AI Conclave

Online

AIOT103

Exploring agents with Amazon Bedrock

Sahil Verma

Senior GTM Specialist Solutions Architect,
Generative AI
AWS India

Agenda

- Amazon Bedrock
- Introduction to agents
- Building agents with Amazon Bedrock
- Deep dive on the agent’s capabilities, solution and patterns
- Demo – Build an agent from scratch
- Resources

Amazon Bedrock

The easiest way to build and scale generative AI applications with powerful tools and foundation models

Choice of leading FMs through a single API

Model import, distillation and fine-tuning

Generative AI Tools – Knowledge Bases (RAG), Guardrails, Flows, and Agents

Security, privacy, and data governance

2025 – The year of agents

What is an **AI Agent?**

Intelligent, autonomous systems

Plan, reason, and act

Access to enterprise data

Ability to use tools

Momentum behind Amazon Bedrock Agents

Investment & medical research

Marketing assistants

Insurance claims processing

Root cause analysis

Customer experience

What generative AI customers are asking for

Help me automate complex workflows

Help me move faster

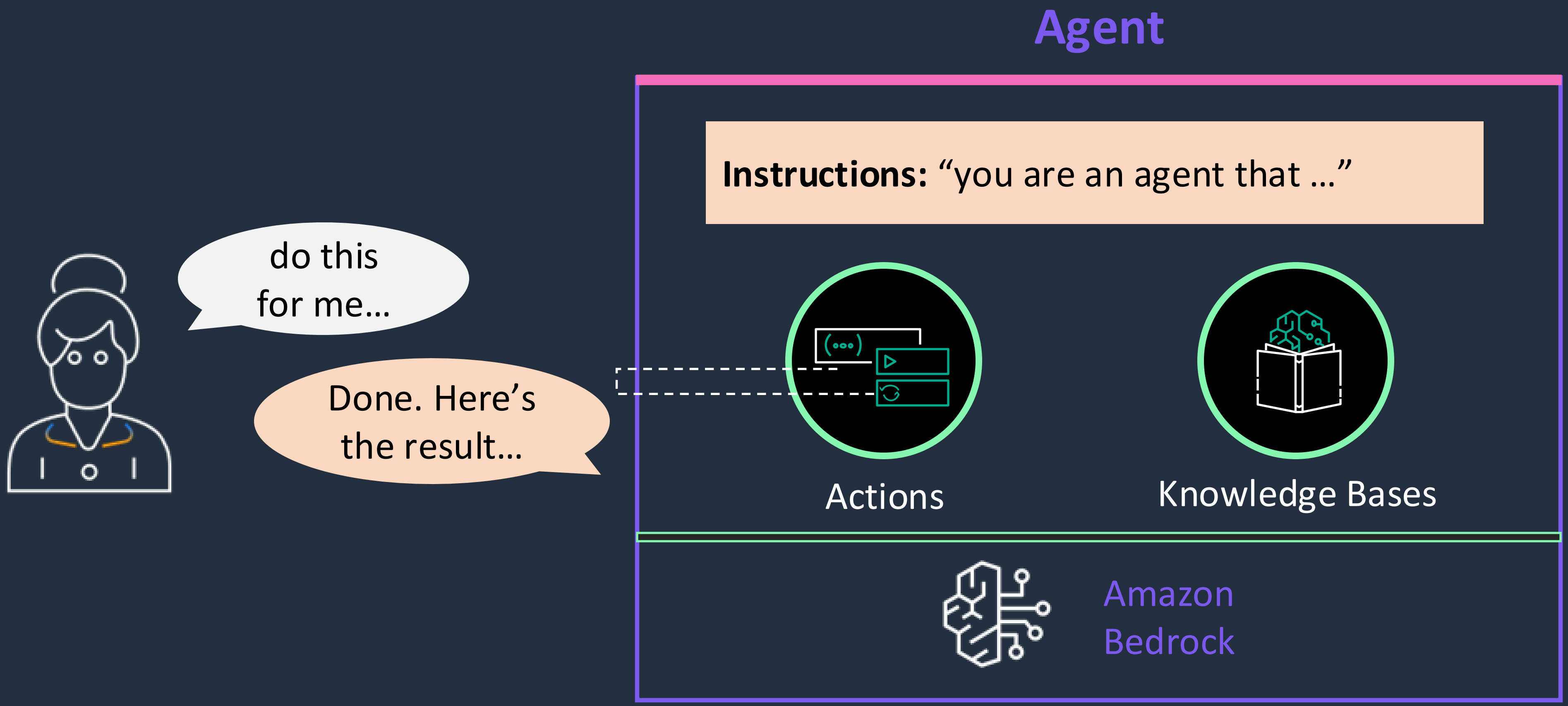
Help me find more robust and scalable solutions

