

# AWS Advance GenAI Immersion Day

## CHUONG HO

Sr. Solutions Architect  
Amazon Web Services  
[chuongho@amazon.com](mailto:chuongho@amazon.com)

## BRITTANY ROBINSON

Solutions Architect  
Amazon Web Services  
[btrobin@amazon.com](mailto:btrobin@amazon.com)



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

## Agenda

- 0900 - 0915 Introduction to AWS team
- 0915 - 0945: AI Trends Overview
- 0945 - 1000: Fine Tuning vs. Augmented Generation
- 1000 - 1010: Break
- 1010 - 1100: Workflows, Agents, and Tools (MCP)
- 1100 - 1110: Break
- 1110 - 1130: Moving to Production and AI Guardrails
- 1130 - 1200: Demos (time permitting)
- 1200 - 1300: Break for Lunch
- 1300 - 1600: Workshop - Building MCP Powered Agents on Amazon Bedrock



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Generative AI Trends



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

# Model Architecture Developments

**Mixture of Experts**

**State Space Models**

**Diffusion Transformers**

**Ring Attention**

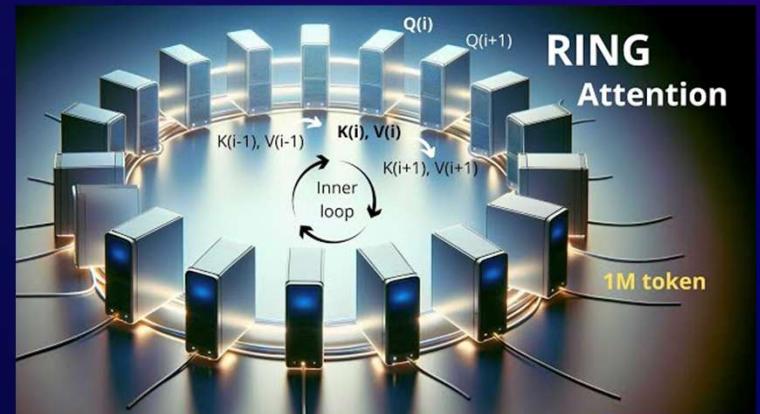


© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Model Innovations

## Ring Attention

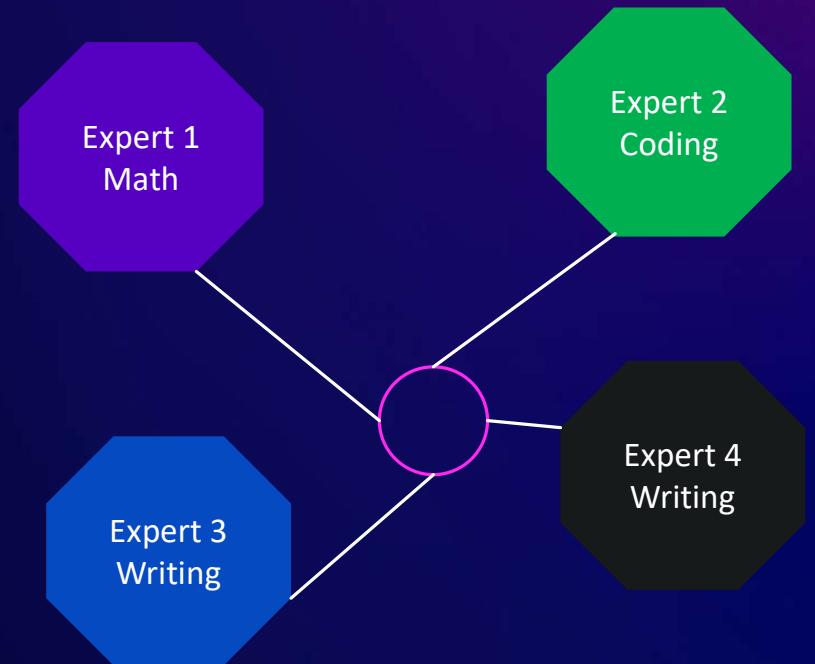
- **Definition:**
  - Transformers handle super long sequences by splitting the attention computation across multiple devices in a ring topology
  - Enables training and inference on extremely long documents, entire codebases, or long conversations that would be impossible with standard attention mechanisms.
- **Benefits:**
  - Memory Efficiency
  - Linear Scaling
  - Full Attention



# Model Innovations

## Mixture of Experts

- **Definition:** Model only activates experts, based on probability the expert matches the input/question
- Input can be further broken down and sent to experts
- **Benefits:** Scalability, Specialization, Efficiency, Performance



# Multimodal Advances

**Audio Generation**

**Vision Generation Models**

**Video Understanding and Creation**

**Ultimate Goal: Any-to-Any Models**



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

# Model Advances

## Audio Generation

- **Current State:**
  - Real-time voice cloning from minimal samples
    - From just seconds of audio!
  - Multi-speaker models that can generate diverse voices from text prompts
  - Real time voice conversation (Notebook LM creates podcasts out of documents)
- **Technical Constraints:**
  - Requires 32GB+ of GPU memory to generate
  - Real time generation slow on personal hardware
  - Artifacting still possible
  - Overfitting: Models may reproduce audio they were trained on (copyright issues!)

# Model Advances

## Video Understanding and Creation

- **Definition:** Understanding Video
  - Object detection and tracking across frames
  - Face recognition and emotion detection
  - Temporal reasoning (cause and effect sequences)
  - Fine-grained activity recognition
- **Current State:**
  - Mainly short clips (Sora from OpenAI)
  - Great for common activities
  - Can answer basic questions – which part of the movie did a guy get hit in the face with a pie?
- **Technical Constraints:**
  - 100 x more expensive than text!
  - Requires massive memory
  - Processing is slow (not real time). So no auto-commentators yet!

# Model Advances Video Understanding



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

# Capabilities Development

**Code Assistance**

**Fine Tuning and Augmented Generation**

**Workflows**

**Agent and Multi-Agent**

**Model Context Protocol**



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

# Capabilities Development

## Code Assistance

- **Current State/Trends:**
  - Integrations into IDE for auto completion
  - Automated security scanning
  - Enhanced support for serverless
  - Using MCP servers boost integrations with cloud services
- Offerings
  - Amazon Kiro
  - Claude Code
  - Cursor
  - Github Copilot

# Capabilities Development Code Assistance - Demo



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

# Fine Tuning vs Augmented Generation

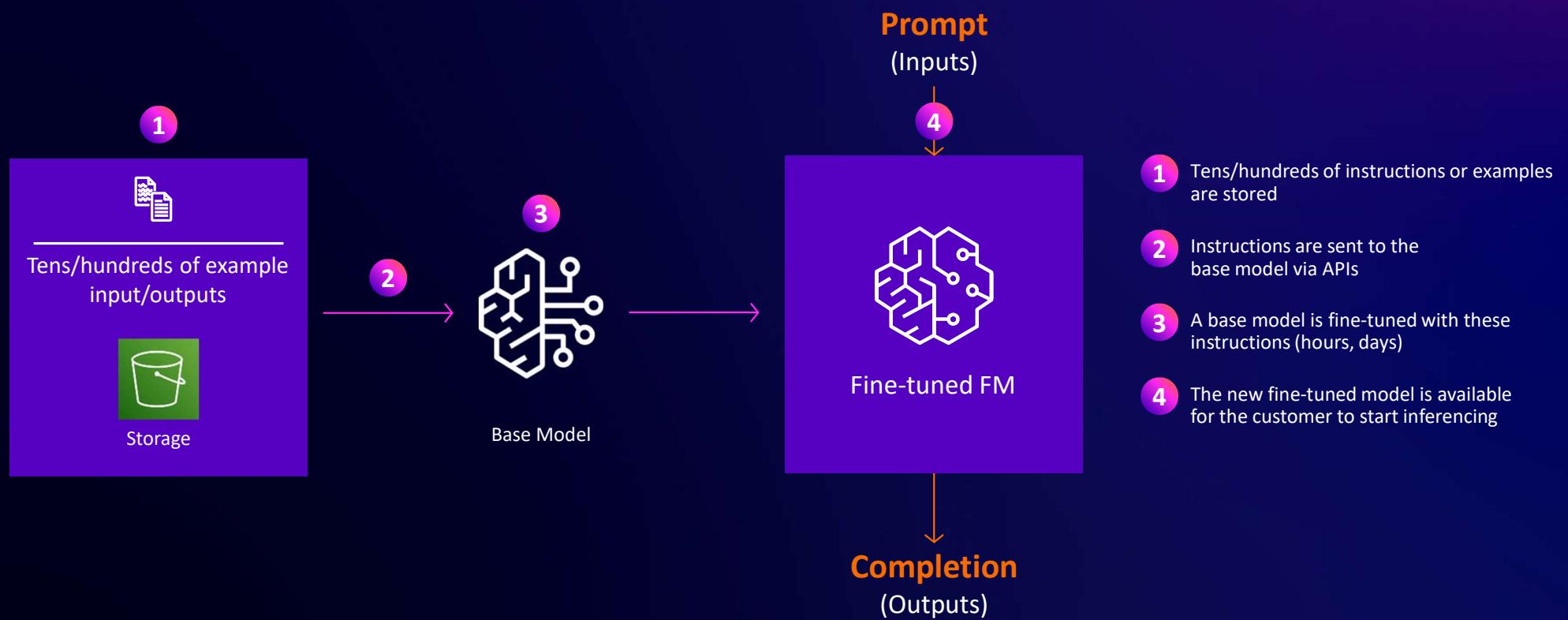


© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

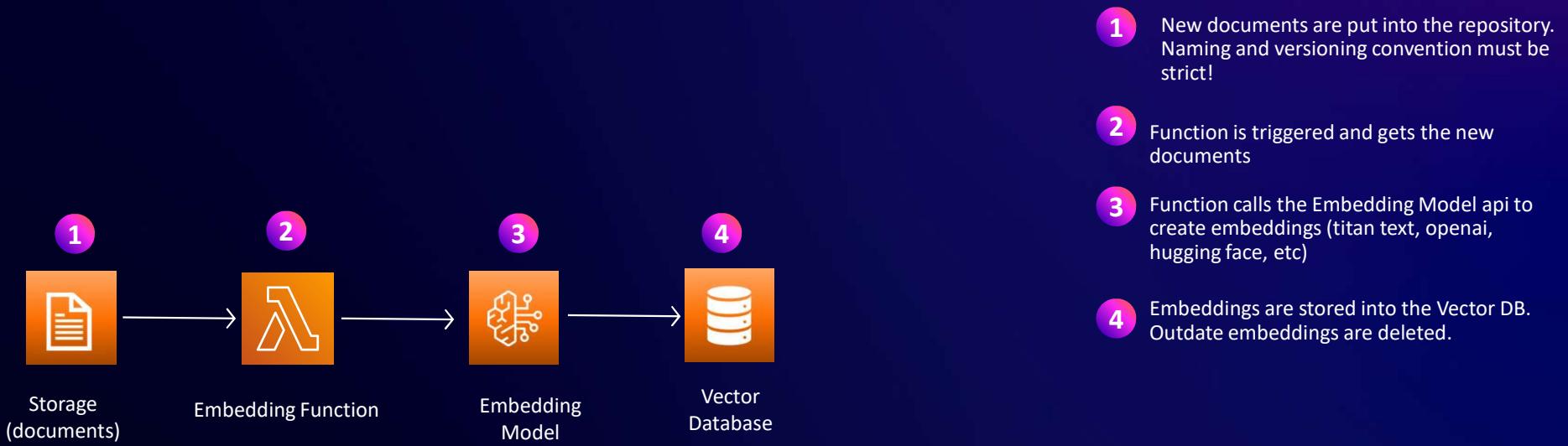
# Strategies for implementation and their trade-offs



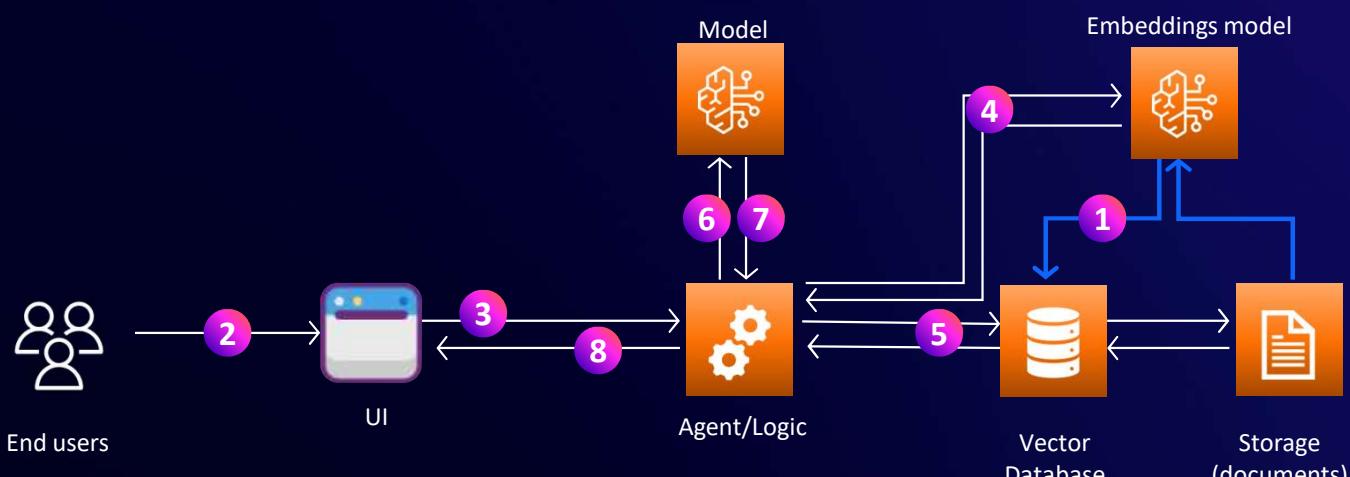
# Fine-tuning



# Retrieval augmented generation (RAG) Pipeline

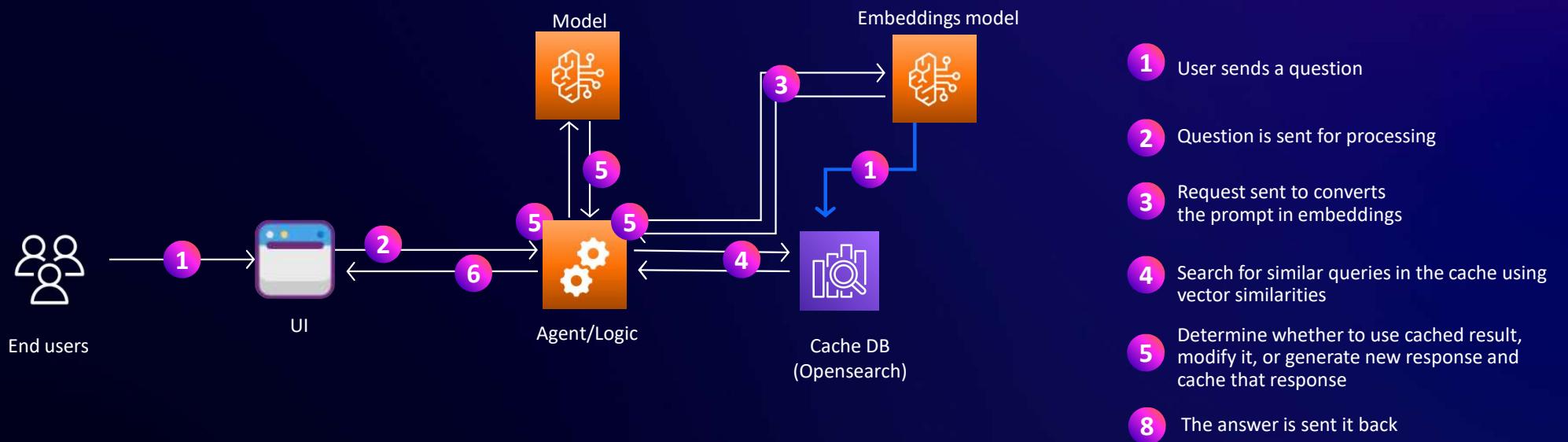


# Retrieval augmented generation (RAG) In Use



1. Documents are converted into vectors using the pipeline in our previous example.
2. User sends a question.
3. Question is sent for processing.
4. Request sent to converts the prompt in embeddings.
5. With the embedding we search the paragraphs that contain the answer to the question.
6. Those paragraphs together with the prompt are sent to the model.
7. The model generates the answer.
8. The answer is sent it back.

# Cache Augmented Generation (RAG) In Use



# Cache Augmented Generation (RAG) In Use

- **Caching Tools**
  - Vector DB with cache extensions
  - Elasticache – can be enabled when using vector search
  - Redis – faster in memory caching
  - Pinecone – Native caching features
  - Weaviate – Automatic caching
- **Caching Techniques**
  - Embedded Caches
  - Result Caching
  - Index Caching
  - Context Caches

# RAG vs CAG

## RAG

### Pros:

- Up to date info
- Lower risk of hallucination
- Verifyable resources
- Can support multiple knowledge domains by using different vector DBs

### Challenges:

- Dependent on good document pipeline
- How to handle conflicting information
- Increased latency
- Increased API/tokens usage
- More infrastructure to manage

## CAG

### Pros:

- Reduced inference for high volume but similar queries
- Faster response time for users
- Consistent answers across different agent
- Reduced API/token usage

### Challenges:

- Maintaining cache: invalidation, updating, storage
- Balancing cache size and hit rates
- Setting similiarity cutoffs
- How to handle for distributed application
- More infrastructure to manage



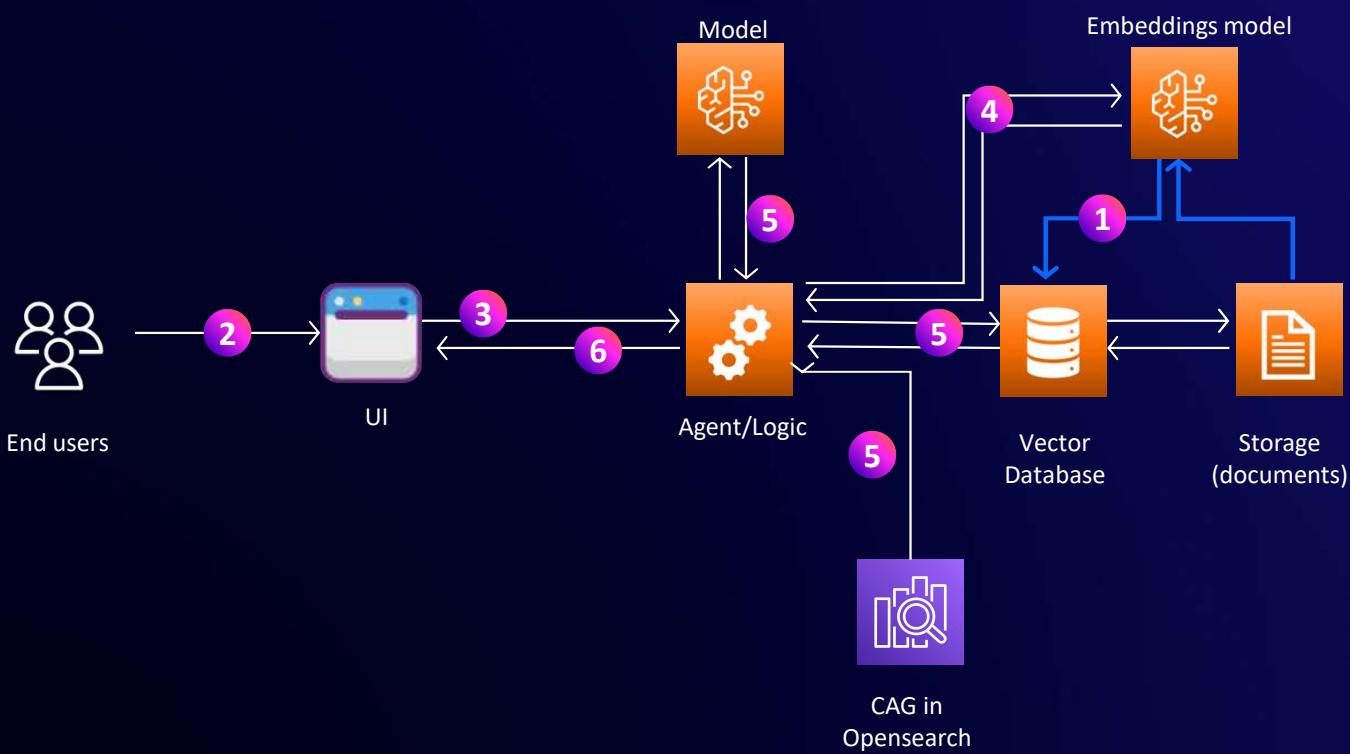
# RAG vs CAG

**But, we can also do this....**



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

# RAG and CAG



- 1 Documents are converted into vectors using the pipeline in our previous example
- 2 User sends a question
- 3 Question is sent for processing
- 4 Request sent to converts the prompt in embeddings
- 5 With the embedding we can then:
  - Search the cache for similar questions
  - If no cache, then go to Vector DB. Then use retrieved vectors and original prompt to send to model
  - Answer is cached
- 6 Answer is returned to user

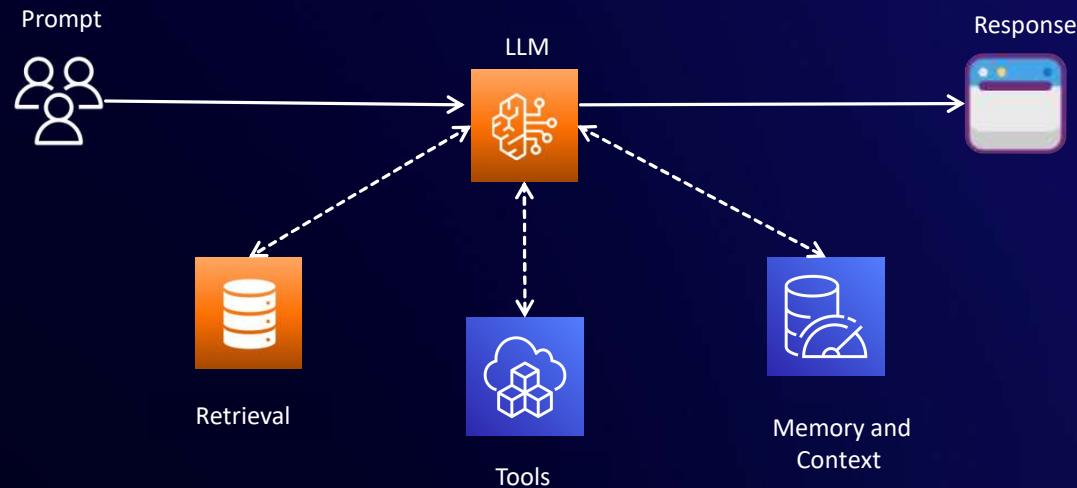
# Agentic AI: Workflow, Agents, and Tools



