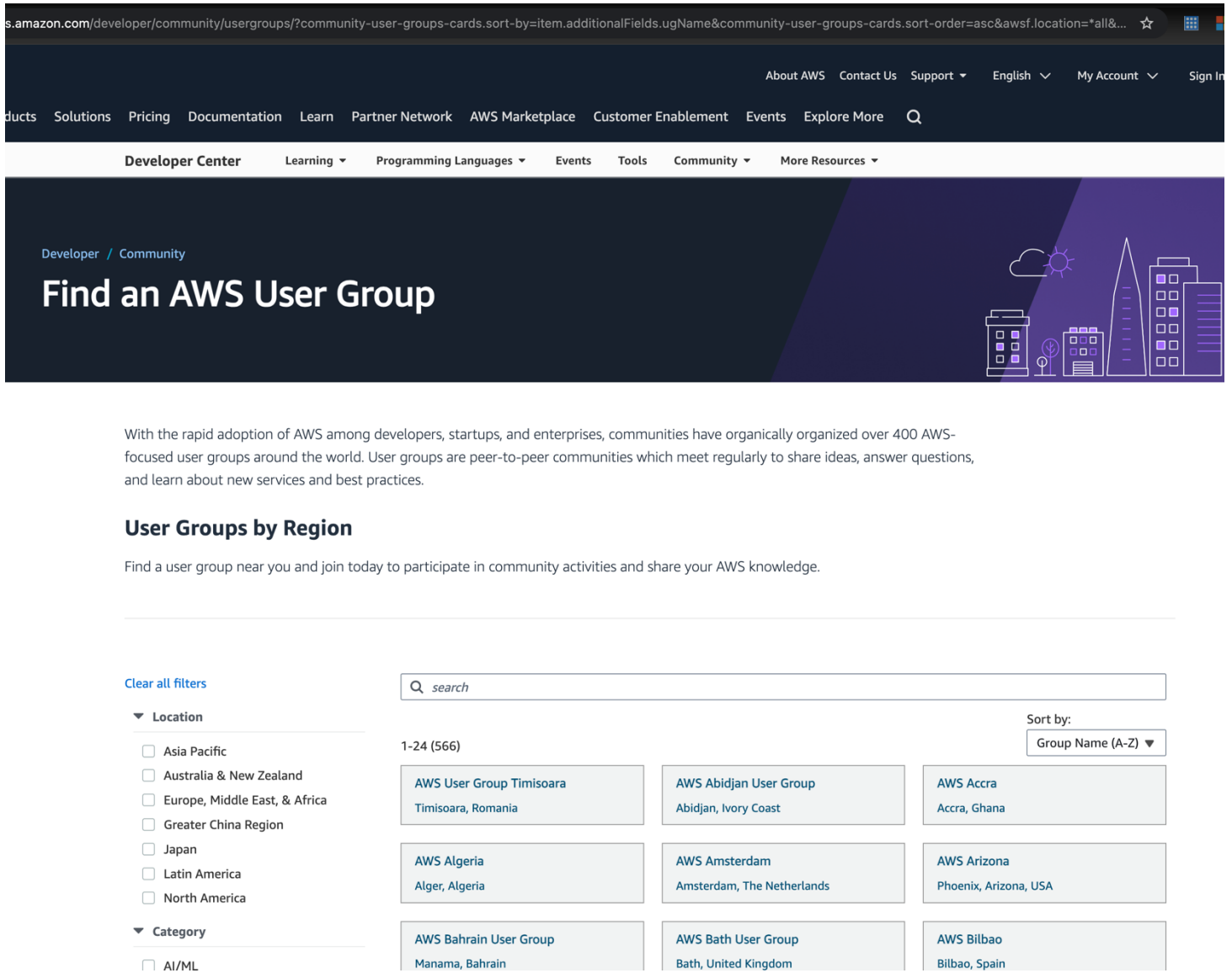
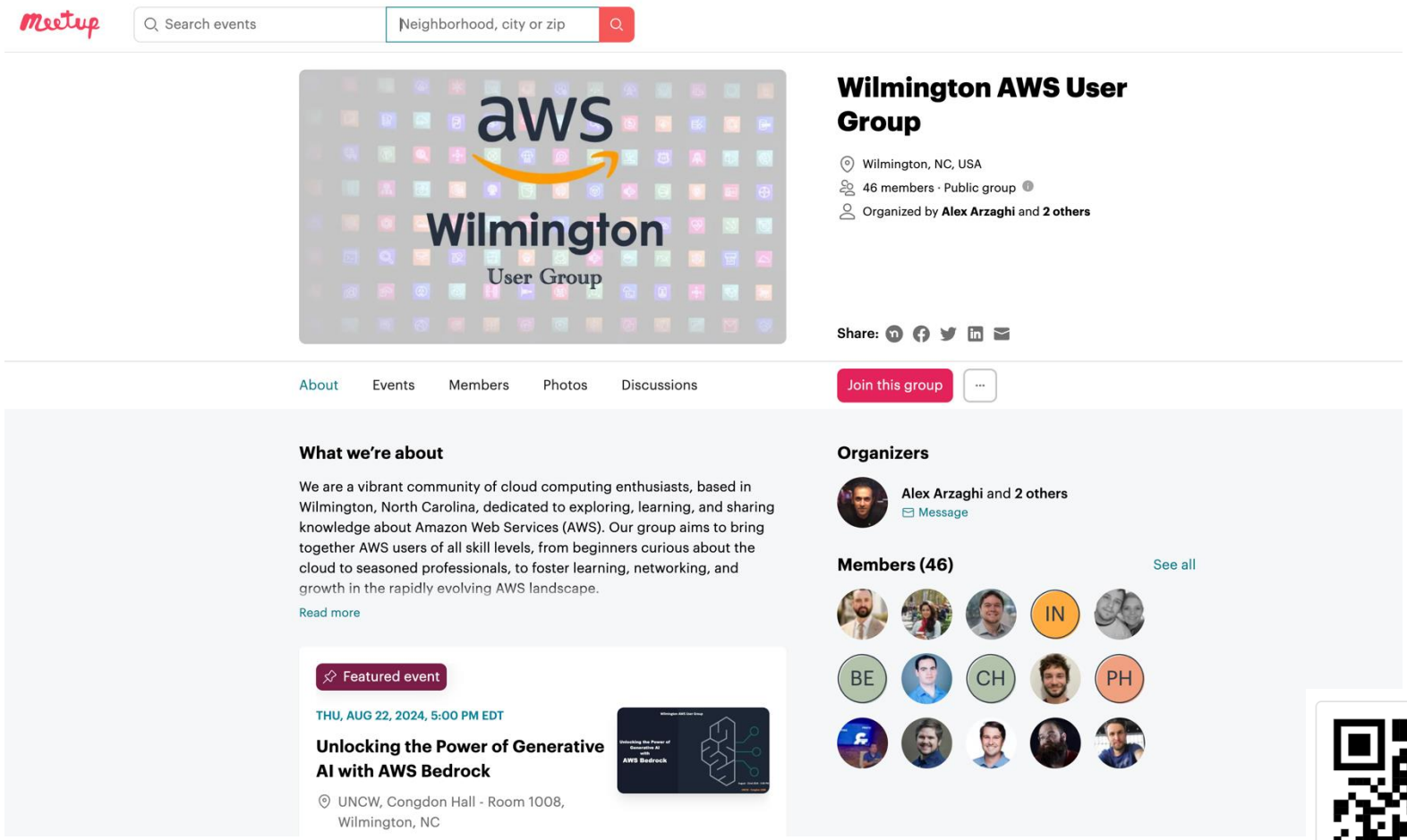
# Agenda

- AWS User Group Intro
- ACM at UNCW Intro
- Introduction to Amazon Bedrock
- Amazon Bedrock Deep Dive
- Showcase

# AWS User Groups

- Over 400 User Groups around the world
- Peer-to-peer communities
- Share ideas, answer questions, and learn




Search events
Neighborhood, city or zip
aws
Wilmington
User Group
Wilmington AWS User Group
Wilmington, NC, USA
46 members · Public group
Organized by Alex Arzaghi and 2 others
Share:
About
Events
Members
Photos
Discussions
Join this group
What we're about
We are a vibrant community of cloud computing enthusiasts, based in Wilmington, North Carolina, dedicated to exploring, learning, and sharing knowledge about Amazon Web Services (AWS). Our group aims to bring together AWS users of all skill levels, from beginners curious about the cloud to seasoned professionals, to foster learning, networking, and growth in the rapidly evolving AWS landscape.
Read more
Featured event
THU, AUG 22, 2024, 5:00 PM EDT
Unlocking the Power of Generative AI with AWS Bedrock
UNCW, Congdon Hall - Room 1008, Wilmington, NC
Organizers
Alex Arzaghi and 2 others
Message
Members (46)
See all
IN
BE
CH
PH


aws
User Group
Member

# Amazon Bedrock

# Agenda

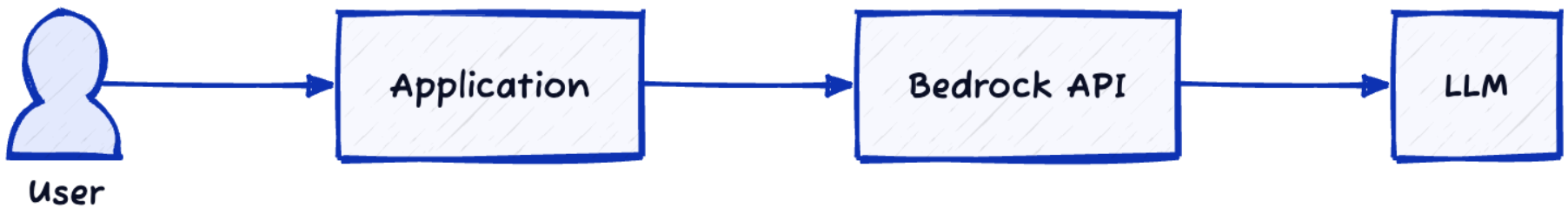
1. Introduction to Amazon Bedrock
2. Amazon Bedrock Deep Dive
    a. Model Invocations
    b. RAG + Knowledge bases
    c. Tools / Function Calling
    d. Agents
3. Showcase

# Introduction to Amazon Bedrock

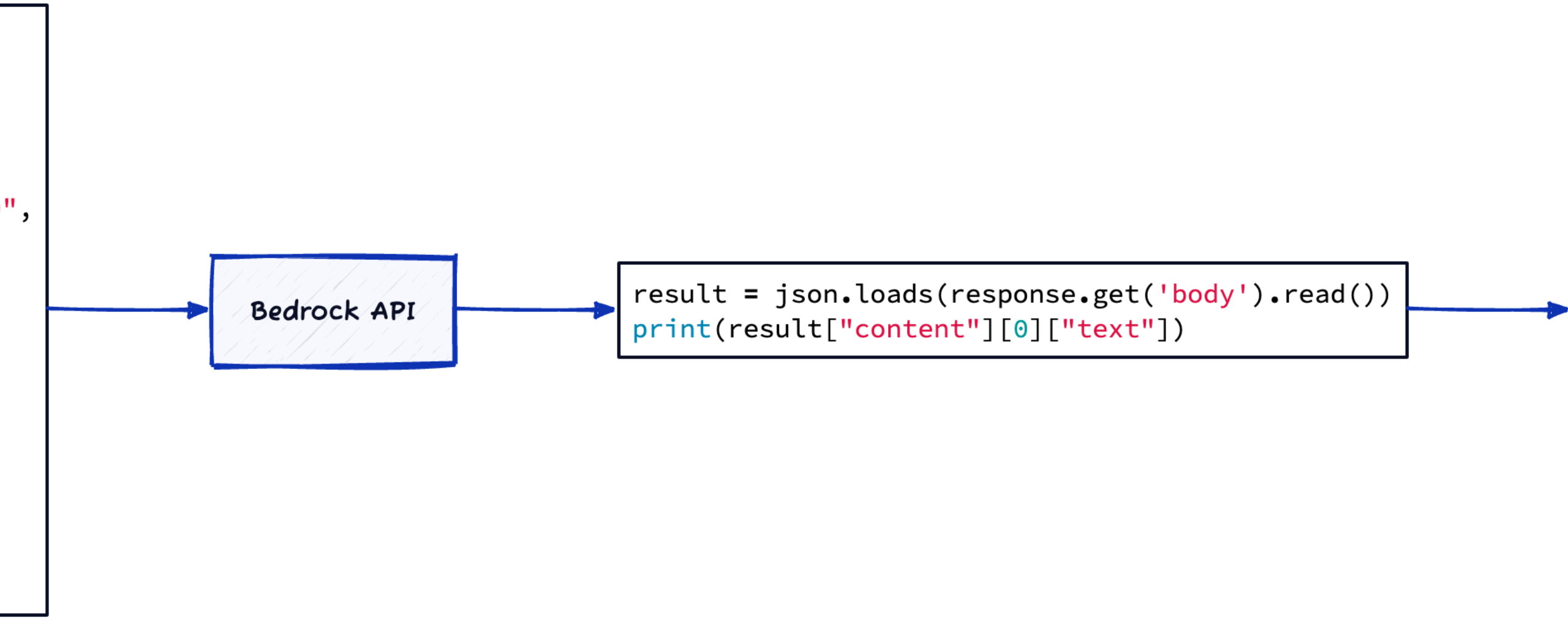
# What is Amazon Bedrock?