Amazon Bedrock Agents

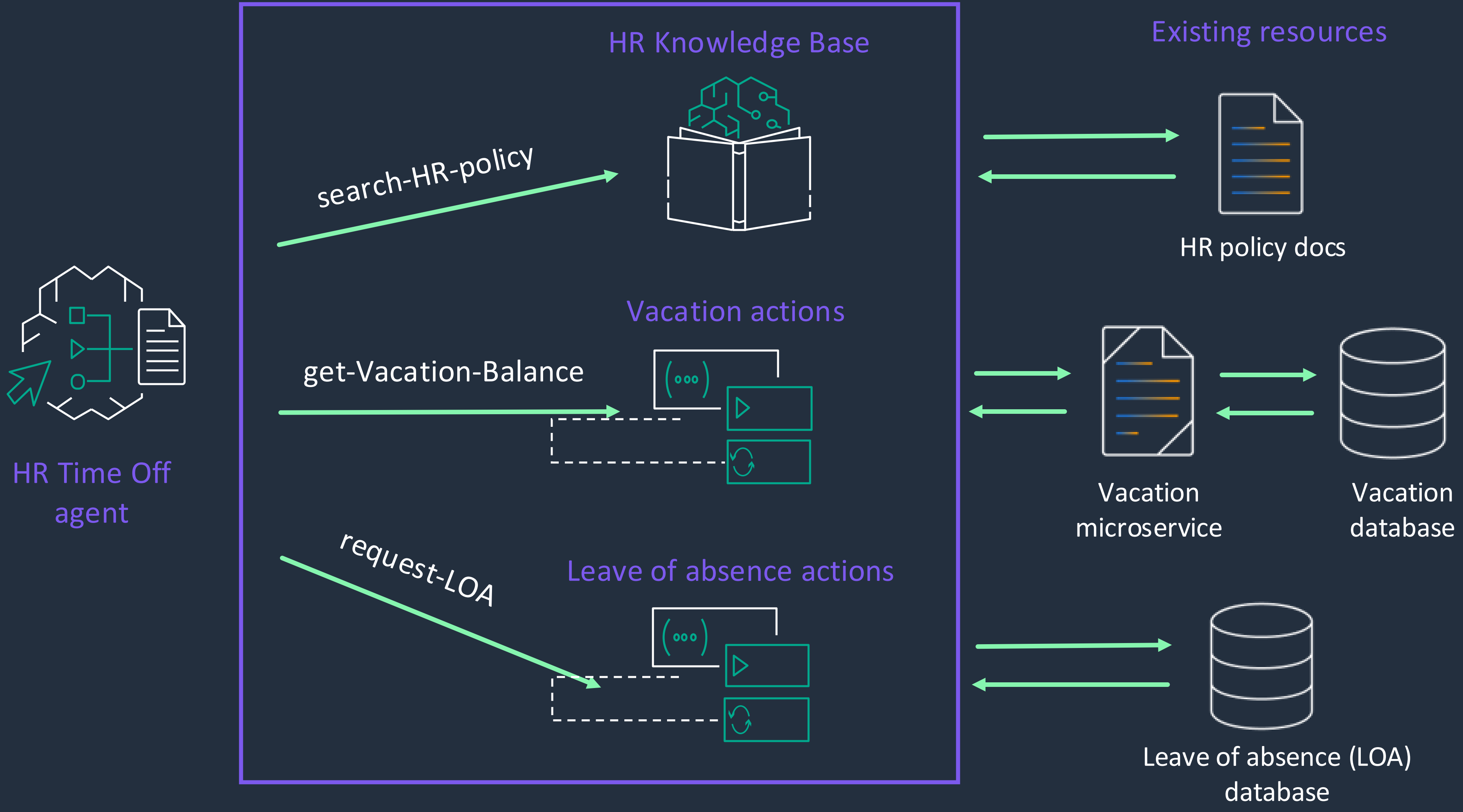
Agentic building blocks

- Choice of foundation models
- Memory, Knowledge Bases, Guardrails
- Tools and action groups
- Trace, debug, and observability
- Multi-agent collaboration NEW

