust the publisher pop-up, choose **Trust Publisher & Install**.
9. At the top right of the lab.ipynb editor, choose Select Kernel again.

10. In the Select Kernel menu, choose Python Environments, then Create Python Environment, then Venv, then choose the newest Python version available from the list.
11. To install Python dependencies, select requirements.txt and choose OK.
 Note: Wait up to a minute for the packages to install.
12. Carefully advance through the lab.ipynb notebook. Run each code cell and review the completed output.
13. To run a cell, select within the cell and press Shift + enter or, at the top of the page, choose the Run button.

When you have finished running the notebook, return here and resume instructions.

 Task complete: You successfully created AnyCompany's first agents capable of a wide array of tasks.

Task 2: Integrate real-time AWS documentation access with an MCP

The *Model Context Protocol (MCP)* is an protocol that standardizes how applications provide context to LLMs. MCP makes the work of building complex and capable agents simple.

In this task, you use the *AWS Documentation MCP* to make an agent thats an expert in up-to-date AWS infrastructure and capabilities.

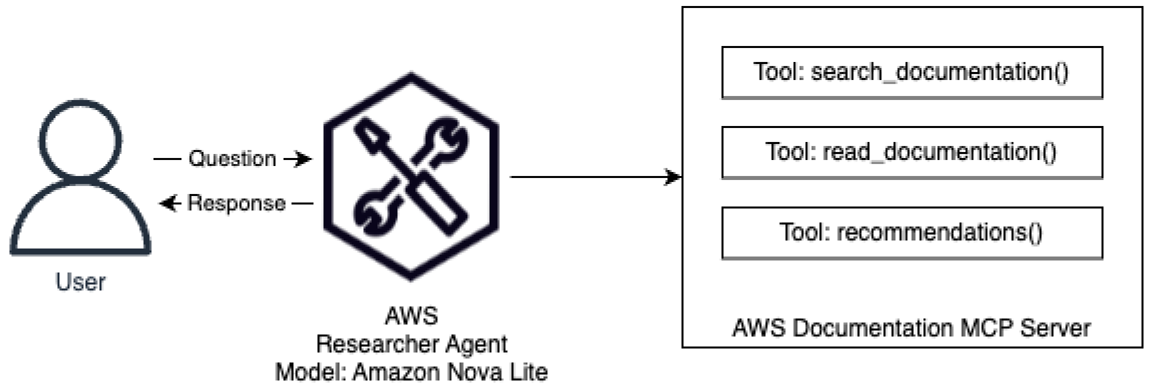

Image description: The agent you create in task 3 uses the tools created by the AWS Documentation MCP.

 Learn more: AWS publishes many MCP servers to assist with useful tasks. Refer to *AWS Documentation MCP Server* in the Additional resources section for more information.

Build this agent in a Python file instead of a notebook.

14. In the Code Editor file explorer, choose agents/aws_researcher.py.

One way of connecting to externally hosted MCP servers is with a *Standard Input/Output (stdio)* server. *UVX* is a Python package manager that has been pre-installed on your Code Editor.

To create a stdio client to the AWS documentation server, within the function, add the following code:

```
documentation_mcp_server = MCPClient(
    lambda: stdio_client(
        StdioServerParameters(
            command="uvx", args=["awslabs.aws-documentation-mcp-server"]
        )
    )
```

15.)

To setup the MCP server's tools, within the function, add the following code:

```
with documentation_mcp_server:
```

16.   tools = documentation_mcp_server.list_tools_sync() + [file_write]

Learn more: The AWS Documentation MCP Server has three tools: *read_documentation*, *search_documentation*, and *recommend* to extend your agent's functionality. Refer to *AWS Documentation MCP Server* in the Additional resources section for more information.