1. Amazon’s answer to LLMs
2. Foundational Models
3. Use YOUR private data
4. AWS’s security & compliance

# Amazon Bedrock Model Invocations


User
Application
Bedrock API
LLM

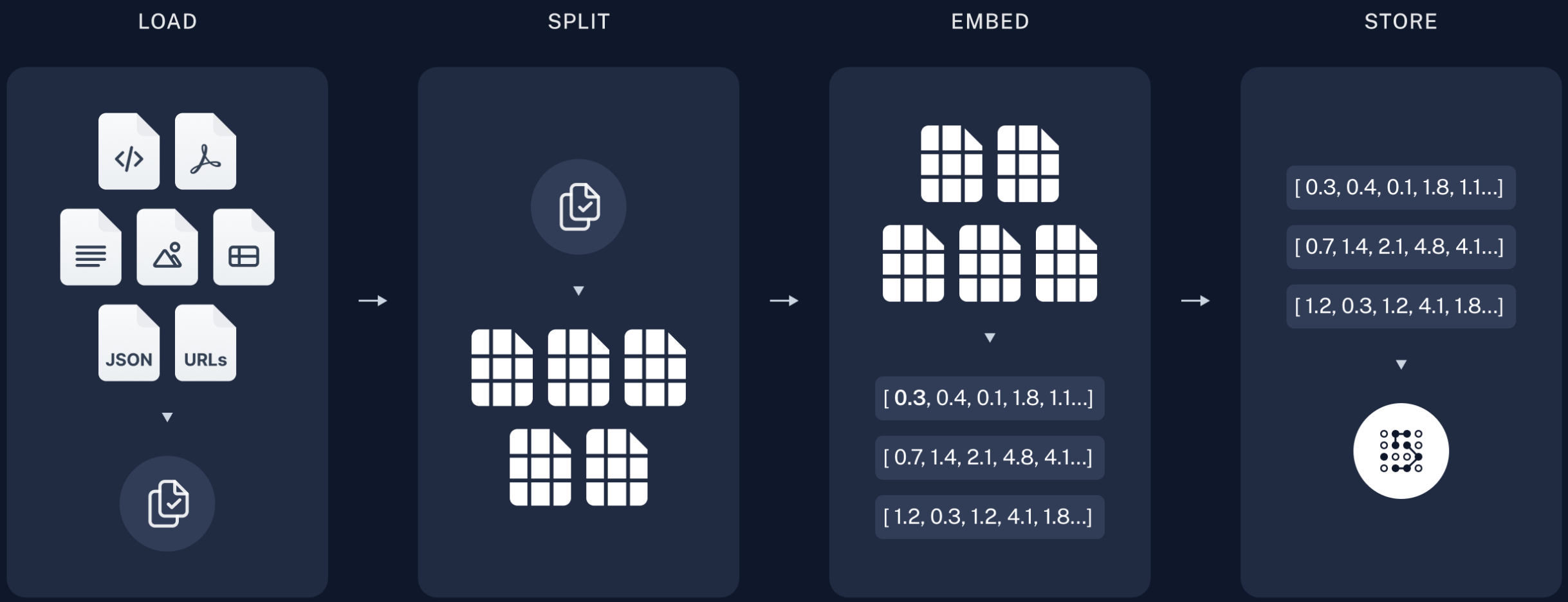
```python
import boto3
import json

client = boto3.client('bedrock-runtime')
response = client.invoke_model(
    modelId="anthropic.claude-3-haiku-20240307-v1:0",
    body=json.dumps({
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 100,
        "messages": [
            {
                "role": "user",
                "content": "Tell me a joke"
            }
        ]
    })
)
```

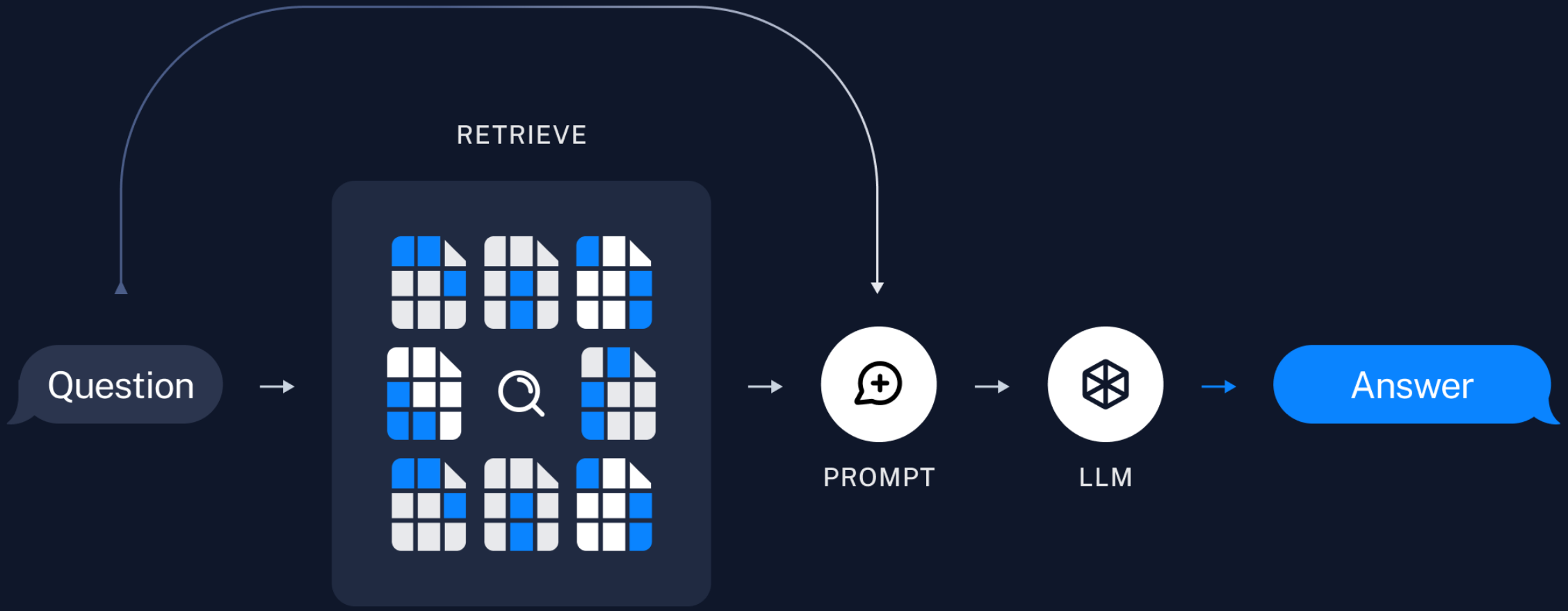

",
Bedrock API
result = json.loads(response.get('body').read())
print(result["content"][0]["text"])

```
onse.get('body').read())
0]["text"])
```

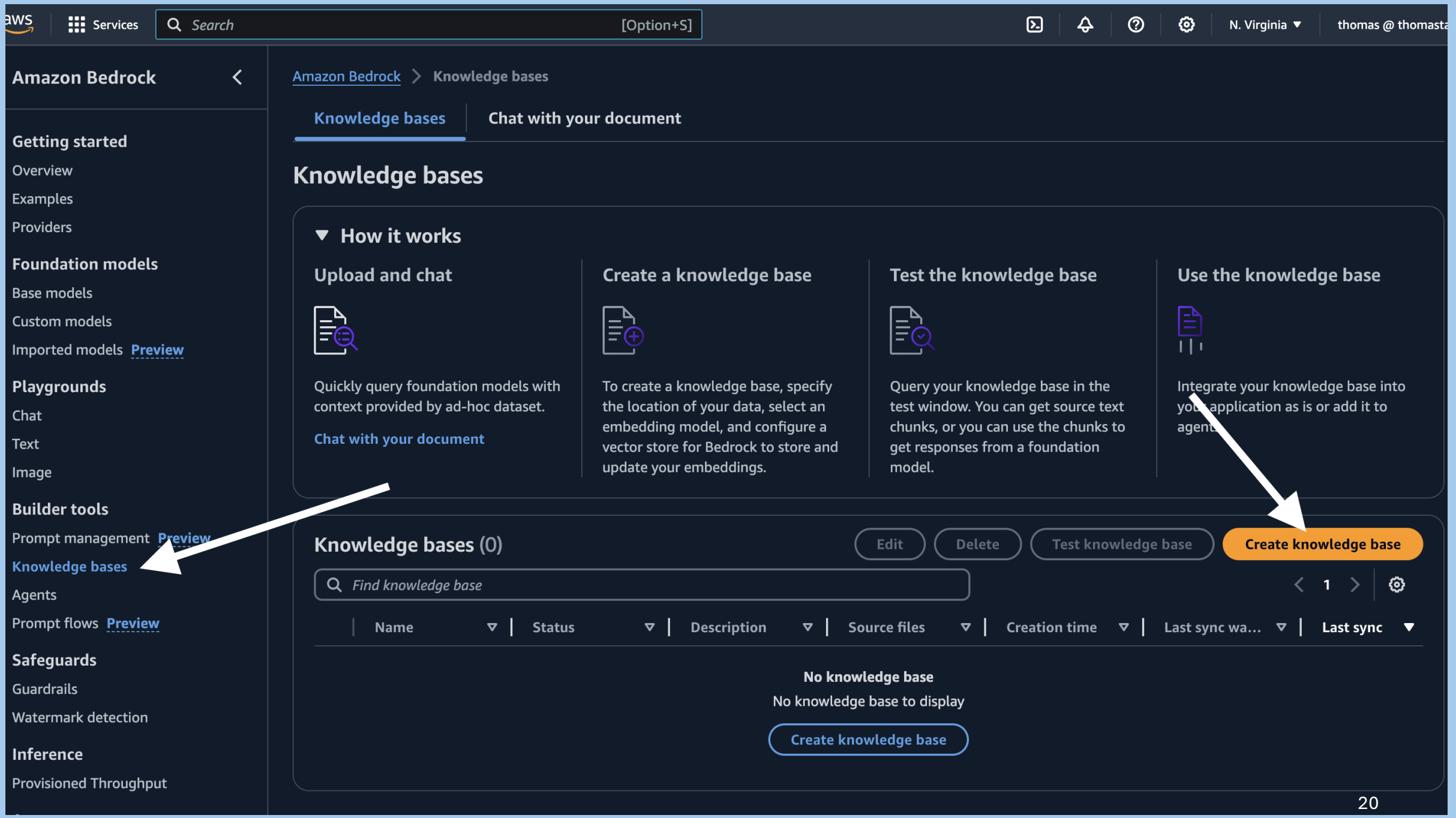
```
Here's a silly joke for you:

Why don't scientists trust atoms?
Because they make up everything!
```

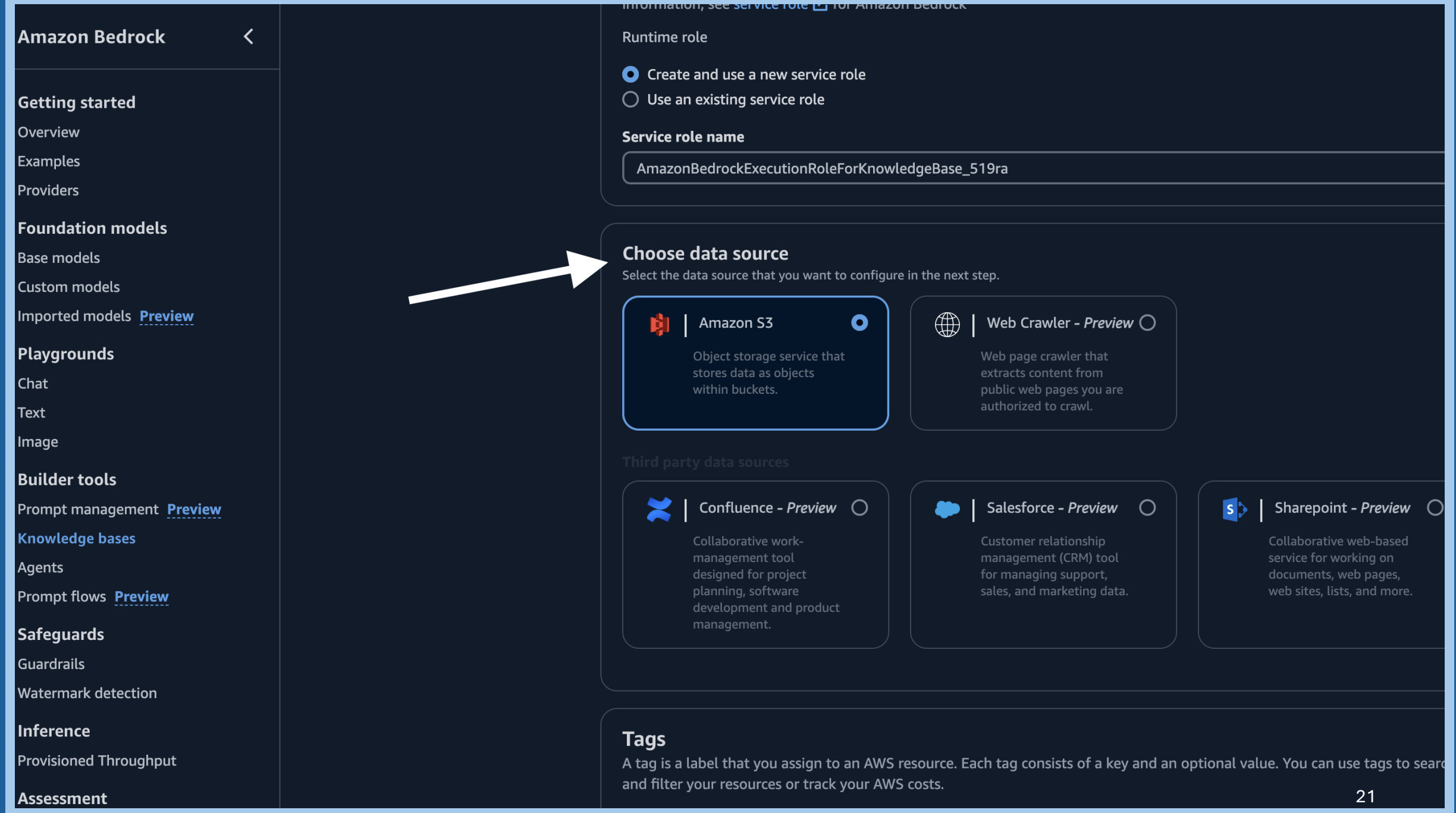
# Retrieval-Augmented Generation (RAG)

