



# Building Serverless AI Agents on AWS with Strands Agents SDK



**Anton Aleksandrov**

Principal Solutions Architect, Serverless  
AWS



# Enterprises are doubling down on agents

33%

of enterprise software apps will include agentic AI by 2028, up from less than 1% in 2024.

Gartner, "Top strategic Technology Trends for 2025," October 2024.

15%

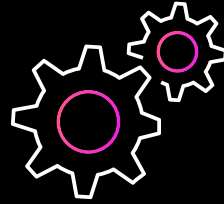
of day-to-day work decisions will be made autonomously through agentic AI by 2028.

Gartner, "Top Strategic Technology Trends: agentic AI – The evolution of Experience" February 2025

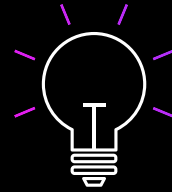
# Businesses are creating value with AI agents



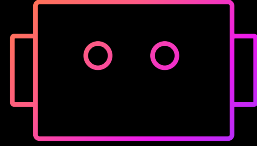
Workplace  
productivity



Business  
workflows



Innovation  
and research



# What are AI Agents?

Autonomous software systems that leverage AI to reason, plan, and complete tasks on behalf of humans or systems

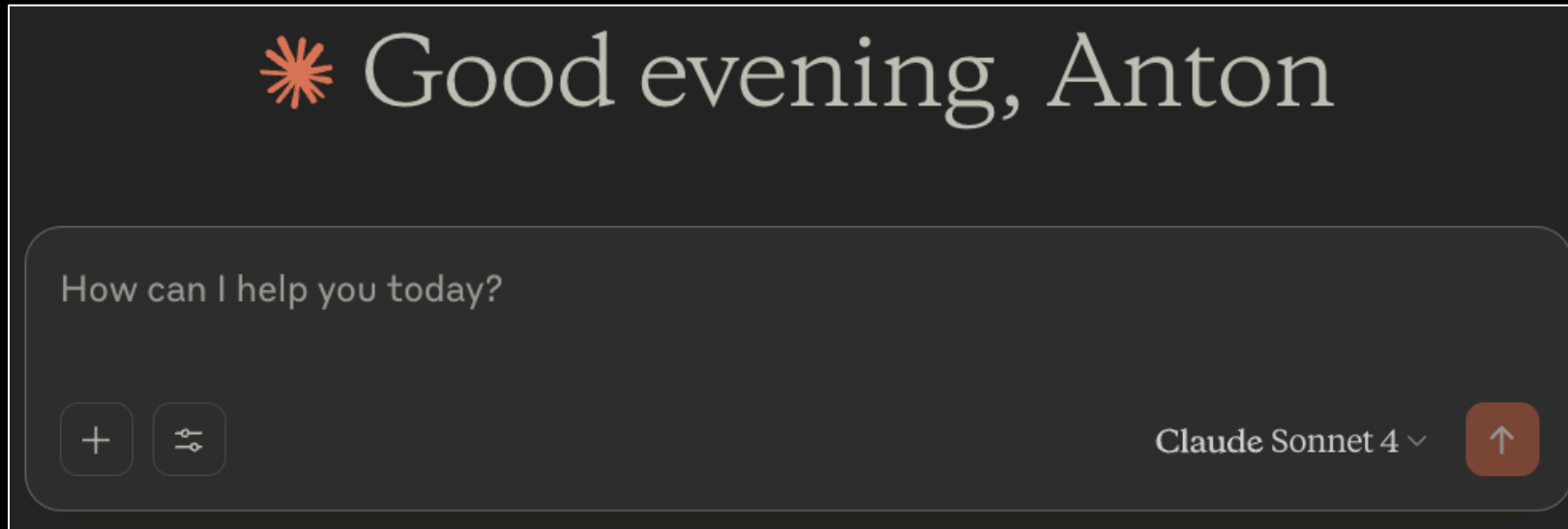


# The GenAI Evolution

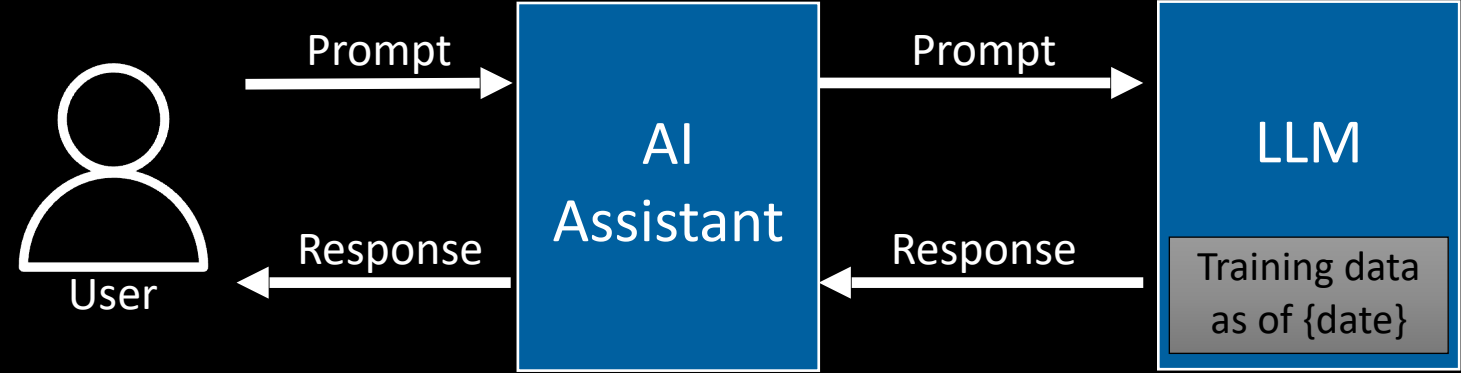
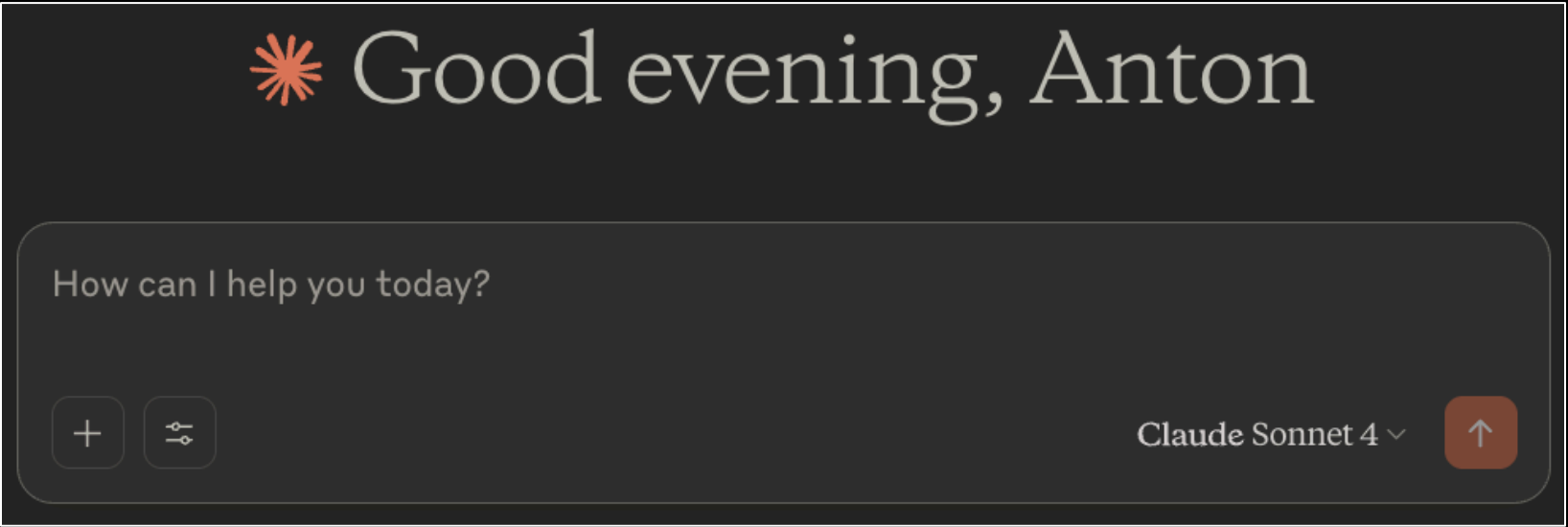
---

Agent? RAG? Tools? LLM? Inference?

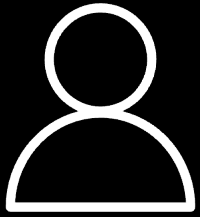
# The Genesis



# The Genesis



# The Genesis



User

Write me a  
poem about  
serverless

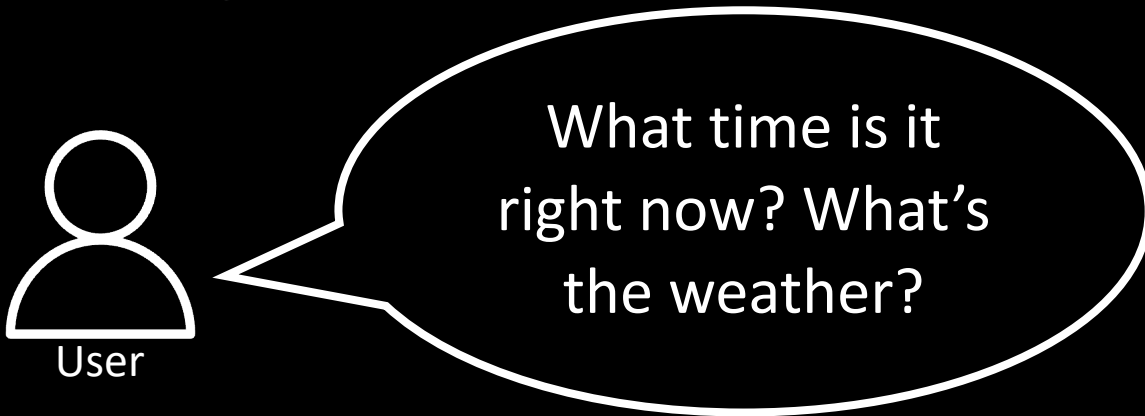
"Events gently flow,  
Your apps simply run.  
No servers to tend—  
It's cloud and it's fun"

- Claude 3.7 Sonnet, 2025



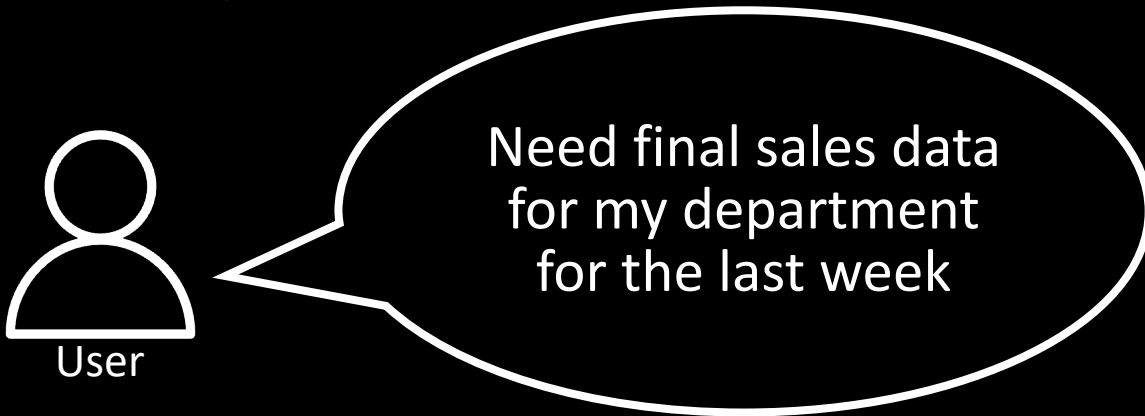
- DALL-E, 2025

# Realizing the limitations



I don't have access to real-time information like the current time. As an AI, I don't have the ability to check the current time or date unless you provide that information to me. If you need to know the current time, you could check your device's clock, a watch, or an online time service.

# Realizing the limitations



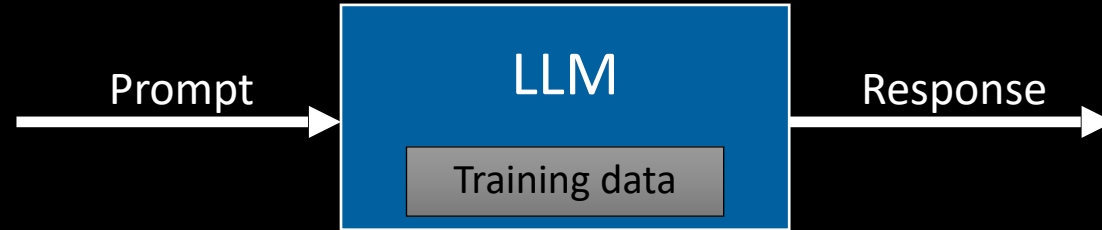
I'd be happy to help you access the final sales data for your department from last week. However, I don't have direct access to your company's sales database or records.

# Realizing the limitations



I'd be happy to help you plan a trip to Paris, but I'm not able to make actual bookings or reservations directly. I can guide you through the process of booking your Paris trip.

# Realizing the limitations



**LLMs don't know what they don't know.**

**LLMs can generate content, but they cannot take actions.**

**LLMs are stateless. They don't retain memory between interactions.**



# Injecting the context

how much is 1+2+3?

1 + 2 + 3 = 6 ✓

# Injecting the context

how much is 1+2+3?

$1 + 2 + 3 = 6$  ✓

assume 1 = apple, 2=peach, and 3 = grape. How much is 1+2+3?

If 1 = apple, 2 = peach, and 3 = grape, then:

$1 + 2 + 3 = \text{apple} + \text{peach} + \text{grape}$

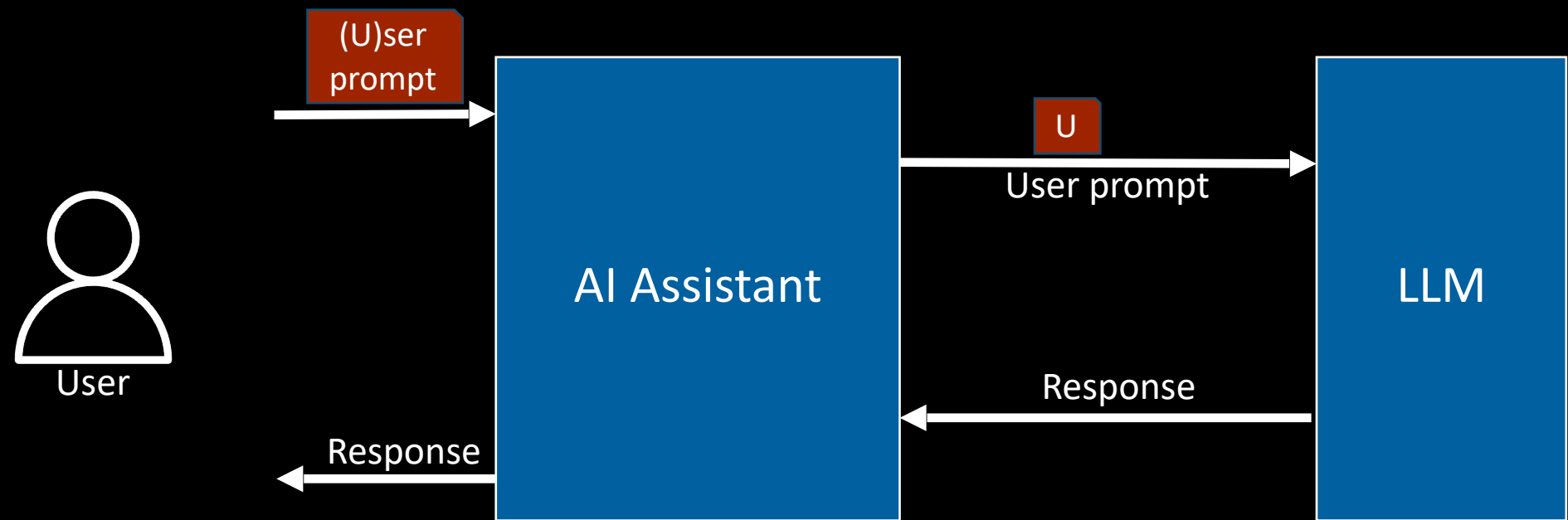
So the answer is a delicious fruit salad 🍌🍏🍑🍇