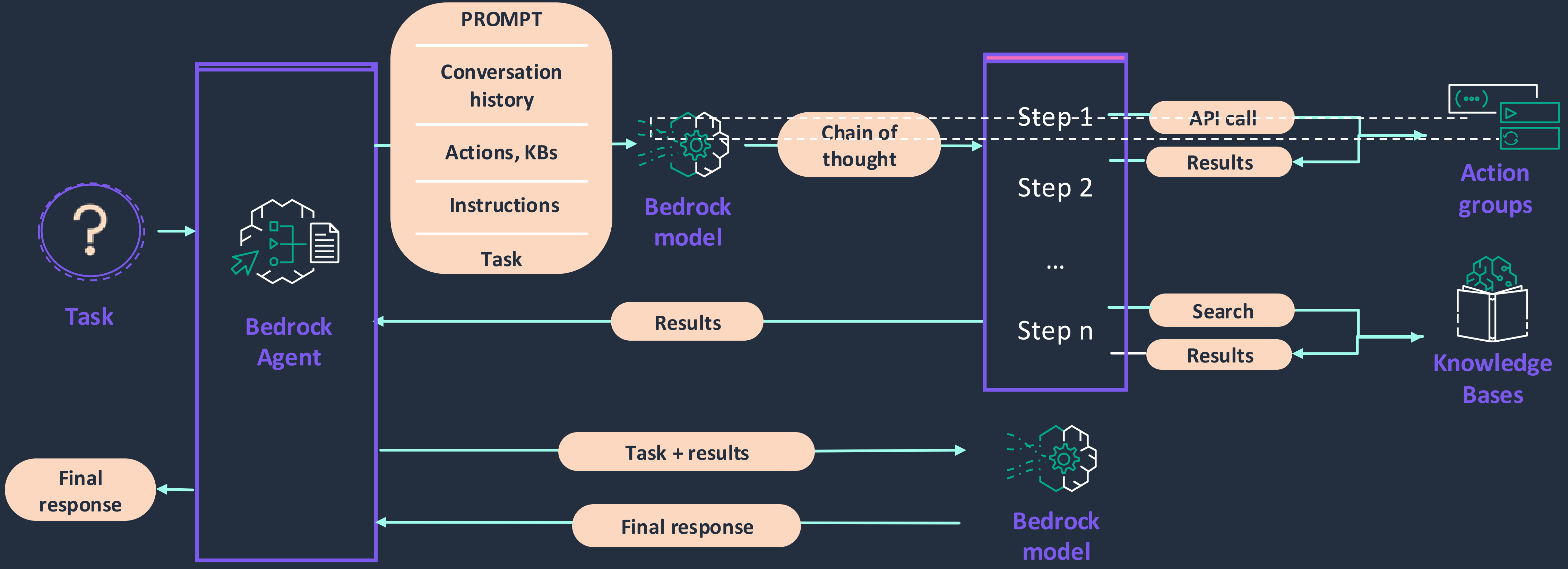
Agent basics



Example - HR Time Off Agent



Agent orchestration - ReAct



Agent breaks task into subtasks, determines the right sequence, and executes actions and knowledge searches on the fly