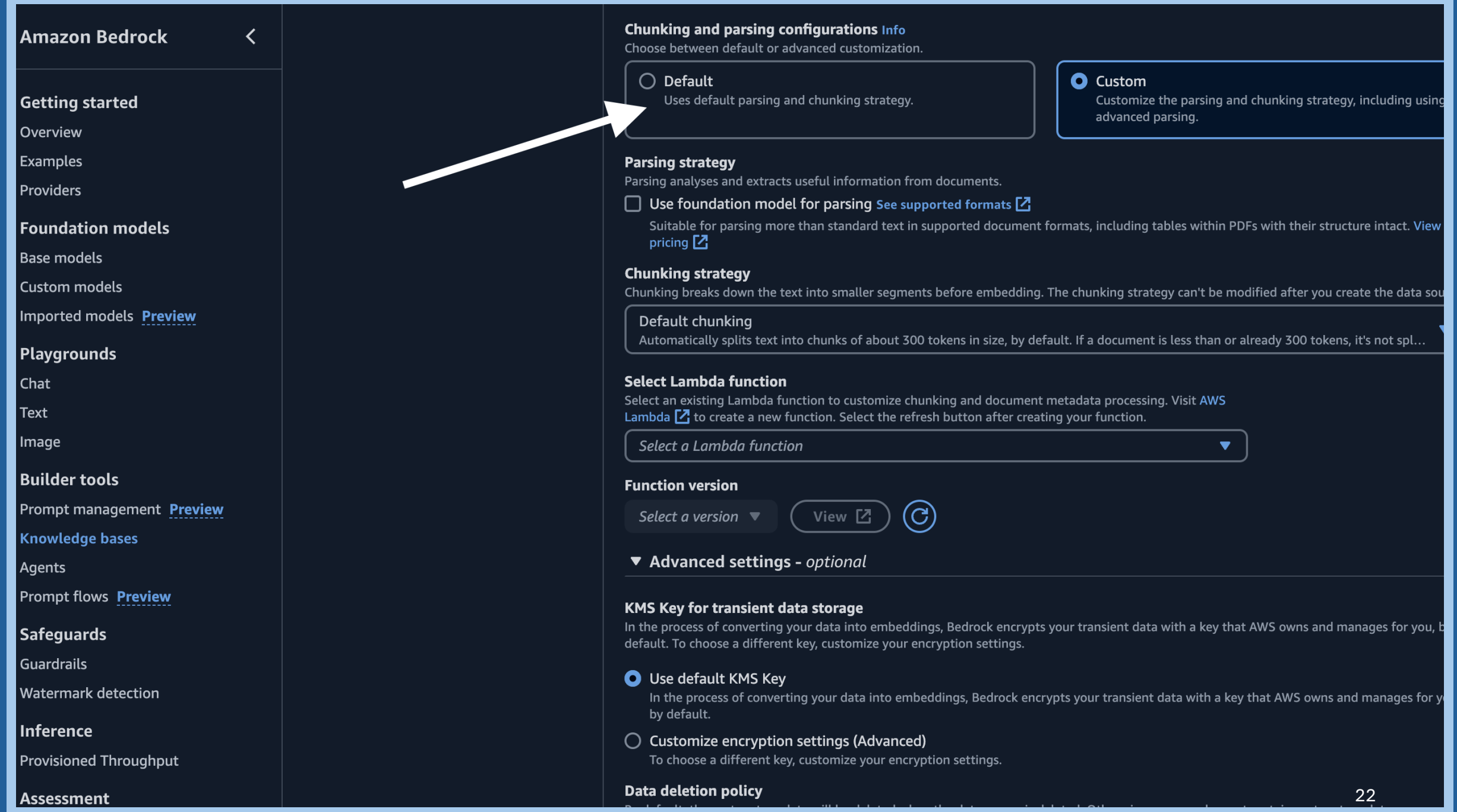

LOAD
SPLIT
EMBED
STORE
JSON
URLs
[ 0.3, 0.4, 0.1, 1.8, 1.1...]
[ 0.7, 1.4, 2.1, 4.8, 4.1...]
[ 1.2, 0.3, 1.2, 4.1, 1.8...]
[ 0.3, 0.4, 0.1, 1.8, 1.1...]
[ 0.7, 1.4, 2.1, 4.8, 4.1...]
[ 1.2, 0.3, 1.2, 4.1, 1.8...]

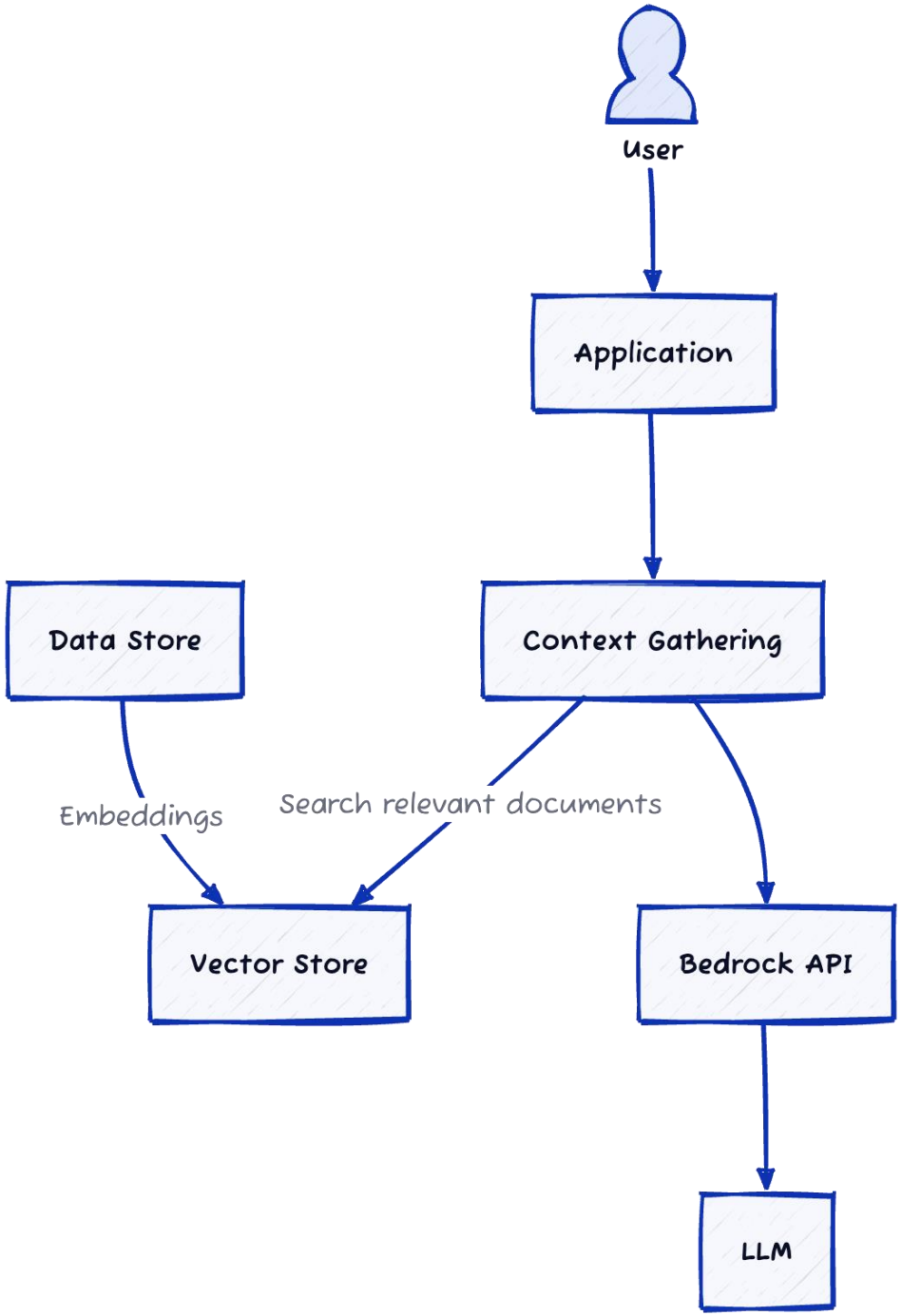

RETRIEVE
Question
PROMPT
LLM
Answer

# Amazon Bedrock: Knowledge Bases


aws
Services
Search
[Option+S]
N. Virginia
thomas @ thomasta
Amazon Bedrock
Getting started
Overview
Examples
Providers
Foundation models
Base models
Custom models
Imported models Preview
Playgrounds
Chat
Text
Image
Builder tools
Prompt management Preview
Knowledge bases
Agents
Prompt flows Preview
Safeguards
Guardrails
Watermark detection
Inference
Provisioned Throughput
Amazon Bedrock > Knowledge bases
Knowledge bases
Chat with your document
Knowledge bases
How it works
Upload and chat
Quickly query foundation models with context provided by ad-hoc dataset.
Chat with your document
Create a knowledge base
To create a knowledge base, specify the location of your data, select an embedding model, and configure a vector store for Bedrock to store and update your embeddings.
Test the knowledge base
Query your knowledge base in the test window. You can get source text chunks, or you can use the chunks to get responses from a foundation model.
Use the knowledge base
Integrate your knowledge base into your application as is or add it to agent
Knowledge bases (0)
Edit
Delete
Test knowledge base
Create knowledge base
Find knowledge base
1
Name
Status
Description
Source files
Creation time
Last sync wa...
Last sync
No knowledge base
No knowledge base to display
Create knowledge base


Amazon Bedrock
Getting started
Overview
Examples
Providers
Foundation models
Base models
Custom models
Imported models Preview
Playgrounds
Chat
Text
Image
Builder tools
Prompt management Preview
Knowledge bases
Agents
Prompt flows Preview
Safeguards
Guardrails
Watermark detection
Inference
Provisioned Throughput
Assessment
Runtime role
Create and use a new service role
Use an existing service role
Service role name
AmazonBedrockExecutionRoleForKnowledgeBase_519ra
Choose data source
Select the data source that you want to configure in the next step.
Amazon S3
Object storage service that stores data as objects within buckets.
Web Crawler - Preview
Web page crawler that extracts content from public web pages you are authorized to crawl.
Third party data sources
Confluence - Preview
Collaborative work-management tool designed for project planning, software development and product management.
Salesforce - Preview
Customer relationship management (CRM) tool for managing support, sales, and marketing data.
Sharepoint - Preview
Collaborative web-based service for working on documents, web pages, web sites, lists, and more.
Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to searc
and filter your resources or track your AWS costs.


Amazon Bedrock
Getting started
Overview
Examples
Providers
Foundation models
Base models
Custom models
Imported models Preview
Playgrounds
Chat
Text
Image
Builder tools
Prompt management Preview
Knowledge bases
Agents
Prompt flows Preview
Safeguards
Guardrails
Watermark detection
Inference
Provisioned Throughput
Assessment
Chunking and parsing configurations Info
Choose between default or advanced customization.
Default
Uses default parsing and chunking strategy.
Custom
Customize the parsing and chunking strategy, including using advanced parsing.
Parsing strategy
Parsing analyses and extracts useful information from documents.
Use foundation model for parsing See supported formats
Suitable for parsing more than standard text in supported document formats, including tables within PDFs with their structure intact. View pricing
Chunking strategy
Chunking breaks down the text into smaller segments before embedding. The chunking strategy can't be modified after you create the data sou
Default chunking
Automatically splits text into chunks of about 300 tokens in size, by default. If a document is less than or already 300 tokens, it's not spl...
Select Lambda function
Select an existing Lambda function to customize chunking and document metadata processing. Visit AWS Lambda to create a new function. Select the refresh button after creating your function.
Select a Lambda function
Function version
Select a version
View
Advanced settings - optional
KMS Key for transient data storage
In the process of converting your data into embeddings, Bedrock encrypts your transient data with a key that AWS owns and manages for you, b
default. To choose a different key, customize your encryption settings.
Use default KMS Key
In the process of converting your data into embeddings, Bedrock encrypts your transient data with a key that AWS owns and manages for y
by default.
Customize encryption settings (Advanced)
To choose a different key, customize your encryption settings.
Data deletion policy


User
Application
Data Store
Context Gathering
Embeddings
Search relevant documents
Vector Store
Bedrock API
LLM

# Claude Long Context Prompting

## Essential tips for long context prompts

- **Put longform data at the top**: Place your long documents and inputs (~20K+ tokens) near the top of your prompt, above your query, instructions, and examples. This can significantly improve Claude's performance across all models.

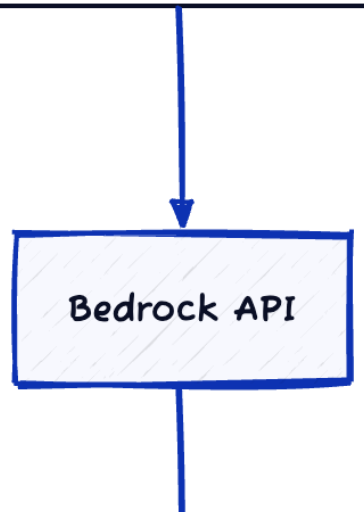
> Queries at the end can improve response quality by up to 30% in tests, especially with complex, multi-document inputs.

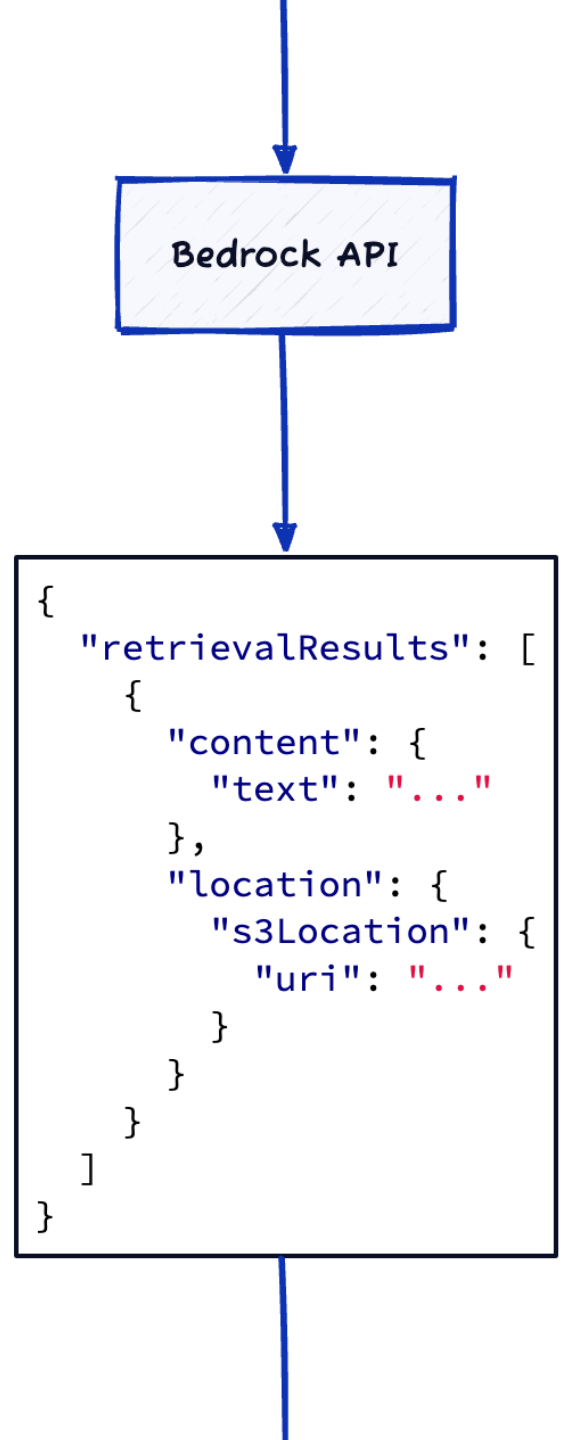
- **Structure document content and metadata with XML tags**: When using multiple documents, wrap each document in `<document>` tags with `<document_content>` and `<source>` (and other metadata) subtags for clarity.