To create the agent and give it a strictly defined purpose, within the with statement, add the following agent declaration, system prompt, and tool set:

```
# Create the research agent with specific capabilities
aws_documentation_agent = Agent(
    model=nova_lite,
    system_prompt="""You are a thorough AWS researcher specialized in
finding accurate
    information online. For each question:

    1. Determine what information you need
    2. Search the AWS Documentation for reliable information
    3. Extract key information and cite your sources
    4. Store important findings in memory for future reference
    5. Synthesize what you've found into a clear, comprehensive answer

    When researching, focus only on AWS documentation. Always provide
citations
    for the information you find.
    """,
    tools=tools,
```

17.)

To invoke the agent using the input query, and return its response after type converting it to an *str*, within the with statement, add the following code:
```
response = aws_documentation_agent(query)
```

18. `return str(response)`
19. Compare aws_researcher.py with the following complete code:
    aws_research.py full code
20. Command: To active the Python virtual environment in the terminal, run the following command:
    ```
    source .venv/bin/activate
    ```

Command: To test the agent, having it give you information from official AWS documentation, run the following command:
```
python -u agents/aws_researcher.py
```
Note: If the agent asks to write a file with the information, enter
`y` or `*` to either allow or deny it.
Expected output
```
************************
**** EXAMPLE OUTPUT ****
************************

...

Tool #1: search_documentation

...

<thinking>I have found the relevant documentation for Amazon Bedrock. The
most relevant information is found in the 'Amazon Bedrock User Guide'. I
will extract key information from this documentation and synthesize it
into a clear, comprehensive answer.</thinking>

Tool #2: read_documentation

...

<thinking>I have extracted the key information from the documentation and
synthesized it into a clear, comprehensive answer. I will now provide the
answer to the User and store important findings in memory for future
reference.</thinking>

**What is Amazon Bedrock?**

Amazon Bedrock is a fully managed service that provides access to
high-performing foundation models (FMs) from leading AI companies and
Amazon. It offers a unified API to choose from a wide range of foundation
models best suited for various use cases. Amazon Bedrock also provides
```

```
capabilities to build generative AI applications with security, privacy,
and responsible AI.
```

21. `**Key Features of Amazon Bedrock:**`

Task complete: You successfully integrated real-time AWS documentation access into your agent using the AWS Documentation MCP server.

Task 3: Automate AWS resource management

In this task, you build and test AWS resource management capabilities with Strands Agents built-in tools and easily usable MCP servers.