Custom orchestration

GRANULAR CONTROL OVER TASK PLANNING, COMPLETION, AND VERIFICATION

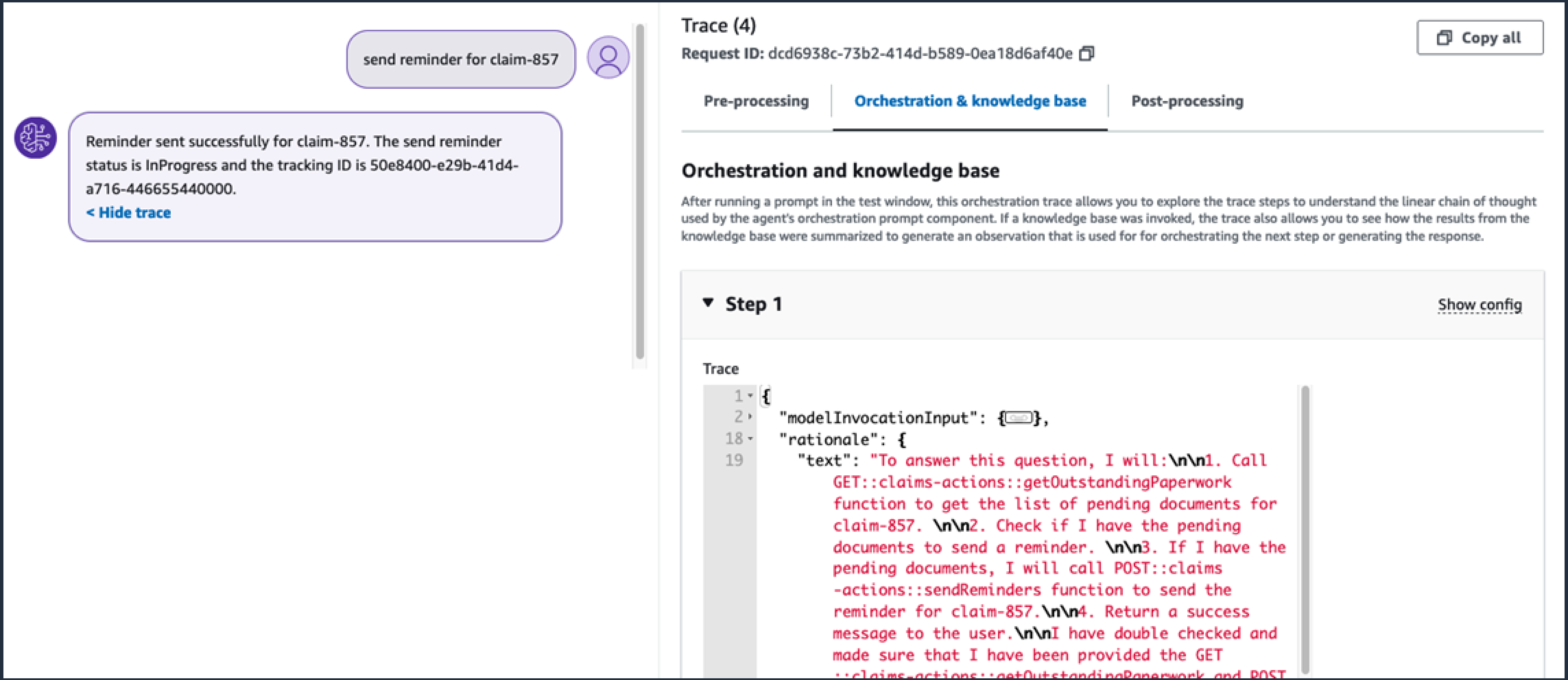
Full control over orchestration strategy

Real-time adjustments

Reusable across use cases

Other orchestration strategies : Plan and Solve, Reason without Observation, Tree of Thought, and Standard Operating Procedures (SOP)

Agent orchestration is transparent – Trace



send reminder for claim-857

Reminder sent successfully for claim-857. The send reminder status is InProgress and the tracking ID is 50e8400-e29b-41d4-a716-446655440000.

< Hide trace

Trace (4)

Request ID: dcd6938c-73b2-414d-b589-0ea18d6af40e

Orchestration and knowledge base

After running a prompt in the test window, this orchestration trace allows you to explore the trace steps to understand the linear chain of thought used by the agent's orchestration prompt component. If a knowledge base was invoked, the trace also allows you to see how the results from the knowledge base were summarized to generate an observation that is used for for orchestrating the next step or generating the response.

▼ **Step 1**

Trace

```
{
  "modelInvocationInput": {…},
  "rationale": {
    "text": "To answer this question, I will:\n\n1. Call
        GET::claims-actions::getOutstandingPaperwork
        function to get the list of pending documents for
        claim-857. \n\n2. Check if I have the pending
        documents to send a reminder. \n\n3. If I have the
        pending documents, I will call POST::claims
        -actions::sendReminders function to send the
        reminder for claim-857.\n\n4. Return a success
        message to the user.\n\nI have double checked and
        made sure that I have been provided the GET
        ::claims-actions::getOutstandingPaperwork and POST
```

Detailed orchestration trace in the console and from the SDK

Each action group has 3 key elements

Action group description

Overview of actions provided – helps agent know when this action group is relevant

API schema

- Rich definition of each action
- Operation name, input parameters, data types, response details
- Helps agents know when to use it, how to call it, and how to use results
- Language agnostic API definition using industry-standard schema

Lambda function

- Implementation of each action
- Contains either business logic or wraps microservices, databases, or tools
- Serverless, scalable, secure
- Choice of programming language (Python, C#, JavaScript, Java, …)

Action group example

UtilityActionGroup

Description: “This action group provides a set of commonly used actions. Use these actions for things like sending emails and getting team member lists.”