▾ **Example multi-document structure**

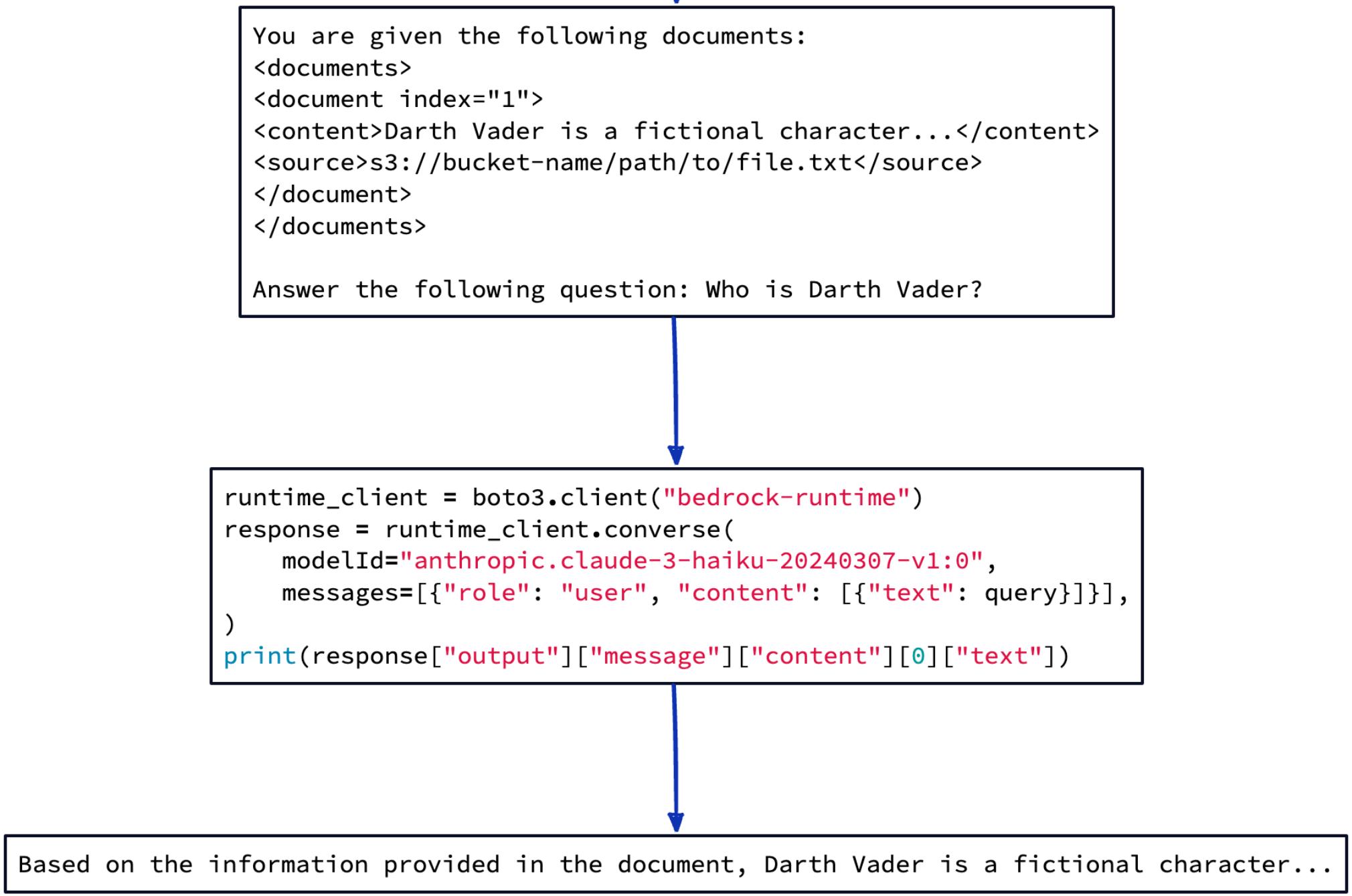
```
<documents>
  <document index="1">
    <source>annual_report_2023.pdf</source>
    <document_content>
      {{ANNUAL_REPORT}}
    </document_content>
  </document>
  <document index="2">
    <source>competitor_analysis_q2.xlsx</source>
    <document_content>
      {{COMPETITOR_ANALYSIS}}
    </document_content>
  </document>
</documents>

Analyze the annual report and competitor analysis. Identify strategic adva
```

```python
import boto3

user_query = "Who is Darth Vader?"

agent_runtime_client = boto3.client("bedrock-agent-runtime")
results = agent_runtime_client.retrieve(
    knowledgeBaseId="...",
    retrievalQuery={"text": user_query},
)
```



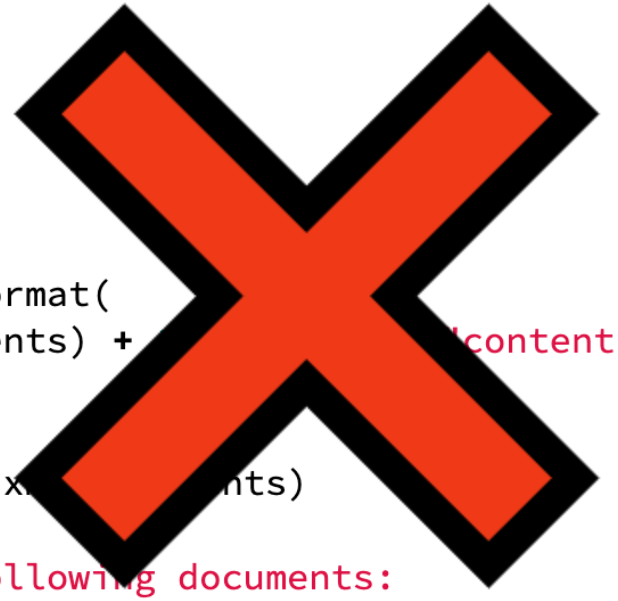

Bedrock API
{
"retrievalResults": [
{
"content": {
"text": "..."
},
"location": {
"s3Location": {
"uri": "..."
}
}
}
]
}

```
documents = [
    {
        "content": c["content"]["text"],
        "source": c["location"]["s3Location"]["uri"],
    }
    for c in results["retrievalResults"]
]

xml_document_template = """<document index="{index}">
<content>{content}</content>
<source>{source}</source>
</document>"""

xml_documents: list[str] = []
for d in documents:
    xml_documents.append(
        xml_document_template.format(
            index=len(xml_documents) + 1, content=d["content"], source=d["source"]
        )
    )
xml_document_string = "\n".join(xml_documents)

query = f"""You are given the following documents:
<documents>
{xml_document_string}
</documents>

Answer the following question: {user_query}
"""
```

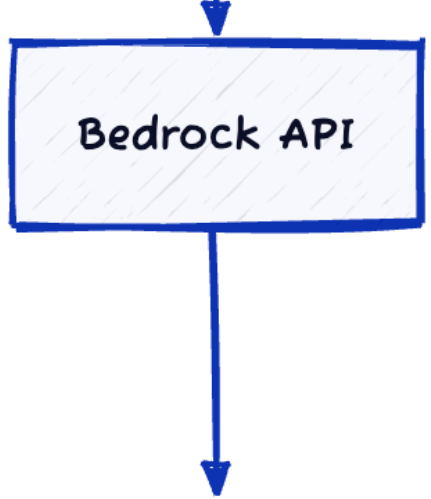

You are given the following documents:
<documents>
<document index="1">
<content>Darth Vader is a fictional character...</content>
<source>s3://bucket-name/path/to/file.txt</source>
</document>
</documents>
Answer the following question: Who is Darth Vader?
runtime_client = boto3.client("bedrock-runtime")
response = runtime_client.converse(
modelId="anthropic.claude-3-haiku-20240307-v1:0",
messages=[{"role": "user", "content": [{"text": query}]}],
)
print(response["output"]["message"]["content"][0]["text"])
Based on the information provided in the document, Darth Vader is a fictional character...

```
documents = [
    {
        "content": c["content"]["text"],
        "source": c["location"]["s3Location"]["uri"],
    }
    for c in results["retrievalResults"]
]

xml_document_template = """<document index="{index}">
<content>{content}</content>
<source>{source}</source>
</document>"""

xml_documents: list[str] = []
for d in documents:
    xml_documents.append(
        xml_document_template.format(
            index=len(xml_documents) + 1, content=d["content"], source=d["source"]
        )
    )
xml_document_string = "\n".join(xml_documents)

query = f"""You are given the following documents:
<documents>
{xml_document_string}
</documents>

Answer the following question: {user_query}
"""
```

```
documents = [
    {
        "content": c["content"]["text"],
        "source": c["location"]["s3Location"]["uri"],
    }
    for c in results["retrievalResults"]
]

xml_document_template = """<document index="{index}">
<content>{content}</content>
<source>{source}</source>
</document>"""

xml_documents: list[str] = []
for d in documents:
    xml_documents.append(
        xml_document_template.format(
            index=len(xml_documents) +                    "content"], source=d["source"]
        )
    )
xml_document_string = "\n".join(x                    nts)

query = f"""You are given the following documents:
<documents>
{xml_document_string}
</documents>

Answer the following question: {user_query}
"""
```



```python
import boto3

user_query = "Who is Darth Vader?"

agent_runtime_client = boto3.client("bedrock-agent-runtime")
response = agent_runtime_client.retrieve_and_generate(
    input={"text": user_query},
    retrieveAndGenerateConfiguration={
        "knowledgeBaseConfiguration": {
            "knowledgeBaseId": "...",
            "modelArn": ".../anthropic.claude-3-haiku-20240307-v1:0",
        },
        "type": "KNOWLEDGE_BASE",
        # Decomposition also exists here
    },
)
```




```
{
  "citations": [
    {
      "generatedResponsePart": {
        "textResponsePart": {
          "text": "..."
        }
      },
      "retrievedReferences": []
    }
  ],
  "output": {
    "text": "Based on the information provided..."
  }
}
```

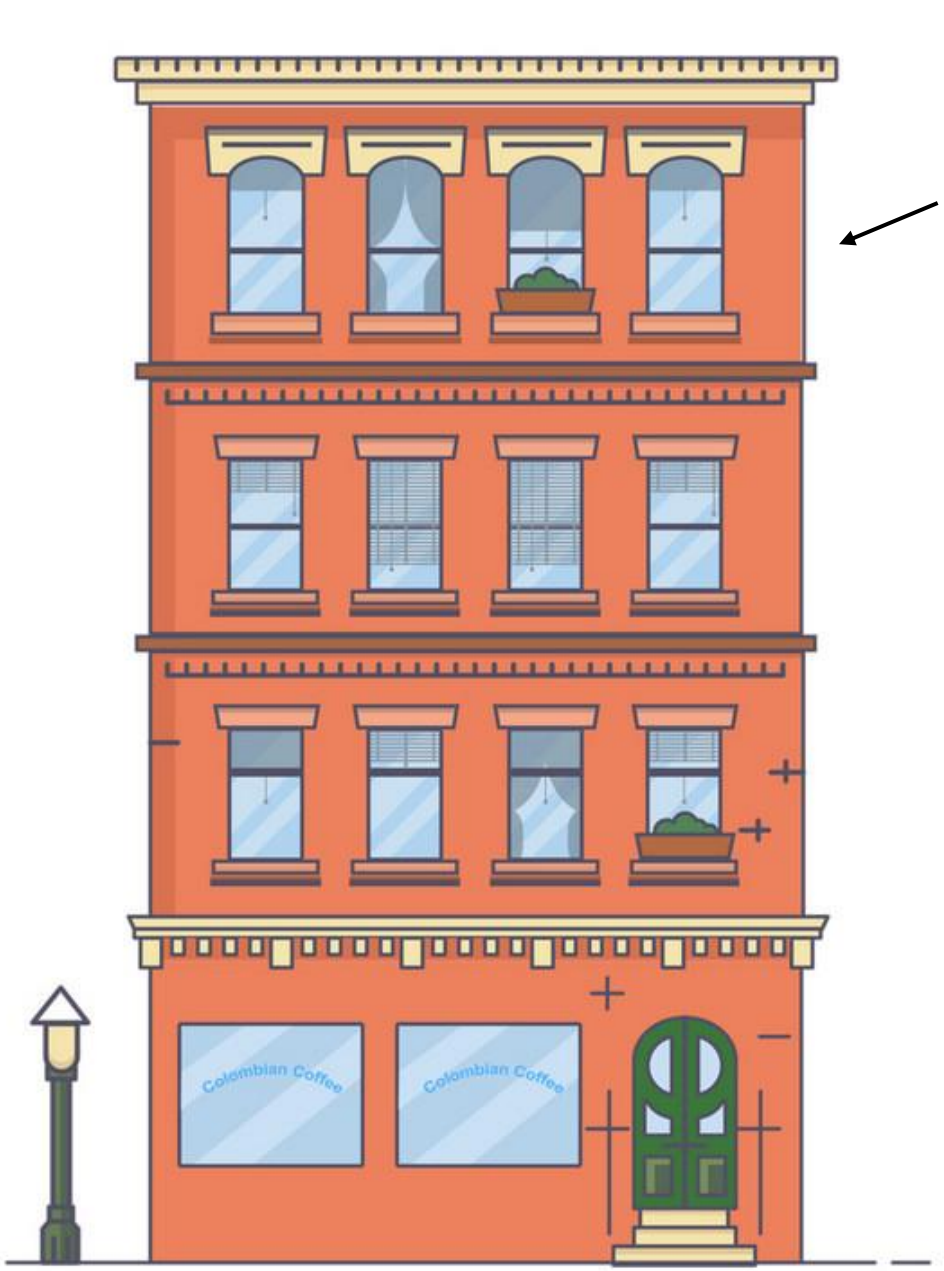
# RAG

1. Vector stores
2. Leverage them with LLMs
3. Bedrock makes it easy…

# Amazon Bedrock:
# Tools / Function Calling


Colombian Coffee
Colombian Coffee


Colombian Coffee
Colombian Coffee


Colombian Coffee
Colombian Coffee

Help me make spaghetti!
Tools: pot, spoon, stove

Put water in pot

I put water in the pot!

Put pot on stove and turn on the stove

Done!

Put the noodles in!