Realizing the limitations

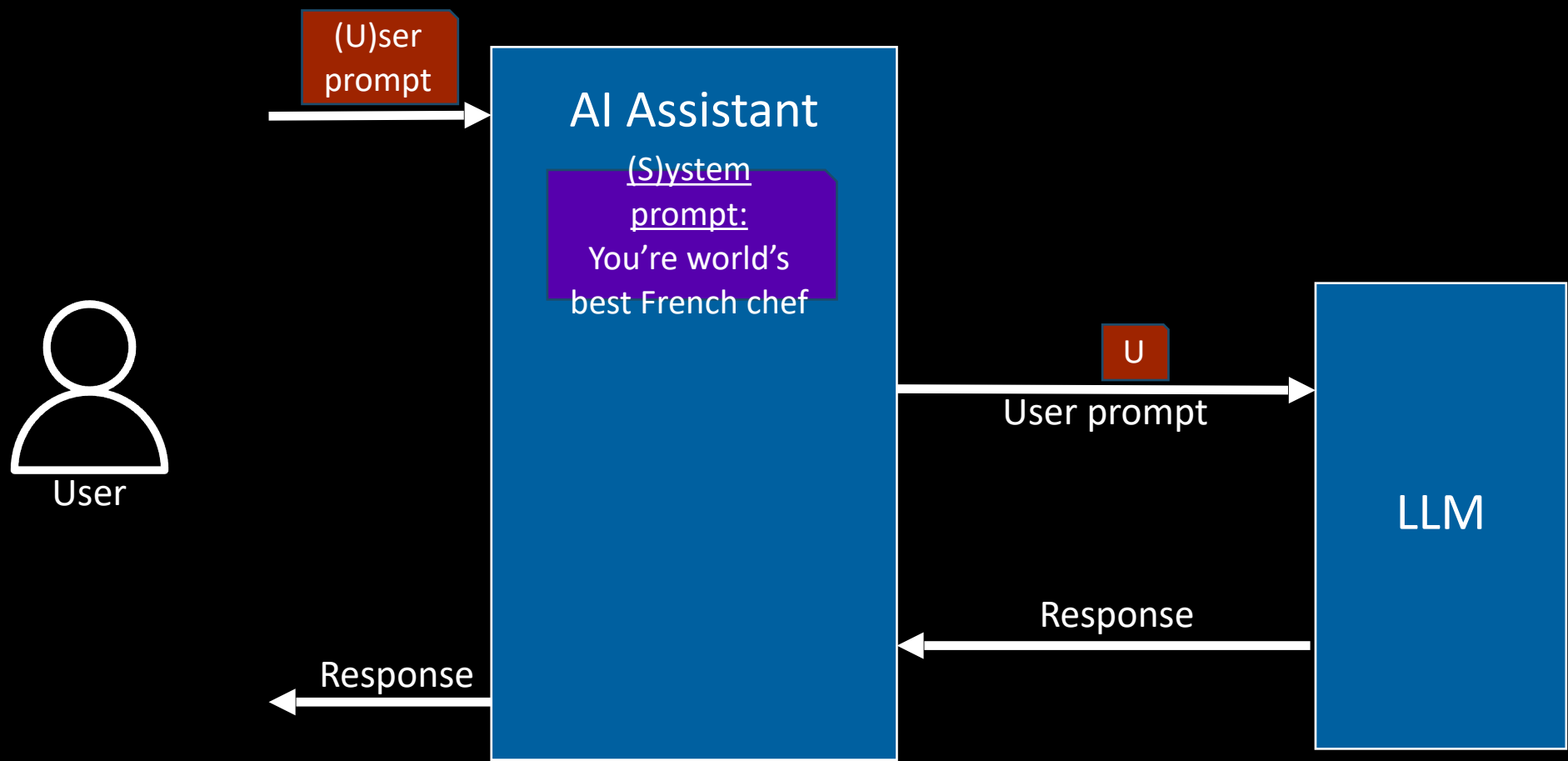
**So... can we inject the context  
into prompts to make LLMs  
smarter!?**

**Absolutely!!!!**

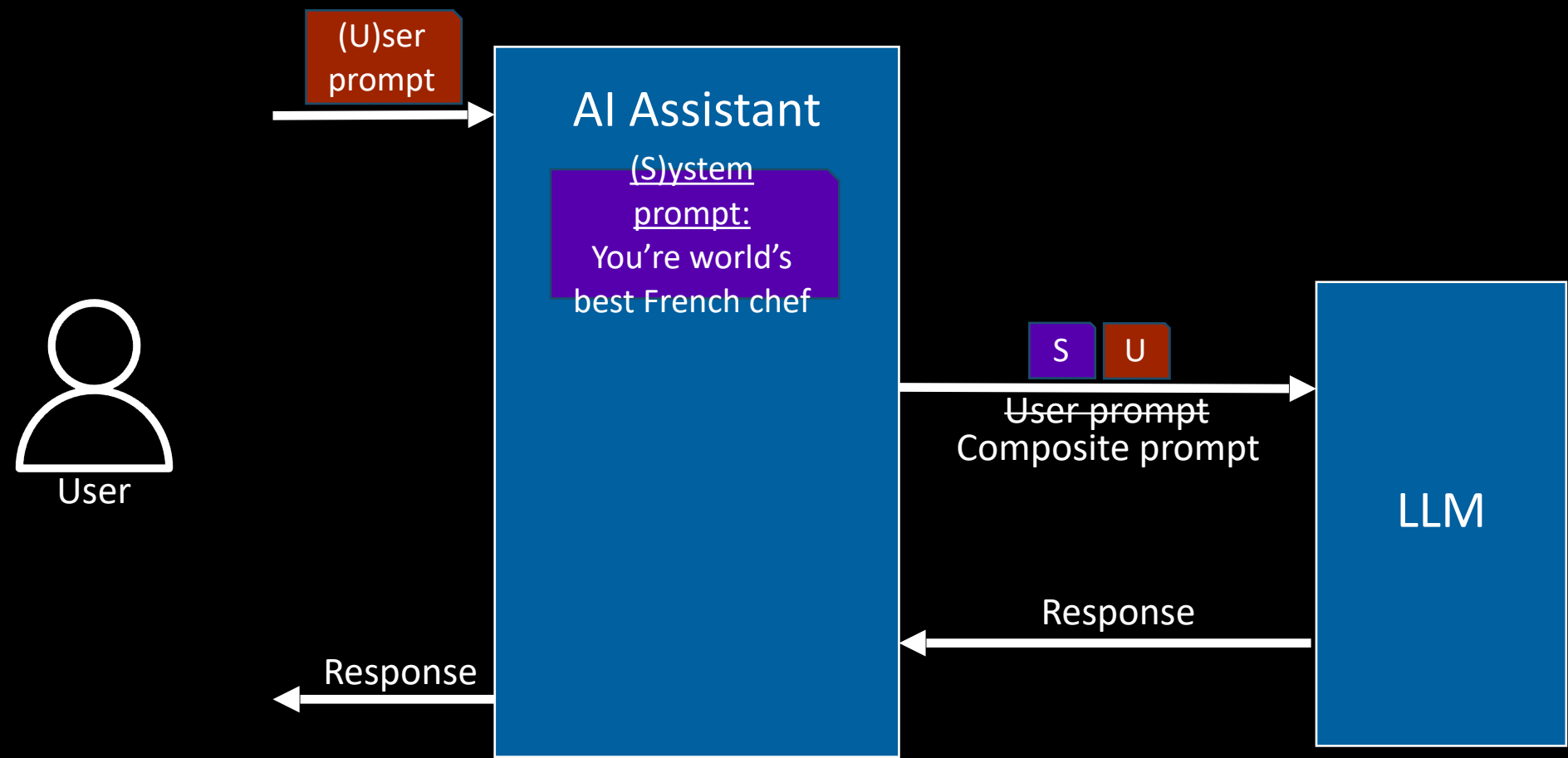
# Giving your assistant a personality



# Giving your assistant a personality



# Giving your assistant a personality



# Giving your assistant a personality

How can you help me?

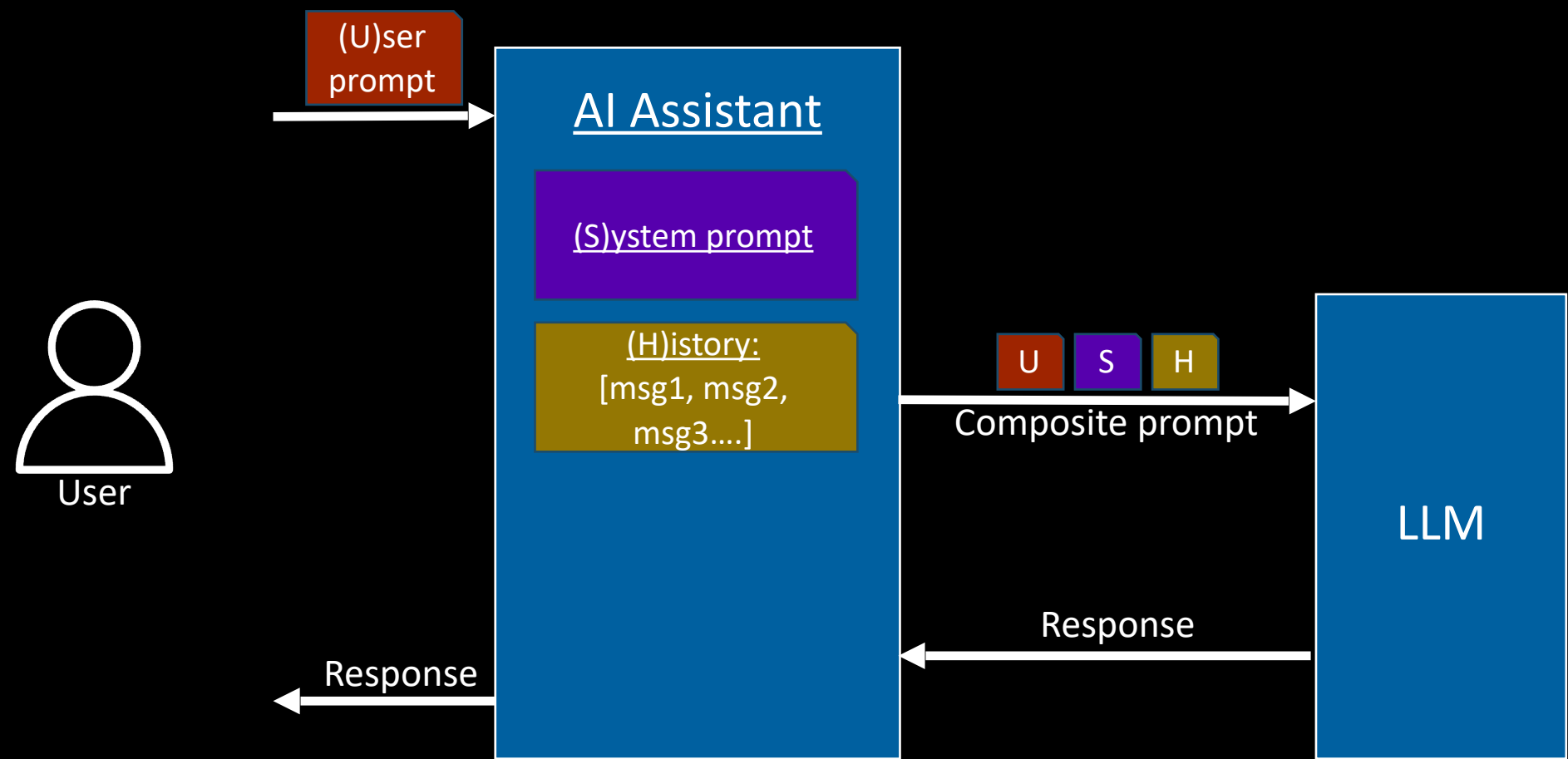
# Giving your assistant a personality

How can you help me?

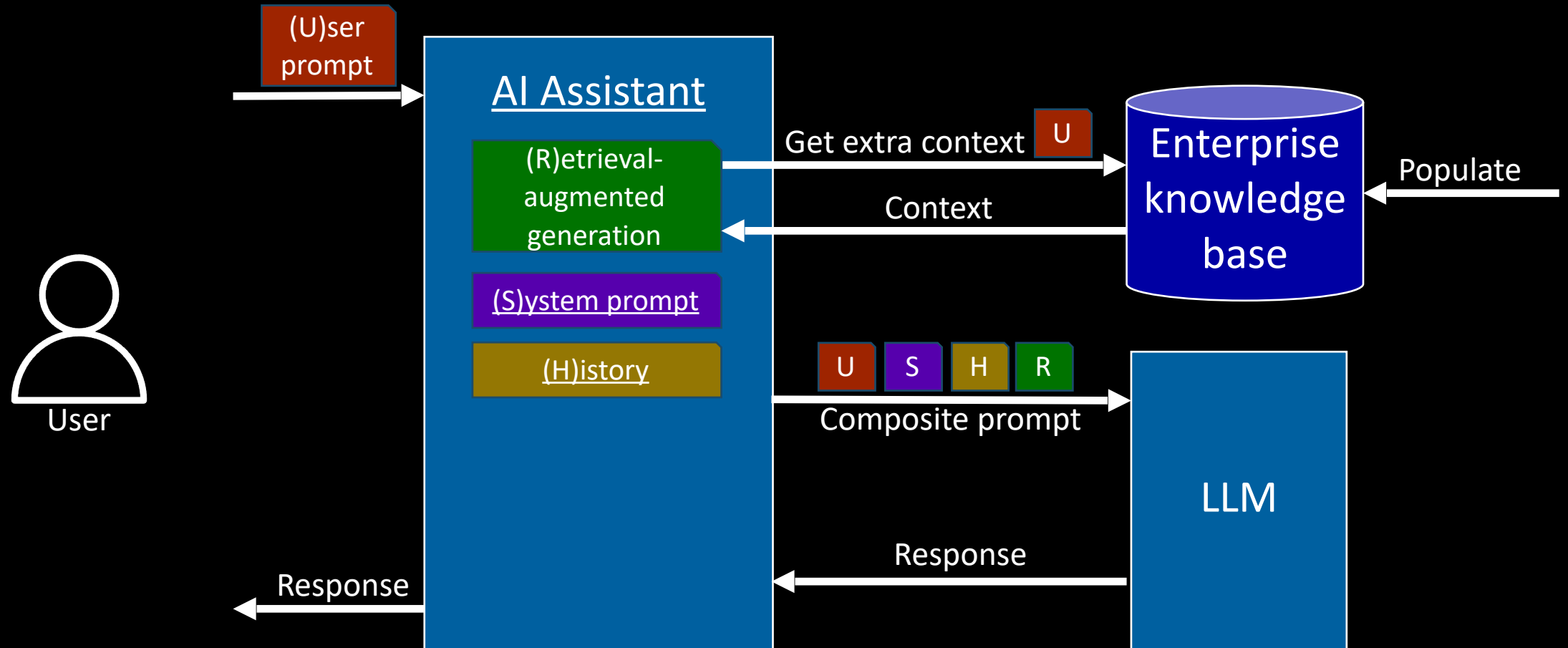
Ah, bonjour monsieur Anton! 🧑🍳 I am Chef Gaston LeFeuvre, at your service — Michelin dreams in every dish, butter in every heartbeat. Ici pour transformer your kitchen into a template of haute cuisine and guide your hands like a maestro conducting a symphony of flavours.



# Giving your assistant memory



# Retrieval-Augmented Generation (RAG)



# Retrieval-Augmented Generation (RAG)

A

Need final sales data for my department for the last week

# Retrieval-Augmented Generation (RAG)

**A** Need final sales data for my department for the last week

Based on your department's sales data for the last week, here's a summary of the final figures:

Total Sales: \$247,835.92

Units Sold: 1,482

Average Transaction Value: \$167.23

Conversion Rate: 24.8%

Top performing products:

- Premium Package: \$82,450 (33.3% of total sales)
- Standard Bundle: \$64,290 (25.9% of total sales)
- Basic Plan: \$43,870 (17.7% of total sales)

The department saw a 12.3% increase compared to the previous week, largely driven by the new promotion on Premium Packages. The eastern region continues to be the strongest performer, accounting for 41% of total sales.

Would you like more specific details about any particular aspect of these sales figures?

# Adding real-time context dynamically

What are the concerts I can go to next weekend?

# Adding real-time context dynamically

What are the concerts I can go to next weekend?

Here are some great concerts happening next weekend (Saturday–Sunday, July 26–27, 2025) in Austin:

## Saturday, July 26

- **The Fray** – *How to Save a Life: 20th Anniversary Tour* at ACL Live at The Moody Theater, 7 PM
- **Mason Lively** with Telander at 3TEN ACL Live, 7 PM
- **Africa Night!** (Afro Jazz) at Sahara Lounge, 7 PM

## Sunday, July 27

- **Chris Isaak** at Paramount Theatre, 7 PM
- **12 Stones** at Come and Take It Live, 6 PM
- **Maye** at Antone's Nightclub, 8 PM

Let me know which ones catch your eye.

Sources

# Adding real-time context dynamically

What are the concerts I can go to next weekend?

Here are some great concerts happening next weekend (Saturday–Sunday, July 26–27, 2025) in Austin:

## Saturday, July 26

- **The Fray** – *How to Save a Life: 20th Anniversary Tour* at ACL Live at The Moody Theater, 7 PM
- **Mason Lively** with Telander at 3TEN ACL Live, 7 PM
- **Africa Night!** (Afro Jazz) at Sahara Lounge, 7 PM

## Sunday, July 27

- **Chris Isaak** at Paramount Theatre, 7 PM
- **12 Stones** at Come and Take It Live, 6 PM
- **Maye** at Antone’s Nightclub, 8 PM

Let me know which ones catch your eye.