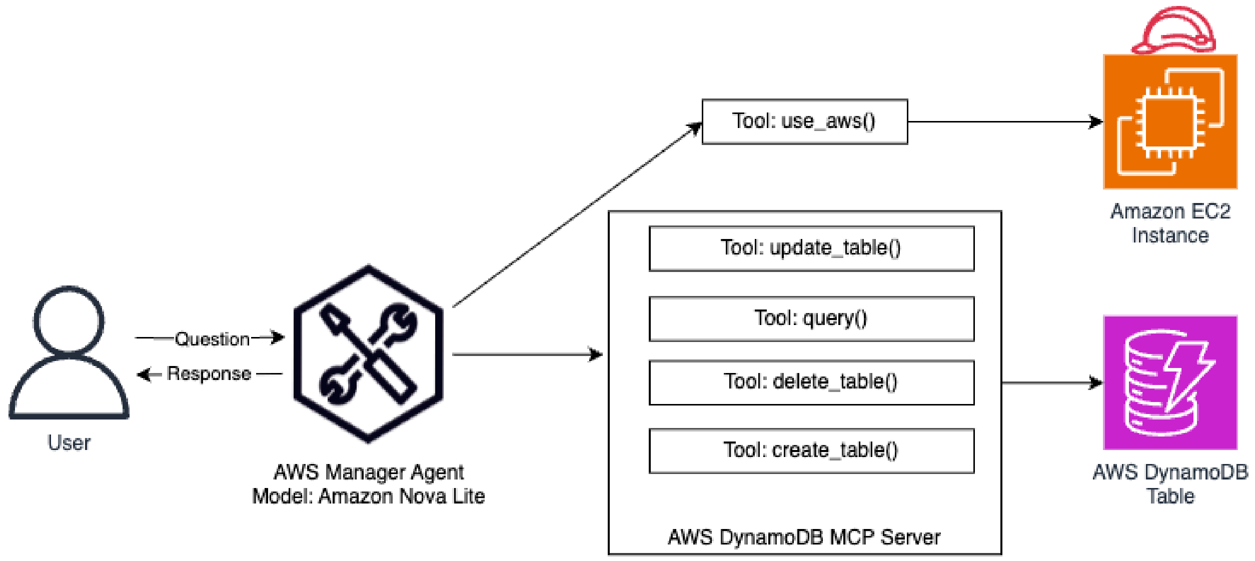

Image description: The agent you create in task 4 uses the tools created by the Amazon DynamoDB MCP and the use_aws tool.

Task 3.1: Create a DynamoDB table with an agent

Use an already created agent to create and populate a DynamoDB table with the flight information created earlier.

22. From the Code Editor file explorer, open agents/aws_manager.py.
23. Command: To clear the terminal, run the following command:
    ```
    clear
    ```

The agent code for utilizing the DynamoDB MCP is written in the file.

24. Command: To run the aws_manager agent, execute the following command:
    ```
    python -u agents/aws_manager.py
    ```

Enter the following prompt, replacing REGION with the value of PrimaryRegion that is to the left of these instructions:

```
Create a dynamodb table called flights in REGION with string FlightNumber
as the partition key. Sleep then check that the table has been
successfully created and then use the data from
/home/ec2-user/environment/project/flights.csv to add items to the table
```

Note: The agent will possibly fail to create the table several times, learning from each mistake the correct parameters to assume and include in an AWS request. If a different system prompt was used, the agent could ask you to provide it with missing parameters. This agent has been given more authority to make assumptions while creating the DynamoDB table.

Expected output

```
***********************
**** EXAMPLE OUTPUT ****
***********************

Tool #1: create_table

Tool #2: sleep

Tool #3: describe_table

Tool #4: file_read

...

Tool #5: create_table

Tool #6: sleep

Tool #7: describe_table

Tool #8: file_read

...

<thinking> The table is still in the 'CREATING' state. I'll need to wait a
bit longer before checking the table status again and then proceed to add
the items from the CSV file. </thinking>
```

```
...

Tool #9: sleep

Tool #10: describe_table

<thinking> The table has been successfully created and is now in the
'ACTIVE' state. I'll now proceed to add the items from the CSV file to the
table using the `put_item` tool. </thinking>
Tool #12: put_item

Tool #13: put_item

Tool #14: put_item

...
```

25. `The items from the `/home/ec2-user/environment/project/flights.csv` file have been added to the DynamoDB table 'flights'. Each item was added using the `put_item` tool, and each operation consumed 1 Capacity Unit as expected.`
    Note: If your agent's output does not imply successful table creation, continue prompting the agent to get the indented result.
26. To verify the table's creation, in a browser different tab, open the AWS Management console, and in the search bar, search for and choose
    `DynamoDB`.
27. In the left navigation pane, choose Tables.
    Note: The flights table should be created. If it has not been or it's apparent the table is not populated in the subsequent steps, continue prompting the agent until it is.
28. Select the flights table.
29. Choose **Explore table items** .
    Note: The flights table should be populated with AnyCompany flights data.
30. After the agent confirms it has finished creating and populating the table, to exit the prompt, enter
    `exit`.

Task 3.2: Add additional AWS capabilities to the agent

To perform general tasks in AWS, add the *use_aws* built-in tool to the aws_manager agent.