# Did the work, no planning

**Backend**

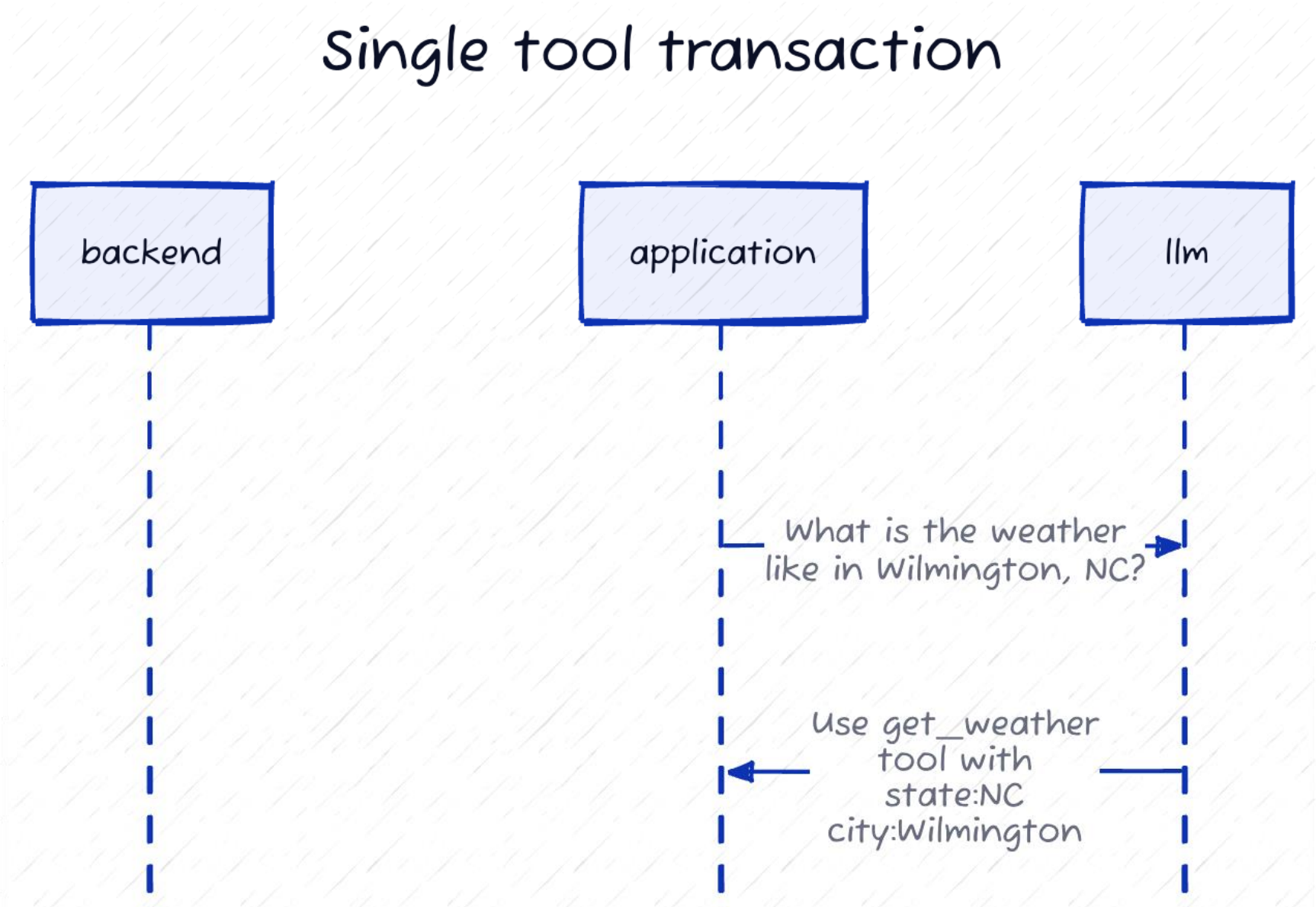
**Application**

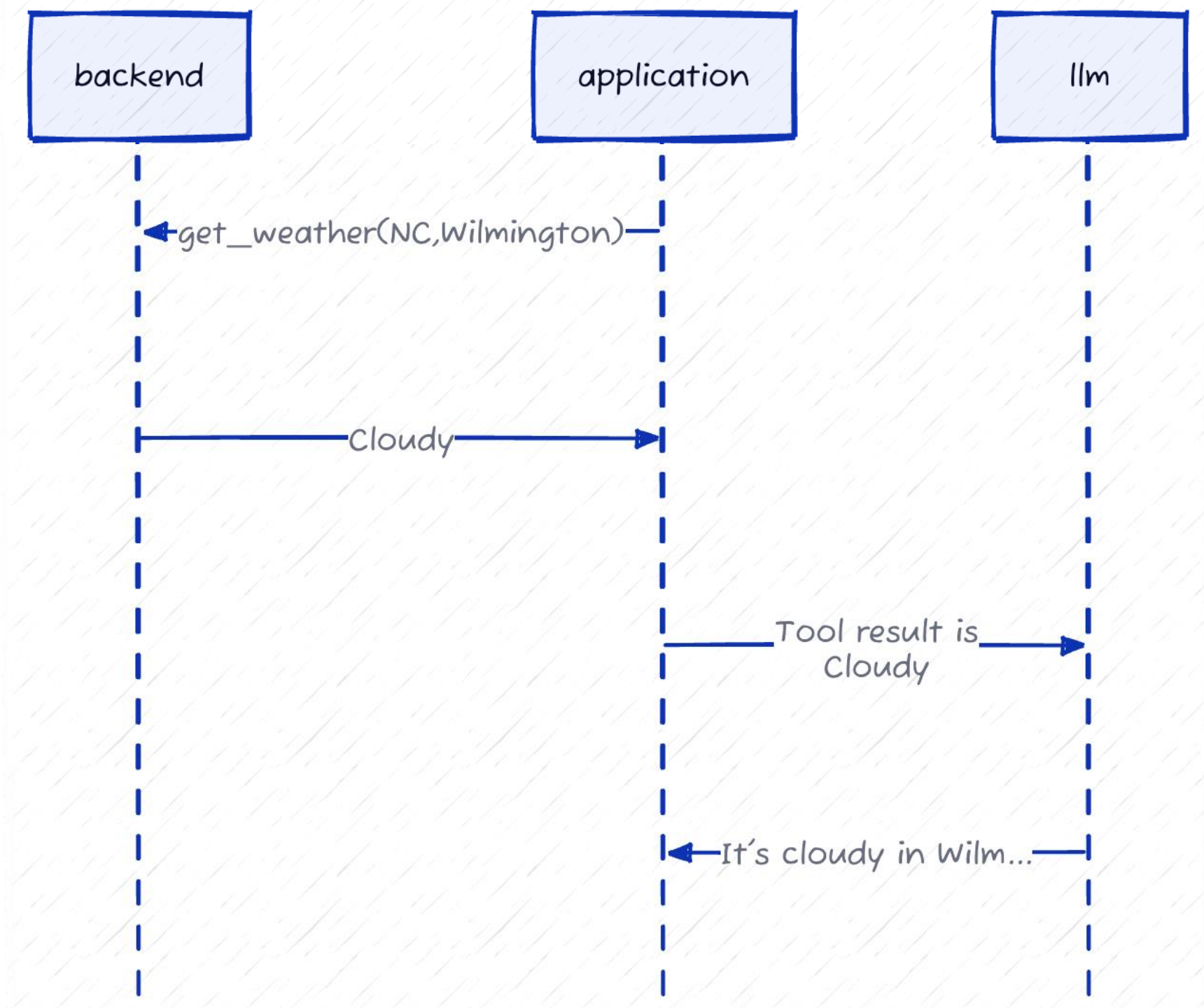
**LLM**

```
tools:
- get_weather
```

# Single tool transaction

```
tools:
- get_weather
```

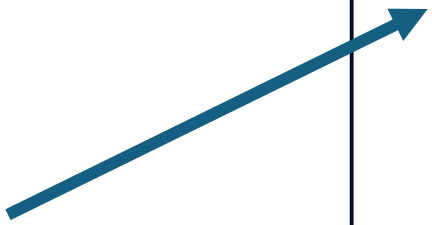



backend
application
llm
get_weather(NC,Wilmington)
Cloudy
Tool result is Cloudy
It's cloudy in Wilm...

1. **Get weather by latitude and longitude**
   **Get latitude and longitude of city / state**

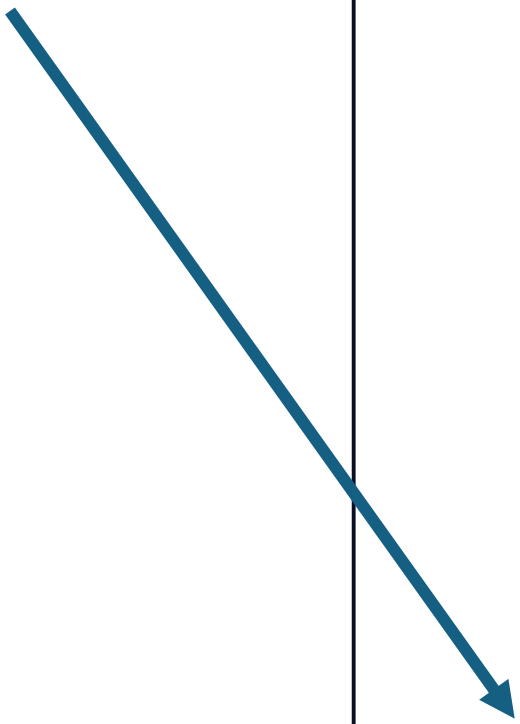
```
{
  "toolSpec": {
    "name": "get_weather",
    "description": "Get the current weather for a given latitude and longitude",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "latitude": {
            "type": "number",
            "description": "The latitude of the location"
          },
          "longitude": {
            "type": "number",
            "description": "The longitude of the location"
          },
          "unit": {
            "type": "string",
            "enum": [
              "celsius",
              "fahrenheit"
            ],
            "description": "The unit of temperature, either 'celsius' or 'fahrenheit'",
            "default": "fahrenheit"
          }
        },
        "required": [
          "latitude",
          "longitude"
        ]
      }
    }
  }
```

```
{
  "toolSpec": {
    "name": "get_weather",
    "description": "Get the current weather for a given latitude and longitude",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "latitude": {
            "type": "number",
            "description": "The latitude of the location"
          },
          "longitude": {
            "type": "number",
            "description": "The longitude of the location"
          },
          "unit": {
            "type": "string",
            "enum": [
              "celsius",
              "fahrenheit"
            ],
            "description": "The unit of temperature, either 'celsius' or 'fahrenheit'",
            "default": "fahrenheit"
          }
        },
        "required": [
          "latitude",
          "longitude"
        ]
      }
    }
  }
```



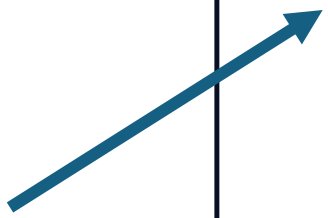
```
{
  "toolSpec": {
    "name": "get_weather",
    "description": "Get the current weather for a given latitude and longitude",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "latitude": {
            "type": "number",
            "description": "The latitude of the location"
          },
          "longitude": {
            "type": "number",
            "description": "The longitude of the location"
          },
          "unit": {
            "type": "string",
            "enum": [
              "celsius",
              "fahrenheit"
            ],
            "description": "The unit of temperature, either 'celsius' or 'fahrenheit'",
            "default": "fahrenheit"
          }
        },
        "required": [
          "latitude",
          "longitude"
        ]
      }
    }
  }
```

```
{
  "toolSpec": {
    "name": "get_weather",
    "description": "Get the current weather for a given latitude and longitude",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "latitude": {
            "type": "number",
            "description": "The latitude of the location"
          },
          "longitude": {
            "type": "number",
            "description": "The longitude of the location"
          },
          "unit": {
            "type": "string",
            "enum": [
              "celsius",
              "fahrenheit"
            ],
            "description": "The unit of temperature, either 'celsius' or 'fahrenheit'",
            "default": "fahrenheit"
          }
        },
        "required": [
          "latitude",
          "longitude"
        ]
      }
    }
  }
```



```
{
  "toolSpec": {
    "name": "get_weather",
    "description": "Get the current weather for a given latitude and longitude",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "latitude": {
            "type": "number",
            "description": "The latitude of the location"
          },
          "longitude": {
            "type": "number",
            "description": "The longitude of the location"
          },
          "unit": {
            "type": "string",
            "enum": [
              "celsius",
              "fahrenheit"
            ],
            "description": "The unit of temperature, either 'celsius' or 'fahrenheit'",
            "default": "fahrenheit"
          }
        },
        "required": [
          "latitude",
          "longitude"
        ]
      }
    }
  }
```

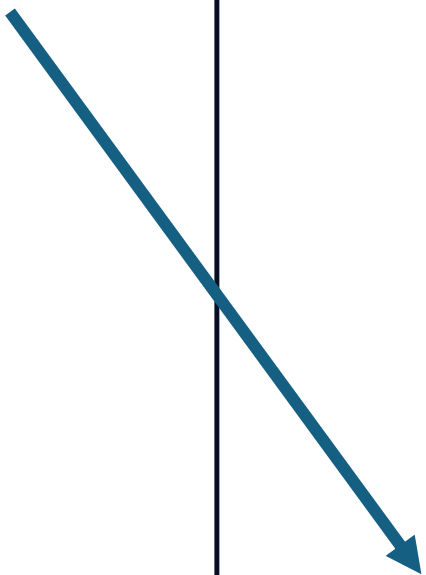
```
{
  "toolSpec": {
    "name": "get_latitude_and_longitude",
    "description": "Get the latitude and longitude for a given city and state",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "state": {
            "type": "string",
            "description": "The state"
          },
          "city": {
            "type": "string",
            "description": "The city"
          }
        },
        "required": [
          "state",
          "city"
        ]
      }
    }
  }
}
```



```
{
  "toolSpec": {
    "name": "get_latitude_and_longitude",
    "description": "Get the latitude and longitude for a given city and state",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "state": {
            "type": "string",
            "description": "The state"
          },
          "city": {
            "type": "string",
            "description": "The city"
          }
        },
        "required": [
          "state",
          "city"
        ]
      }
    }
  }
}
```

```
{
  "toolSpec": {
    "name": "get_latitude_and_longitude",
    "description": "Get the latitude and longitude for a given city and state",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "state": {
            "type": "string",
            "description": "The state"
          },
          "city": {
            "type": "string",
            "description": "The city"
          }
        },
        "required": [
          "state",
          "city"
        ]
      }
    }
  }
}
```