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

# Agentic AI

## What is agentic AI?



**It is a model, supplied with additional tools, that can choose to mix and match the tools to answer the question.**

Reference: <https://www.anthropic.com/engineering/building-effective-agents>



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

# Workflow vs Agents

**Workflows:** Workflows are systems where LLMs and tools are orchestrated through predefined code paths.

Workflows offer predictability and consistency for well-defined tasks

**Agents:** Agents, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.

Agents are a better option when flexibility and model-driven decision-making are needed at scale

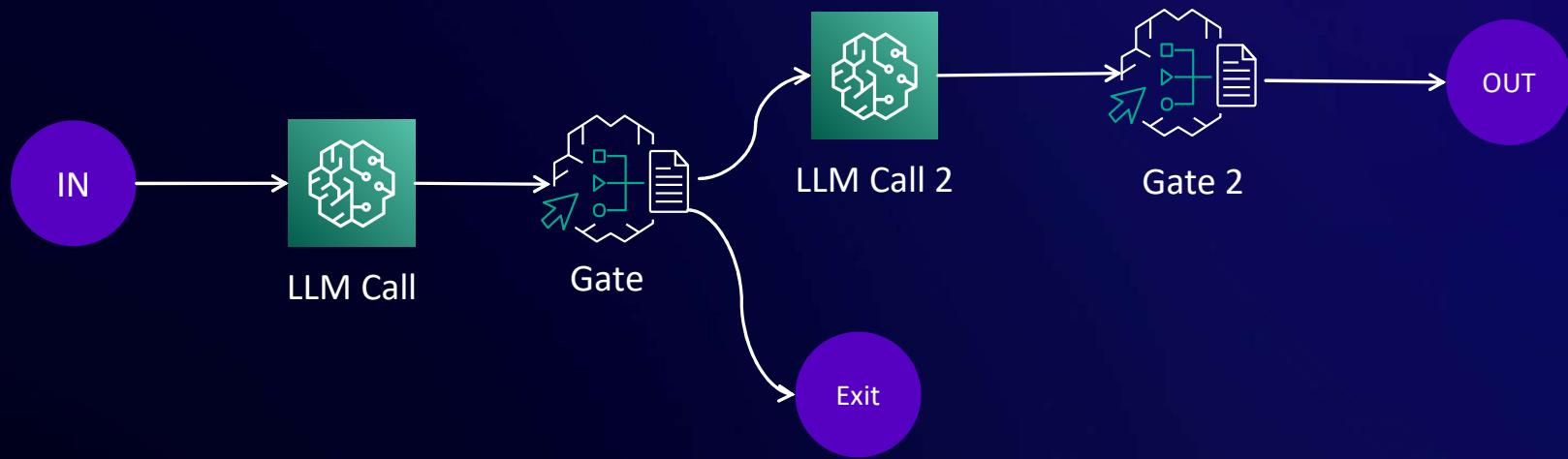


# Types of Workflows:

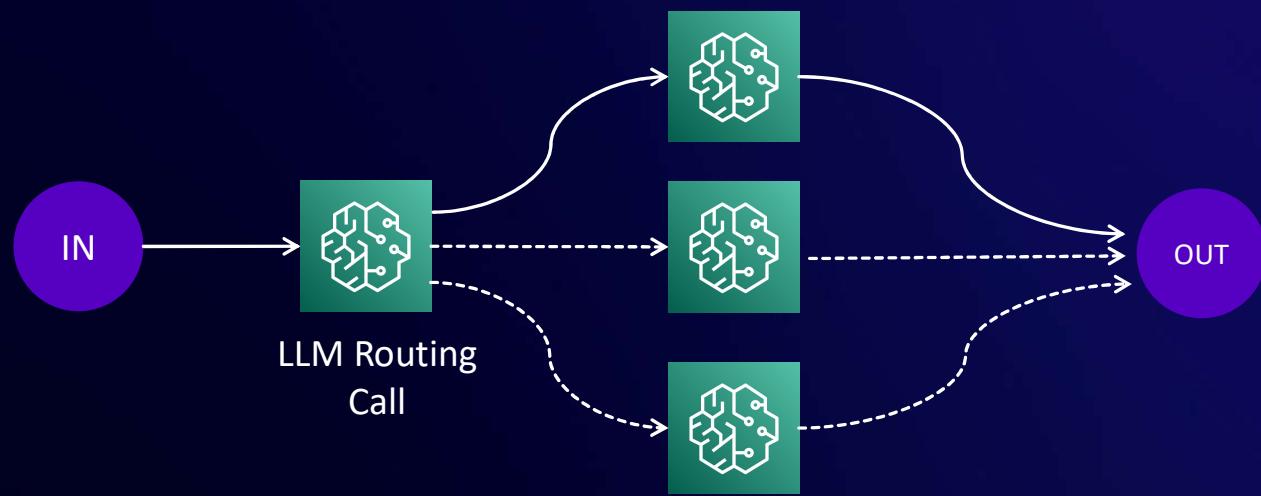
## Workflows:

- **Prompt Chaining**
- **Routing**
- **Parallelization/Aggregation**
- **Orchestrator**
- **Evaluator/Optimizer**

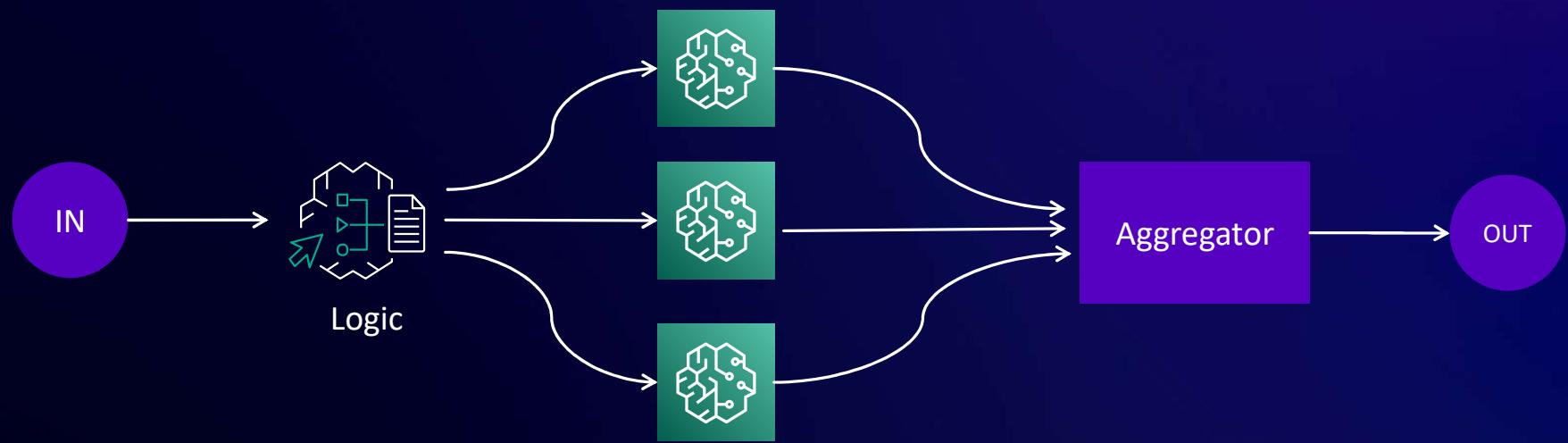
# Types of Workflows: Prompt Chaining



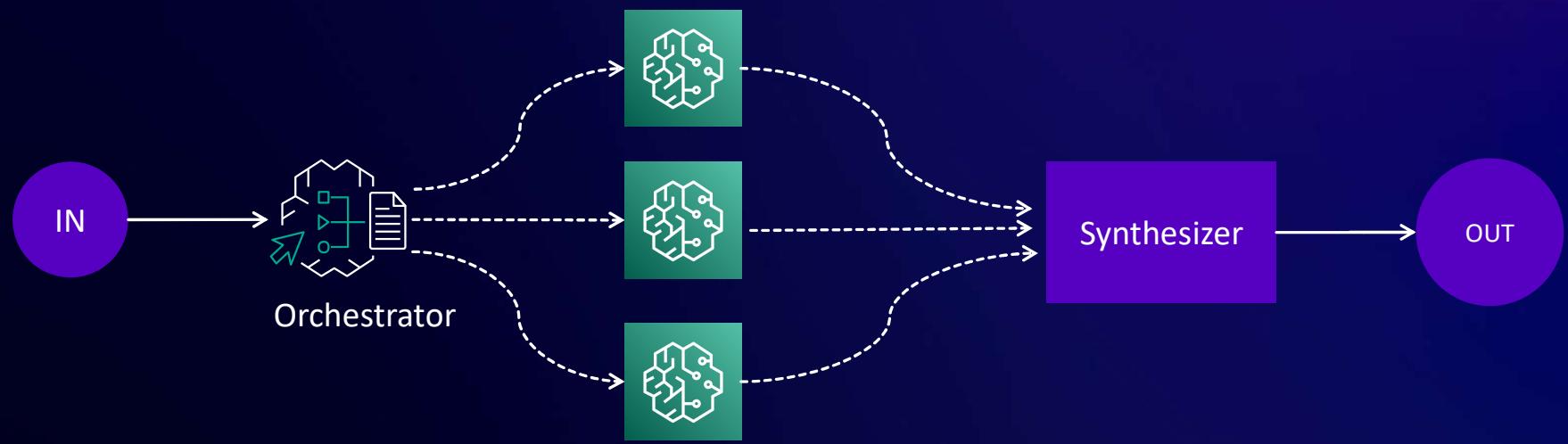
# Types of Workflows: Routing



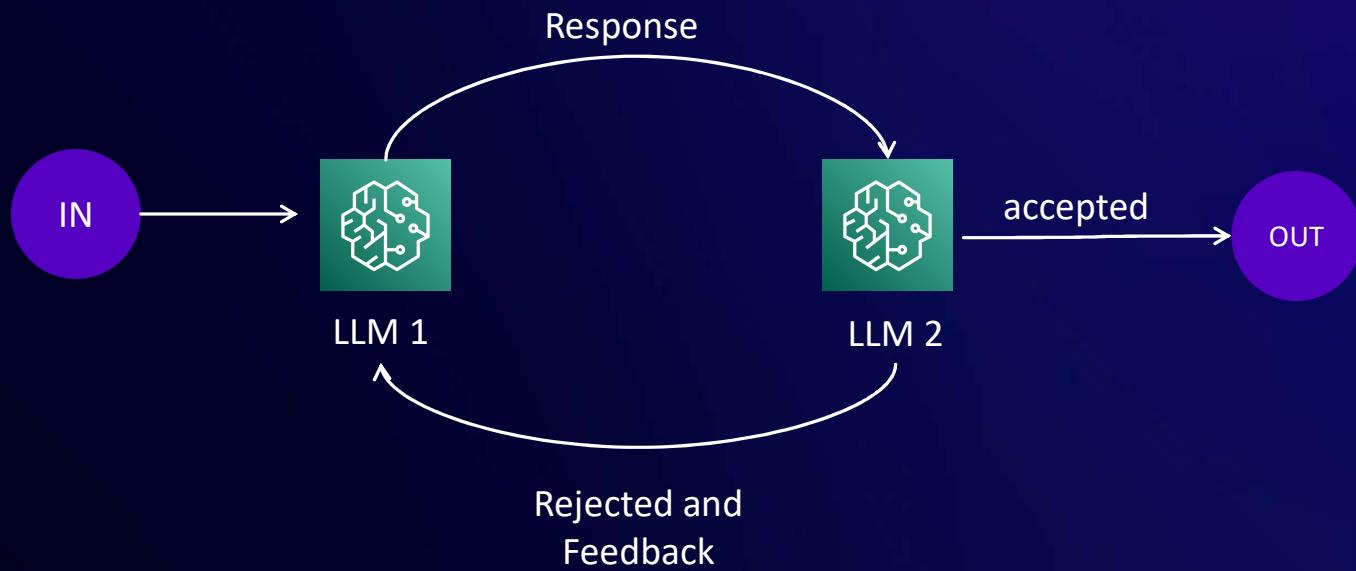
# Types of Workflows: Parallelization/Aggregation



# Types of Workflows: Orchestrator



# Types of Workflows: Evaluator/Optimizer



# Workflow Demo

[Go To DEMO](#)



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

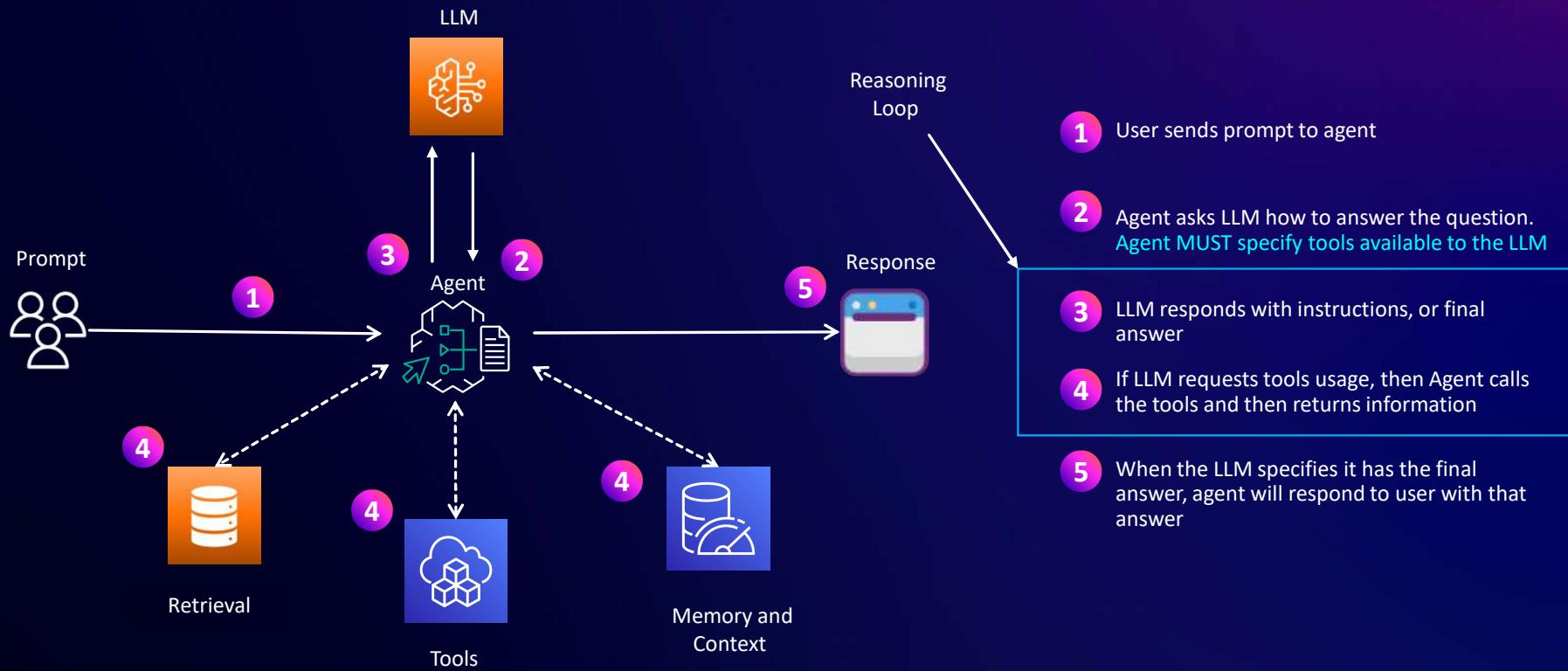
# Agents

**Agents are very different from workflows:**

- Open Ended
- Feedback/reasoning Loops
- No Fixed Path
- Options to use available tools



# Agent Workflows



# AI Agents - Code Example

IN THE BEGINNING...

Create tool for agent, this is a sql query

```
# Simple tool function using if-else logic
def get_pto_info(query_type):
    """
    Simple function to get PTO information based on query type
    Uses if-else statements to handle different queries
    """
    if query_type == "available_hours":
        return f"You have {PTO_HOURS_AVAILABLE} PTO hours available."
    elif query_type == "available_days":
        days = PTO_HOURS_AVAILABLE / 8
        return f"You have {days} PTO days available ({PTO_HOURS_AVAILABLE} hours)."
    elif query_type == "can_take_day":
        if PTO_HOURS_AVAILABLE >= 8:
            return f"Yes, you can take a full day off. You have {PTO_HOURS_AVAILABLE} hours available."
        else:
            return f"No, you don't have enough PTO for a full day. You only have {PTO_HOURS_AVAILABLE} hours available."
    elif query_type == "can_take_half_day":
        if PTO_HOURS_AVAILABLE >= 4:
            return f"Yes, you can take a half day off. You have {PTO_HOURS_AVAILABLE} hours available."
        else:
            return f"No, you don't have enough PTO for a half day. You only have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"You have {PTO_HOURS_AVAILABLE} PTO hours available. You can ask about available hours, days, or if you can take time off."
```



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

# AI Agents - Code Example

## Define the Tools in OpenAI format

```
# Define the tool for OpenAI function calling
tools = [
    {
        "type": "function",
        "function": {
            "name": "get_pto_info",
            "description": "Get information about PTO (Paid Time Off) hours",
            "parameters": {
                "type": "object",
                "properties": {
                    "query_type": {
                        "type": "string",
                        "enum": ["available_hours", "available_days", "can_take_day", "can_take_half_day", "general"],
                        "description": "The type of PTO information requested"
                    }
                },
                "required": ["query_type"]
            }
        }
    }
]
```



# AI Agents - Code Example

Tell Model that we have tools when sending prompt

```
def pto_agent(user_query):
    """
    Simple AI agent that handles PTO queries using if-else statements
    """
    print(f"User: {user_query}\n")

    # Create messages for the conversation
    messages = [
        {
            "role": "system",
            "content": "You are a helpful HR assistant that helps employees check their PTO (Paid Time Off) hours. Be friendly and concise."
        },
        {
            "role": "user",
            "content": user_query
        }
    ]

    # Get response from OpenAI
    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=messages,
        tools=tools,
        tool_choice="auto"
    )

    response_message = response.choices[0].message

    # Check if the model wants to call a function using if-else
    if response_message.tool_calls:
```



# AI Agents - Agent Demo

[Go To DEMO](#)



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

# AI Agents - Agent Frameworks

**Agent frameworks helps us code faster!**

- **Strands Agent SDK – AWS**
- **Crew AI**
- **LangChain**
- **LlamaIndex (Meta)**
- **AutoGen (Microsoft)**

# AI Agents - Strands Agent SDK example

## “@tools” used to define what the LLM can use

```
# Define tools using if-else logic with the @tool decorator
@tool
def get_available_pto_hours() -> str:
    """Get the total number of PTO hours available."""
    return f"You have {PTO_HOURS_AVAILABLE} PTO hours available."

@tool
def get_available_pto_days() -> str:
    """Convert PTO hours to days (assuming 8-hour workday)."""
    days = PTO_HOURS_AVAILABLE / 8
    return f"You have {days} PTO days available ({PTO_HOURS_AVAILABLE} hours)."

@tool
def can_take_full_day() -> str:
    """Check if employee has enough PTO for a full day off (8 hours)."""
    if PTO_HOURS_AVAILABLE >= 8:
        return f"Yes, you can take a full day off. You have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"No, you don't have enough PTO for a full day. You only have {PTO_HOURS_AVAILABLE} hours available."

@tool
def can_take_half_day() -> str:
    """Check if employee has enough PTO for a half day off (4 hours)."""
    if PTO_HOURS_AVAILABLE >= 4:
        return f"Yes, you can take a half day off. You have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"No, you don't have enough PTO for a half day. You only have {PTO_HOURS_AVAILABLE} hours available."

print("/ PTO tools defined with if-else logic")
```



# AI Agents - Strands Agent SDK example

## Tell the LLM about the tools available

```
# Define the system prompt for the PTO agent
system_prompt = """You are a helpful HR assistant that helps employees check their PTO (Paid Time Off) hours.
Be friendly, concise, and professional. Use the available tools to provide accurate information about PTO balances."""

# Create the PTO agent with tools
pto_agent = Agent(
    model=model,
    system_prompt=system_prompt,
    tools=[
        get_available_pto_hours,
        get_available_pto_days,
        can_take_full_day,
        can_take_half_day
    ]
)

print("✓ PTO Agent created successfully with 4 tools")
```

Python

```
## Example Usage
```



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

# AI Agents - Strands Agent SDK example

“@tools” used to define what the LLM can use

```
# Define tools using if-else logic with the @tool decorator
@tool
def get_available_pto_hours() -> str:
    """Get the total number of PTO hours available."""
    return f"You have {PTO_HOURS_AVAILABLE} PTO hours available."

@tool
def get_available_pto_days() -> str:
    """Convert PTO hours to days (assuming 8-hour workday)."""
    days = PTO_HOURS_AVAILABLE / 8
    return f"You have {days} PTO days available ({PTO_HOURS_AVAILABLE} hours)."

@tool
def can_take_full_day() -> str:
    """Check if employee has enough PTO for a full day off (8 hours)."""
    if PTO_HOURS_AVAILABLE >= 8:
        return f"Yes, you can take a full day off. You have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"No, you don't have enough PTO for a full day. You only have {PTO_HOURS_AVAILABLE} hours available."

@tool
def can_take_half_day() -> str:
    """Check if employee has enough PTO for a half day off (4 hours)."""
    if PTO_HOURS_AVAILABLE >= 4:
        return f"Yes, you can take a half day off. You have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"No, you don't have enough PTO for a half day. You only have {PTO_HOURS_AVAILABLE} hours available."

print("✓ PTO tools defined with if-else logic")
```



# AI Agents - Strands Agent SDK

**What else does using an agent SDK do for us?**

- Easier to invoke the agent, less if/else statements
- Saving state: agent state, message history
- Tool connection methods: MCP, http stream, server sent events
- Automate model provider methods – query then becomes agent('your query here')

# Strands Agents | tools

 File Operations

 Shell Integration

 Memory

 HTTP Client

 Slack Client

 Python Execution

 Mathematical Tools

 AWS Integration

 Image Processing

 Video Processing

 Audio Output

 Environment Management

 Journaling

 Task Scheduling

 Advanced Reasoning

 Swarm Intelligence

# Strands Agent Demo

[Go To DEMO](#)