31. At the top of aws_manager.py, add code to import the tool use_aws from strands_tools.

32. To add the use_aws tool to the agent's tools parameter, modify it to look like the following:
    ```
    tools=stdio_mcp_client.list_tools_sync() + [file_read, sleep, 
    use_aws],
    ```

To prompt the agent to use the new tool when applicable, change the system prompt to the following:

```
system_prompt="""You are an AWS operations assistant with access to 
multiple AWS services.

When handling requests:
- For DynamoDB operations: Use the specialized DynamoDB tools
- For all other AWS services: Use the use_aws tool
    - For ec2 actions: use the service name "ec2". EC2 instances do not 
need a key pair. Use AMI ID from launch template.
    - s3: Bucket and object operations
    - iam: User and role management
    - Other AWS services
- For file operations: Use the file_read tool
- To sleep: Use the sleep tool

Example uses:
- "Create an EC2 instance" → use the use_aws tool
- "Query DynamoDB table" → use DynamoDB-specific tools
- "Read configuration file" → use file_read tool
```

33. `- "Sleep for 10 seconds" → use sleep tool""",`
34. Command: To run the aws_manager, execute the following command:
    ```
    python -u agents/aws_manager.py
    ```

Note: If there was an issue with running the agent, check your code against the following completed code.

aws_manager.py full code

To list s3 buckets in your account, enter the following prompt:

```
Tell me what S3 buckets I have in us-east-1
```

Expected output

```
************************
**** EXAMPLE OUTPUT ****
************************
Tool #1: use_aws
You have the following S3 buckets in the us-east-1 region:
```

35. `...`
36. To exit the prompt, enter
    `exit`.

Task complete: You successfully automated AWS resource management by creating and populating a DynamoDB table and adding general AWS capabilities to your agent.

Task 4: Orchestrate multi-agent AWS operations

In this task, you create the final AWS DevOps agent which combines the capabilities of all the agents you have created. You use an orchestrator agent to create an EC2 instance of flight data for customers to view.

Task 4.1: Create the orchestrator agent

Add the necessary code to the tool agents and orchestrator agent for them to integrate.

Agents as Tools is an architectural pattern in AI systems where specialized AI agents are wrapped as callable functions (tools) that can be used by other agents. This creates a hierarchical structure where:

- A primary *orchestrator agent* handles user interaction and determines which specialized agent to call
- Specialized *tool agents* perform domain-specific tasks when called by the orchestrator

Rather than a single agent trying to handle everything, tasks are delegated to the most appropriate specialized agent.

Note: The agents you created in *lab.ipynb*, coder and alarm_manager, are added as *.py* files in the */agents* directory.

To be able to use both tools and the agents you created as tools, add the `@tool` decorator to the four functions contained in your tool agents' Python files.

coder.py

```
@tool
def coder(query: str) -> str:
```

alarm_manager.py

```
@tool
def alarms(query: str) -> str:
```

aws_researcher.py

```
@tool
def aws_researcher(query: str) -> str:
```

aws_manager.py

```
@tool
```

37. `def aws_manager(query: str) -> str:`
38. In the Code Editor file explorer, choose agents/orchestrator.py.

Agents can use different models depending on what works best for their given task. Unlike the tool agents, the orchestrator agent uses *Anthropic Claude 3.5 Haiku*.

Add the following code to the top of the file to import the *tool agents*:

```
from coder import coder
from aws_researcher import aws_researcher
from alarm_manager import alarms
```

39. `from aws_manager import aws_manager`

To create a sufficiently prescriptive system prompt for the orchestrator agent, telling it what tool to use to perform each task its capable of, add the following code:

```
ORCHESTRATOR_SYSTEM_PROMPT = """
You are an assistant that routes queries to specialized agents:
- When asked to code or read a local file → Use the coder tool
- When asked to check on AWS account health → Use the alarms tool
- When asked to perform research on AWS → Use the aws_researcher
- When asked to perform an action in AWS -> Use aws_manager
- For simple questions not requiring specialized knowledge → Answer
directly
Always select the most appropriate tool based on the user's query.
Do not prompt the user again until a task is complete.
```

40. `"""`
41. To add the tools you imported to the agent, add the following code to the agent creation:
    ```
    tools=[coder, alarms, aws_researcher, aws_manager]
    ```

Task 4.2: Display the available flights on a site for customers

Use multiple tool agents of the orchestrator agent to create an EC2 hosted website with flight information from DynamoDB.