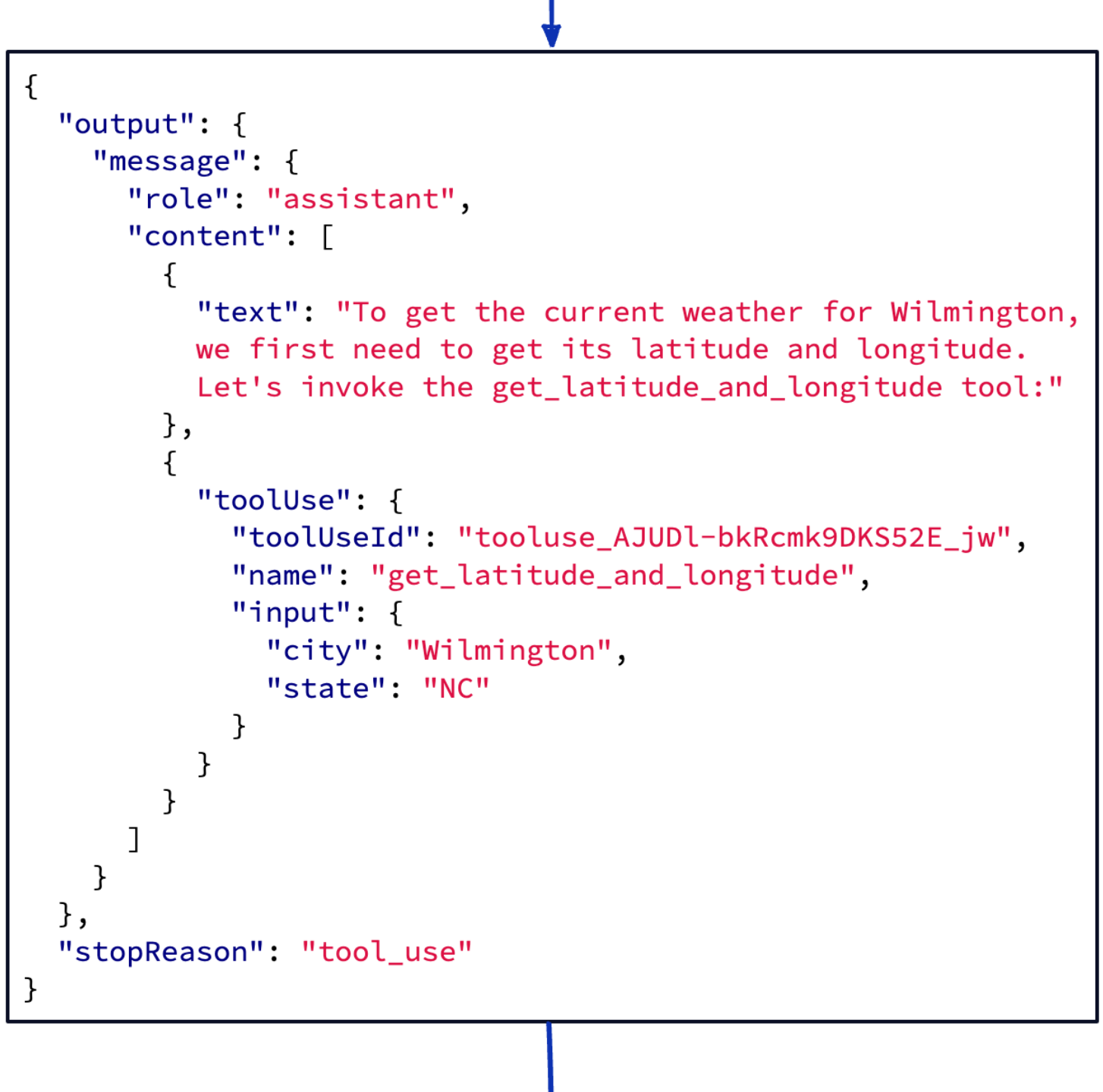
```
{
  "toolSpec": {
    "name": "get_latitude_and_longitude",
    "description": "Get the latitude and longitude for a given city and state",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "state": {
            "type": "string",
            "description": "The state"
          },
          "city": {
            "type": "string",
            "description": "The city"
          }
        },
        "required": [
          "state",
          "city"
        ]
      }
    }
  }
}
```

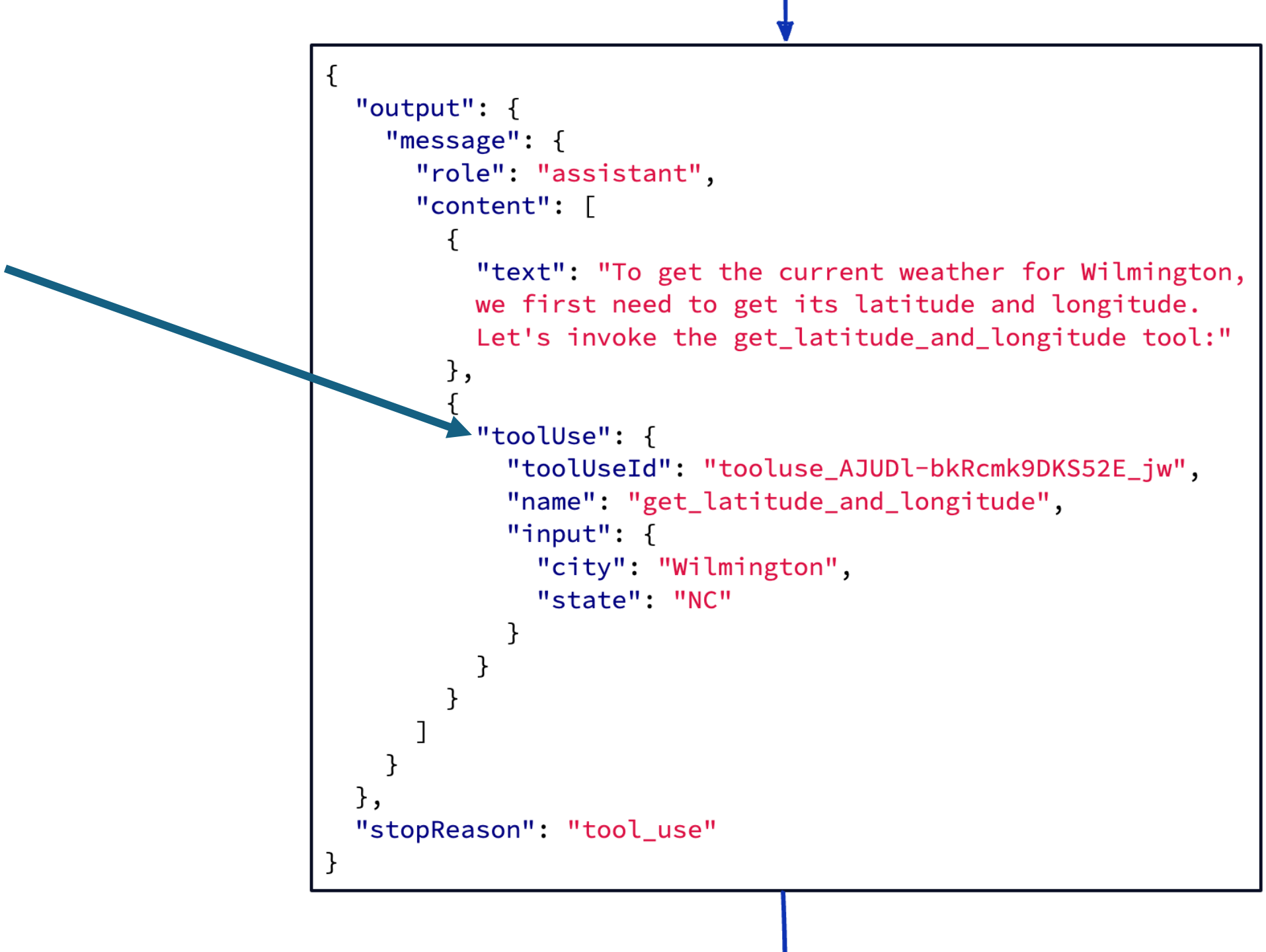


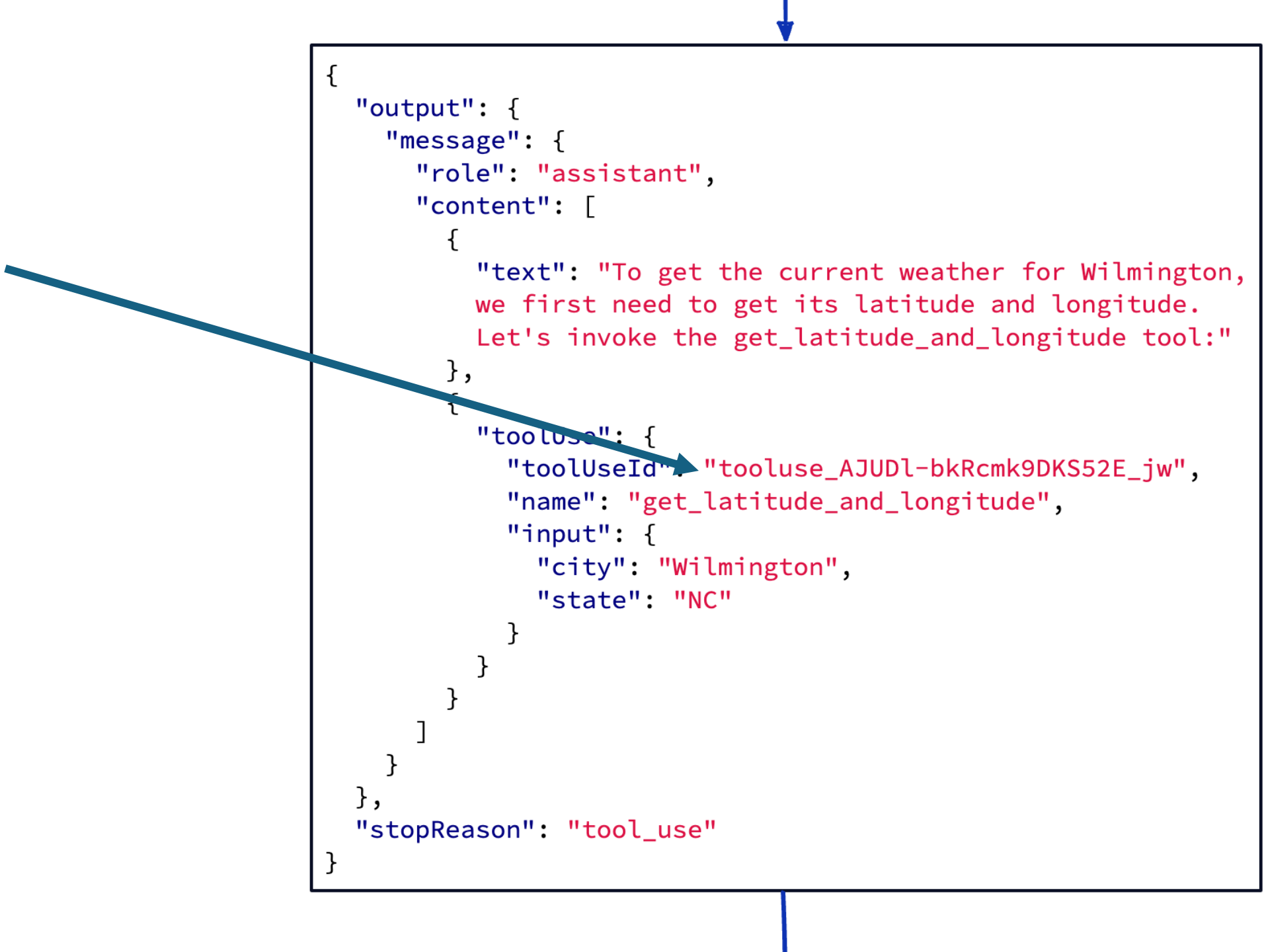
```
{
  "toolSpec": {
    "name": "get_latitude_and_longitude",
    "description": "Get the latitude and longitude for a given city and state",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "state": {
            "type": "string",
            "description": "The state"
          },
          "city": {
            "type": "string",
            "description": "The city"
          }
        },
        "required": [
          "state",
          "city"
        ]
      }
    }
  }
}
```

```
response = bedrock.converse(
    modelId="anthropic.claude-3-sonnet-20240229-v1:0",
    messages=[
        {
            "role": "user",
            "content": [{"text": "What is the weather right now in Wilmington?"}],
        }
    ],
    toolConfig={"tools": tools},  # The two tools are here
)
```

```
{
  "output": {
    "message": {
      "role": "assistant",
```


{
"output": {
"message": {
"role": "assistant",
"content": [
{
"text": "To get the current weather for Wilmington,
we first need to get its latitude and longitude.
Let's invoke the get_latitude_and_longitude tool:"
},
{
"toolUse": {
"toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
"name": "get_latitude_and_longitude",
"input": {
"city": "Wilmington",
"state": "NC"
}
}
}
]
}
},
"stopReason": "tool_use"
}


{
"output": {
"message": {
"role": "assistant",
"content": [
{
"text": "To get the current weather for Wilmington,
we first need to get its latitude and longitude.
Let's invoke the get_latitude_and_longitude tool:"
},
{
"toolUse": {
"toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
"name": "get_latitude_and_longitude",
"input": {
"city": "Wilmington",
"state": "NC"
}
}
}
]
}
},
"stopReason": "tool_use"
}

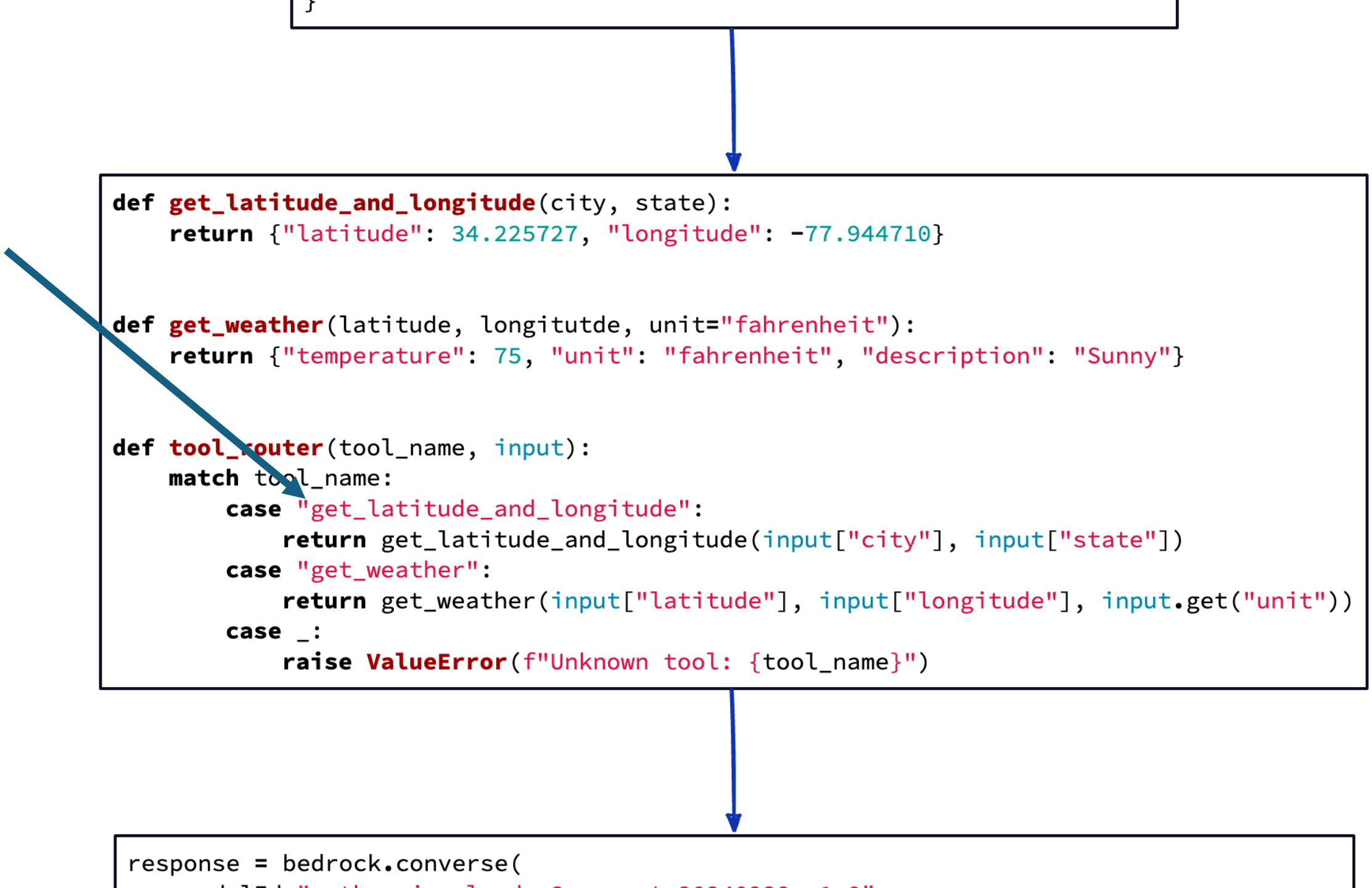

{
"output": {
"message": {
"role": "assistant",
"content": [
{
"text": "To get the current weather for Wilmington,
we first need to get its latitude and longitude.
Let's invoke the get_latitude_and_longitude tool:"
},
{
"toolUse": {
"toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
"name": "get_latitude_and_longitude",
"input": {
"city": "Wilmington",
"state": "NC"
}
}
}
]
}
},
"stopReason": "tool_use"
}