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

# Model Context Protocol Server

## What is the MCP Server?

- It is a unified way of declaring tools that the agent/model can use.
- Instead of us writing and copying this over and over:

```
# Define tools using if-else logic with the @tool decorator
@tool
def get_available_pto_hours() -> str:
    """Get the total number of PTO hours available."""
    return f"You have {PTO_HOURS_AVAILABLE} PTO hours available."

@tool
def get_available_pto_days() -> str:
    """Convert PTO hours to days (assuming 8-hour workday)."""
    days = PTO_HOURS_AVAILABLE / 8
    return f"You have {days} PTO days available ({PTO_HOURS_AVAILABLE} hours)."

@tool
def can_take_full_day() -> str:
    """Check if employee has enough PTO for a full day off (8 hours)."""
    if PTO_HOURS_AVAILABLE >= 8:
        return f"Yes, you can take a full day off. You have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"No, you don't have enough PTO for a full day. You only have {PTO_HOURS_AVAILABLE} hours available."

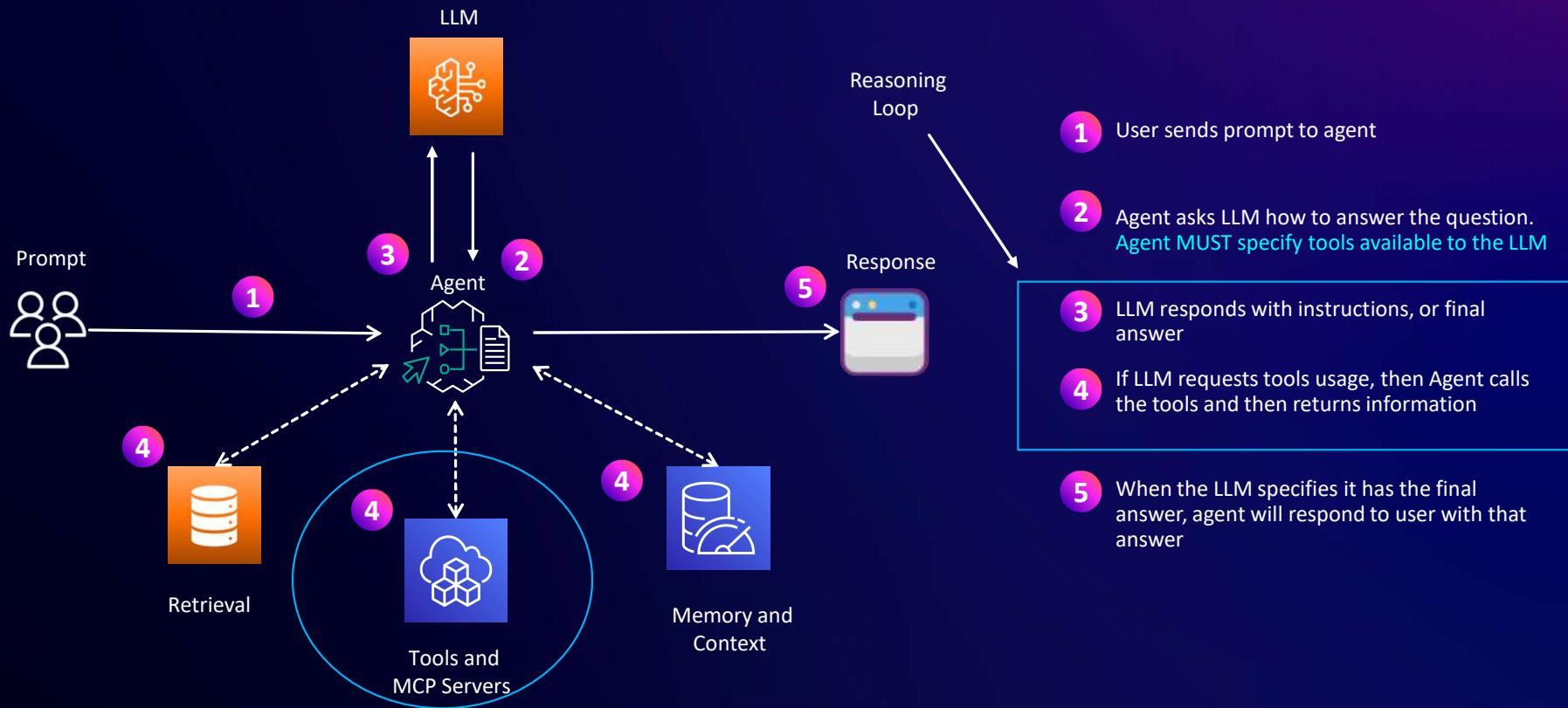
@tool
def can_take_half_day() -> str:
    """Check if employee has enough PTO for a half day off (4 hours)."""
    if PTO_HOURS_AVAILABLE >= 4:
        return f"Yes, you can take a half day off. You have {PTO_HOURS_AVAILABLE} hours available."
    else:
        return f"No, you don't have enough PTO for a half day. You only have {PTO_HOURS_AVAILABLE} hours available.

print(~ PTO tools defined with if-else logic")
```

- We do it once on the MCP server and then other agents can use by calling the API.
- Think of it like a “Menu” where the model can see what is available, and how to use it!



# Agent and MCP Workflows



# MCP demo

**Go To DEMO**



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

# MCP Server Registries

Docker MCP Catalogue – containerized MCP servers

GitHub MCP Registry

Smithery – Cloud for MCP. Contains code and

# Agents and MCP security

## Challenges:

- Unpredictable path
  - When to stop?
- Unpredictable output
  - How to parse output, unexpected output
- Unpredictable costs
  - If process loops continually, we must set limits
- Security
  - What do the agents/mcp/tools have access to?  
Is it based on the user? How do we pass in  
authorization from the user to the tool?

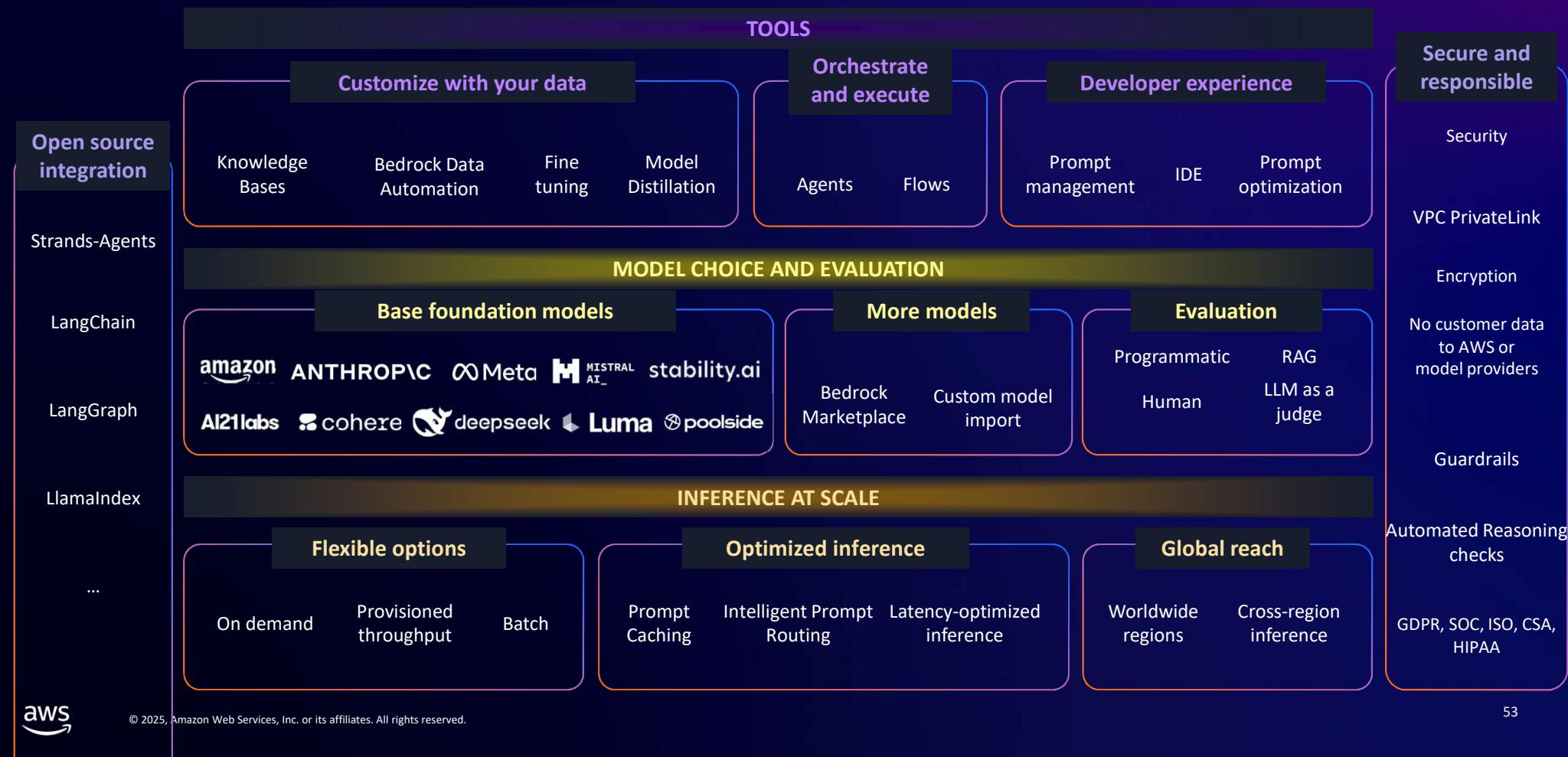
# Agents and MCP Security

```
class AuthenticatedMCPSServer:
    def __init__(self):
        self.user_context = {
            'id': os.getenv('USER_ID'),
            'token': os.getenv('USER_TOKEN'),
            'permissions': self.load_user_permissions()
        }

    @async def execute_tool(self, tool_name, params):
        # All tools automatically have user context
        return await self.tools[tool_name](self.user_context, params)
```

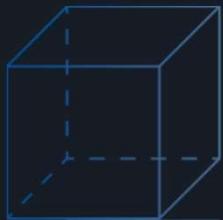
**Get token from central Identity provider, then supply it to agents, tools, and MCP servers**

# Amazon Bedrock

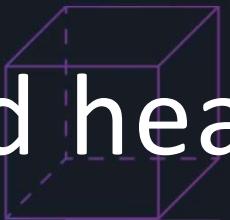
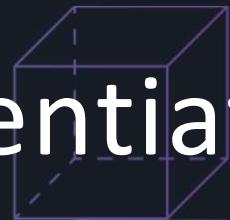
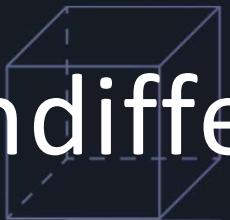


GETTING AGENTS TO PRODUCTION CAN BE CHALLENGING

Bedrock AgentCore can help!



# Undifferentiated heavy lifting



Securely execute  
and scale agent  
code

Remember past  
interactions &  
learning

Identity and  
access controls  
for all agents and  
tools

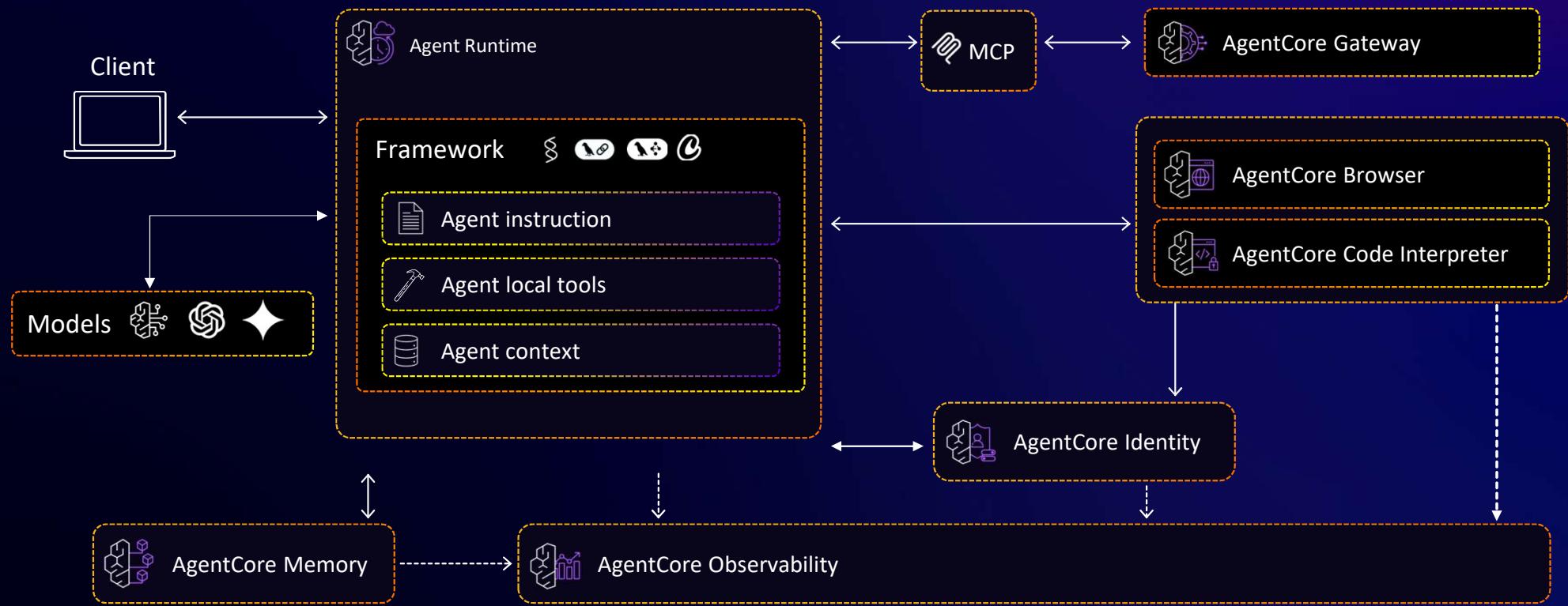
Agentic tool use for  
executing complex  
workflows

Discover and  
connect with  
custom tools and  
resources

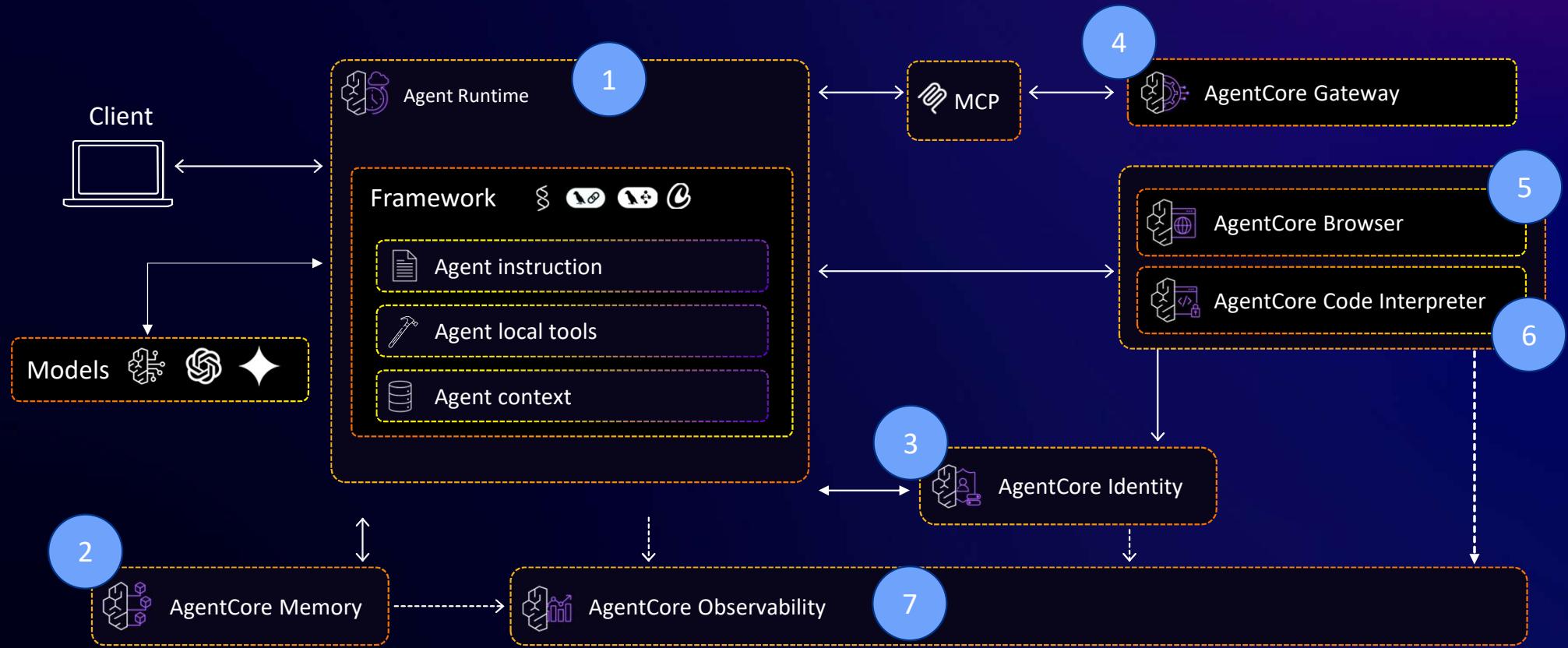
Understand and  
audit every  
interaction

(Optional slides)

# Amazon Bedrock AgentCore for production ready agents



# Amazon Bedrock AgentCore for production ready agents



# AgentCore Python SDK

- **Keep your agent logic** - Works with any framework
- **Zero infrastructure management** - No servers, containers, or scaling related work
- **Command line and python** - Cloudformation, CDK coming soon
- **Composable services** – work with Runtime, Gateway, Tools, Identity

bedrock-agentcore 0.1.1

pip install bedrock-agentcore

```
# Build your agent with the SDK
from bedrock_agentcore import BedrockAgentCoreApp

app = BedrockAgentCoreApp( )

@app.entrypoint
def my_agent(request):
    # Implement agent logic
    return response

app.run( )
```

```

%%writefile strands_claude.py
from strands import Agent, tool
from strands_tools import calculator # Import the calculator tool
import argparse
import json
from bedrock_agentcore.runtime import BedrockAgentCoreApp
from strands.models import BedrockModel

app = BedrockAgentCoreApp()

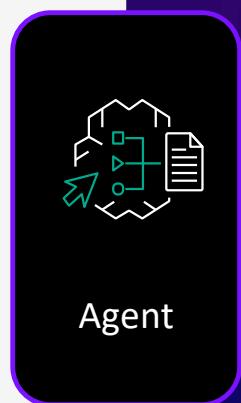
# Create a custom tool
@tool
def weather():
    """ Get weather """ # Dummy implementation
    return "sunny"

model_id = "us.anthropic.claude-3-7-sonnet-20250219-v1:0"
model = BedrockModel(
    model_id=model_id,
)
agent = Agent(
    model=model,
    tools=[calculator, weather],
    system_prompt="You're a helpful assistant. You can do simple math calculation, and tell the weather."
)

@app.entrypoint
def strands_agent_bedrock(payload):
    """
    Invoke the agent with a payload
    """
    user_input = payload.get("prompt")
    print("User input:", user_input)
    response = agent(user_input)
    return response.message['content'][0]['text']

if __name__ == "__main__":
    app.run()

```



[https://github.com/awslabs/amazon-bedrock-agentcore-samples/blob/main/01-tutorials/06-AgentCore-observability/01-Agentcore-runtime-hosted/runtime\\_with\\_strands\\_and\\_bedrock\\_models.ipynb](https://github.com/awslabs/amazon-bedrock-agentcore-samples/blob/main/01-tutorials/06-AgentCore-observability/01-Agentcore-runtime-hosted/runtime_with_strands_and_bedrock_models.ipynb)



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

# AgentCore Runtime

## Deploying the agent to AgentCore Runtime

```
from bedrock_agentcore_starter_toolkit import Runtime
from boto3.session import Session
boto_session = Session()
region = boto_session.region_name

agentcore_runtime = Runtime()
agent_name = "strands_claude_getting_started"
response = agentcore_runtime.configure(
    entrypoint="strands_claude.py",
    auto_create_execution_role=True,
    auto_create_ecr=True,
    requirements_file="requirements.txt",
    region=region,
    agent_name=agent_name
)
response
```

### Launching

```
launch_result = agentcore_runtime.launch()
launch_result
```

### Invoking

```
invoke_response = agentcore_runtime.invoke({"prompt": "How is the weather now?"})
invoke_response
```



# AgentCore Runtime

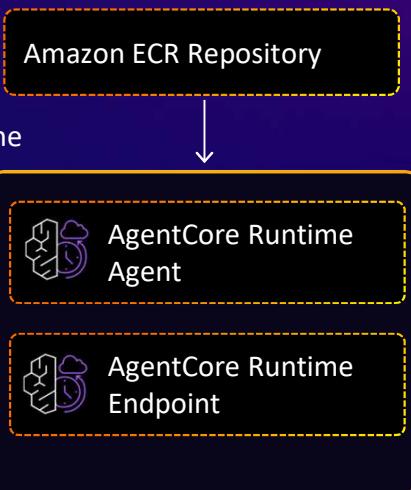
Agent or tool code



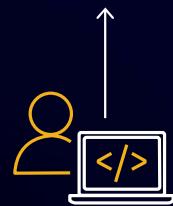
configure



AgentCore Runtime



launch



User

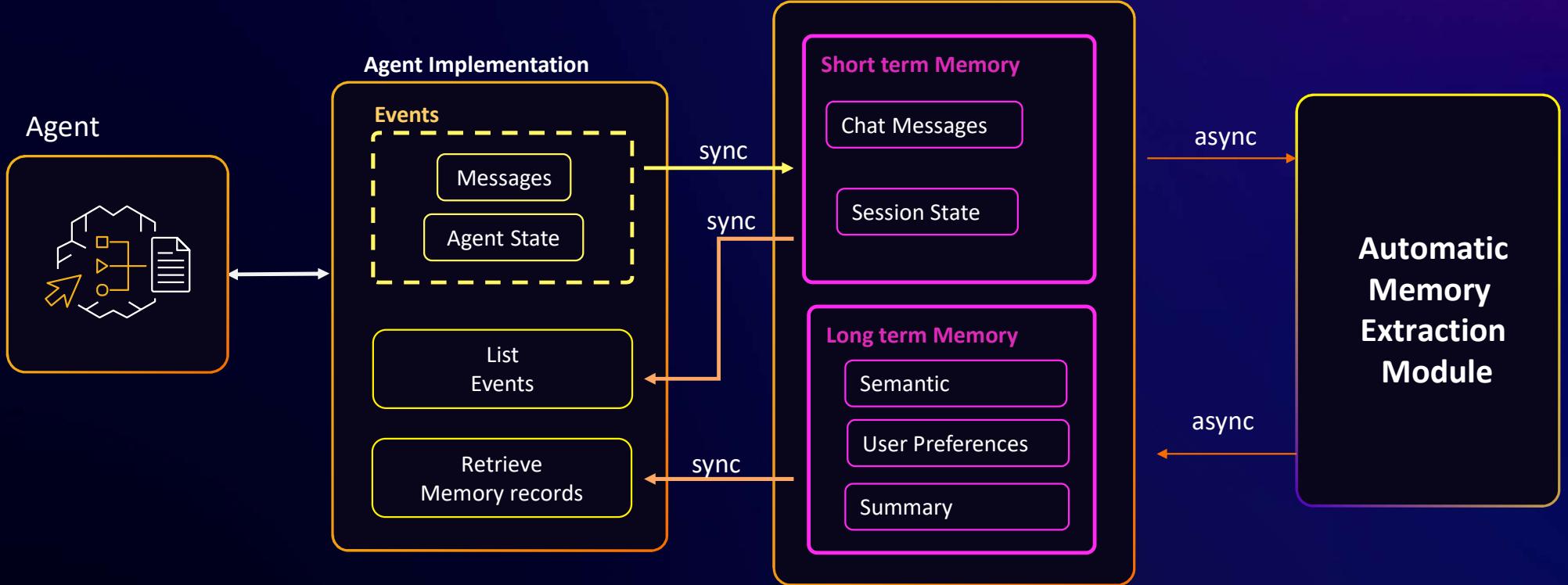


Application

invoke

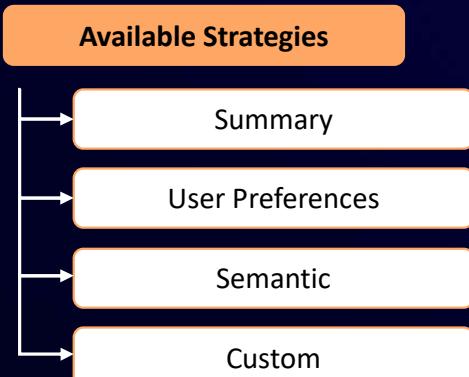


# AgentCore Memory



# AgentCore Memory - Long-term memory

KEY CONCEPTS: STRATEGIES



**Session Summaries:** Creates condensed representations of interaction content and outcomes.

**User Preferences:** Stores and learns recurring patterns in user behavior, interaction styles, and choices.

**Semantic Facts:** Maintains knowledge of facts and domain-specific information, technical concepts, and their relationships.