API schema

```
{ ”openapi”: “3.0.0”,
  ”info”: { “title”: “Utility Actions”,
            “description”: “…” },
  ”paths”: {
     “/sendEmail”: {
       “post”: {
          ”description”: “This operation …”
          ”operationId”: “sendEmail”,
          ”requestBody”: { … },
          “responses”: {
             “200”: {
                “text/plain”: {…}
     “/getTeam”: { … }
  }
}
```

Lambda function

```
def lambda_handler(event, context):
   if event[‘apiPath’] == ‘/sendEmail’:
      result = sendEmail(event)
   elif event[‘apiPath’] == ‘/getTeam’:
      result = getTeam(event)

   response_body = {'application/json':
            {'body': result}}
   action_response = { …
            ‘responseBody’: response_body

   return {‘messageVersion’: ‘1.0’,
           ‘response’: action_response}
```

Code interpretation for Amazon Bedrock Agents

Allow agents to generate and execute code to perform complex analysis

Enable agents to generate and execute code to answer questions and solve problems

Automatically generate charts and analysis using generated code

Analyze files automatically with generated code, including CSV, XLS, JSON, DOC, HTML, TXT, and PDF

Run code in an isolated environment, with built-in guardrails enabled

Memory retention for Amazon Bedrock Agents

Enable agents to keep and use summaries of prior interactions over time

Build agents that learn from previous interactions for more seamless conversations over time