Sources

Showlist Austin

### Showlist Austin

Saturday, July 26th 2025. Africa Night! featuring Afro Jazz, Cazayoux ... Sunday, July 27th 2025. Candler...

Do512

### Live music in Austin on July 26th, 2025 - Do512

In 2 days — The Fray - How To Save A Life: The 20t... ACL Live at the Moody Theater. 7:00PM ; Radiohead ...

acllive.com

### July 2025 - Events | Austin City Limits Live

July 25, 2025. ACL Live at 3TEN. Aubrey Logan. with special guest Jo James. Get TicketsMore Info · More...

Songkick

### Austin Concerts, Festivals, Tickets & Tour Dates 2025 & 2026

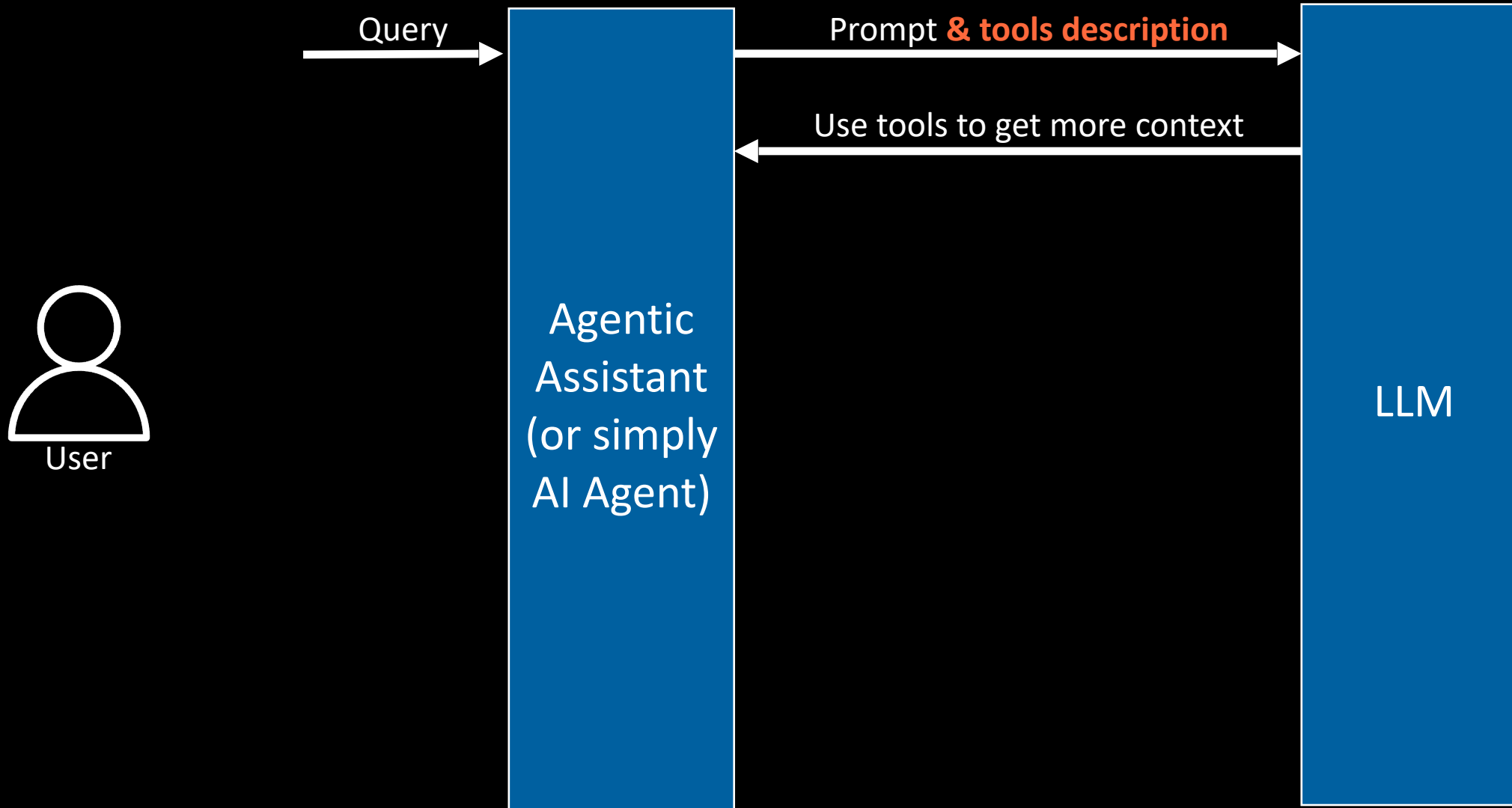
Lit Lounge, Austin, TX, US. Saturday 26 July 2025 – Saturday 26 July 2025. Grandmaster's Gala 2025...

austinconcerts2025.com

### Austin Concerts 2025 - Calendar of Events, Tickets, & More!

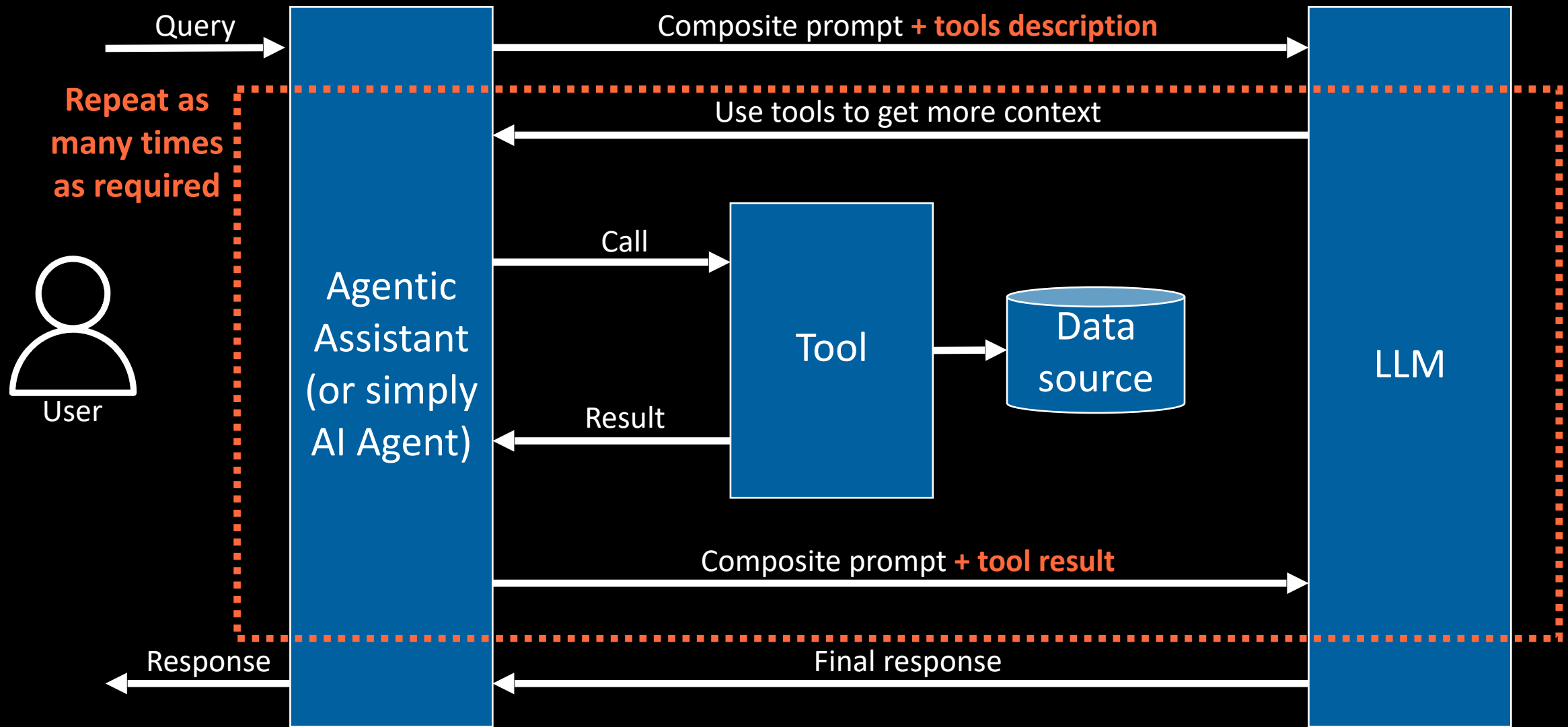
Sat, Jul 26, 2025 7:00 PM · Get Tickets · The Fray ACL Live At The Moody Theater ... Sun, Jul 27, 2025 6:00

# A simple agentic workflow

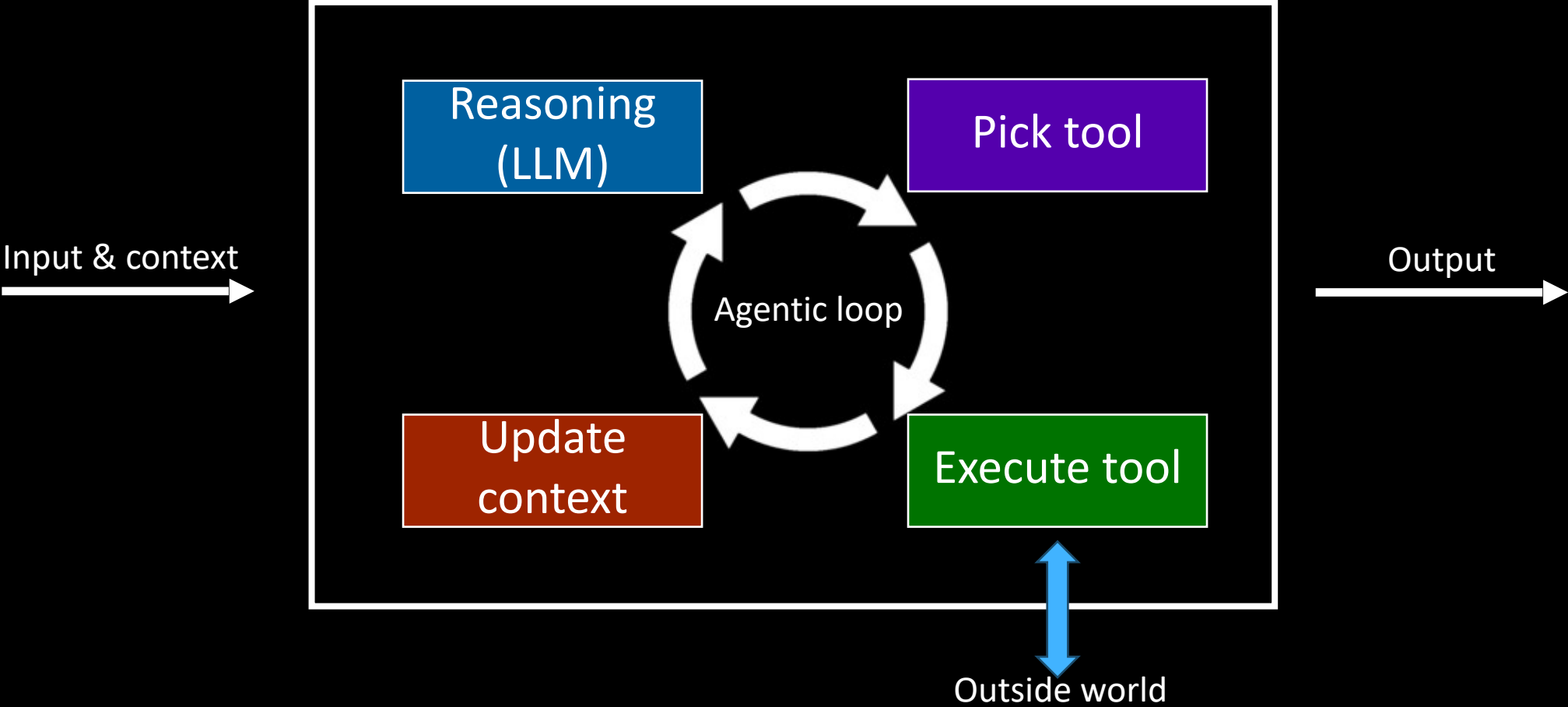




# A simple agentic workflow



# Agentic AI system in a nutshell



# Building an AI Agent

---

“Book me a business trip to NYC”

# Building your first AI Agent



**I want to build a Travel Agent that helps my employees to book business travel while enforcing corporate travel policies.**

# Implementing a simple AI Agent workflow

```
def ai_agent(user_prompt):
```

# Implementing a simple AI Agent workflow

```
def ai_agent(user_prompt):  
    system_prompt = "You're an AcmeCorp Corporate Travel Agent...."  
    rag_context = get_rag_context(user_prompt)  
    tools = get_tools()  
    composite_prompt = [system_prompt, history, rag_context, tools, user_prompt]
```

# Implementing a simple AI Agent workflow

```
def ai_agent(user_prompt):  
    system_prompt = "You're an AcmeCorp Corporate Travel Agent...."  
    rag_context = get_rag_context(user_prompt)  
    tools = get_tools()  
    composite_prompt = [system_prompt, history, rag_context, tools, user_prompt]  
  
    while True:  
        llm_response = call_llm(composite_prompt)  
        history.append(user_prompt, llm_response)
```

# Implementing a simple AI Agent workflow

```
def ai_agent(user_prompt):  
    system_prompt = "You're an AcmeCorp Corporate Travel Agent...."  
    rag_context = get_rag_context(user_prompt)  
    tools = get_tools()  
    composite_prompt = [system_prompt, history, rag_context, tools, user_prompt]  
  
    while True:  
        llm_response = call_llm(composite_prompt) ←  
        history.append(user_prompt, llm_response)  
  
        if llm_response.status == "CALL_TOOL":  
            tool_result = call_tool(llm_response.tool_name,  
                                   llm_response.tool_params)  
  
            composite_prompt = [system_prompt, history,  
                               rag_context, tools,  
                               user_prompt, tool_result]
```



# Implementing a simple AI Agent workflow

```
def ai_agent(user_prompt):  
    system_prompt = "You're an AcmeCorp Corporate Travel Agent...."  
    rag_context = get_rag_context(user_prompt)  
    tools = get_tools()  
    composite_prompt = [system_prompt, history, rag_context, tools, user_prompt]  
  
    while True:  
        llm_response = call_llm(composite_prompt)  
        history.append(user_prompt, llm_response)  
  
        if llm_response.status == "CALL_TOOL":  
            tool_result = call_tool(llm_response.tool_name,  
                                   llm_response.tool_params)  
  
            composite_prompt = [system_prompt, history,  
                               rag_context, tools,  
                               user_prompt, tool_result]  
        else:  
            break
```

# Implementing a simple AI Agent workflow

```
def ai_agent(user_prompt):  
    system_prompt = "You're an AcmeCorp Corporate Travel Agent...."  
    rag_context = get_rag_context(user_prompt)  
    tools = get_tools()  
    composite_prompt = [system_prompt, history, rag_context, tools, user_prompt]  
  
    while True:  
        llm_response = call_llm(composite_prompt)  
        history.append(user_prompt, llm_response)  
  
        if llm_response.status == "CALL_TOOL":  
            tool_result = call_tool(llm_response.tool_name,  
                                   llm_response.tool_params)  
  
            composite_prompt = [system_prompt, history,  
                               rag_context, tools,  
                               user_prompt, tool_result]  
        else:  
            break  
  
    return llm_response
```

# Implementing a simple AI Agent workflow

# BOILERPLATE

```
def ai_agent(user_prompt):
    system_prompt = "You're an AcmeCorp Corporate Travel Agent..."
    rag_context = get_rag_context(user_prompt)
    tools = get_tools()
    composite_prompt = [system_prompt, history, rag_context, tools, user_prompt]

    while True:
        llm_response = call_llm(composite_prompt)
        history.append(user_prompt, llm_response)

        if llm_response.status == "CALL_TOOL":
            tool_result = call_tool(llm_response.tool_name,
                                   llm_response.tool_params)

            composite_prompt = [system_prompt, history,
                               rag_context, tools,
                               user_prompt, tool_result]
        else:
            break

    return llm_response
```

# CODE

# Strands Agents SDK - Built for builders

Who value flexibility, speed, and simplicity



## Model & deployment choice

- ✓ Model choice
- ✓ Custom model providers
- ✓ Deploy anywhere

## Highly flexible

- ✓ Safeguard with guardrails
- ✓ Native observability
- ✓ Monitoring
- ✓ Evaluation

## Broad selection of tools

- ✓ MCP integration
- ✓ Custom tools
- ✓ Coordinate multiple agents
- ✓ Multi-modal
- ✓ Fetch web data
- ✓ Read and write files
- ✓ Interpret code

## Integrations

- ✓ Use AWS services
- ✓ LiteLLM
- ✓ Mem0
- ✓ RAGAS
- ✓ Tavily
- ✓ Langfuse



# Creating your first agent

```
from strands.models import BedrockModel
from strands import Agent

# Default is Claude Sonnet 3.7 in us-west-1
model = BedrockModel(
    region_name="us-east-1",
    model_id="us.anthropic.claude-3-5-haiku-20241022-v1:0"
)
```

Configure model

# Creating your first agent

```
from strands.models import BedrockModel
from strands import Agent
```

```
# Default is Claude Sonnet 3.7 in us-west-1
model = BedrockModel(
    region_name="us-east-1",
    model_id="us.anthropic.claude-3-5-haiku-20241022-v1:0"
)
```

Configure model

```
agent = Agent(
    model=model
)
```

Create agent

# Creating your first agent

```
from strands.models import BedrockModel
from strands import Agent
```

```
# Default is Claude Sonnet 3.7 in us-west-1
model = BedrockModel(
    region_name="us-east-1",
    model_id="us.anthropic.claude-3-5-haiku-20241022-v1:0"
)
```

Configure model

```
agent = Agent(
    model=model
)
```

Create agent

```
agent("How can you help me?")
```

Prompt

# Creating your first agent

```
> python app.py
```

I can help you with a wide range of tasks, such as:

1. Answering questions on various topics
2. Providing writing assistance (essays, emails, etc.)
3. Explaining complex concepts
4. Helping with research
5. Offering problem-solving advice
6. Providing coding help and explanations
7. Brainstorming ideas
8. Proofreading and editing
9. Math and calculation support
10. Language translation

What specific task or topic would you like help with today?