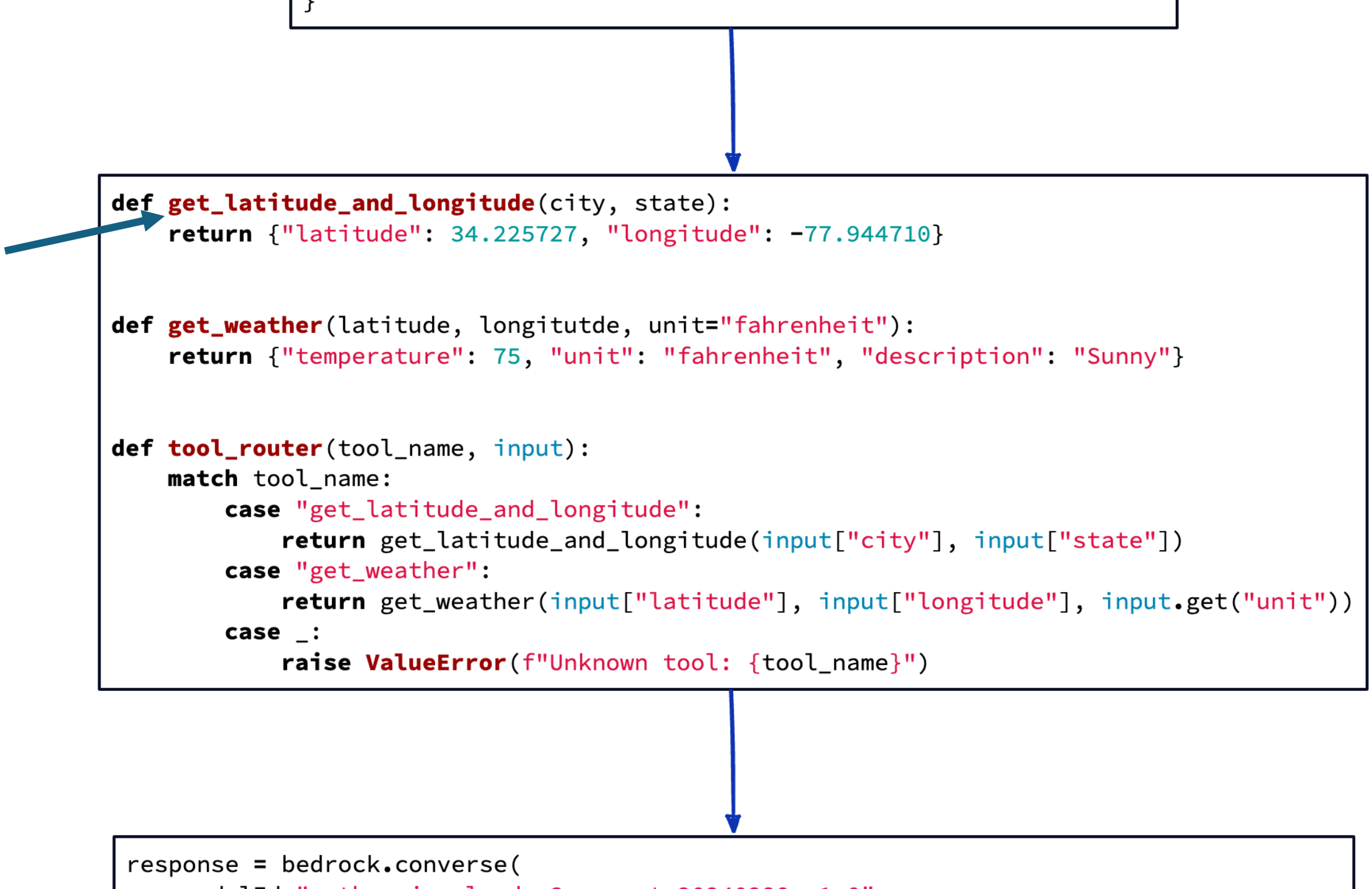

{
"output": {
"message": {
"role": "assistant",
"content": [
{
"text": "To get the current weather for Wilmington,
we first need to get its latitude and longitude.
Let's invoke the get_latitude_and_longitude tool:"
},
{
"toolUse": {
"toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
"name": "get_latitude_and_longitude",
"input": {
"city": "Wilmington",
"state": "NC"
}
}
}
]
}
},
"stopReason": "tool_use"
}

```
}
```

```python
def get_latitude_and_longitude(city, state):
    return {"latitude": 34.225727, "longitude": -77.944710}


def get_weather(latitude, longitutde, unit="fahrenheit"):
    return {"temperature": 75, "unit": "fahrenheit", "description": "Sunny"}


def tool_router(tool_name, input):
    match tool_name:
        case "get_latitude_and_longitude":
            return get_latitude_and_longitude(input["city"], input["state"])
        case "get_weather":
            return get_weather(input["latitude"], input["longitude"], input.get("unit"))
        case _:
            raise ValueError(f"Unknown tool: {tool_name}")
```

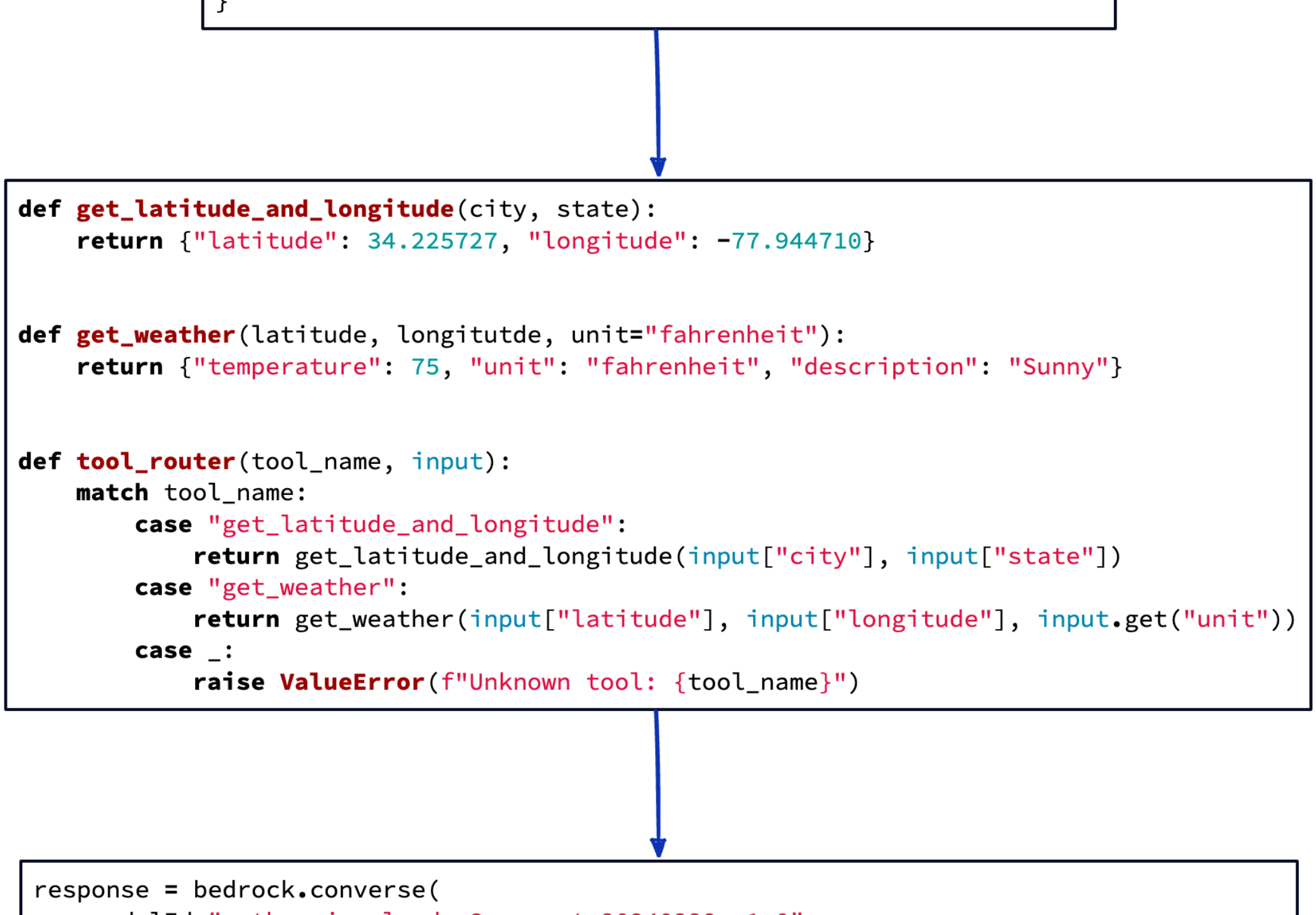
```python
response = bedrock.converse(
```


}
def get_latitude_and_longitude(city, state):
return {"latitude": 34.225727, "longitude": -77.944710}
def get_weather(latitude, longitutde, unit="fahrenheit"):
return {"temperature": 75, "unit": "fahrenheit", "description": "Sunny"}
def tool_router(tool_name, input):
match tool_name:
case "get_latitude_and_longitude":
return get_latitude_and_longitude(input["city"], input["state"])
case "get_weather":
return get_weather(input["latitude"], input["longitude"], input.get("unit"))
case _:
raise ValueError(f"Unknown tool: {tool_name}")
response = bedrock.converse(

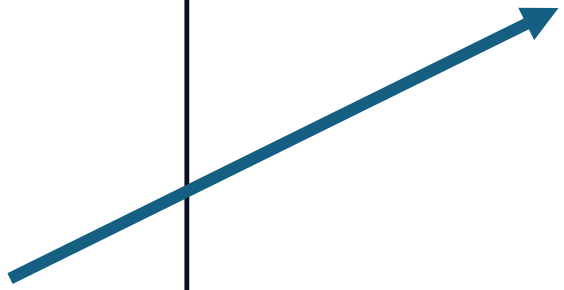

}
def get_latitude_and_longitude(city, state):
return {"latitude": 34.225727, "longitude": -77.944710}
def get_weather(latitude, longitutde, unit="fahrenheit"):
return {"temperature": 75, "unit": "fahrenheit", "description": "Sunny"}
def tool_router(tool_name, input):
match tool_name:
case "get_latitude_and_longitude":
return get_latitude_and_longitude(input["city"], input["state"])
case "get_weather":
return get_weather(input["latitude"], input["longitude"], input.get("unit"))
case _:
raise ValueError(f"Unknown tool: {tool_name}")
response = bedrock.converse(

**ANSWER**

**{"latitude": 34.225727, "longitude": -77.944710}**


}
def get_latitude_and_longitude(city, state):
return {"latitude": 34.225727, "longitude": -77.944710}
def get_weather(latitude, longitutde, unit="fahrenheit"):
return {"temperature": 75, "unit": "fahrenheit", "description": "Sunny"}
def tool_router(tool_name, input):
match tool_name:
case "get_latitude_and_longitude":
return get_latitude_and_longitude(input["city"], input["state"])
case "get_weather":
return get_weather(input["latitude"], input["longitude"], input.get("unit"))
case _:
raise ValueError(f"Unknown tool: {tool_name}")
response = bedrock.converse(

```
                {
                  "text": "To get the current weather for Wilmington,
                  we first need to get its latitude and longitude.
                  Let's invoke the get_latitude_and_longitude tool:"
                },
                {
                    "toolUse": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "name": "get_latitude_and_longitude",
                        "input": {"city": "Wilmington", "state": "NC"},
                    }
                },
            ],
        },
        {
            "role": "user",
            "content": [
                {
                    "toolResult": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "content": [
                            {"json": {"latitude": 34.225727, "longitude": -77.944710}}
                        ],
                    }
                }
            ],
        },
    ],
    toolConfig={"tools": tools},  # The two tools are here
```



```
                {
                  "text": "To get the current weather for Wilmington,
                  we first need to get its latitude and longitude.
                  Let's invoke the get_latitude_and_longitude tool:"
                },
                {
                    "toolUse": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "name": "get_latitude_and_longitude",
                        "input": {"city": "Wilmington", "state": "NC"},
                    }
                },
            ],
        },
        {
            "role": "user",
            "content": [
                {
                    "toolResult": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "content": [
                            {"json": {"latitude": 34.225727, "longitude": -77.944710}}
                        ],
                    }
                }
            ],
        },
    ],
    toolConfig={"tools": tools},  # The two tools are here
```

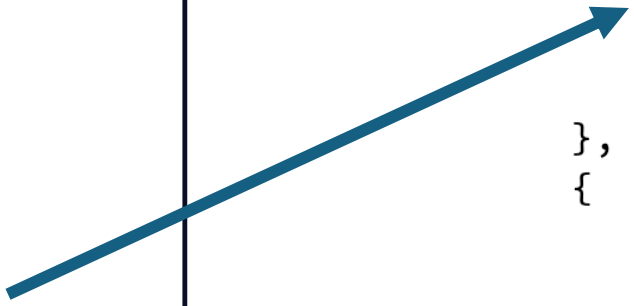
```
                {
                  "text": "To get the current weather for Wilmington,
                  we first need to get its latitude and longitude.
                  Let's invoke the get_latitude_and_longitude tool:"
                },
                {
                    "toolUse": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "name": "get_latitude_and_longitude",
                        "input": {"city": "Wilmington", "state": "NC"},
                    }
                },
            ],
        },
        {
            "role": "user",
            "content": [
                {
                    "toolResult": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "content": [
                            {"json": {"latitude": 34.225727, "longitude": -77.944710}}
                        ],
                    }
                }
            ],
        },
    ],
    toolConfig={"tools": tools},  # The two tools are here
```

```
                {
                  "text": "To get the current weather for Wilmington,
                  we first need to get its latitude and longitude.
                  Let's invoke the get_latitude_and_longitude tool:"
                },
                {
                    "toolUse": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "name": "get_latitude_and_longitude",
                        "input": {"city": "Wilmington", "state": "NC"},
                    }
                },
            ],
        },
        {
            "role": "user",
            "content": [
                {
                    "toolResult": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "content": [
                            {"json": {"latitude": 34.225727, "longitude": -77.944710}}
                        ],
                    }
                }
            ],
        },
    ],
    toolConfig={"tools": tools},  # The two tools are here
```