43. Command: To open the orchestrator agent prompt, run the following command:
    ```
    python -u agents/orchestrator.py
    ```

Note: If there was an issue with running the agent, check your code against the following completed code.

orchestrator.py full code

To have the orchestrator agent check on system status and then create a website showing AnyCompany flights status, enter the following prompt in the terminal, replacing SUBNET_ID with the value of PublicSubnet, and REGION with the value of PrimaryRegion, that is to the left of these instructions.

```
Check if an alarm indicates the client system is down. If it is, create a
single backup EC2 instance in REGION using the AnyCompanyFlightTracker
launch template in subnet SUBNET_ID
```

Consider: Notice the agents' ability to think deeply and retry requests to AWS as it makes mistakes, eventually arriving at the correct requests.

Expected output

```
************************
**** EXAMPLE OUTPUT ****
************************

I'll help you with that. I'll break this down into steps:

1. First, I'll check the alarms to see if the client system is down:
Tool #1: alarms
The alarm indicates that the client system is indeed down.

2. Now, I'll create a backup EC2 instance using the AWS Manager:
Tool #2: aws_manager
[09/10/25 18:12:03] INFO     Processing request of type ListToolsRequest
server.py:624
<thinking>To create an EC2 instance using a launch template, I will use
the `use_aws` tool with the `run_instances` operation for the EC2 service.
I will specify the launch template name, the subnet ID, and the region.
EC2 instances do not need a key pair as per the instructions, and I will
use the AMI ID from the launch template.</thinking>

...

<response>The EC2 instance has been successfully created with the instance
ID `i-025f5dd28a8988b34`. It is currently in the 'pending' state and
should be available shortly. The instance is associated with the launch
template `AnyCompanyFlightTracker` and is located in the subnet
`subnet-01d4cb50830eca35a` in the `us-east-1` region.</response>I've
completed both tasks:
1. Confirmed that the client system is down via the alarm
2. Created a backup EC2 instance using the specified launch template in
the requested subnet and region

Is there anything else you would like me to do?Response: I've completed
both tasks:
1. Confirmed that the client system is down via the alarm
2. Created a backup EC2 instance using the specified launch template in
the requested subnet and region
```

44. `Is there anything else you would like me to do?`

Note: If your agent's output does not imply successful site creation, continue prompting the agent to get the indented result.

If the agent did not tell you the public IP address

48. Wait up to five minutes for the instance status to become ready.
49. Copy the Public IPv4 address and paste it into a new browser tab. Press enter.

If the agents, did their job the flights data should be displayed on this internet page.

Task complete: You successfully orchestrated multi-agent AWS operations by creating a comprehensive DevOps agent that combines all specialized agents to deploy and manage AWS infrastructure through natural language requests.

Conclusion

You successfully did the following:

- Implemented intelligent file analysis and code generation capabilities using built-in SDK tools.
- Built specialized monitoring solutions that integrate with AWS CloudWatch for real-time system health assessment.
- Constructed research-enabled agents that access current AWS documentation through MCP servers.
- Developed agents capable of managing AWS services and automating resource deployment workflows.
- Designed coordinated systems that translated natural language requests into complex operational workflows.

End lab

Follow these steps to close the console and end your lab.

50. Return to the AWS Management Console.
51. At the upper-right corner of the page, choose AWSLabsUser, and then choose Sign out.
52. Choose End Lab and then confirm that you want to end your lab.

Additional Resources

- AWS Documentation MCP Server
- Strands Agents User Guide

For more information about AWS Training and Certification, see *https://aws.amazon.com/training/*.

Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.