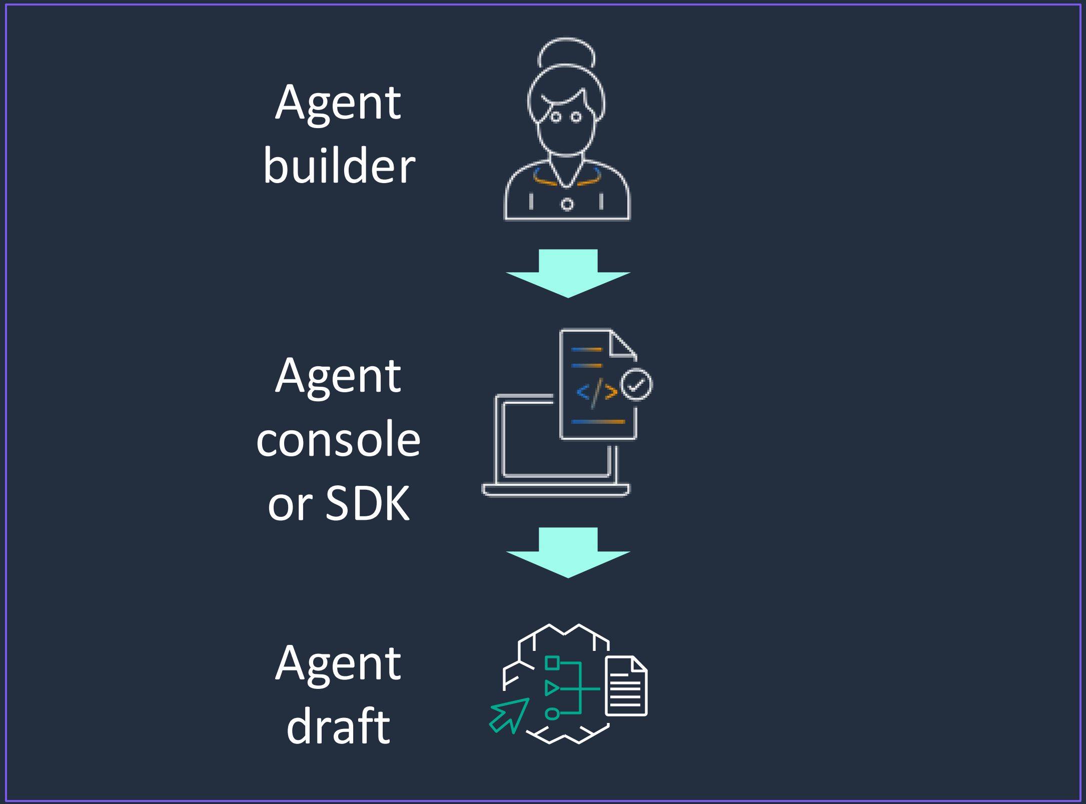
Enable more personalized experiences

Use unique memory IDs for separation between users

Retain the summaries for up to 30 days

Agents can be deployed and invoked from any app

Building and testing agents

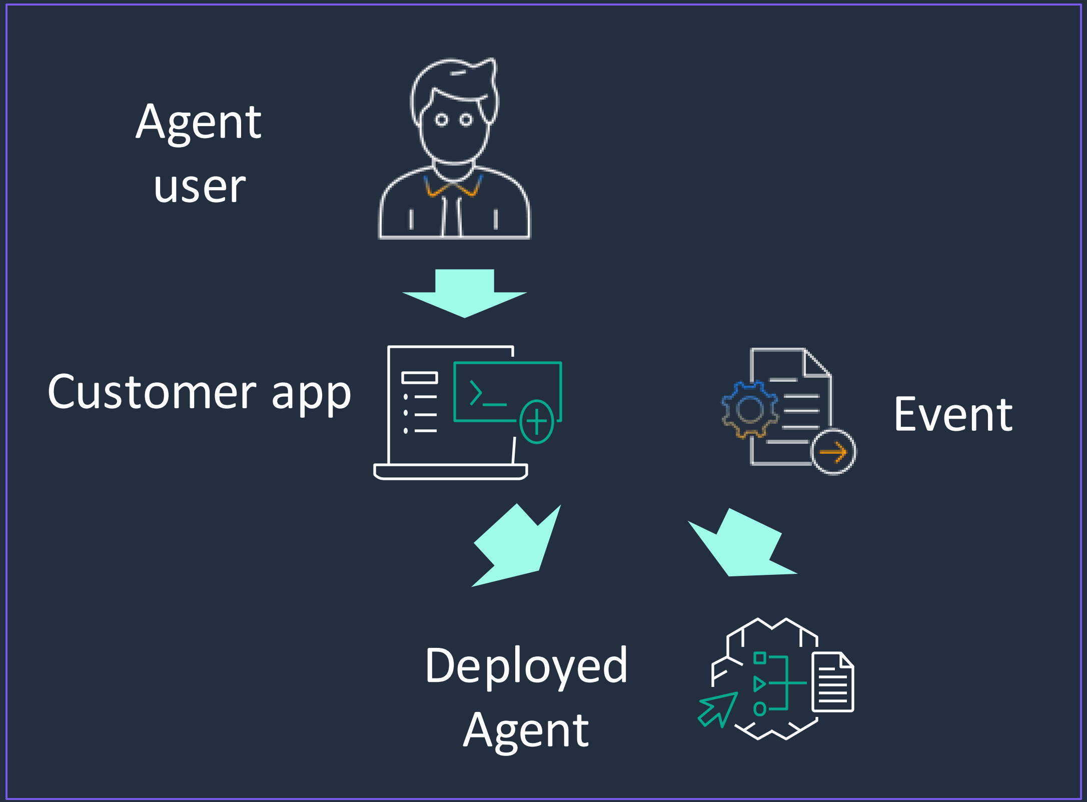


Agent builder

Agent console or SDK

Agent draft

Using production agents



Agent user

Customer app

Event

Deployed Agent

To deploy an agent, you create a new alias, and optionally a new version

Invoking an Agent via the SDK

Invoke agent

```
response = client.invoke_agent(
    inputText=‘<user request>',
    agentId=agent_id,
    agentAliasId=agent_alias_id,
    sessionId=str(uuid.uuid1()),
    enableTrace=True )
```

Process the response stream

```
for event in response['completion']:
    if 'chunk' in event:
        data = event['chunk']['bytes’]
        answer = data.decode('utf8')
        print(f”Answer:\n{answer}")
    elif 'trace' in event:
        print(json.dumps(event['trace’],
                    indent=2))
```

Inline agents

DYNAMICALLY CONFIGURE AGENTS AT RUNTIME

Dynamic agent configuration

Modify agent's instructions, action groups, knowledge bases and other parameters on the fly

Flexible integration

Easily incorporate external APIs and other tooling as needed for each interaction