**Custom:** Lets you override the prompts and choose the LLM to tailor the memory extraction and consolidation to your specific domain or use case.

# Strands + Memory (via tool)

## Step 1: Define a AgentCore Memory

```
from bedrock_agentcore.memory import MemoryClient

client = MemoryClient(region_name="us-west-2")

memory = client.create_memory_and_wait(
    name="MyAgentMemory",
    strategies=[{
        "userPreferenceMemoryStrategy": {
            "name": "UserPreference",
            "namespaces": ["/users/{actorId}"]
        }
    }]
)
```

## STEP 2: ADD MEMORY TO AN AGENT

```
from strands import tool, Agent
from strands_tools.agent_core_memory import
AgentCoreMemoryToolProvider

strands_provider = AgentCoreMemoryToolProvider(
    memory_id=memory.get("id"),
    actor_id="PersonX",
    session_id="sess1",
    namespace="/users/PersonX",
    region="us-west-2")

agent = Agent(tools=strands_provider.tools)

agent("I'm vegetarian")
agent("I'm in the mood for Italian cuisine.")
...
agent("Do you remember my preferences?")
```



# Agentic AI Auth Basics

As you're building an AI agent...

Your users need to access the agent through an application



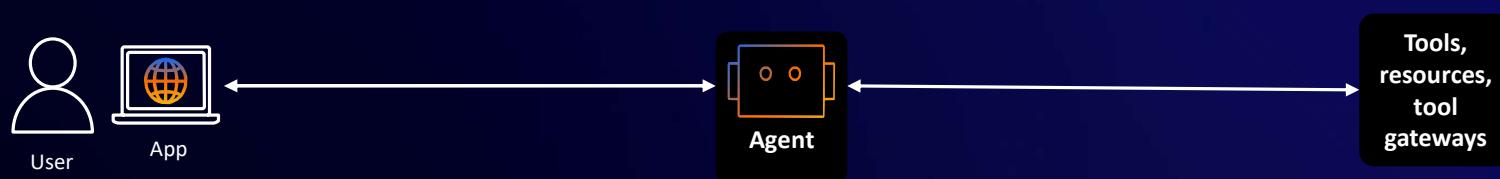
Inbound Auth

The agent needs to access resources



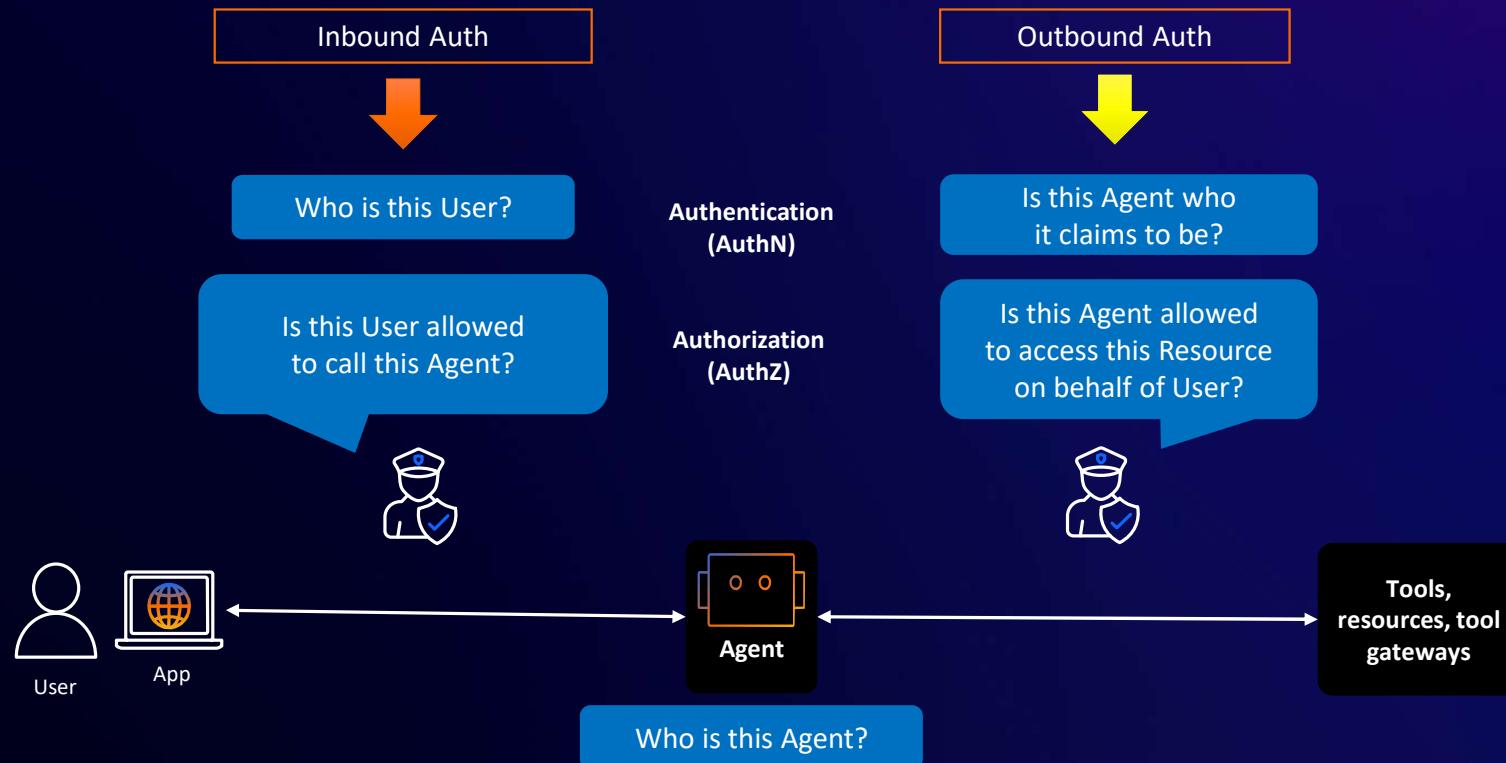
Outbound Auth

**Bedrock AgentCore Identity** can help you with both...



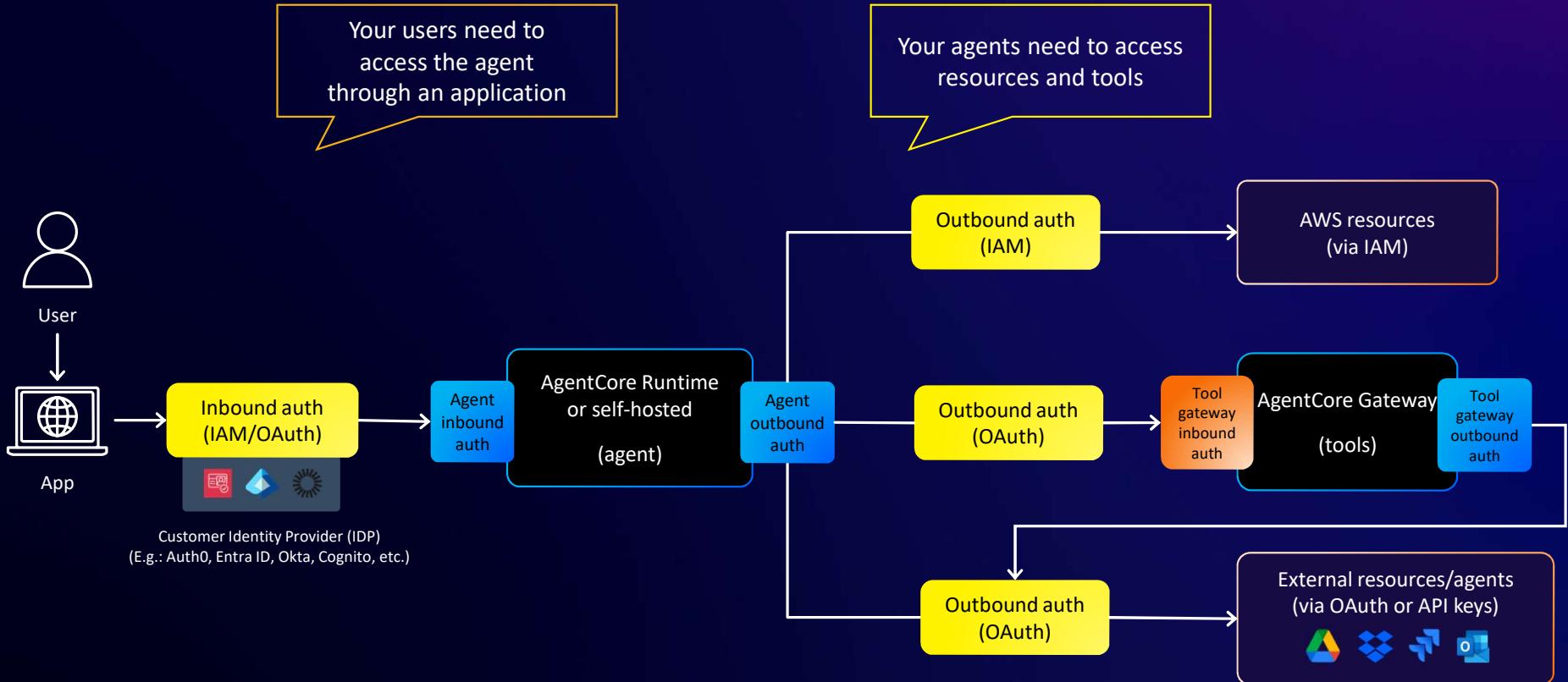
...and we can also host your agent code

# Agentic AI Auth Basics (cont.)



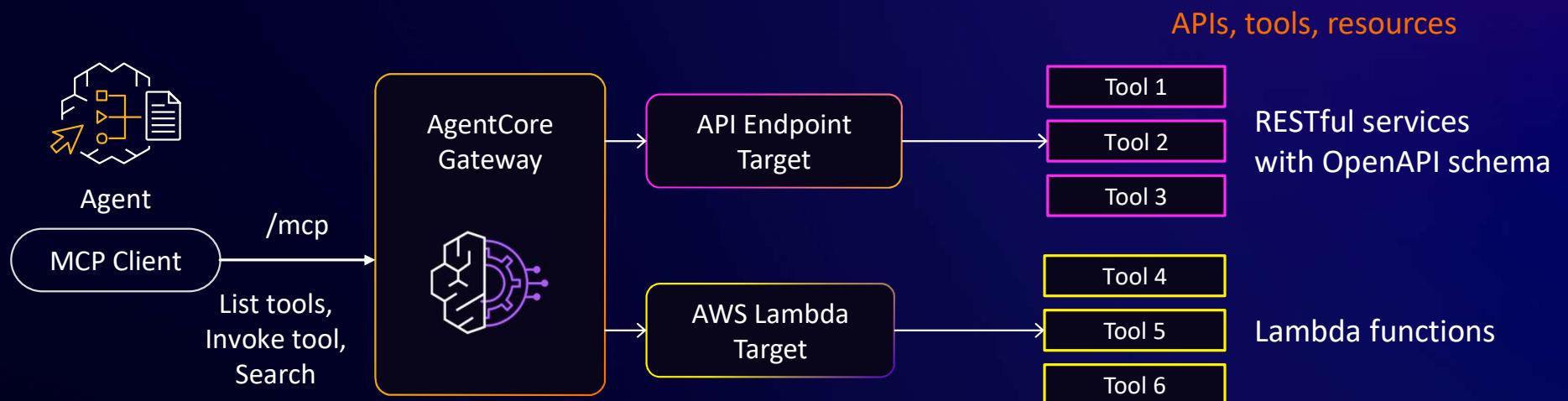


# Auth with AgentCore Identity





# AgentCore Gateway





# AgentCore Browser



1. Query  
("Buy shoes on Amazon")



2. Invoke LLM



3. Tool use



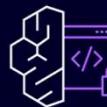
4. Command Translation



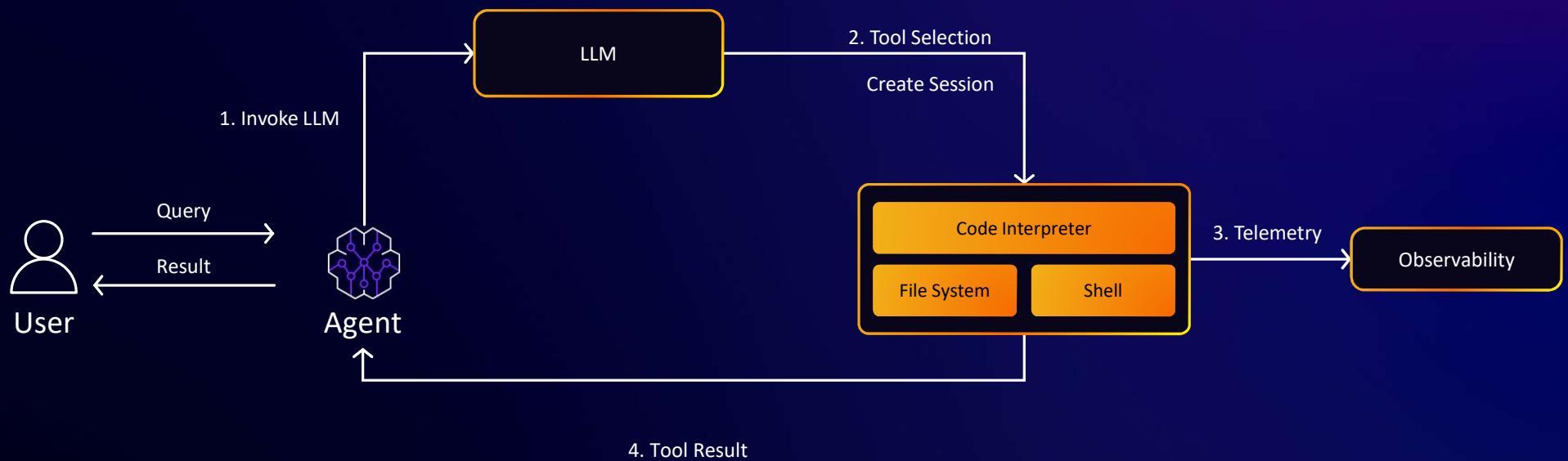
5. Command Execution



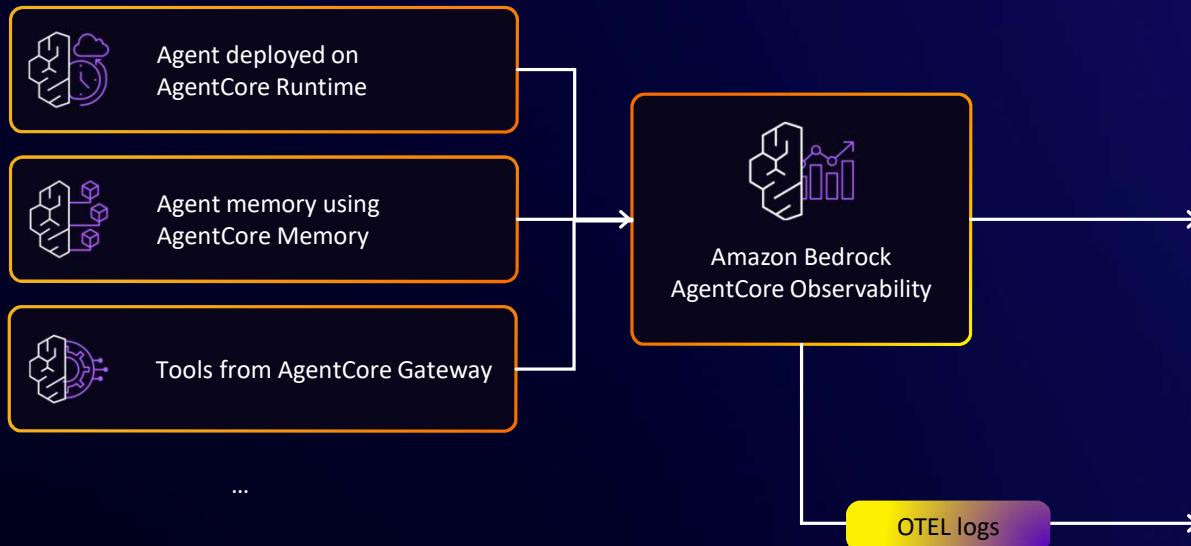
6. Screenshot



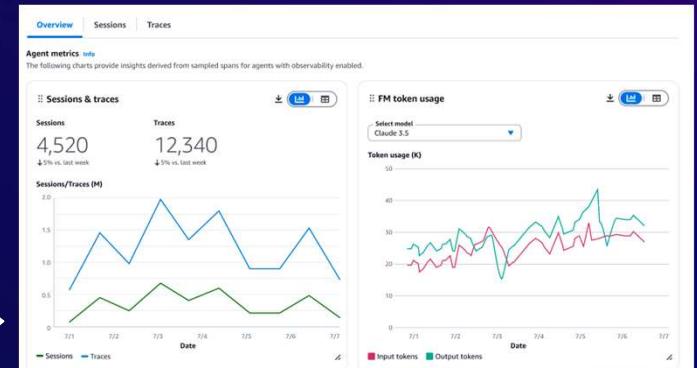
# AgentCore Code Interpreter



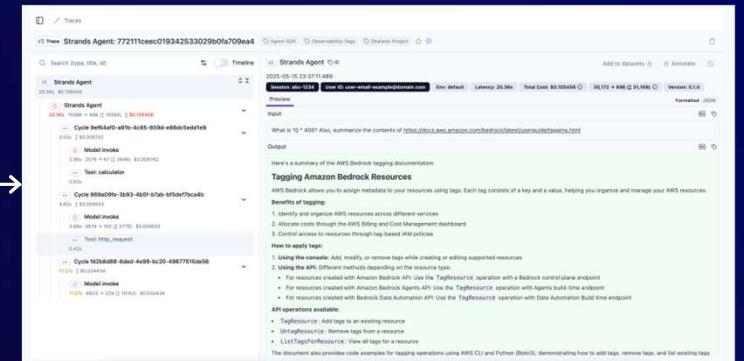
# AgentCore Observability



AgentCore Observability dashboards



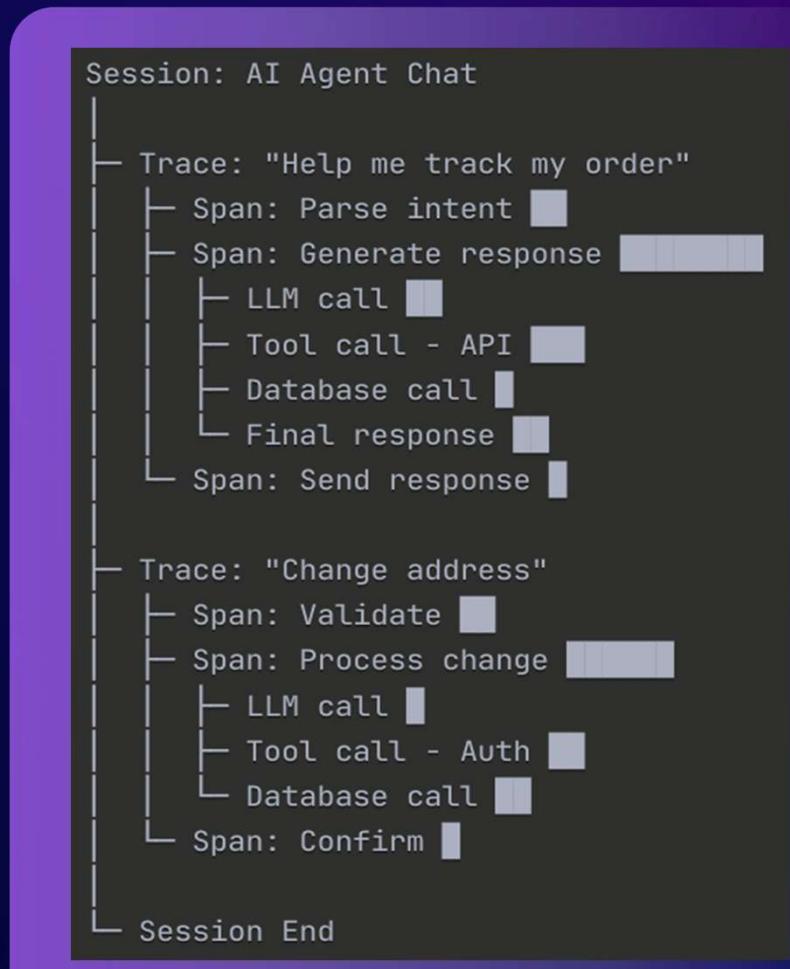
Third-party observability dashboards





# Key Concepts

- Session
- Trace
- Span
- Sub-Span



## ▼ Runtime metrics

These metrics provide insights into all agents deployed on Runtime

### Runtime Agents/Aliases

14/14

### Runtime sessions

120

↑ 700.0% vs previous period

### Runtime invocations

259

↑ 324.6% vs previous period

### Runtime errors

31

↓ 6.1% vs previous period

### Runtime throttles

0

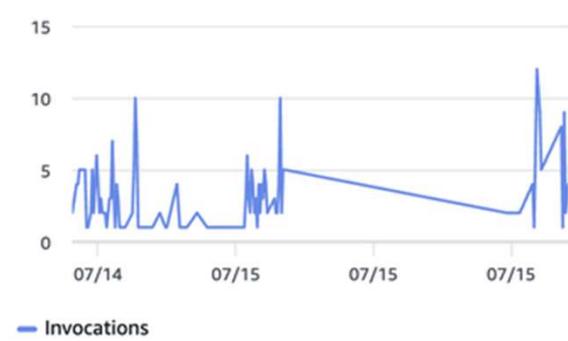
↑ 0.0% vs previous period

## ▼ View details

### Runtime sessions



### Runtime invocations



### Runtime errors



## Runtime metrics

These charts provide insights into all agents deployed on Runtime.