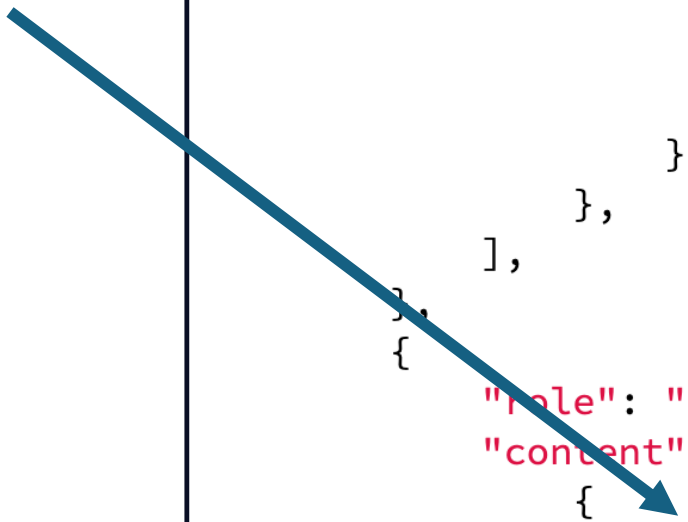
```
                {
                  "text": "To get the current weather for Wilmington,
                  we first need to get its latitude and longitude.
                  Let's invoke the get_latitude_and_longitude tool:"
                },
                {
                    "toolUse": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "name": "get_latitude_and_longitude",
                        "input": {"city": "Wilmington", "state": "NC"},
                    }
                },
            ],
        },
        {
            "role": "user",
            "content": [
                {
                    "toolResult": {
                        "toolUseId": "tooluse_AJUDl-bkRcmk9DKS52E_jw",
                        "content": [
                            {"json": {"latitude": 34.225727, "longitude": -77.944710}}
                        ],
                    }
                }
            ],
        },
    ],
    toolConfig={"tools": tools},  # The two tools are here
```

```
{
  "output": {
    "message": {
      "role": "assistant",
      "content": [
        {
          "text": "Now that we have the latitude and longitude for Wilmington, NC, we
           can use those coordinates to get the current weather with the
           get_weather tool:"
        },
        {
          "toolUse": {
            "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
            "name": "get_weather",
            "input": {
              "latitude": 34.225727,
              "longitude": -77.94471
            }
          }
        }
      ]
    }
  },
  "stopReason": "tool_use"
}
```

```
        "content": [
            {
                "text": "Now that we have the latitude and longitude
                 for Wilmington, NC, we can use those coordinates to
                 get the current weather with the get_weather tool:"
            },
            {
                "toolUse": {
                    "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                    "name": "get_weather",
                    "input": {"latitude": 34.225727, "longitude": -77.94471},
                }
            },
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "toolResult": {
                    "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                    "content": [
                        {
                            "json": {
                                "temperature": 75,
                                "unit": "fahrenheit",
                                "description": "Sunny",
                            }
                        }
```



```
        "content": [
            {
                "text": "Now that we have the latitude and longitude
                 for Wilmington, NC, we can use those coordinates to
                 get the current weather with the get_weather tool:"
            },
            {
                "toolUse": {
                    "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                    "name": "get_weather",
                    "input": {"latitude": 34.225727, "longitude": -77.94471},
                }
            },
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "toolResult": {
                    "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                    "content": [
                        {
                            "json": {
                                "temperature": 75,
                                "unit": "fahrenheit",
                                "description": "Sunny",
                            }
                        }
```



```
"content": [
    {
        "text": "Now that we have the latitude and longitude
         for Wilmington, NC, we can use those coordinates to
         get the current weather with the get_weather tool:"
    },
    {
        "toolUse": {
            "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
            "name": "get_weather",
            "input": {"latitude": 34.225727, "longitude": -77.94471},
        }
    },
],
},
{
    "role": "user",
    "content": [
        {
            "toolResult": {
                "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                "content": [
                    {
                        "json": {
                            "temperature": 75,
                            "unit": "fahrenheit",
                            "description": "Sunny",
                        }
                    }
```

```
        "content": [
            {
                "text": "Now that we have the latitude and longitude
                 for Wilmington, NC, we can use those coordinates to
                 get the current weather with the get_weather tool:"
            },
            {
                "toolUse": {
                    "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                    "name": "get_weather",
                    "input": {"latitude": 34.225727, "longitude": -77.94471},
                }
            },
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "toolResult": {
                    "toolUseId": "tooluse_26KH3B67TreBks2aj4XxWg",
                    "content": [
                        {
                            "json": {
                                "temperature": 75,
                                "unit": "fahrenheit",
                                "description": "Sunny",
                            }
                        }
```

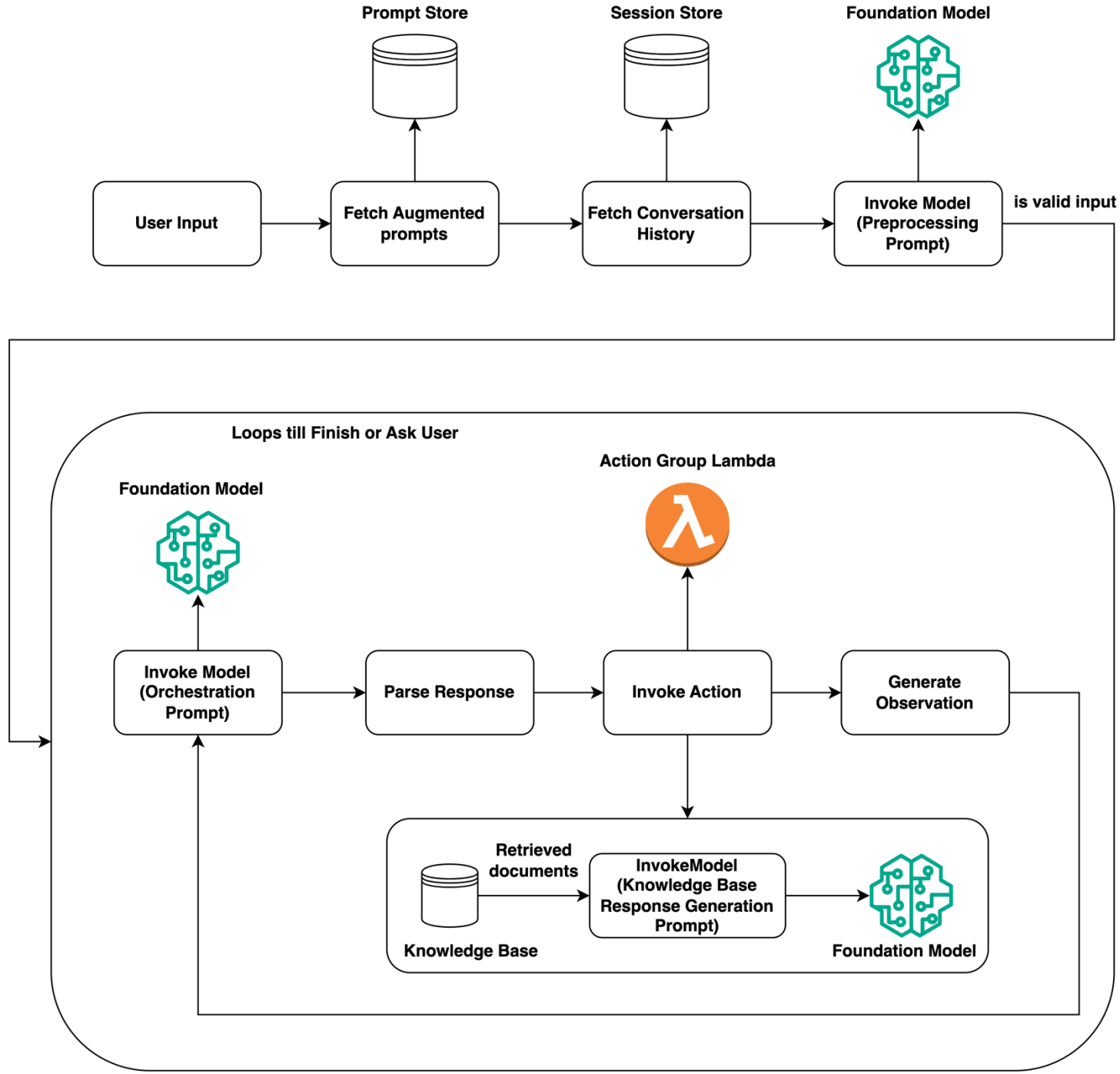
```
    toolConfig={"tools": tools},  # The two tools are here
)
```

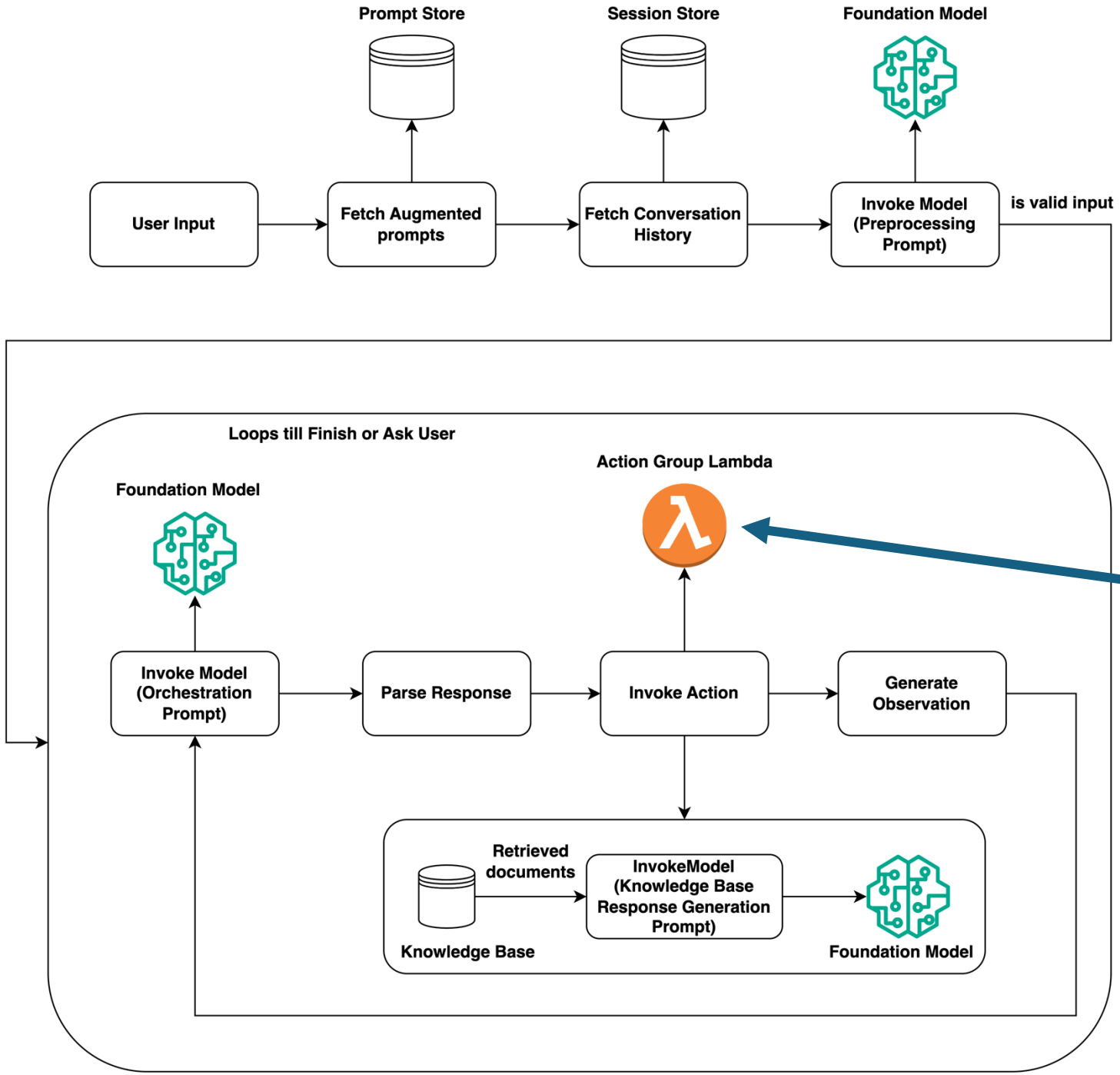
```
{
  "output": {
    "message": {
      "role": "assistant",
      "content": [
        {
          "text": "The current weather in Wilmington, NC is sunny and 75°F."
        }
      ]
    }
  },
  "stopReason": "end_turn"
}
```

# Agent Use Cases

1. Virtual assistants
2. Customer service automation
3. Email assistants
4. Lead generation

# Amazon Bedrock Agents


Prompt Store
Session Store
Foundation Model
User Input
Fetch Augmented prompts
Fetch Conversation History
Invoke Model (Preprocessing Prompt)
is valid input
Loops till Finish or Ask User
Action Group Lambda
Foundation Model
Invoke Model (Orchestration Prompt)
Parse Response
Invoke Action
Generate Observation
Retrieved documents
InvokeModel (Knowledge Base Response Generation Prompt)
Knowledge Base
Foundation Model




```python
def get_latitude_and_longitude(city, state):
    return {"latitude": 34.225727, "longitude": -77.944710}


def get_weather(latitude, longitutde, unit="fahrenheit"):
    return {"temperature": 75, "unit": "fahrenheit", "description": "Sunny"}


def tool_router(tool_name, input):
    match tool_name:
        case "get_latitude_and_longitude":
            return get_latitude_and_longitude(input["city"], input["state"])
        case "get_weather":
            return get_weather(input["latitude"], input["longitude"], input.get("unit"))
        case _:
            raise ValueError(f"Unknown tool: {tool_name}")
```

# Take aways

Amazon Bedrock grants access to many leading foundation models

“Knowledge bases” does the heavy lifting, you focus on getting data

You can easily orchestrate difficult tasks using function calling

# Amazon Bedrock is developer friendly

# Questions?

# Appendix

# Production Learnings

1. Rate limits are harsh
2. Region availability is limited
3. Testing is difficult
4. Hallucinations are real
    1. Feedback
    2. Guardrails
5. Customers want HITL
6. Provisioned throughput is $$$
7. Lack of trust in OpenAI
8. Design with retries in mind
9. Price by usage