# Adding system prompt

```
agent = Agent(  
    model=model,  
    system_prompt=  
        """You are an enterprise travel agent for AcmeCorp.  
        Your job is to help employees book business travel  
        that complies with company policies."""  
)  
  
agent("How can you help me?")
```

# Adding system prompt

```
agent = Agent(  
    model=model,  
    system_prompt=  
        """You are an enterprise travel agent for AcmeCorp.  
        Your job is to help employees with their travel needs  
        that align with company policies.  
        """  
)  
  
agent("How can I help you with your travel today?")
```

> python app.py

As an enterprise travel agent for AcmeCorp, I can help you with:

- Booking business travel arrangements
- Understanding company travel policies
- Finding approved hotels and transportation
- Ensuring travel expenses comply with corporate guidelines
- Answering questions about travel reimbursement
- Recommending cost-effective travel options

What type of business travel do you need assistance with today?

# Validating the session persistence

```
agent = Agent(  
    model=model,  
    system_prompt=  
        """You are an enterprise travel agent for AcmeCorp.  
        Your job is to help employees book business travel  
        that complies with company policies."""  
)  
  
agent("I always dreamt of visiting Tokyo")  
agent("What language do they speak there?")
```

# Validating the session persistence

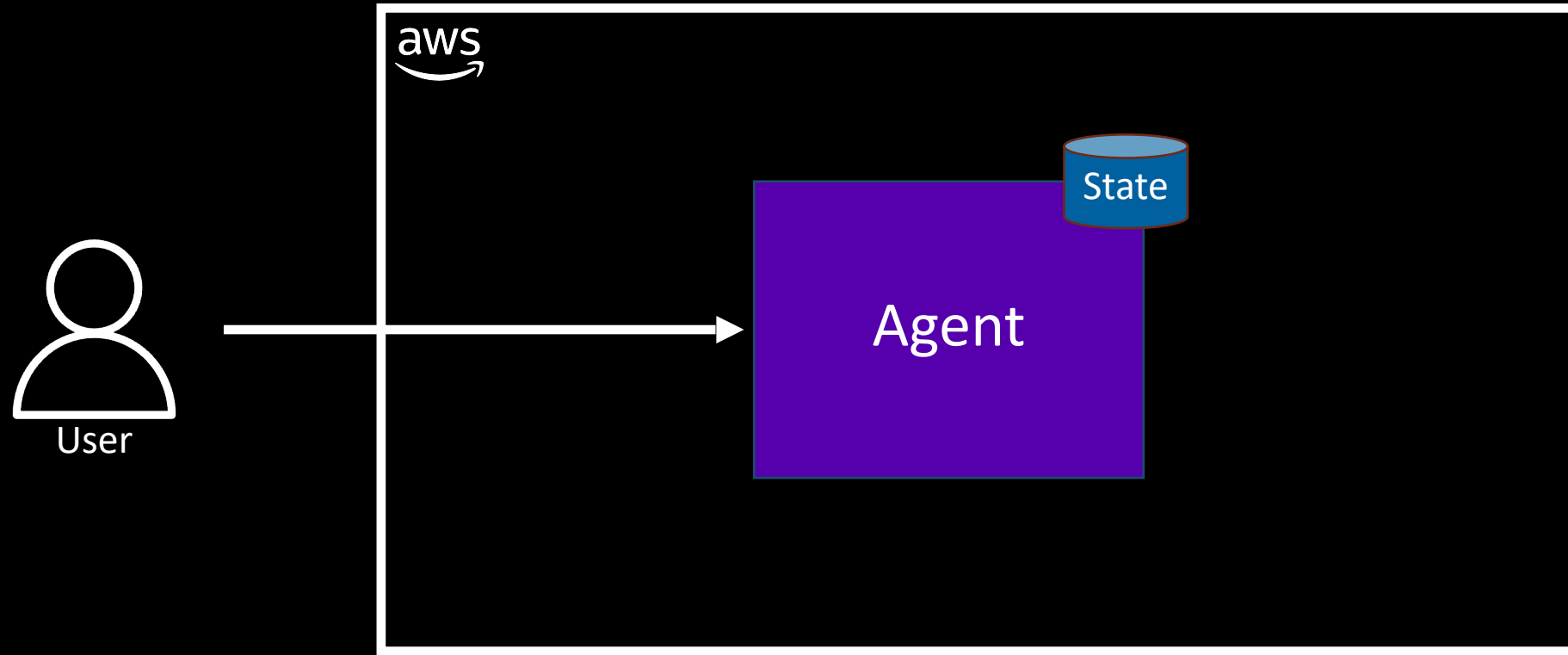
```
agent = Agent(  
    model=model,  
    system_prompt=  
        """"You are an enterprise travel agent for AcmeCorp.  
        Your job is to help employees book business travel  
        that co  
        """  
)  
  
agent("I al  
agent("What
```

› python app.py

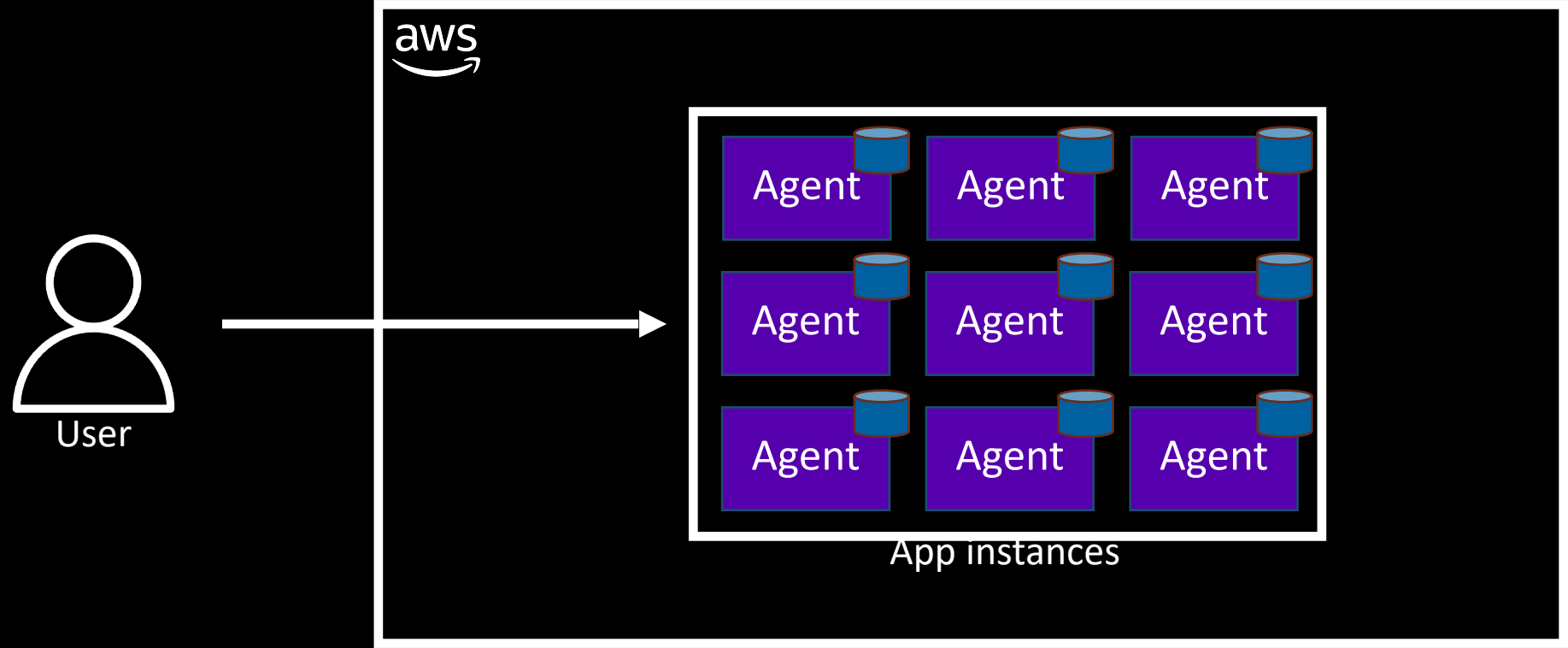
As an enterprise travel agent, I can help you plan a business trip to Tokyo while ensuring compliance with AcmeCorp's travel policies.

In Tokyo, and throughout Japan, the primary language spoken is Japanese (日本語, Nihongo).

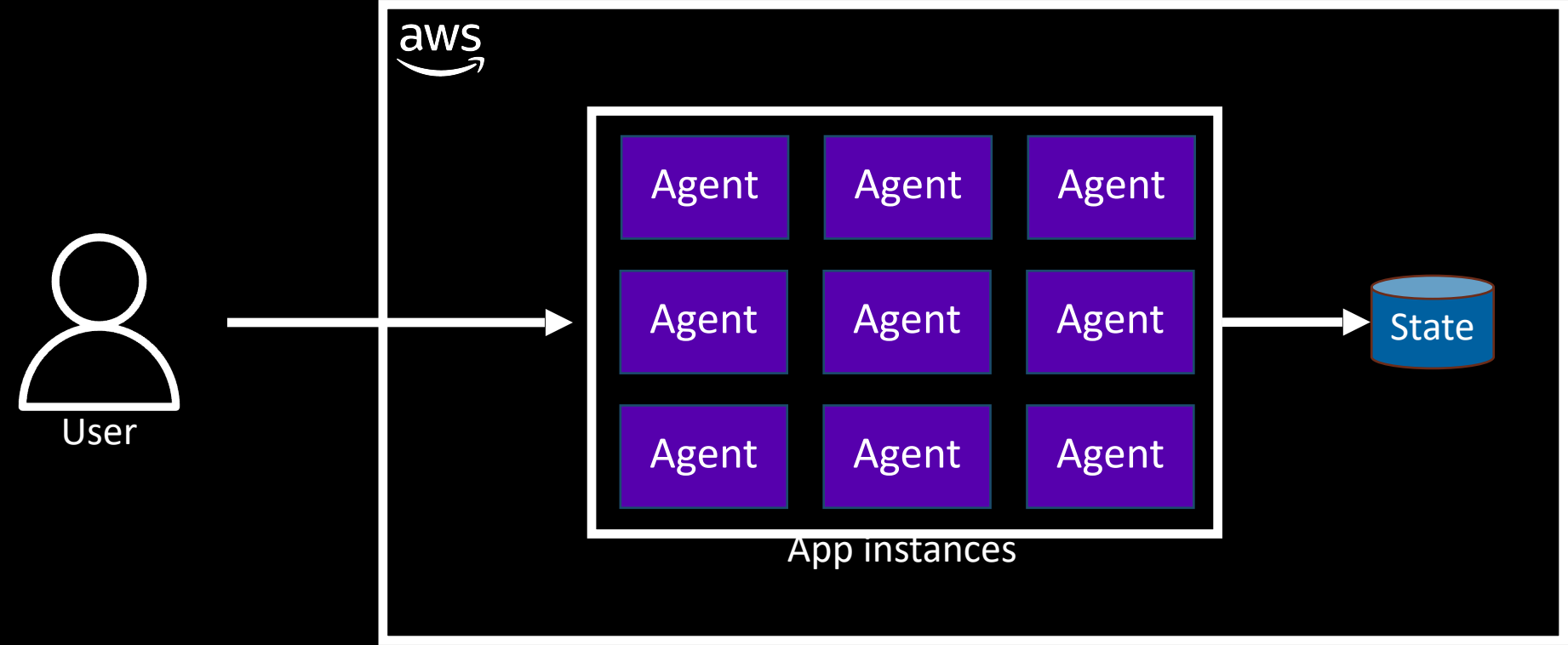
# The problem with stateful apps



# The problem with stateful apps




# Externalizing the state



# Externalizing the session

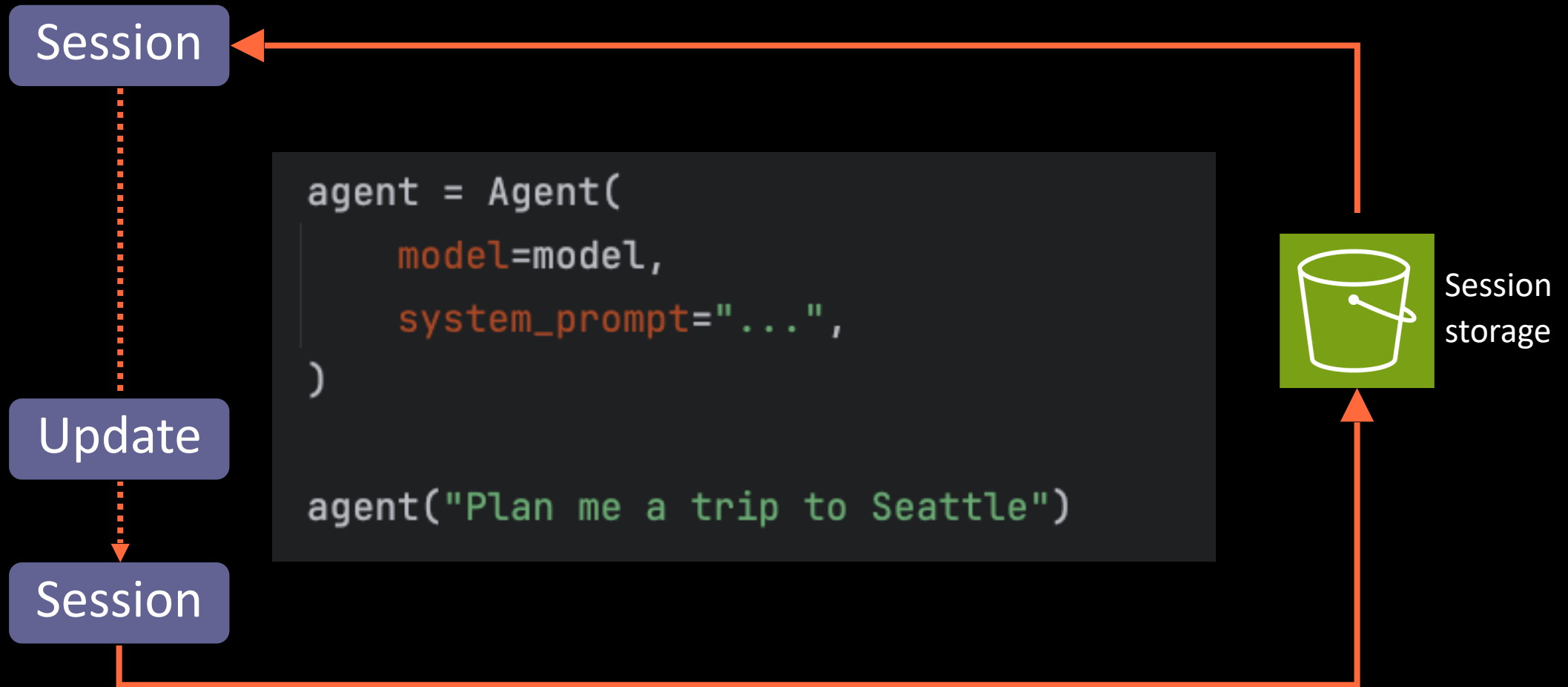
```
agent("Book me a trip to Tokyo")  
print(agent.messages)
```



```
[  
  {  
    "role": "user",  
    "content": [{  
      "text": "Book me a trip to Tokyo"  
    }]  
  },  
  {  
    "role": "assistant",  
    "content": [  
      {  
        "text": "As a business travel agent....."  
      }  
    ]  
  }  
]
```