### \_Runtime sessions & invocations

Total sessions

9

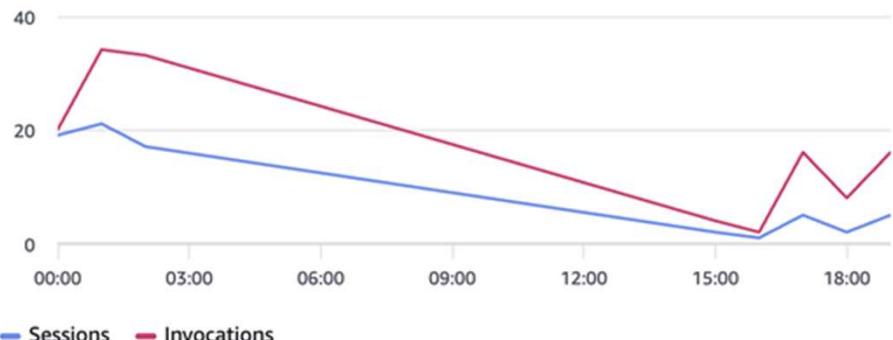
↓ 78.0% vs previous period

Total invocations

16.625

↓ 53.8% vs previous period

#### Session/Invocations



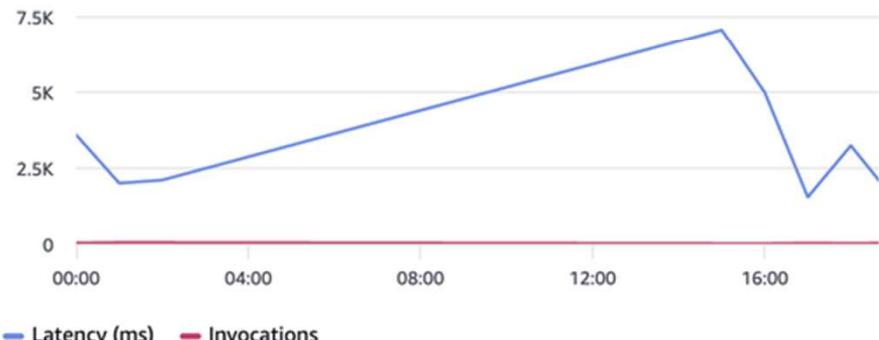
### Runtime latency

Avg runtime latency

3.2 secs

↓ 23.9% vs previous period

#### Invocations



Overview Sessions Traces

## Sessions (73)

View all your sessions across all resources in your account. Click on any session to view a detailed dashboard for that session.



<input type="checkbox"/> Session ID	▼ Traces	▼ Errors	▼ Throttles	▼ P95 Latency (ms)
<a href="#">2f112a96-19fa-467c-8000-3ad...</a>	4	0	0	2470.30
<a href="#">3b1c0e19-9b82-4313-9a0e-b6f...</a>	2	0	0	3615.21
<a href="#">54d9118c-795a-4916-9aed-f96...</a>	2	0	0	4179.02
<a href="#">6aa68698-91c4-43c9-8a22-f53...</a>	6	0	0	3029.01
<a href="#">563dbe02-40ba-423e-b57b-69...</a>	2	0	0	4141.75
<a href="#">223582a6-27d5-4b88-be24-78...</a>	7	0	0	2948.37
<a href="#">9057e1e1-9590-4160-8e45-36...</a>	1	0	0	4547.59
<a href="#">40b3dbdc-c6ca-485d-9637-005...</a>	7	0	0	3261.54
<a href="#">93b1d8bd-d09d-4387-adec-9dc...</a>	1	0	0	2517.02
<a href="#">698251bb-5d8c-4dda-a94c-47d...</a>	3	0	0	3374.48



## Traces (10000)

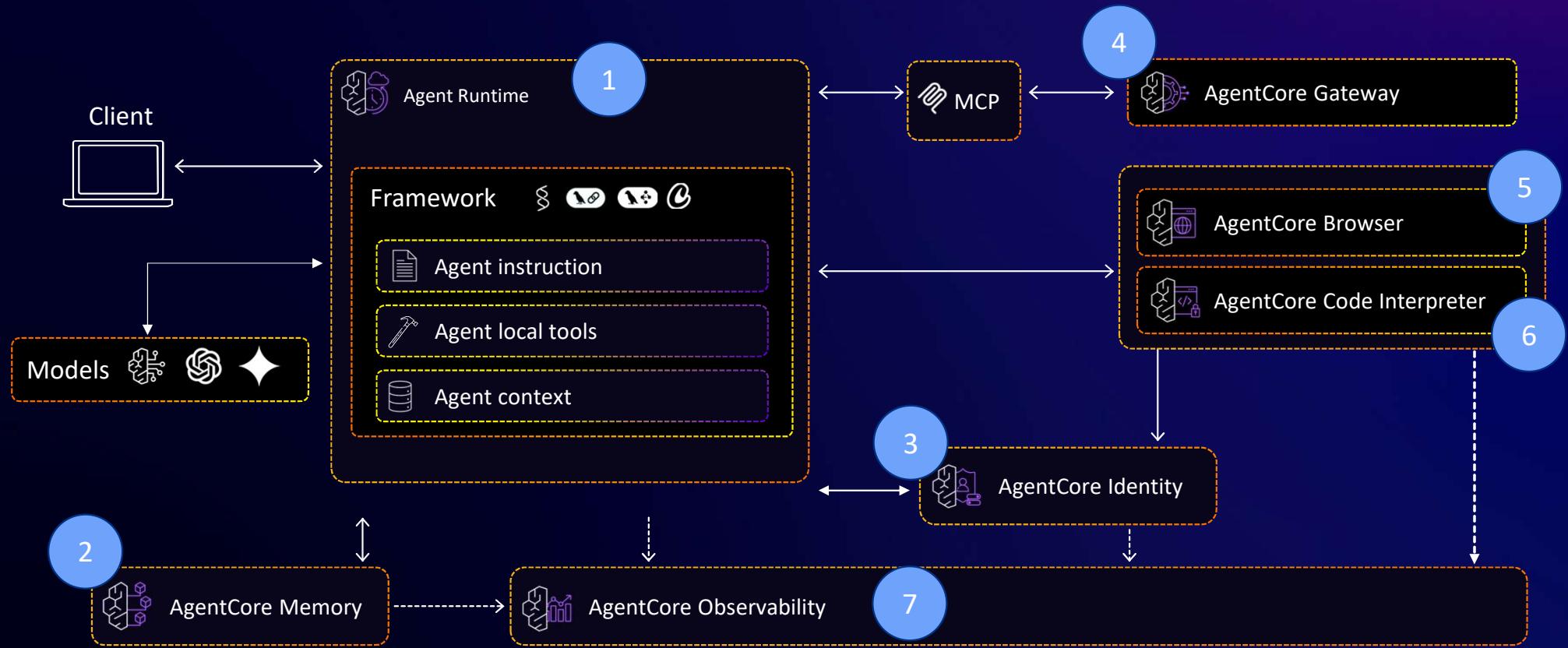
View all the traces for an agent and a session. Click any trace to view a detailed dashboard.

Filter traces

< 1 2 3 4 5 6 7

<input type="checkbox"/>	Trace ID	▼	Spans	▼	Errors	▼	Latency (ms)	▼	Start time	▼	P95 span la
<input type="checkbox"/>	<a href="#">6877f1485a445a40f212...</a>		3		0		17.63		07/16/2025 6:36 PM UTC		17.64
<input type="checkbox"/>	<a href="#">6877f14622d3b48a5a68...</a>		2		0		64.96		07/16/2025 6:36 PM UTC		64.96
<input type="checkbox"/>	<a href="#">6877ec945c4faf387c484...</a>		3		0		31.67		07/16/2025 6:16 PM UTC		31.67
<input type="checkbox"/>	<a href="#">6877ec94cb4fcc5a46c14...</a>		3		0		21.77		07/16/2025 6:16 PM UTC		21.77
<input type="checkbox"/>	<a href="#">6877ec92e1b1bb84c5e3...</a>		2		0		68.55		07/16/2025 6:16 PM UTC		68.55
<input type="checkbox"/>	<a href="#">6877ec923c480410acd9...</a>		2		0		68.80		07/16/2025 6:16 PM UTC		68.80
<input type="checkbox"/>	<a href="#">6877eb86c324657fd63d...</a>		3		0		20.89		07/16/2025 6:12 PM UTC		20.89

# Amazon Bedrock AgentCore for production ready agents



# How to move to Production



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

# Everyone talks about models...



Gen AI Applications  
Foundation Models

# But that's only the tip of the iceberg!



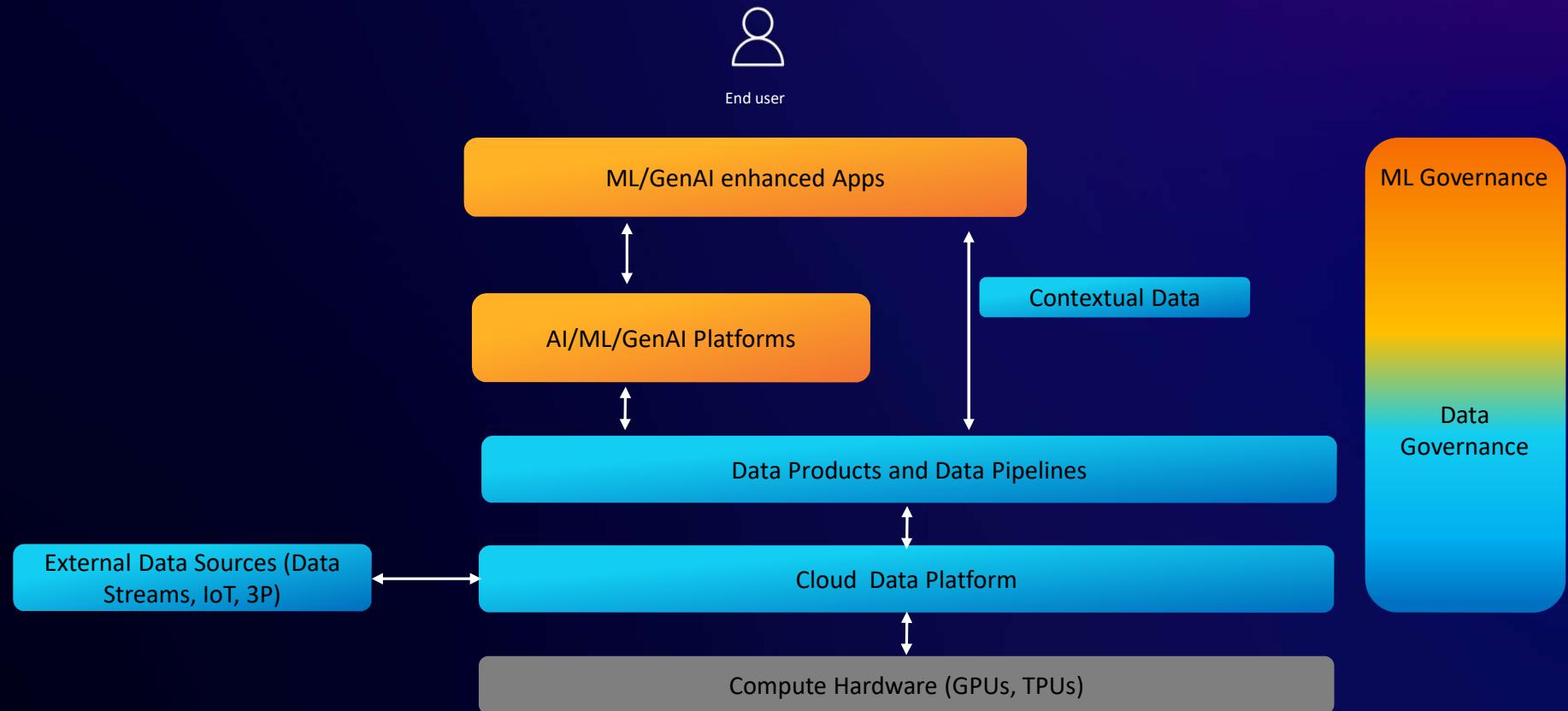
Gen AI Applications  
Foundation Models

Scale Considerations

**Data Quality**  
**Compliance**  
**Security**  
**Cost to value**  
**Privacy**  
**Repeatability**  
**Resilience**  
**Governance**  
**Responsible AI**

Key: AI/ML Data

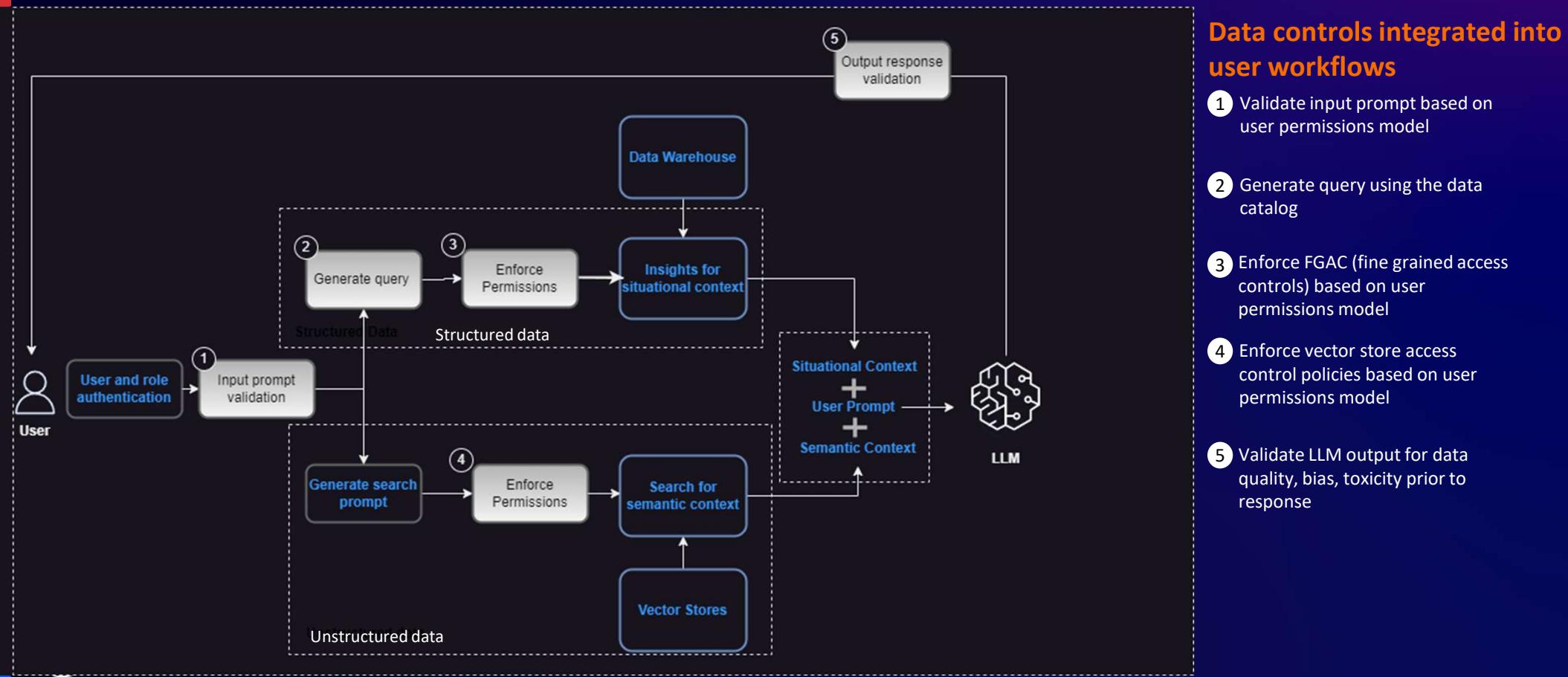
# GenAI and Data Governance



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

# SECURITY IS JOB NUMBER 1!!!!

## Enforce data permission controls



# **How to move to Production Skills/People and Processes**



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

# Key Roles and Skills

Advance Analytics Team  
Data Lake



## Data Engineer

Prepare & Ingest data building ETL pipelines

Data Science Team  
Experimentation & MLOps



## Data Scientist

Create the best ML models to solve business problems

Platform Team  
Secure Cloud/Data/ML Platform



## MLOps Engineer/Admin

Standardize CI/CD, user/service role, model consumption, testing and deployment methodology

Business  
Viz Dashboards, ML Adoption, & ROI



## Product Owners

Define business problem, business KPIs, and make business decisions

Data Owners  
Manage data sharing and provide access



ML Engineer  
Collaborate with DS to productionize ML



Security & Architects  
Assess data, user, and service access creating policies and infrastructure

Business Stakeholder  
Budget approval and key decision making



Labeler Team  
Data Preparation at Scale



Data Labelers/Editors  
Label or edit billions of Data for FM models and hundreds of data for fine tuning interacting with data lake using a dedicated website

Data Science Team Extension  
Context Adaptation



Fine Tuners  
Select the corresponding FM, evaluate the model & design the deployment method/infrastructure

Application Developer Team  
Integrate GenAI models in applications



## Generative AI Developers, AppDev, & Prompt Engineers/Testers

Design prompt inputs, create examples of prompt input/outputs, and test the engineered prompts, develop the GenAI application and front-end

End-Users  
Consume Generative AI applications

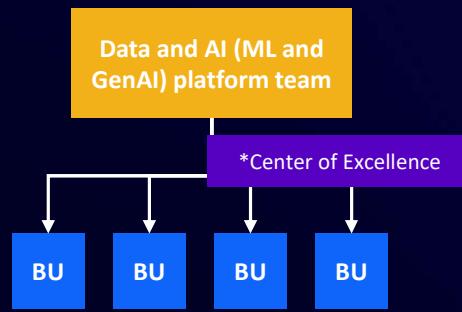


Generative AI Users  
Consume AI solutions as black box, share data and rate the quality of output



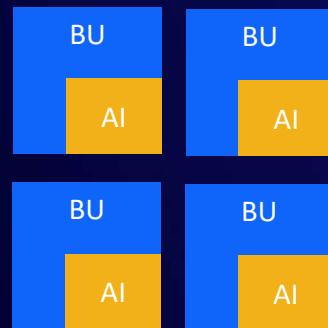
# Operating Model

## CHOOSING THE BEST OPERATING MODEL FOR AN ORGANIZATION



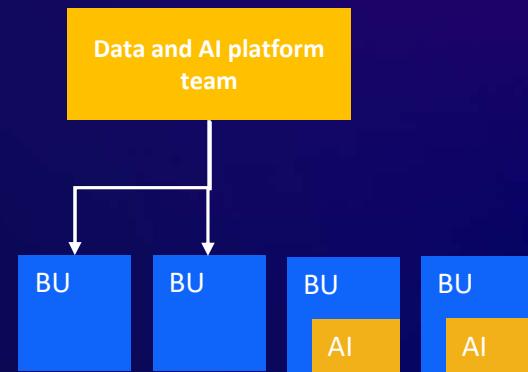
**Centralized Platform Team**

All data science activities are centralized within a single team or organization



**Decentralized**

Each BU is free to choose and own how they build data, ML and GenAI platforms

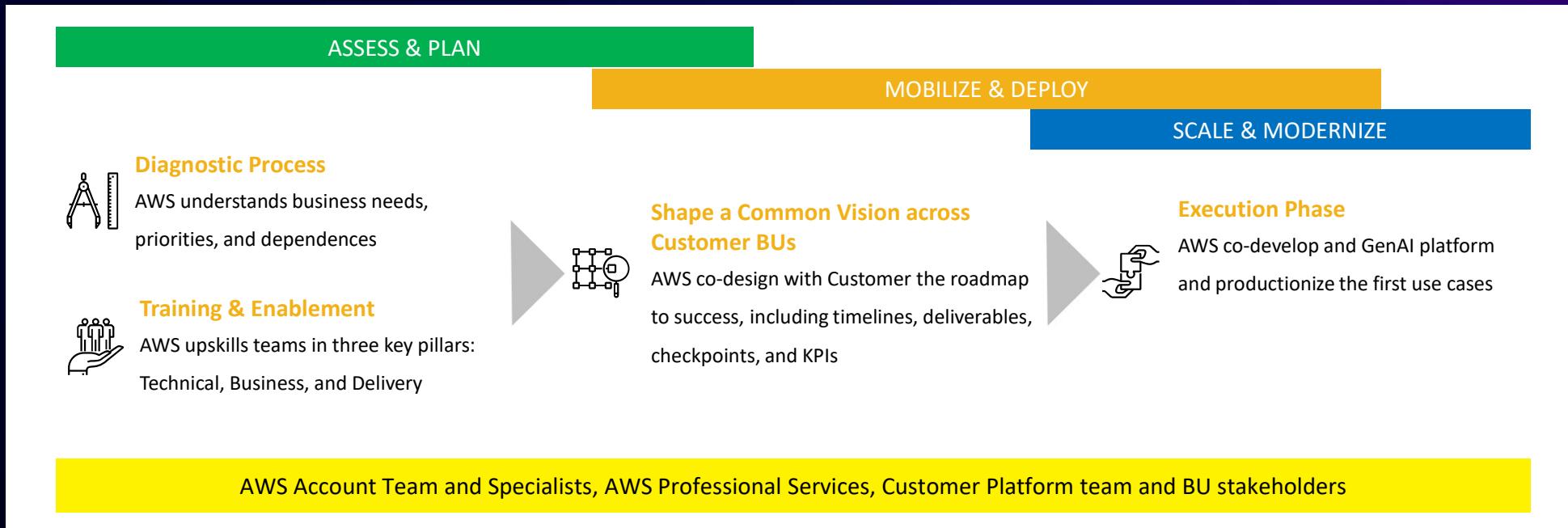


**Mixed**

Smaller BUs depend on central teams, largest BUs have their own but try to adhere to some general policies and guidelines

# Change Management – Road to success

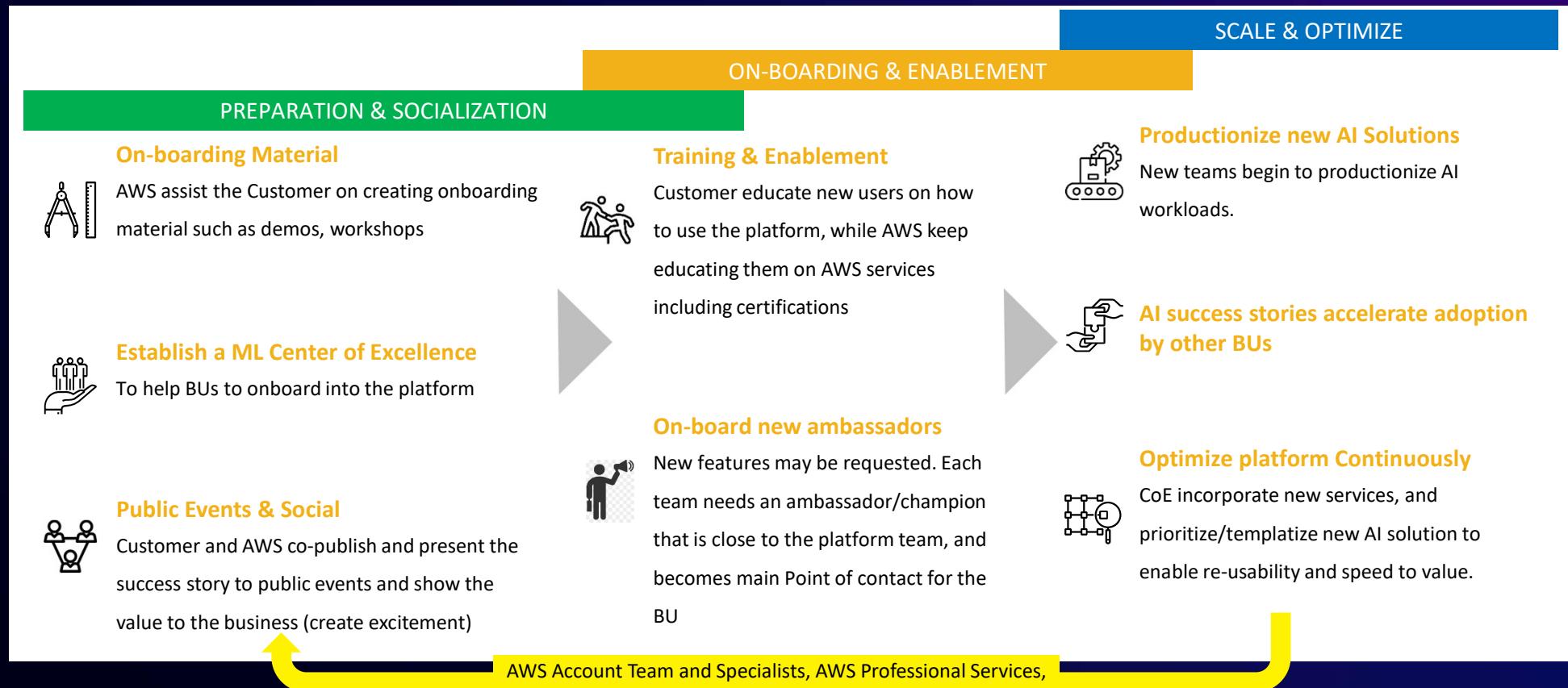
PRIOR TO THE PLATFORM DELIVERY



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

# Change Management – Road to success

AFTER THE MLOPS PLATFORM DELIVERY



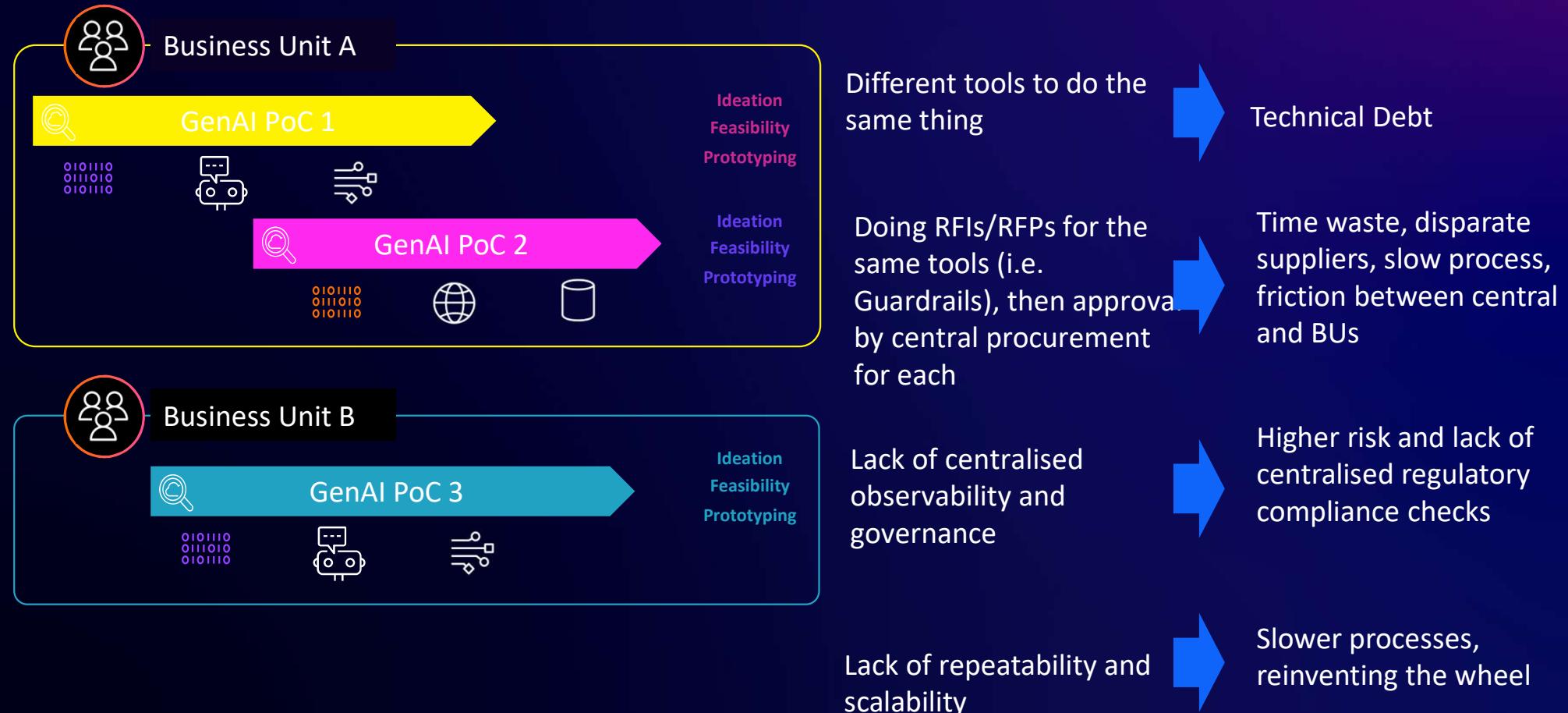
# How to move to Production GenAI/ML Operations



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

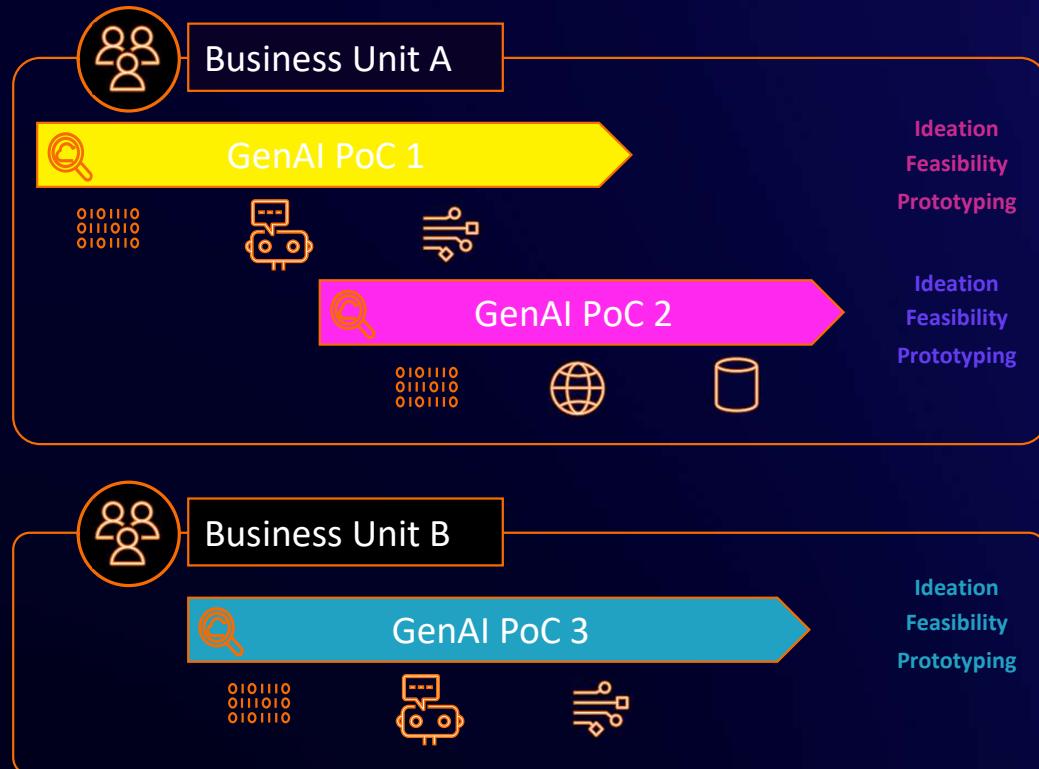
# Barriers to GenAI Production Deployments

SILOED EXPERIMENTATION, RISK MANAGEMENT, REUSABILITY



# Barriers to Generative AI Production Deployments

SILOED EXPERIMENTATION, RISK MANAGEMENT, REUSABILITY



## Path to Production Challenges



Experimentation & Proof of Concepts

Pilot / MVP

Production - Value Realization



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

# Key Benefits GenAI Platform

AN END-TO-END AUTOMATION FRAMEWORK FOR FAST-MOVE TO PRODUCTION



## Cost Controls

Control and reduce costs with optimized infrastructure



## Speed to Value

Faster to innovation and business value



## Scalability/Repeatability

Automation and reusable components



## Governance

Security, Safety and Compliance



## Admin and BUs bespoke access

Fulfill the needs for Observability and control for Admin and the out-of-the box approved tools for building and deploying GenAI applications



## Model Agnostic and Modular

Any model from any cloud provider and modularity for adapting to your environment to new GenAI tools

# GenAI Platform Operational model - Key Personas

ENTERPRISE READY WITH CONSISTENT GOVERNANCE AND CONTROLS



Platform Admin

Platform administrators that manage tenants & users, and can deploy & configure shared foundation blueprints. This persona is typically part of a Gen AI Centre of Excellence or centralized products or technology team. They build, deploy and maintain the GenAI Platform. They approve models and perform security checks. Control Governance.



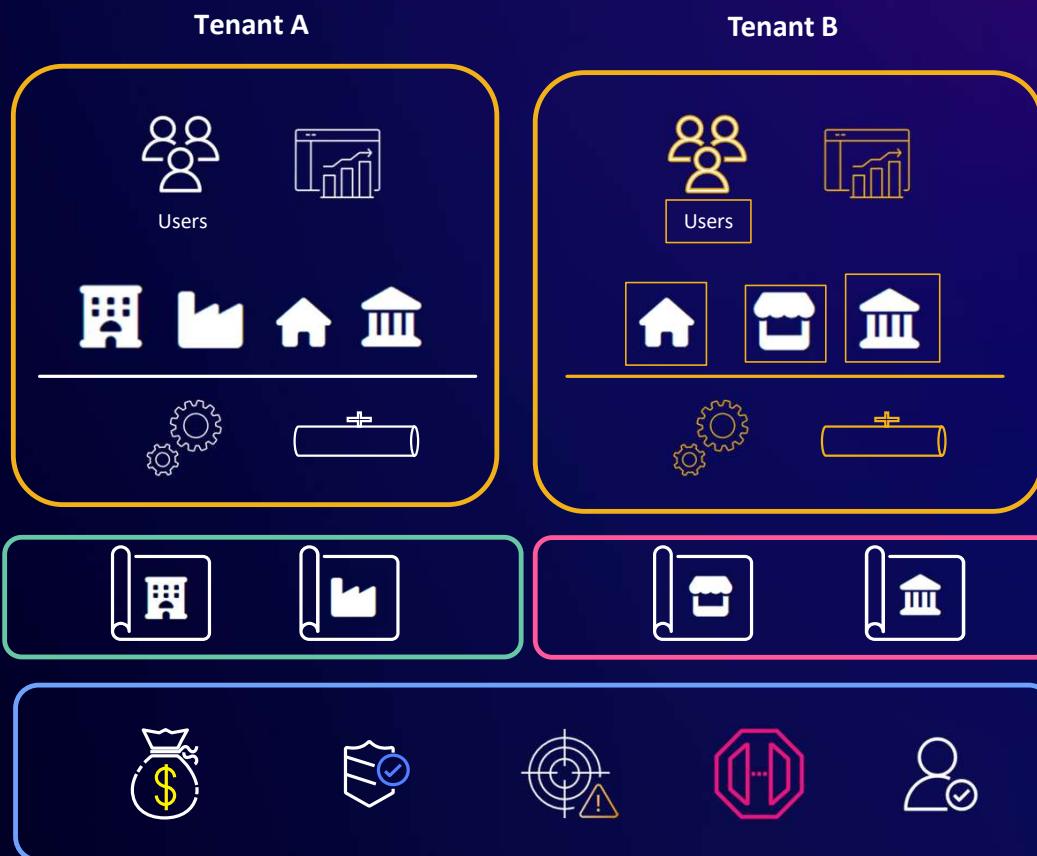
Tenant

Users that can deploy, configure and manage use case blueprints in their own Tenant environment and access directly the AI Platform microservices to build GenAI applications. Have a clear path to production and pre-approved services, achieving 80% faster Go-Live on their POCs.

# GenAI Platform Operational Model – Lifecycle illustration

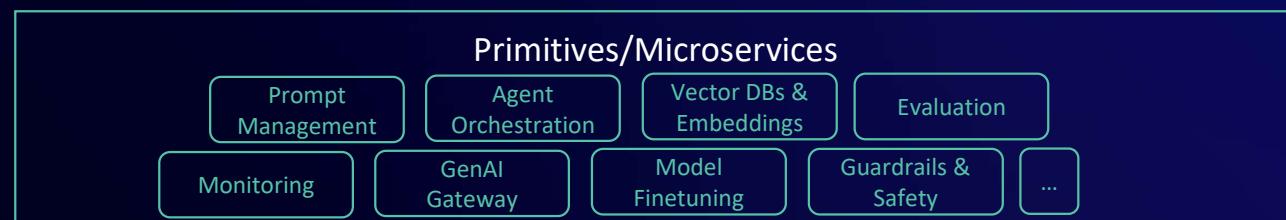
## CREATING CUSTOMER VALUE

-  Tenant onboarding
  - 6 Scale to new tenants
  - 5 New blueprints maybe created, these are shared with Platform team and added to the catalogue
  - 4 Tenants start building POCs, running evaluations and moving applications to production
  - 3 Tenant onboarding
    - Get access to the platform
    - Training and enablement of developer
  - 2 Curate building blocks catalog
    - Configure microservices
    - Add blueprints to the catalogue
  - 1 Setup Foundations
    - Set up multi-tenant infrastructure
    - FinOps
    - Establish governance model
    - Configure microservices
    - Tenant onboarding
    - Observability



# GenAI Platform Overview

## CORE COMPONENTS



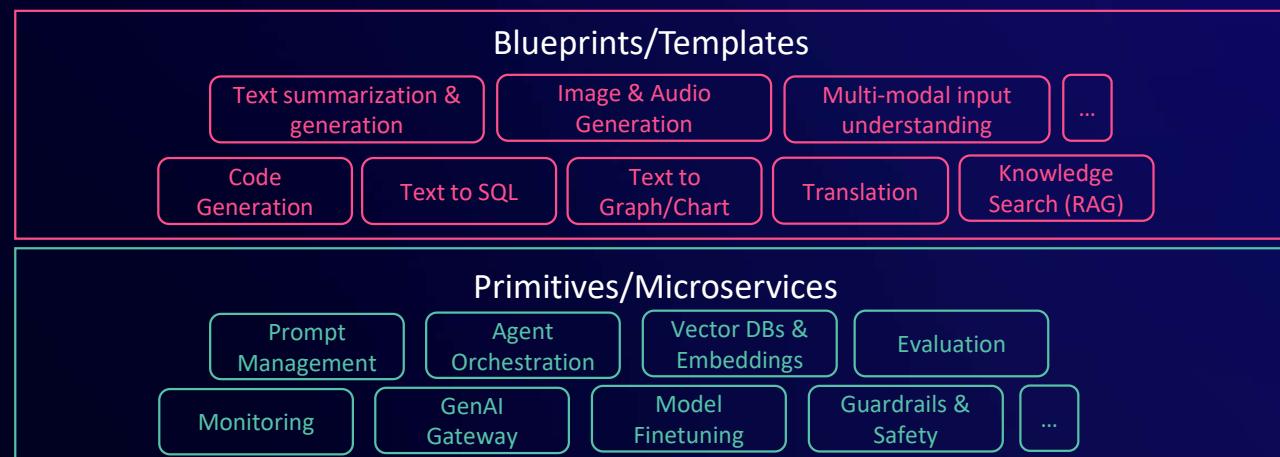
# GenAI Platform Microservices – Modular approach

## BUILDING BLOCKS FOR GENAI APPLICATION DEVELOPERS

GenAI Gateway	Access to pre-approved models from any vendor and OS, access controls, cost tracking
Evaluation	Allows to compare model outputs and choose the model best suited for downstream generative AI applications. You can use automatic model evaluations, human-in-the-loop evaluations or LLM-as-a-judge
Guardrails & Safety	Safeguards that can be applied as generic defaults for users in your organization or can be specific to each use case
Vector DBs & Embeddings	Implement and expose different templates for performing RAG search that help improve the quality of the retrieved documents
Agent Orchestration	Create, manage, access, and share agent implementations
Monitoring	Evaluation of model performance against predefined metrics. It gathers request and response data, runs evaluation jobs against preset baselines, saves the outputs, and sends an alert in case of issues.
Model Finetuning	Fine-tunning via API capabilities includind model distillation
Prompt Management	For storing, versioning, tracking, and sharing prompts. It also lets you automate your evaluation process in your pre-production environments.
...	Capability to expand to future needs as the technology evolves

# GenAI Platform Overview

## CORE COMPONENTS



# GenAI Platform – Blueprints and Templates

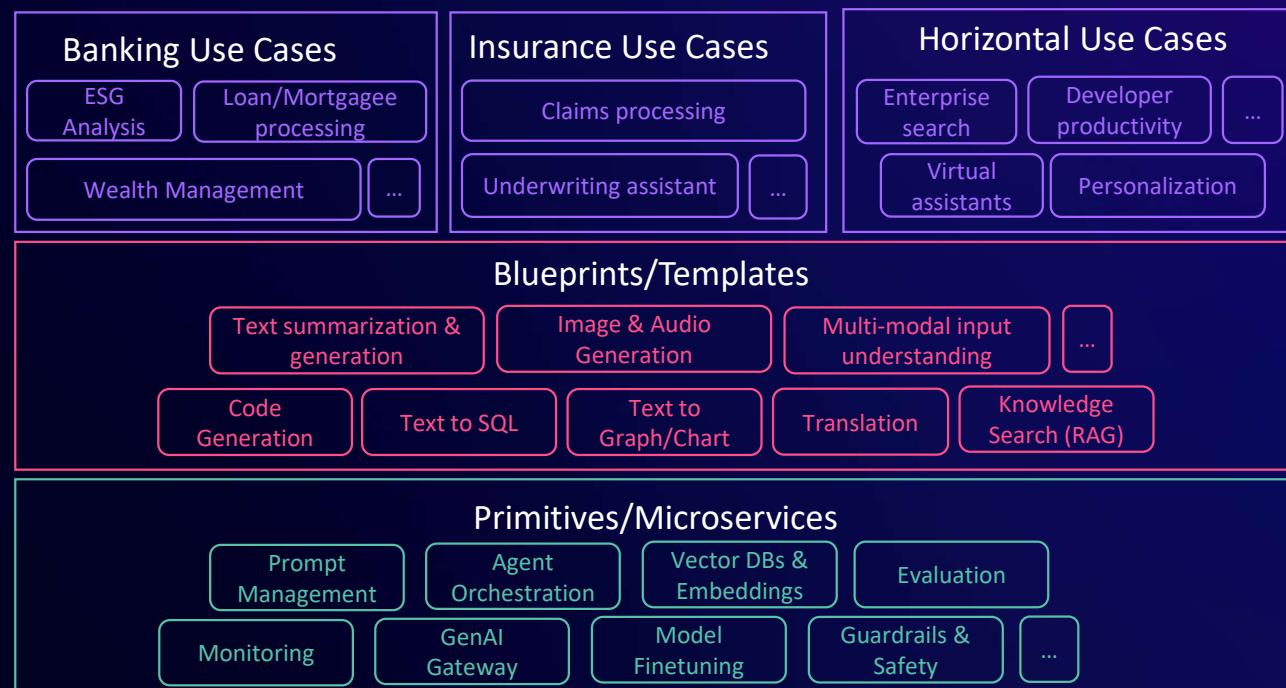
INFRASTRUCTURE AS A CODE FOR MOST COMMON USE CASES PRIMITIVES

Text summarization & generation	Text summarization & text generation from text files
Chatbot	Chat template to easily build chatbots into new or existing applications
Text to SQL	Easily deploy Text to SQL
Image & Video Generation	For sales or marketing use cases
Translation	Translation template
Knowledge Search (RAG)	Pre-configured RAG to common data formats such as for insurance policies, emails, graphs, financial reports, websites, etc
...	