# Externalizing the session



# Externalizing the session

```
session_manager = S3SessionManager(  
    session_id="my_session",  
    bucket=SESSION_STORE_BUCKET_NAME,  
    prefix="agent_sessions"  
)
```

```
agent = Agent(  
    model=model,  
    system_prompt="...",  
)
```

```
agent("Plan me a trip to Seattle")
```



Session  
storage

# Externalizing the session

```
session_manager = S3SessionManager(  
    session_id="my_session",  
    bucket=SESSION_STORE_BUCKET_NAME,  
    prefix="agent_sessions"  
)
```

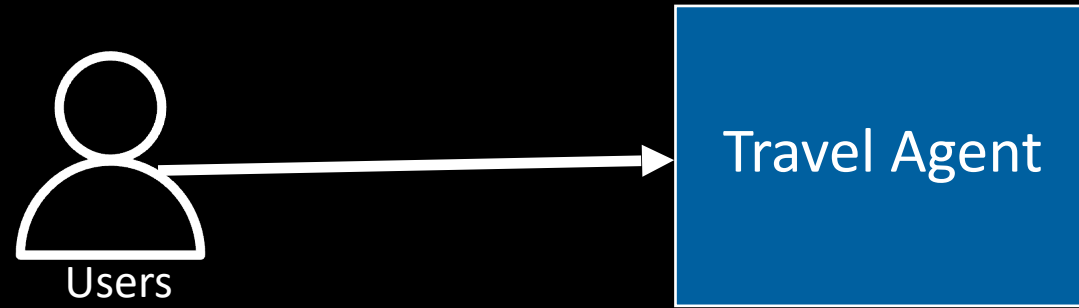
```
agent = Agent(  
    model=model,  
    system_prompt="...",  
    session_manager=session_manager,  
)
```

```
agent("Plan me a trip to Seattle")
```

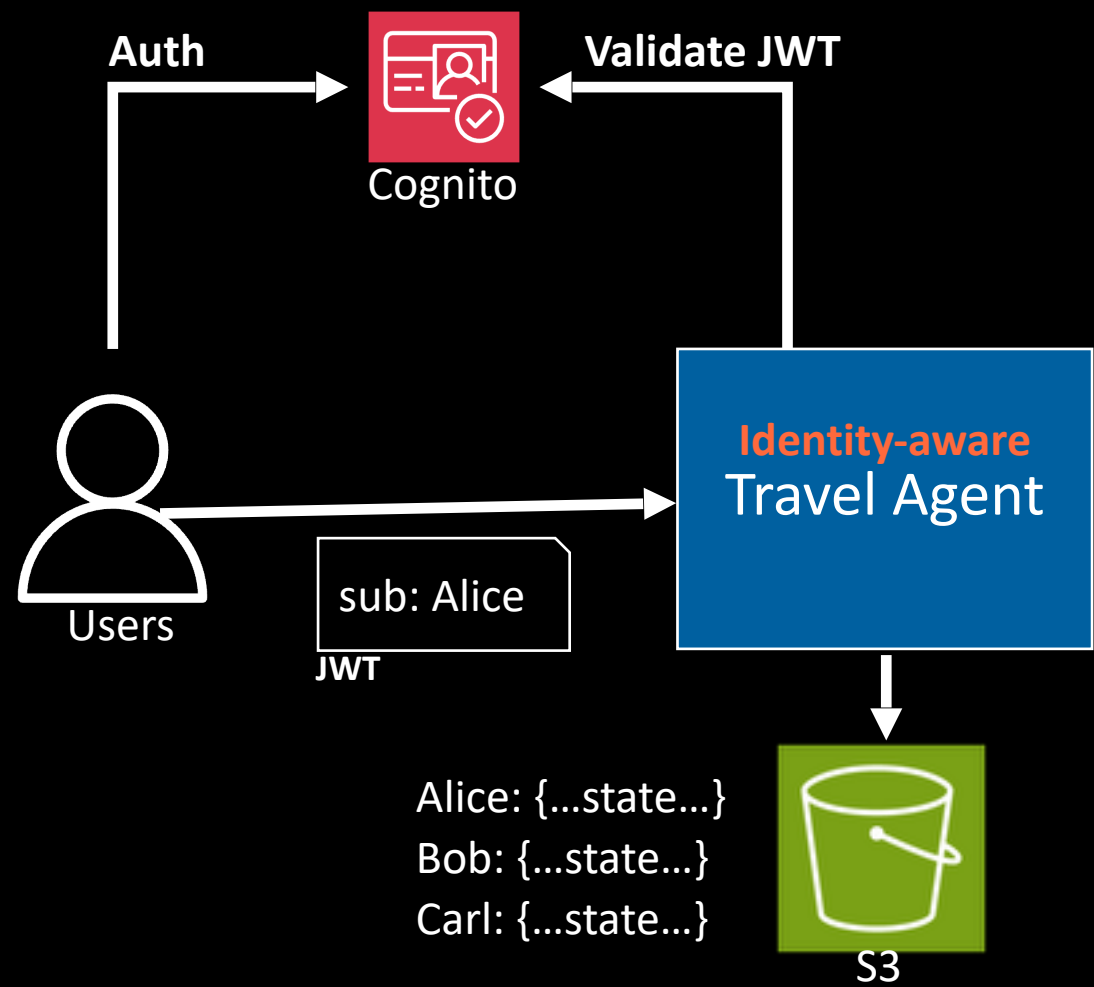


Session  
storage

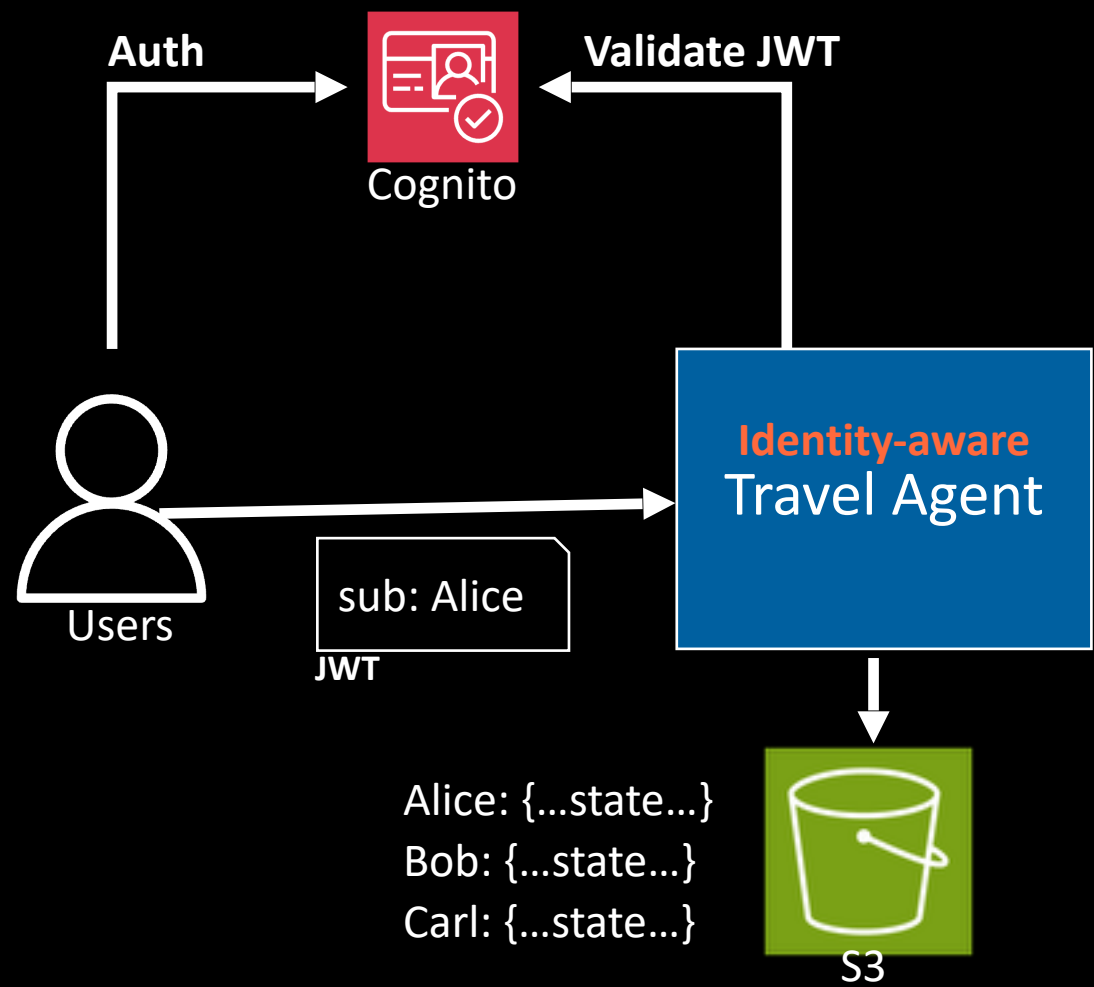
# What's missing?



# Authentication and authorization



# Authentication and authorization



```
session_manager = S3SessionManager(  
    session_id=f"session_{user.id}",  
    bucket=SESSION_STORE_BUCKET_NAME,  
    prefix="agent_sessions"  
)
```

Hi Alice, I'm your friendly corporate travel agent! I'm here to make booking your next business trip easier. Tell me how I can help.

# Deploying to AWS

---

“Book me a business trip to NYC”

# Deploying to AWS



AWS Lambda



Amazon Elastic  
Container Service (ECS)



Amazon Elastic  
Kubernetes Service (EKS)

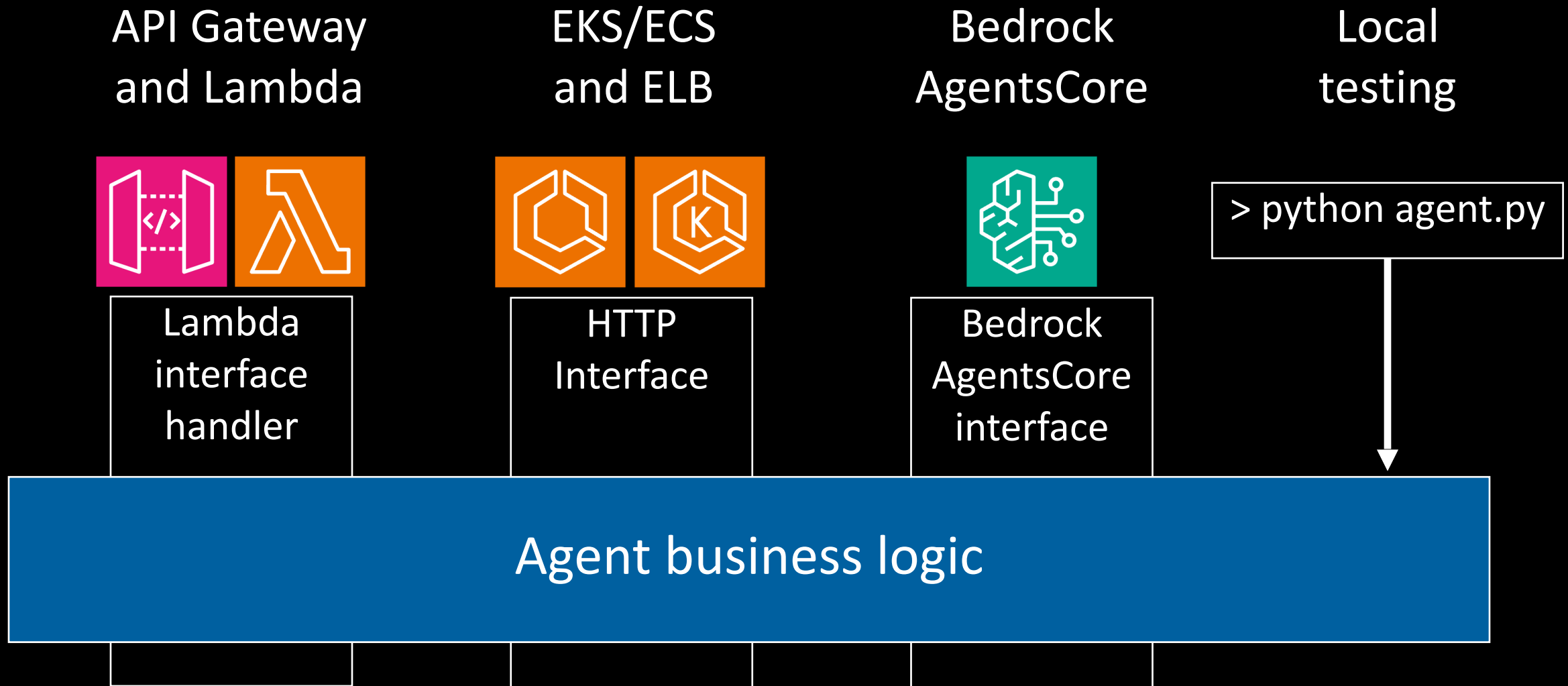


Amazon Bedrock  
AgentCore (preview)

**(or ANY other compute type, your Agent is just a Python app)**



# Building portable agents



# Building portable agents

lambda\_handler.py:

```
def handler(event: dict, ctx):  
    user_id = extract_user_id(event)  
    user_prompt = json.loads(event["body"])["prompt"]  
    llm_response = call_agent(user_id, user_prompt)  
  
    return {  
        "statusCode": 200,  
        "body": json.dumps({  
            "text": llm_response  
        })  
    }
```

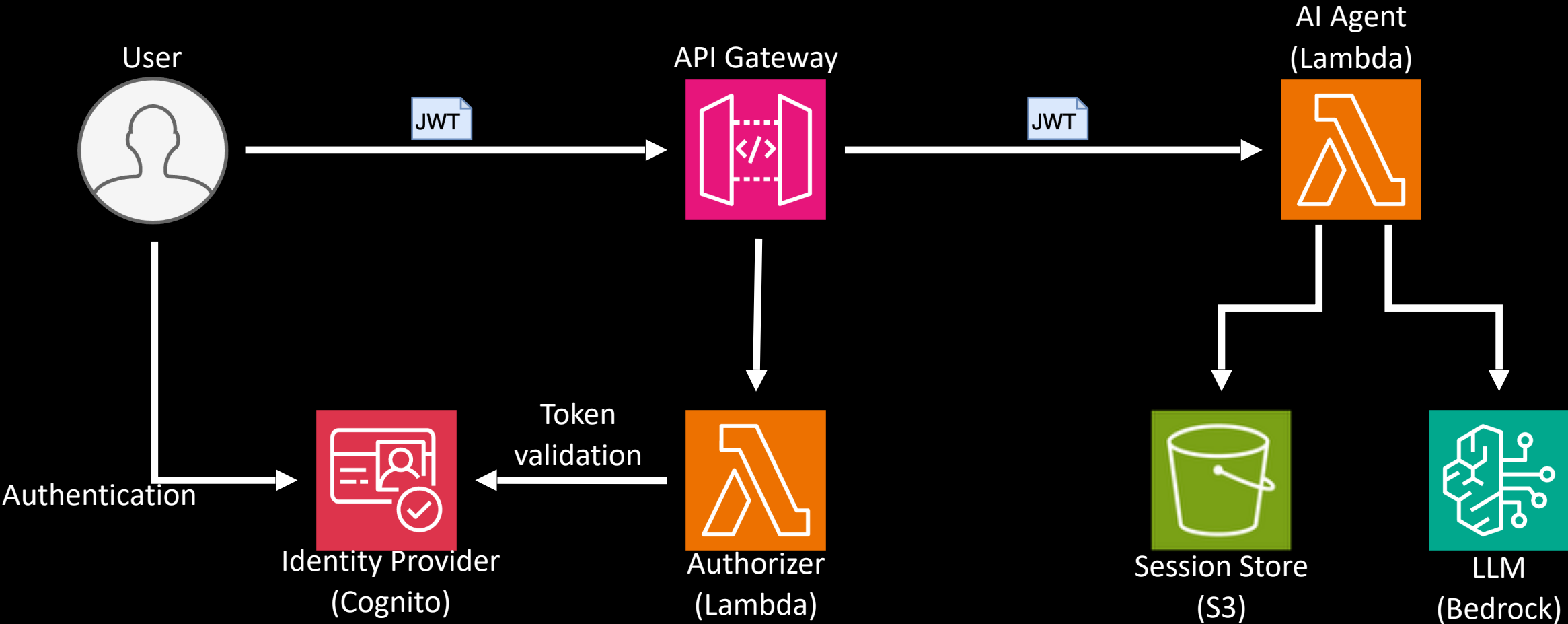
http\_handler.py:

```
@app.post("/chat")  
async def chat(request: Request,  
               prompt_request: PromptRequest):  
    user_id = extract_user_id(request)  
    user_prompt = prompt_request.prompt  
    llm_response = call_agent(user_id, user_prompt)  
  
    return {"text": llm_response}
```

my\_awesome\_agent.py:

```
def call_agent(user_id, user_prompt):  
    prev_messages = retrieve_state(user_id)  
  
    agent = Agent(  
        system_prompt = "...",  
        tools = [...],  
        messages = prev_messages,  
    )  
  
    llm_response = agent(user_prompt)  
  
    save_state(user_id, agent.messages)  
  
    return llm_response
```