80% faster Go-Live because the template is pre-configured and pre-approved. Only configure the access to data the sources and integration to the application

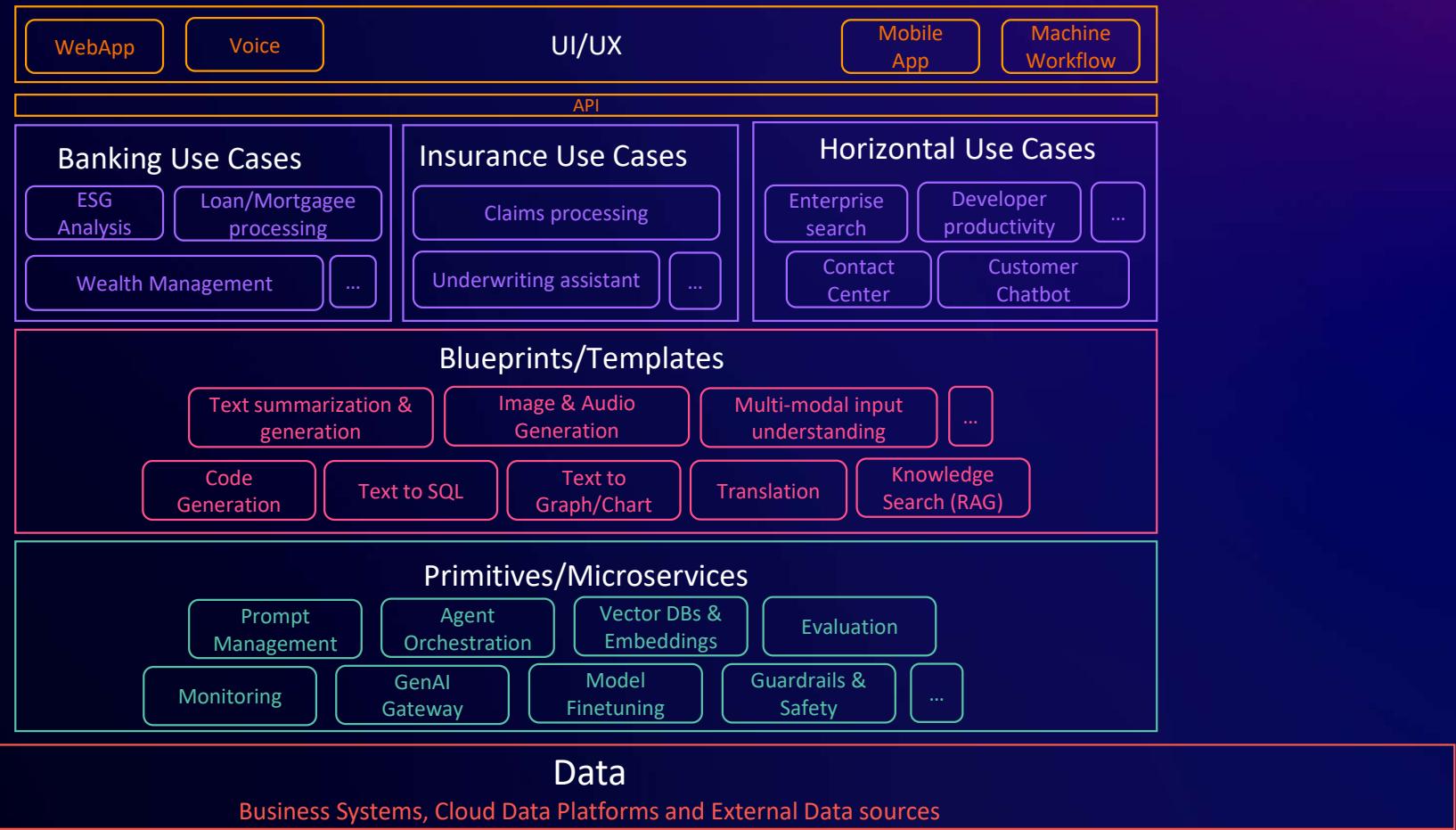
# GenAI Platform Overview

## CORE COMPONENTS



# GenAI Platform Overview

EXTENDED VIEW



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

# GenAI Platform – Governance and GenAI Ops

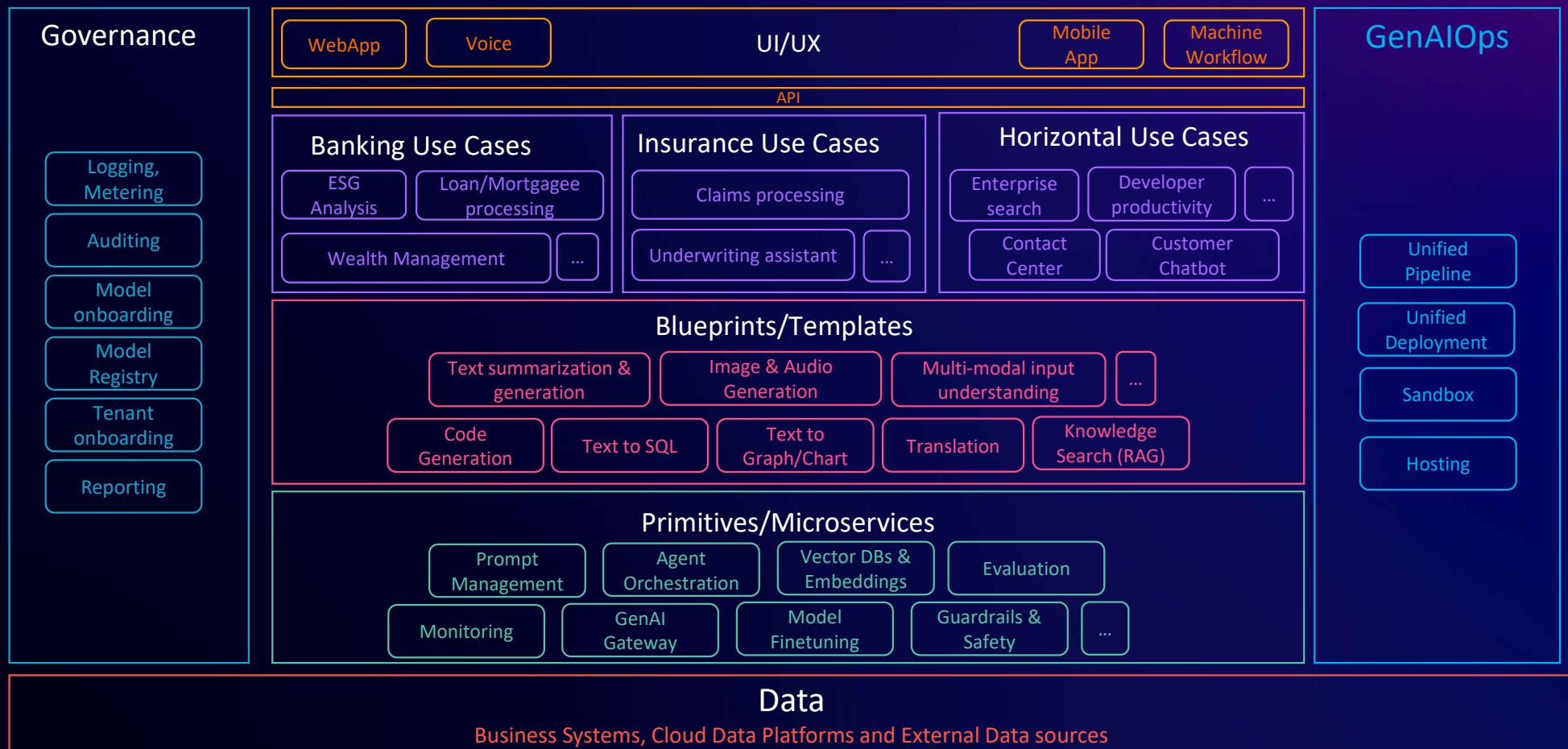
INFRASTRUCTURE AS A CODE FOR MOST COMMON USE CASES PRIMITIVES

Logging, Metering	Model invocations, latency, costs, for oversight
Auditing	Data lineage, data transforms, and visualizations
Model onboarding	Enablement and configuration of the models
Model Registry	Tracking of models in production and experimentation phases and its statutes/owners
Tenant onboarding	Create and manage users, permissions and policies, deploy tenant environment
Reporting	Visualizations and Insights
...	



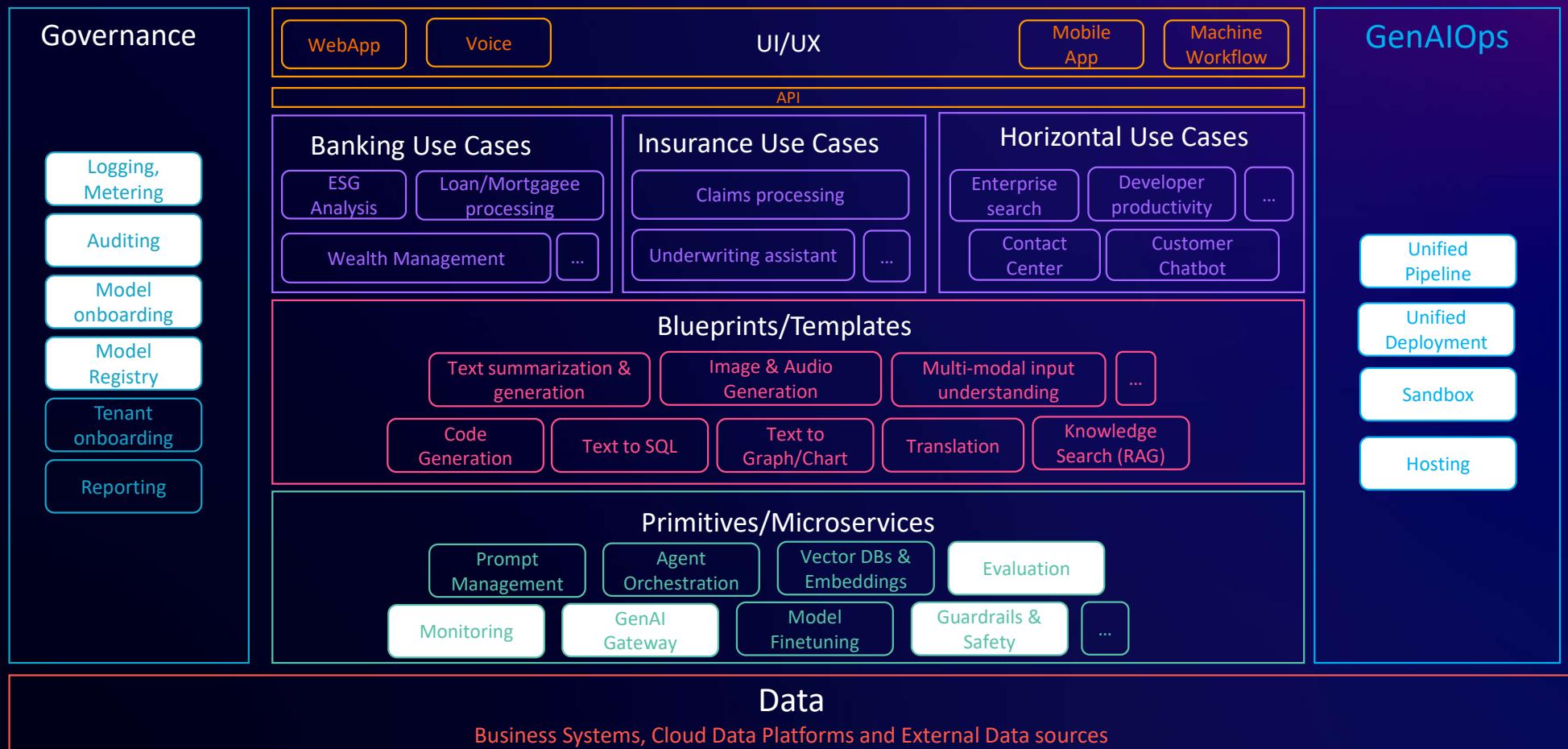
# GenAI Platform Overview

## EXTENDED VIEW



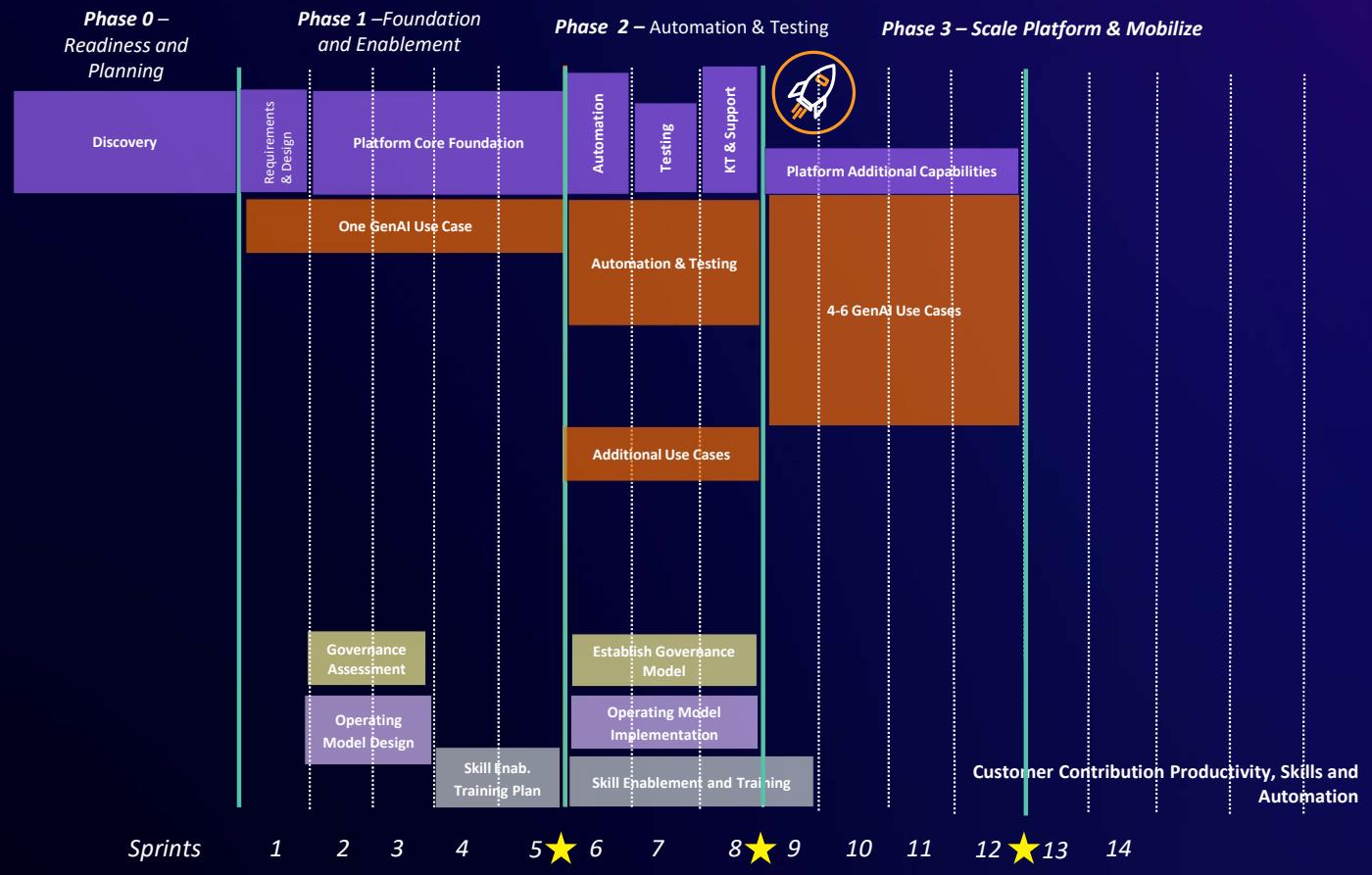
# GenAI Platform Overview – Starter Kit

## EXTENDED VIEW



# Typical Phased Effort Implementation

STREAMLINED & PHASED OUTCOME BASED APPROACH



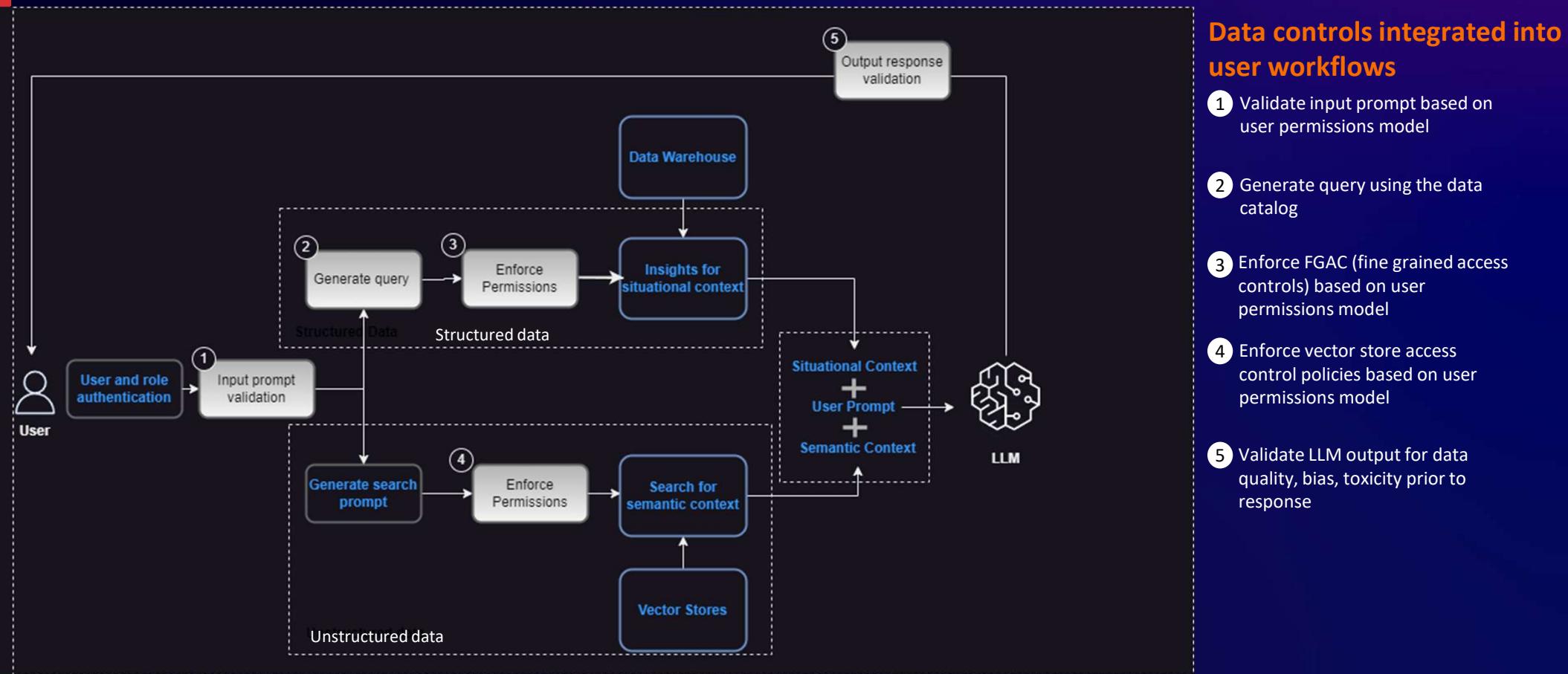
# Guardrails and Security



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

# SECURITY IS JOB NUMBER 1!!!!

## Enforce data permission controls



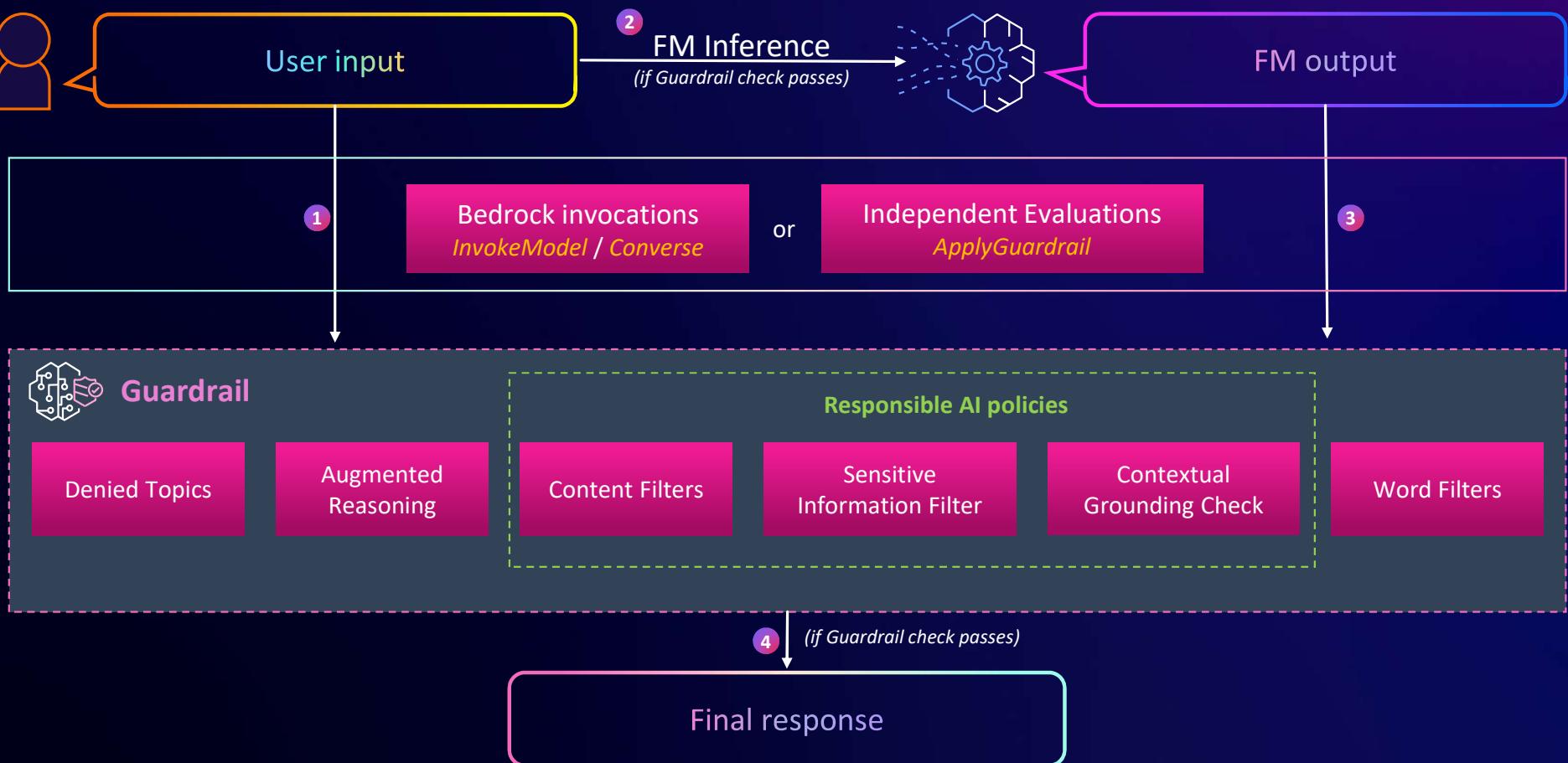
# Guardrails vs Security

- **Security** is access and authorization to the AI platform, and the data (documents, DB, other artifacts)
- **Guardrails** are safety mechanisms and controls designed to ensure AI systems behave responsibly and within acceptable boundaries. They prevent harmful, biased, or unintended outputs

# Types of Guardrails

- **Input Guardrails**
  - Inappropriate or malicious prompts
  - Prompt Injection
- **Output Guardrails**
  - Sensitive or PII
  - Prevent harmful content (violence, hate)
  - Factual accuracy
- **Behavioral**
  - Limit AI to approved functions
  - Require Human approval for critical decisions
  - Enforce compliance rules
- **Technical Implementation**
  - Content Filtering
  - Confidence Thresholds
  - Rate limiting
  - Monitoring and logging

# How Guardrails work



# Content filters

FILTER HARMFUL CONTENT ACROSS CATEGORIES

- Hate
- Insults
- Sexual
- Violence
- Misconduct

Configure harmful categories filters

Use the same harmful categories filters for responses

Enable [?](#)

Enable all

Text

Image

Enable all

Text

Image

Enable all

Text

Image

Enable all

Text

Image

[Reset thresholds](#)

Category	Guardrail action	Set threshold
Hate	Block	None Low Medium High
Insults	Block Prevents content from being processed when this category's policy conditions are met.  Detect (no action) Set up as diagnostic tool without taking action on input or output.	Medium High
Sexual	Block	None Low Medium High
Violence	Block	None Low Medium High
Misconduct	Block	None Low Medium High



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

# Configure Guardrails' actions

For each policy, select between:

- **Block** – For preventing content from being delivered when conditions are met
- or
- **Detect** – For monitoring only, without intervention in the actual content flow

Available for all configurations, including content filters, denied topics, sensitive information filters, word filters and contextual grounding.