# Deploying to AWS



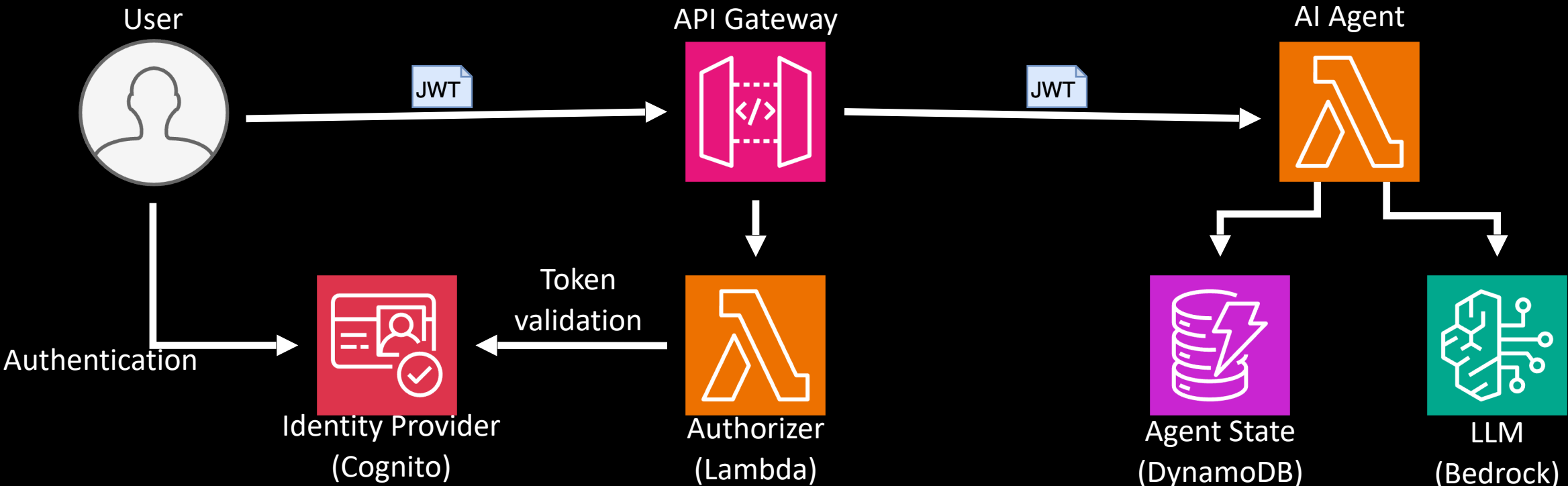
# Deploying to AWS

But what about observability?  
Security? Governance? CI/CD?

---

**Everything you know about building  
applications and APIs on AWS is still  
applicable!**

# Deploying to AWS



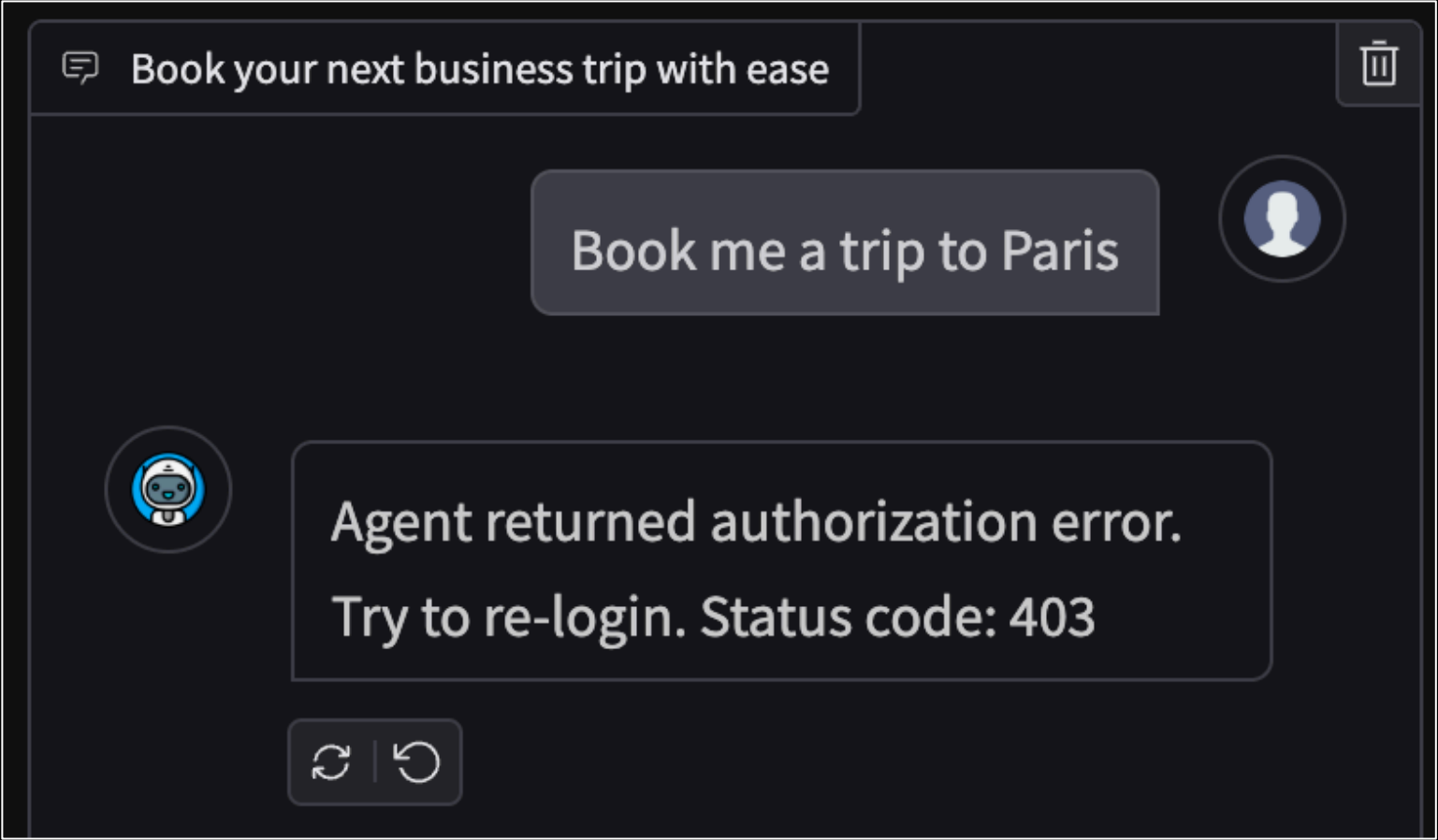
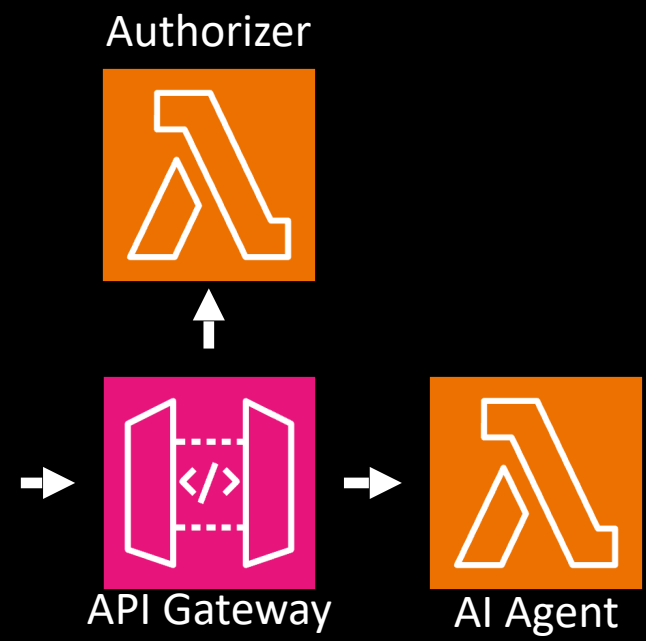
# Observability



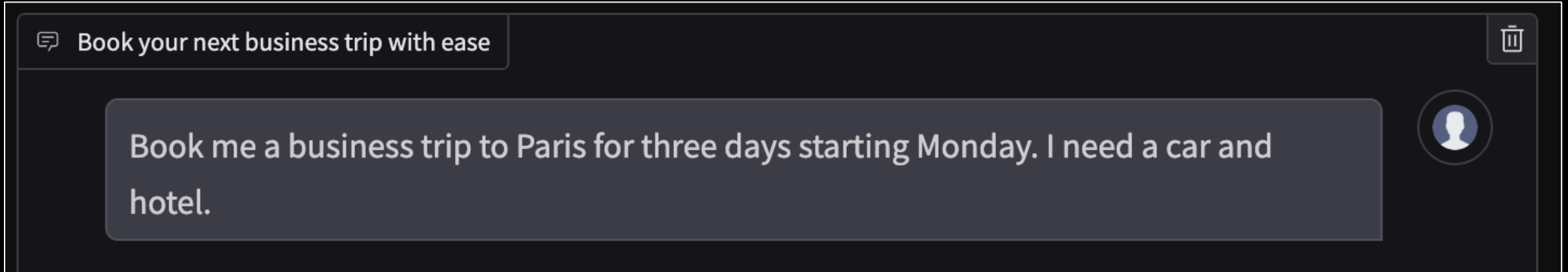
```
2025-06-20 18:05:10,757 INFO app.py:29 :: jwt parsed. user.id=74c8a428-7001-70db-d017-5725bd9efc91 user.name=Alice
2025-06-20 18:05:10,757 INFO app.py:43 :: composite_prompt=User name: Alice
User IP: 70.113.53.201
User prompt: Book me a trip to Paris
2025-06-20 18:05:10,757 INFO agent.py:10 :: user.id=74c8a428-7001-70db-d017-5725bd9efc91, user.name=Alice
2025-06-20 18:05:10,757 INFO agent_state_manager.py:22 :: restoring agent state for user.id=74c8a428-7001-70db-d017-5725bd9efc91
```



# Authorization



# Let's test it!





# Let's test it!

Book your next business trip with ease



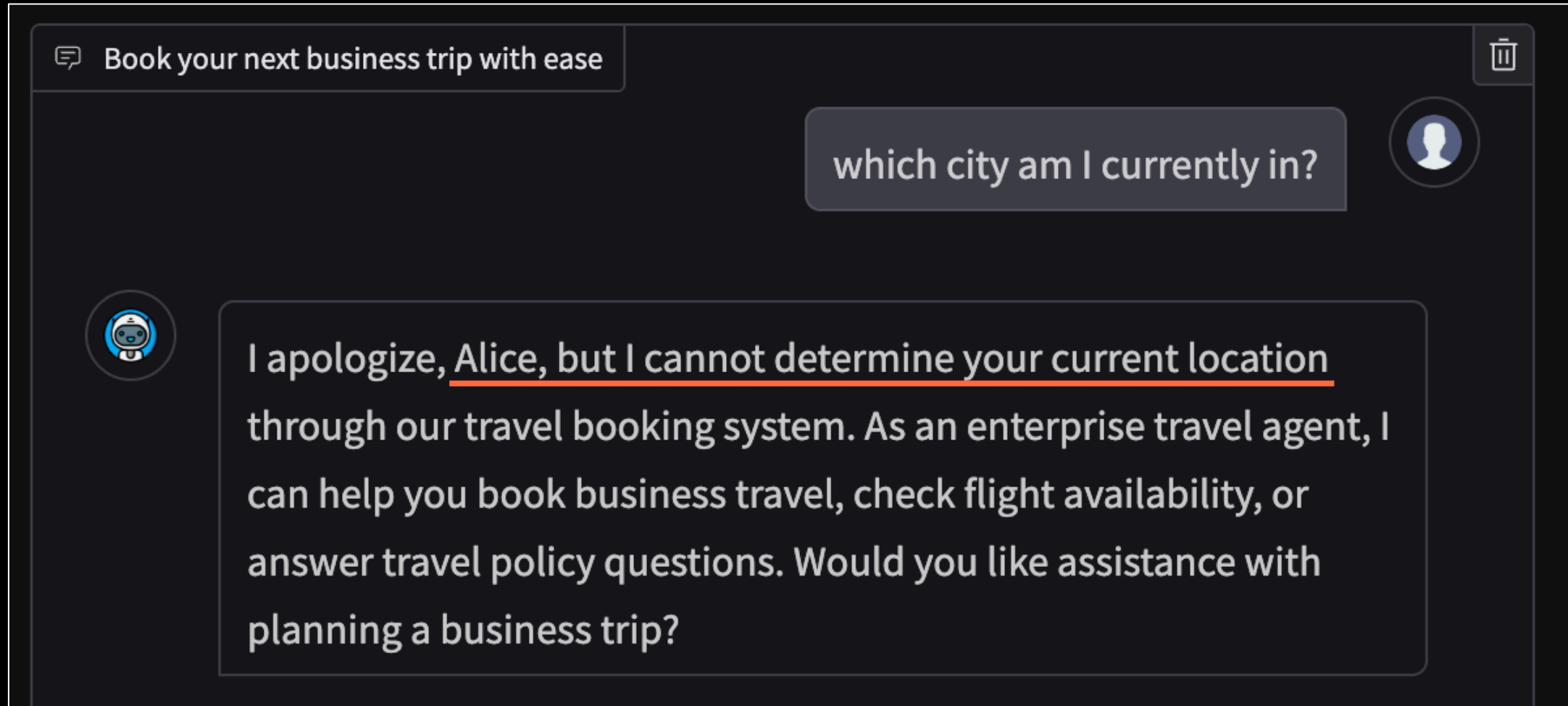
Book me a business trip to Paris for three days starting Monday. I need a car and hotel.



Thank you, Alice. I'll proceed with searching for travel arrangements based on the details you've provided.

However, I've discovered that I cannot actually complete the full booking at this moment. While I can discuss travel plans, my current tools do not allow me to directly book international flights, hotels, or car rentals.

# Let's test it!



# Creating your first tool

```
def get_user_location(ip: str) -> str:
    resp = request.urlopen(f"http://ip-api.com/json/{ip}").read()

    resp = json.loads(resp.decode('utf-8'))

    addr = f"{resp['city']} {resp['region']}, {resp['country']}"

    return addr
```

# Creating your first tool

```
@tool(name="get_user_location",
      description="Retrieves user's address based on the IP address.")
def get_user_location(ip: str) -> str:
    resp = request.urlopen(f"http://ip-api.com/json/{ip}").read()

    resp = json.loads(resp.decode('utf-8'))

    addr = f"{resp['city']} {resp['region']}, {resp['country']}"

    return addr
```

# Creating your first tool

```
@tool(name="get_user_location",
      description="Retrieves user's address based on the IP address.")
def get_user_location(ip: str) -> str:
    resp = request.urlopen(f"http://ip-api.com/json/{ip}").read()

    resp = json.loads(resp.decode('utf-8'))

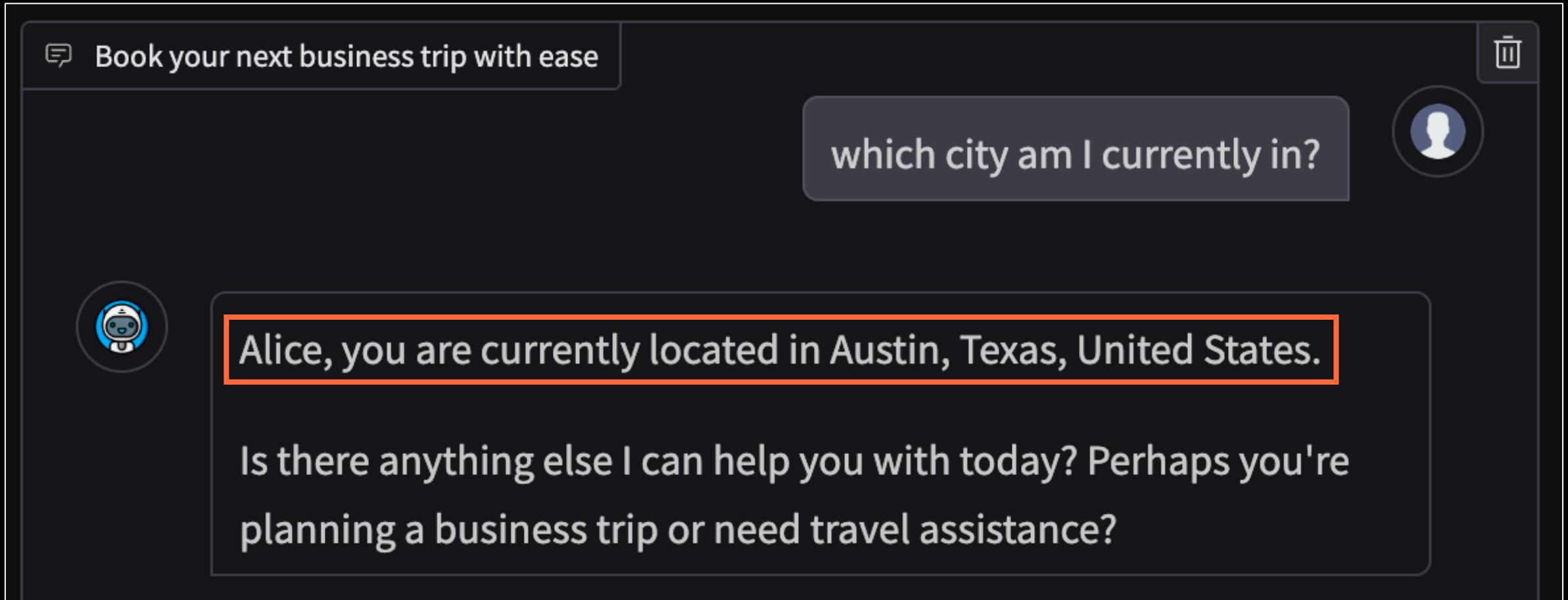
    addr = f"{resp['city']} {resp['region']}, {resp['country']}"

    return addr
```


















```
agent = Agent(
    system_prompt = "...",
    messages = prev_messages,
    tools = [get_user_location]
)

composite_prompt = [..., user_ip]
llm_response = agent(composite_prompt)
```