The screenshot shows the AWS Guardrails configuration interface. It includes sections for 'Harmful categories' and 'Prompt attacks'. In the 'Harmful categories' section, there are five categories: Hate, Insults, Sexual, Violence, and Misconduct. Each category has a 'Guardrail action' dropdown and a 'Strength' slider. The 'Misconduct' category's 'Guardrail action' dropdown is set to 'Detect (no action)' and its strength is set to 'Medium'. A red box highlights this configuration. In the 'Prompt attacks' section, there is one category: Prompt Attack, with a 'Guardrail action' dropdown set to 'Detect (no action)' and a strength slider set to 'Medium'.

**Harmful categories**  
Enable detection and blocking of harmful user inputs and model responses. Use a higher filter strength to help improve the filtering of harmful content in each category.  
 Configure harmful categories filters

**Filters for prompts**  
 Use the same harmful categories filters for responses

Category	Guardrail action	Strength
Hate	Detect (no action)	None Low Medium High
Insults	Block	None Low Medium High
Sexual	Detect (no action)	None Low Medium High
Violence	Block	None Low Medium High
Misconduct	Detect (no action)	None Low Medium High

**Prompt attacks**  
Enable to detect and block user inputs attempting to override system instructions. To avoid misclassifying system prompts as a prompt attack and ensure that the filters are selectively applied to user inputs, use input tagging.  
 Configure prompt attacks filter

Category	Guardrail action	Strength
Prompt Attack	Detect (no action)	None Low Medium High

# Multi-Modal Toxicity Detection

- ✓ Content filters now supports toxic image detection, in addition to text
- ✓ Enhance user experiences and manage risk by detecting and filtering undesirable content across different media
- ✓ Content moderation for social media, e-commerce and advertising, financial fraud detection, educational and healthcare applications, among other applications
- ✓ Industry-leading, up to 88% harmful multimodal content detection rate

**Harmful categories**

Enable detection and blocking of harmful user inputs and model responses. Use a higher filter strength to help improve the filtering of harmful content in each category.

Configure harmful categories filters

**Filters for prompts**

Use the same harmful categories filters for responses

Enable

Category	Guardrail action	Strength
Hate	Detect (no action)	None → High
Insults	Block	None → High
Sexual	Detect (no action)	None
Violence	Block	None
Misconduct	Block	None

**Prompt attacks**

Enable to detect and block user inputs attempting to override system instructions. To avoid misclassifying filters are selectively applied to user inputs, use input tagging.

Configure prompt attacks filter

Enable

Category	Guardrail action	Strength
Prompt Attack	Block	None

**Guardrail trace**

**Content filters**

Category	Test result	Details
Sexual	No action taken	Text Strength: High Confidence: None
Prompt attack	No action taken	Text Strength: High Confidence: None
Hate	No action taken	Text Strength: High Confidence: None
Insults	No action taken	Text Strength: High Confidence: None
Misconduct	No action taken	Text Strength: High Confidence: None
Violence	Blocked	Text Strength: High Confidence: Medium

**AI Claude 3.5 Son... v2**

Use ApplyGuardrail API

Assess any content using your pre-configured Amazon Bedrock Guardrails, without invoking the foundation models. [Learn more](#)

**Choose source**

Check how a Guardrail performs for prompts and model responses.

Input

Prompts, uploaded text, images, and content to be sent to a model.

Output

Content a model would generate in response to a user prompt.

**Content**

Describe this image.

**Guardrail action**

⚠ Intervened (1 instances) [View trace](#)

**Guardrails Response**

Sorry, the model cannot answer this question. -

**Run**



# Prompt attacks detection

Similar to harmful categories, prompt attacks are detected based on classification confidence

Amazon Bedrock > Guardrails > Create guardrail

Step 1  
Provide guardrail details

Step 2 - optional  
 Configure content filters

Step 3 - optional  
 Add denied topics

Step 4 - optional  
 Add word filters

Step 5 - optional  
 Add sensitive information filters

Step 6 - optional  
 Add contextual grounding check

Step 7  
 Review and create

**Configure content filters - optional**

Configure content filters by adjusting the degree of filtering to detect and block harmful user inputs and model responses that violate your usage policies.

**Harmful categories**

Enable to detect and block harmful user inputs and model responses. Use a higher filter strength to increase the likelihood of filtering harmful content in a given category.

Enable harmful categories filters

**Prompt attacks**

Enable to detect and block user inputs attempting to override system instructions. To avoid misclassifying system prompts as a prompt attack and ensure that the filters are selectively applied to user inputs, use input tagging.

Enable prompt attacks filter

Prompt Attack 

**Note:** If you are using InvokeModel or InvokeModelResponseStream for model inference, use input tags to apply prompt attack filtering on user inputs. For Converse and ConverseStream APIs, input tags are not required.

[Cancel](#) [Skip to Review and create](#) [Previous](#) [Next](#)

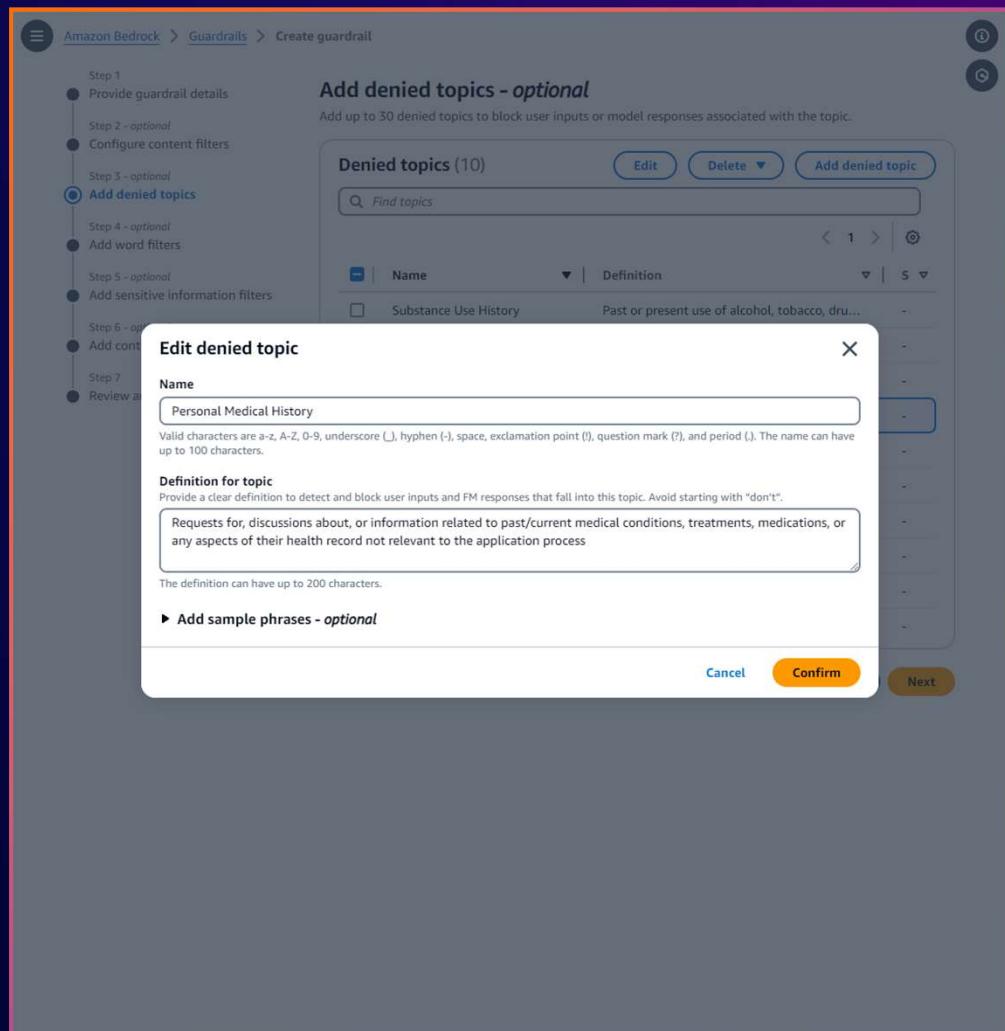


# Denied topics

Topics are defined in simple language and compared against user queries/requests to determine similarity

## Examples:

- **Substance use history**—past or present use of alcohol, tobacco, drugs, or medications outside the scope defined in the application process
- **Financial information**—income, assets, debts, credit score, or financial details not directly relevant to the insurance product applied for



# Guardrails for Amazon Bedrock – Denied Topic

Name  
preview-test-sports

Valid characters are a-z, A-Z, 0-9, underscore (\_), hyphen (-), space, exclamation point (!), question mark (?), and period (.). The name can have up to 100 characters.

**Definition for topic**  
Provide a clear definition of the topic. This definition will be used to classify an input or a response to a topic.

This topic refers to anything to deal with sports, and sporting events. This includes an activity involving physical exertion and skill in which an individual or team competes against another or others for entertainment.

The definition can have up to 1000 characters.

**Example phrases**  
Representative phrases that refer to the topic. These phrases can represent a user input or a model response. Add up to 5 examples.  
An example phrase can have up to 1000 characters.

Who won the basketball game? X

What is the size of a football field X

The olympics X

**AI Claude v2.1 | ODT**

**Prompt**  
When is the AFC championship playoff game?

**Model response**  
The AFC Championship Game is scheduled for Sunday, January 29, 2023. It will feature the winner of the Chiefs-Jaguars divisional round game against the winner of the Bills-Bengals divisional round game. The game will determine the AFC's representative in Super Bowl LVII.

**Final response**  
Sorry, this input was blocked by your custom guardrail.

**Guardrail check**  
✖ 1 Failed [View trace >](#)

**AI Claude v2.1 | ODT**

**Prompt**  
When do the Dolphins play?

**Model response**  
Unfortunately, I do not have information on the Miami Dolphins' upcoming game schedule. As an AI assistant without direct access to sports schedules, I don't have specifics on when particular teams are playing. I'd suggest checking the Dolphins' website or sports news sites for their latest game details.

**Final response**  
Sorry, this input was blocked by your custom guardrail.

**Guardrail check**  
✖ 1 Failed [View trace >](#)

# Guardrails for Amazon Bedrock – Denied Topic

Name

Valid characters are a-z, A-Z, 0-9, underscore (\_), hyphen (-), space, exclamation point (!), question mark (?), and period (.). The name can have up to 100 characters.

Definition for topic

Provide a clear definition of the topic. This definition will be used to classify an input or a response to a topic.

This topic refers to anything to deal with sports, and sporting events. This includes an activity involving physical exertion and skill in which an individual or team competes against another or others for entertainment.

The definition can have up to 1000 characters.

Example phrases

Representative phrases that refer to the topic. These phrases can represent a user input or a model response. Add up to 5 examples.

An example phrase can have up to 1000 characters.

Who won the basketball game? X

What is the size of a football field X

The olympics X

**AI Claude v2.1 | ODT** [Change](#)

**Prompt**

You are an aquarium help bot. Answer the sample guest questions with generative, believable data.

When do the dolphins play ?

**Final response**

The dolphin shows are scheduled twice daily, at 11am and 3pm. Our talented dolphin pod puts on an amazing display of jumps, spins, and clever tricks. Make sure not to miss these fun, family-friendly shows that are included with aquarium admission!

**Guardrail check**

✓ Passed [View trace >](#)

# Sensitive information filter

PROTECT YOUR SENSITIVE DATA

- ✓ Detect, redact or filter PII in FM responses to protect user privacy
- ✓ Detect, redact or filter PII in user inputs
- ✓ Select from a variety of pre-configured PII types, based on application requirements
- ✓ Or define your own sensitive information, by using regular expressions (regex)
- ✓ Apply actions for prompts and/or responses independently

NEW!

Add new PII

PII type

Choose PII type

General

- Phone
- Age
- Username
- Password
- Driver ID
- License plate
- Vehicle Identification number (VIN)

Finance

- CVV
- Credit/Debit card expiry
- Credit/Debit card number
- Personal identification number (PIN)
- International Bank Account Number
- SWIFT code

IT

- IPv4 address
- Media access control (MAC) address
- Web address
- AWS access key
- AWS secret key

to users

Confirm

PII types (3)		
	Input action	Output action
<input type="checkbox"/> Choose PII type	Detect (no action)	Mask
<input type="checkbox"/> NAME	Mask	Mask
<input type="checkbox"/> EMAIL	Block	Block
<input type="checkbox"/> ADDRESS		

Add regex pattern

Name

Label to identify the pattern. Shown as an identifier if PII is masked, e.g. [BOOKING\_ID].

Example - Booking ID

Regex name can have up to 100 characters.

Regex pattern

Example - ^ID\d{3}[A-Z]{3}\$

**Input**

Enable

Input action

Choose what action the guardrail should take on user inputs before they reach the model

Block

**Output**

Enable

Output action

Choose what action the guardrail should take on model outputs before displayed to users

Block

▶ Add description - optional

Cancel Confirm



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

# Word filters

## BLOCK SPECIFIC WORDS OR PHRASES

- Filter profane words
- Define a set of custom words to block in user input and FM responses

The screenshot shows the AWS Guardrail Word Filters configuration interface. It includes three main sections: 'Filter profanity' (with a checked checkbox), 'Add custom words and phrases' (with three options: 'Add words and phrases manually' selected, 'Upload from a local file', and 'Upload from S3 object'), and 'View and edit words and phrases (0)' (which is currently empty).

**Filter profanity**

Filter profanity  
Enable this feature to block profane words in user inputs and model responses. The list of words is based on conventional definitions of profanity and is subject to change.

**Add custom words and phrases** Info

Specify up to 10,000 words or phrases (up to 3 words each) to be blocked by the guardrail. A blocked message will show if user input or model responses contain these words or phrases.

Add words and phrases manually  
Manually add words and phrases to the following table.

Upload from a local file  
Populate the following table with words and phrases from a .txt or .csv file from your computer.

Upload from S3 object  
Populate the following table with words and phrases from an S3 object.

**View and edit words and phrases (0)**

Find words and phrases Show all ▾ Delete all ▾ < 1 >  Action

No words or phrases added  
Upload from file or add manually in the console

**Add a word or phrase** ▼

# Contextual Grounding checks & relevance

REDUCE HALLUCINATION BY FILTERING UNGROUNDED AND IRRELEVANT RESPONSES

- Filter hallucinations in RAG and summarization applications
- Check response accuracy based on your enterprise data
- Check if the responses are relevant to user's query or instruction

**Grounding threshold :** Based on the information in your RAG response and the FM answer, Grounding evaluates the correctness of the answer

**Relevance threshold:** Evaluate the FM response and the user query. If the confidence that the response answers the query is below the threshold, a blocked message is returned to the user

The screenshot shows the 'Create guardrail' wizard in the Amazon Bedrock console. The current step is 'Step 6 - optional: Add contextual grounding check'. A sub-section titled 'Grounding' is visible, which includes a description of validating model responses against a reference source and a slider for setting the 'Grounding score threshold' between 0 and 0.99, with 0.7 selected. Below this is another section titled 'Relevance' with a similar threshold slider set at 0.7. Navigation buttons 'Cancel', 'Previous', and 'Next' are at the bottom.

model hallucination.

source, and block responses

use has a lower score than the defined threshold, the more responses. [View examples](#)

should of relevance.

ore than the defined threshold, the yes. [View examples](#)



Generative example of FAQ serving as reference materials

## Enterprise Acquisition Portal

Home Resources FAQ Account

### Quick Navigation

- Multiple Award Schedules
- Technology Products
- Transportation
- Professional Services
- Vehicle Acquisition
- Travel Services
- Supply Chain

Home > Resources > Frequently Asked Questions

Last Updated: October 15, 2023 | Version 2.3.4

Search FAQs...

## Federal Acquisition Resource Portal FAQ

Welcome to the comprehensive FAQ section of the Federal Acquisition Resource Portal. This resource provides authoritative guidance on procurement procedures, policies, and best practices for federal acquisition professionals. Browse by category or use the search function to find specific information.

**Note:** Information is regularly updated to reflect current policies and procedures.

### Multiple Award Schedules (MAS)

Updated: October 1, 2023

What is a Multiple Award Schedule?

A Multiple Award Schedule is a long-term contract that allows government agencies to purchase products and services at pre-negotiated prices from approved vendors.

### Technology Products and Services

Updated: October 10, 2023

# RAG ChatBot based on FAQ

What's the minimum sales requirement to keep a contract active?



Can you help me understand the total approval process for an IT purchase of \$250,000?



What's the best pizza topping for a company party?



# Implementation #1: Integrated guardrail

SYSTEM PROMPT: { }

REFERENCE MATERIALS:

{

.... (entire FAQ) ...

A: The typical timeline for MAS approval is 120-180 days, which includes a 60-day initial review period, 30 days for clarifications, and 30-90 days for final processing and system setup.

...  
}



Can I get expedited approvals done in 2 months?



[Blocked]

Your response has been blocked due to insufficient confidence in the model.

# Implementation #2: Augmentation

SYSTEM PROMPT: { }

REFERENCE MATERIALS:

{

.... (entire FAQ) ...

A: The typical timeline for MAS approval is 120-180 days, which includes a 60-day initial review period, 30 days for clarifications, and 30-90 days for final processing and system setup.

...  
}



Can I get expedited approvals done in 2 months?



1. **\*\*Initial Review Period (60 days)\*\*:** ...
  2. **\*\*Clarifications (30 days)\*\*:** ...
  3. **\*\*Final Processing and System Setup (30-90 days)\*\*:** ...
- ### Challenges in Expediting the Process:...
- ### Tips to Potentially Speed Up the Process:...

While it is unlikely to reduce the approval timeline to just 2 months, being well-prepared, thorough, and responsive can help ensure that the process proceeds as smoothly and quickly as possible within the standard timeframe.

Grounding score:

0.04

# Implementation #2: Augmentation (w/ system prompt)

**SYSTEM PROMPT:** { Using only the references provided. Do not elaborate. Just answer the question based on the references or say you do not have enough information" }

## REFERENCE MATERIALS:

{  
.... (entire FAQ) ...  
A: The typical timeline for MAS approval is 120-180 days, which includes a 60-day initial review period, 30 days for clarifications, and 30-90 days for final processing and system setup.  
...  
}



Can I get expedited approvals done in 2 months?



The provided reference does not mention the possibility of expedited approvals within 2 months. The typical timeline is 120-180 days.

Grounding score: 0.99

# Implementation #3: Grounding score as a metric

QUESTION 1 OF 20: "CAN I GET EXPEDITED APPROVALS DONE IN 2 MONTHS?"

Model	Original		Edited	
	Grounding	Relevance	Grounding	Relevance
Haiku	0.02	1.00	0.94	1.00
3 Sonnet	0.02	1.00	0.99	1.00
Opus	0.03	1.00	0.99	1.00
3.5 Sonnet	0.85	1.00	0.98	1.00
3.5 Sonnet v2	0.75	1.00	0.97	1.00
Nova Pro	0.04	1.00	0.99	1.00
Lite	0.30	1.00	0.91	1.00
Llama 3.2 11B	0.06	1.00	0.85	.98
Llama 3.2 90B	0.22	1.00	1.00	.99

**Edited includes system prompt:**  
**SYSTEM PROMPT:** { Using only the references provided. Do not elaborate. Just answer the question based on the references or say you do not have enough information" }

# Guardrails

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

# Create a Bedrock model with guardrail configuration
bedrock_model = BedrockModel(
    model_id="anthropic.claude-3-5-sonnet-20241022-v2:0",
    guardrail_id="your-guardrail-id",          # Your Bedrock guardrail ID
    guardrail_version="1",                    # Guardrail version
    guardrail_trace="enabled",               # Enable trace info for debugging
)

# Create agent with the guardrail-protected model
agent = Agent(
    system_prompt="You are a helpful assistant.",
    model=bedrock_model,
)

# Use the protected agent for conversations
response = agent("Tell me about financial planning.")

# Handle potential guardrail interventions
if response.stop_reason == "guardrail_intervened":
    print("Content was blocked by guardrails, conversation context overwritten!")

print(f"Conversation: {json.dumps(agent.messages, indent=4)}")
```