# Creating your first tool



# Strands Agents tools support - built-in tools

-  **File Operations** - Read, write, and edit files with syntax highlighting and intelligent modifications
-  **Shell Integration** - Execute and interact with shell commands securely
-  **Memory** - Store user and agent memories across agent runs to provide personalized experiences with both Mem0 and Amazon Bedrock Knowledge Bases
-  **HTTP Client** - Make API requests with comprehensive authentication support
-  **Slack Client** - Real-time Slack events, message processing, and Slack API access
-  **Python Execution** - Run Python code snippets with state persistence, user confirmation for code execution, and safety features
-  **Mathematical Tools** - Perform advanced calculations with symbolic math capabilities
-  **AWS Integration** - Seamless access to AWS services
-  **Image Processing** - Generate and process images for AI applications
-  **Video Processing** - Use models and agents to generate dynamic videos
-  **Audio Output** - Enable models to generate audio and speak
-  **Environment Management** - Handle environment variables safely
-  **Journaling** - Create and manage structured logs and journals
-  **Task Scheduling** - Schedule and manage cron jobs
-  **Advanced Reasoning** - Tools for complex thinking and reasoning capabilities
-  **Swarm Intelligence** - Coordinate multiple AI agents for parallel problem solving with shared memory
-  **Multiple tools in Parallel** - Call multiple other tools at the same time in parallel with Batch Tool

# Strands Agents tools support – Custom tools

## Internal tools

Implemented as part of the Agentic application package

```
@tool
def sum(a: int, b: int) -> int:
    """Adds two numbers.

    Args:
        a: first number
        b: second number
    """
    return a+b

agent = Agent(tools=[sum])
```

## External tools

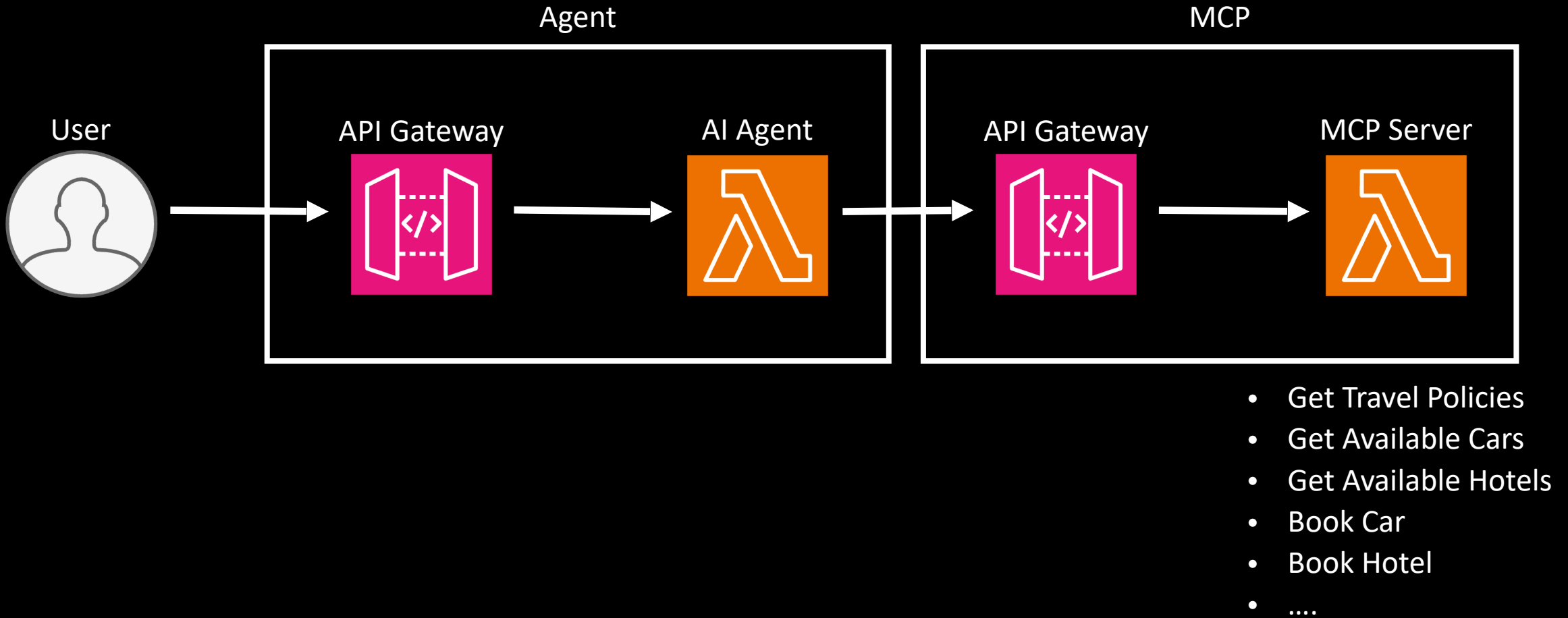
Consumed via Model Context Protocol

```
mcp_client = MCPClient(
    lambda: streamablehttp_client(
        "https://some-mcp-server.com/mcp"
    )
)

with mcp_client:
    tools = mcp_client.list_tools_sync()
    agent = Agent(tools=tools)
```



# Adding a remote MCP Server

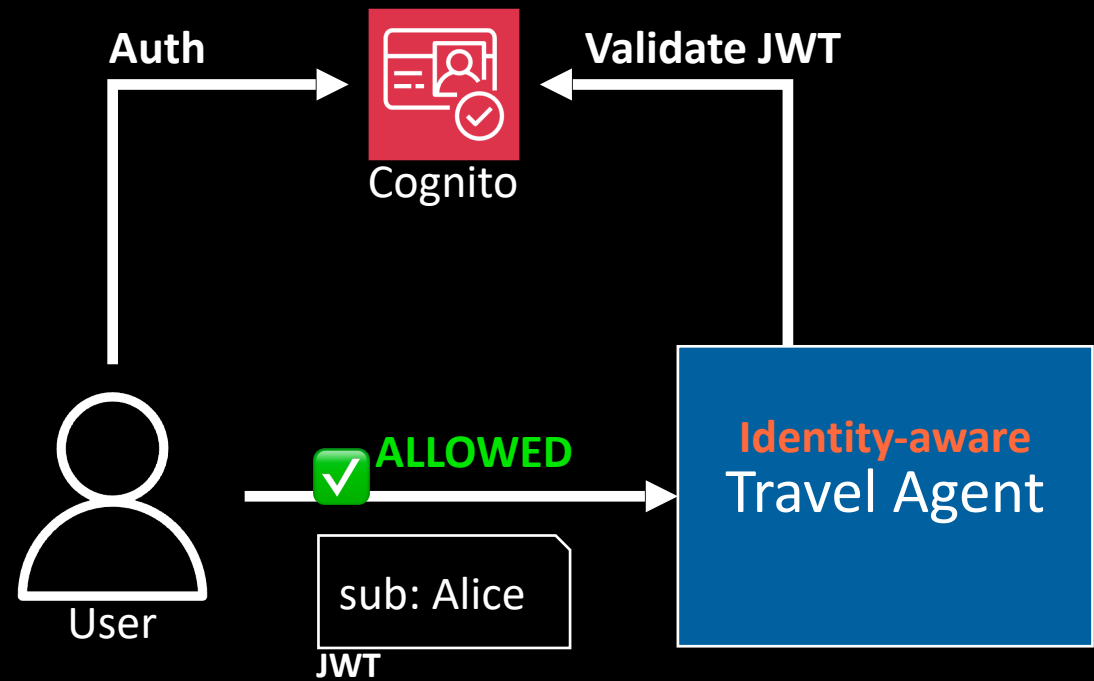


# MCP Server – quick glance

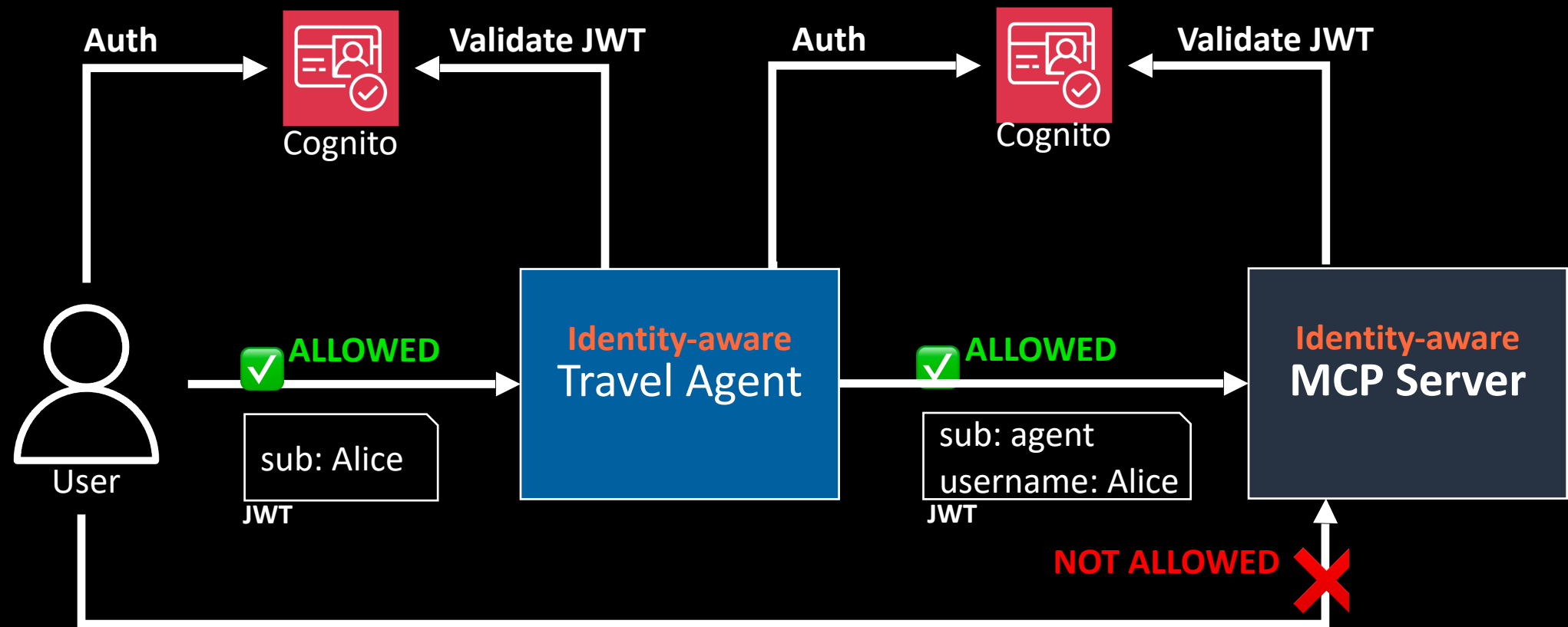
Building AI Agents on AWS



# Authentication and authorization



# Authentication and authorization



# Building Serverless MCP Servers on AWS

```
const TOOL = [  
  `get-travel-policies`,  
  `This tool returns corporate travel policies.  
  Travel agents must ALWAYS follow this guidance and restrictions.`,  
  async (ctx) => {  
    const userName = ctx.authInfo.user_name;  
    return {  
      content: [  
        {  
          type: "text",  
          text: `Here are the travel policies for ${userName}:  
1. Employees can only book travel within the United States of America.  
2. Employees are not allowed to book luxury cars.  
3. Employees cannot travel for more than 5 days.  
4. Employees can book business travel only, no leisure or personal travel is supported.  
`  
        }  
      ]  
    }  
  }  
];
```

Tool name

# Building Serverless MCP Servers on AWS

```
const TOOL = [  
  `get-travel-policies`,  
  `This tool returns corporate travel policies.  
  Travel agents must ALWAYS follow this guidance and restrictions.`,  
  async (ctx) => {  
    const userName = ctx.authInfo.user_name;  
    return {  
      content: [  
        {  
          type: "text",  
          text: `Here are the travel policies for ${userName}:  
1. Employees can only book travel within the United States of America.  
2. Employees are not allowed to book luxury cars.  
3. Employees cannot travel for more than 5 days.  
4. Employees can book business travel only, no leisure or personal travel is supported.  
`  
        }  
      ]  
    }  
  }  
];
```

Tool description

# Building Serverless MCP Servers on AWS

```
const TOOL = [  
  `get-travel-policies`,  
  `This tool returns corporate travel policies.  
  Travel agents must ALWAYS follow this guidance and restrictions.`,  
  async (ctx) => {  
    const userName = ctx.authInfo.user_name;  
    return {  
      content: [  
        {  
          type: "text",  
          text: `Here are the travel policies for ${userName}:  
1. Employees can only book travel within the United States of America.  
2. Employees are not allowed to book luxury cars.  
3. Employees cannot travel for more than 5 days.  
4. Employees can book business travel only, no leisure or personal travel is supported.  
`  
        }  
      ]  
    }  
  }  
];
```

Tool logic (user-aware)

# Building Serverless MCP Servers on AWS

```
const TOOL = [  
  `book-hotel`,  
  `Use this tool to book hotels`,  
  {  
    city: z.string(),  
    date: z.string(),  
    nights: z.number()  
  },  
  async ({ city, date, nights }, ctx) => {  
    const userName = ctx.authInfo.user_name;  
    return {  
      content: [  
        {  
          type: "text",  
          text: `Booked hotel in ${city} for ${userName} for  
                ${nights} nights. Check-in date is ${date}.`  
        }  
      ]  
    }  
  }  
];
```

Tool name and description



# Building Serverless MCP Servers on AWS

```
const TOOL = [  
  `book-hotel`,  
  `Use this tool to book hotels`,  
  {  
    city: z.string(),  
    date: z.string(),  
    nights: z.number()  
  },  
  async ({ city, date, nights }, ctx) => {  
    const userName = ctx.authInfo.user_name;  
    return {  
      content: [  
        {  
          type: "text",  
          text: `Booked hotel in ${city} for ${userName} for  
                ${nights} nights. Check-in date is ${date}.`  
        }  
      ]  
    }  
  }  
];
```

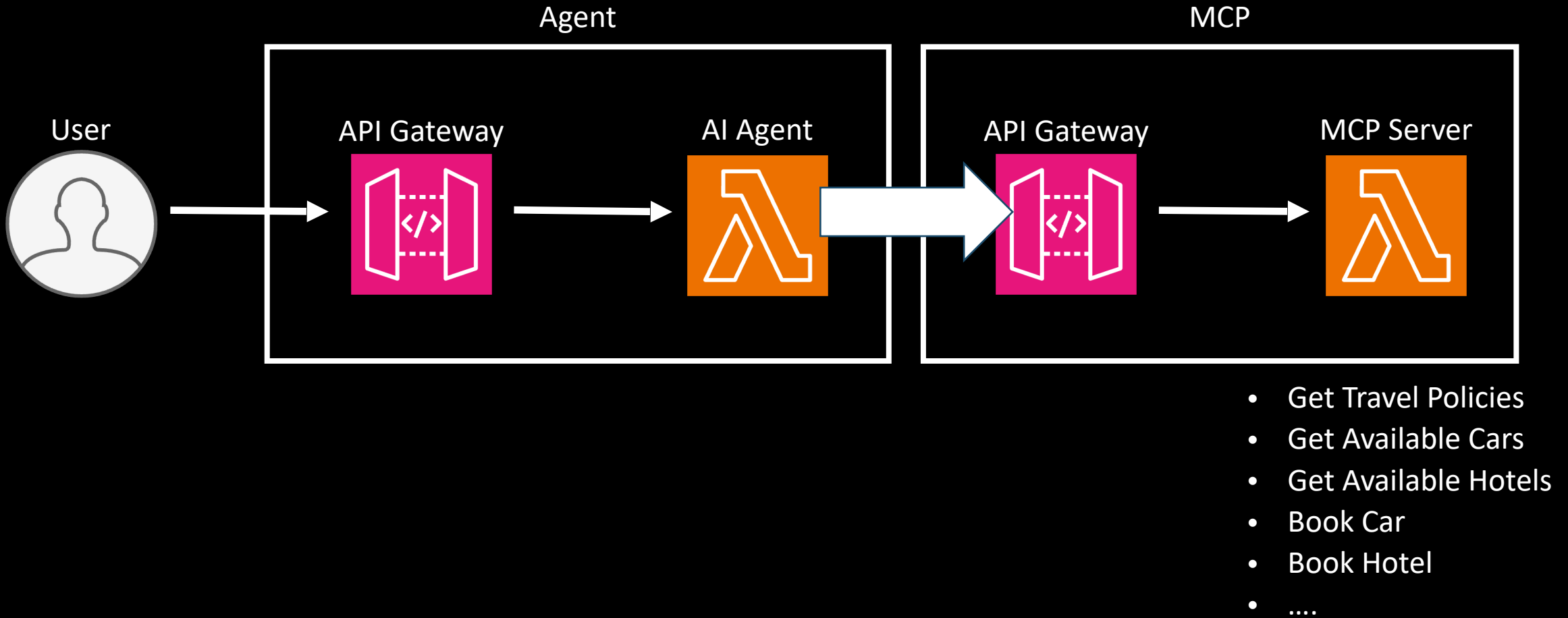
Tool arguments

# Building Serverless MCP Servers on AWS

```
const TOOL = [  
  `book-hotel`,  
  `Use this tool to book hotels`,  
  {  
    city: z.string(),  
    date: z.string(),  
    nights: z.number()  
  },  
  async ({ city, date, nights }, ctx) => {  
    const userName = ctx.authInfo.user_name;  
    return {  
      content: [  
        {  
          type: "text",  
          text: `Booked hotel in ${city} for ${userName} for  
                ${nights} nights. Check-in date is ${date}.`  
        }  
      ]  
    }  
  }  
];
```

Tool logic (user-aware)

# Connecting AI Agent with MCP Server



# Connecting AI Agent with MCP Server

```
def get_mcp_tools_for_user(user: User):  
  
    mcp_token = get_token_for_user(user)  
    mcp_client = MCPCClient(lambda: streamablehttp_client(  
        url=mcp_endpoint,  
        headers={"Authorization": f"Bearer {mcp_token}"},  
    ))  
  
    mcp_client.start()  
    return mcp_client.list_tools_sync()
```

# Connecting AI Agent with MCP Server

```
def get_mcp_tools_for_user(user: User):
```

```
mcp_tools = mcp_client_manager.get_mcp_tools_for_user(user)
```

```
return Agent(  
    model = __model,  
    system_prompt=__system_prompt,  
    messages = messages,  
    tools=[internal_tools] + mcp_tools,  
)
```

strands-on-lambda.auth.us-east-1.amazoncognito.com

## Sign in with your username and password

Username

Password

[Forgot your password?](#)

**Sign in**

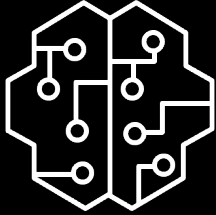
# In conclusion

---

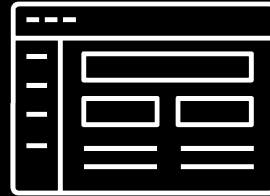
Building AI Agents on AWS



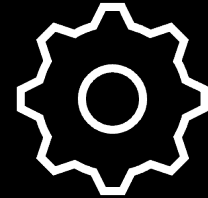
# Agentic AI Systems



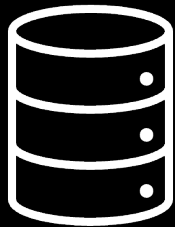
**Plan and sequence actions  
to achieve a goal**



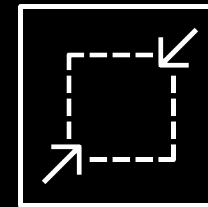
**Perceive and process context  
aware information**



**Use tools to perform  
tasks efficiently**



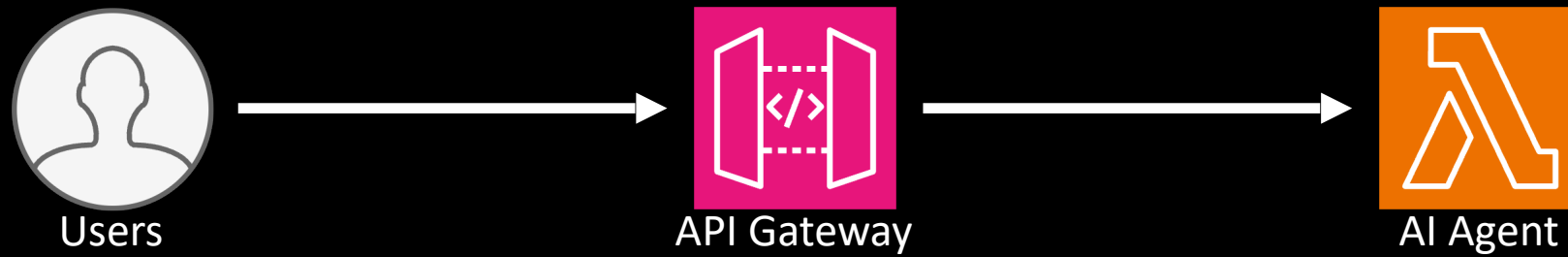
**Remember past interactions  
and behaviors**



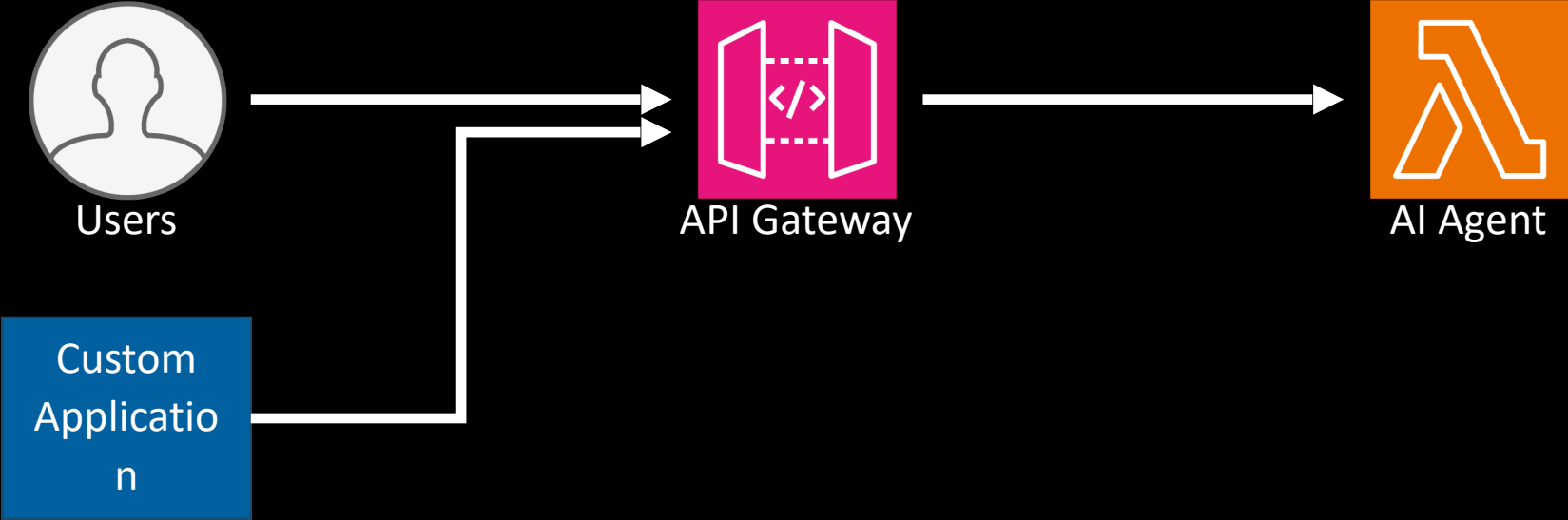
**Makes decisions and execute  
actions based on goals**



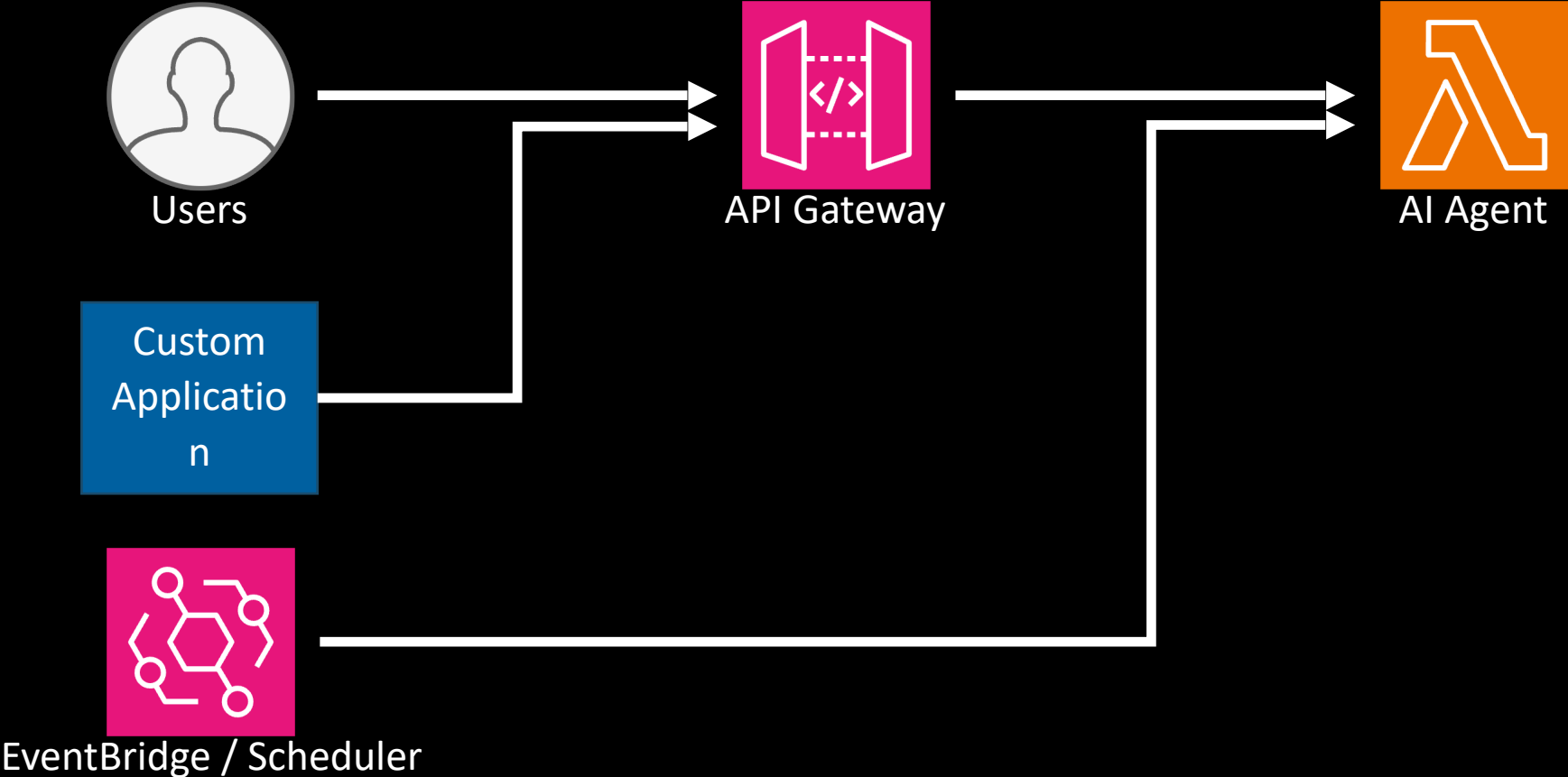
# Serverless AI Agents on AWS



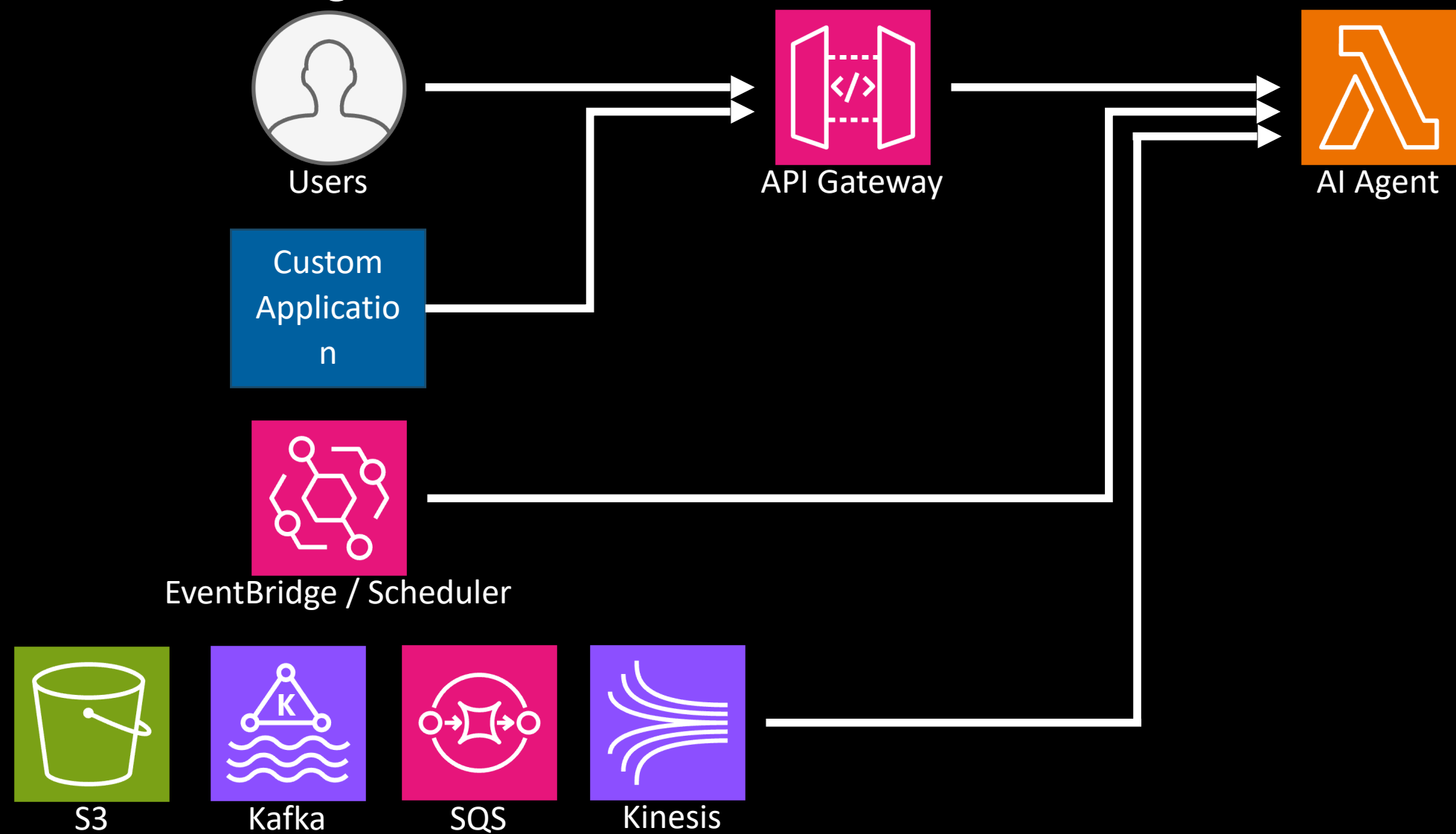
# Serverless AI Agents on AWS



# Serverless AI Agents on AWS



# Serverless AI Agents on AWS



# Final thoughts

# 1

**Your existing knowledge is your accelerator.**

# 2

**Great context leads to great answers.**

# 3

**Do not reinvent the wheel. Keep it simple with battle-tested services/frameworks.**



## Useful links & action items

- Strands Agents SDK documentation ([link](#))
- Running Agents on [Lambda](#), [ECS](#), [EKS](#), [EC2](#)
- Amazon Bedrock [AgentCore](#) (preview)
- Operating Agents in Production ([link](#))
- E2E Travel Agent sample implementation ([link](#))
- Building and scaling Agentic AI Workflows workshop ([link](#))
- Generative AI and Serverless paved path ([link](#))
- Solution Patterns with Amazon Bedrock ([link](#))
- Patterns to Automatically Scale your Generative AI Solutions ([link](#))



# Thank you!



**Anton Aleksandrov**  
Principal Solutions Architect  
AWS Serverless