Contextual adaptation

Adjust the agent's responses based on the user role, preferences or specific scenarios

Amazon Bedrock Guardrails

Apply safeguards customized to your gen AI application requirements and responsible AI policies

Configure thresholds to filter undesirable and potentially harmful text

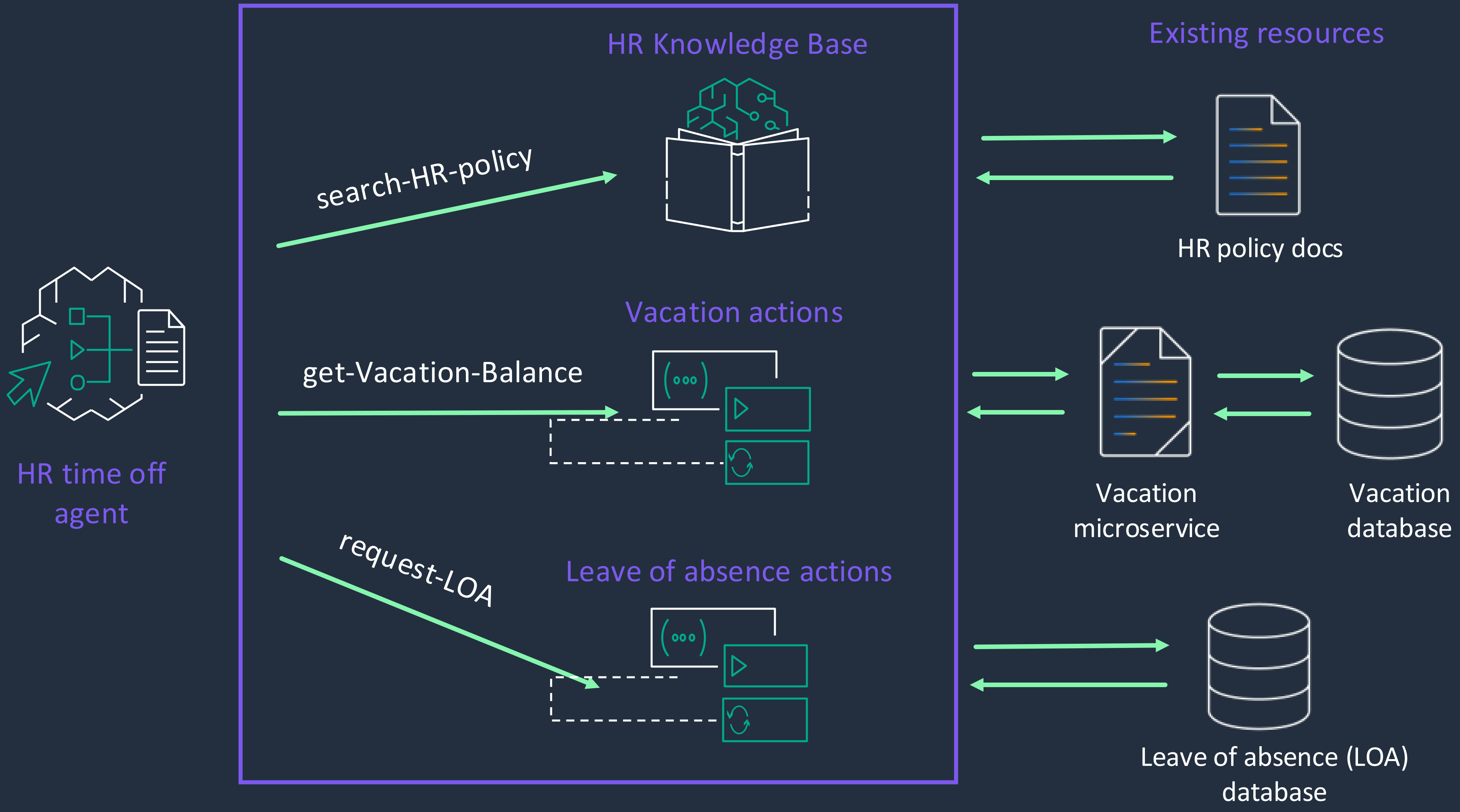
Define and disallow denied topics with short natural language descriptions

Remove personally identifiable information (PII) and sensitive information in gen AI apps

Define a set of words to detect and block in user inputs and model responses

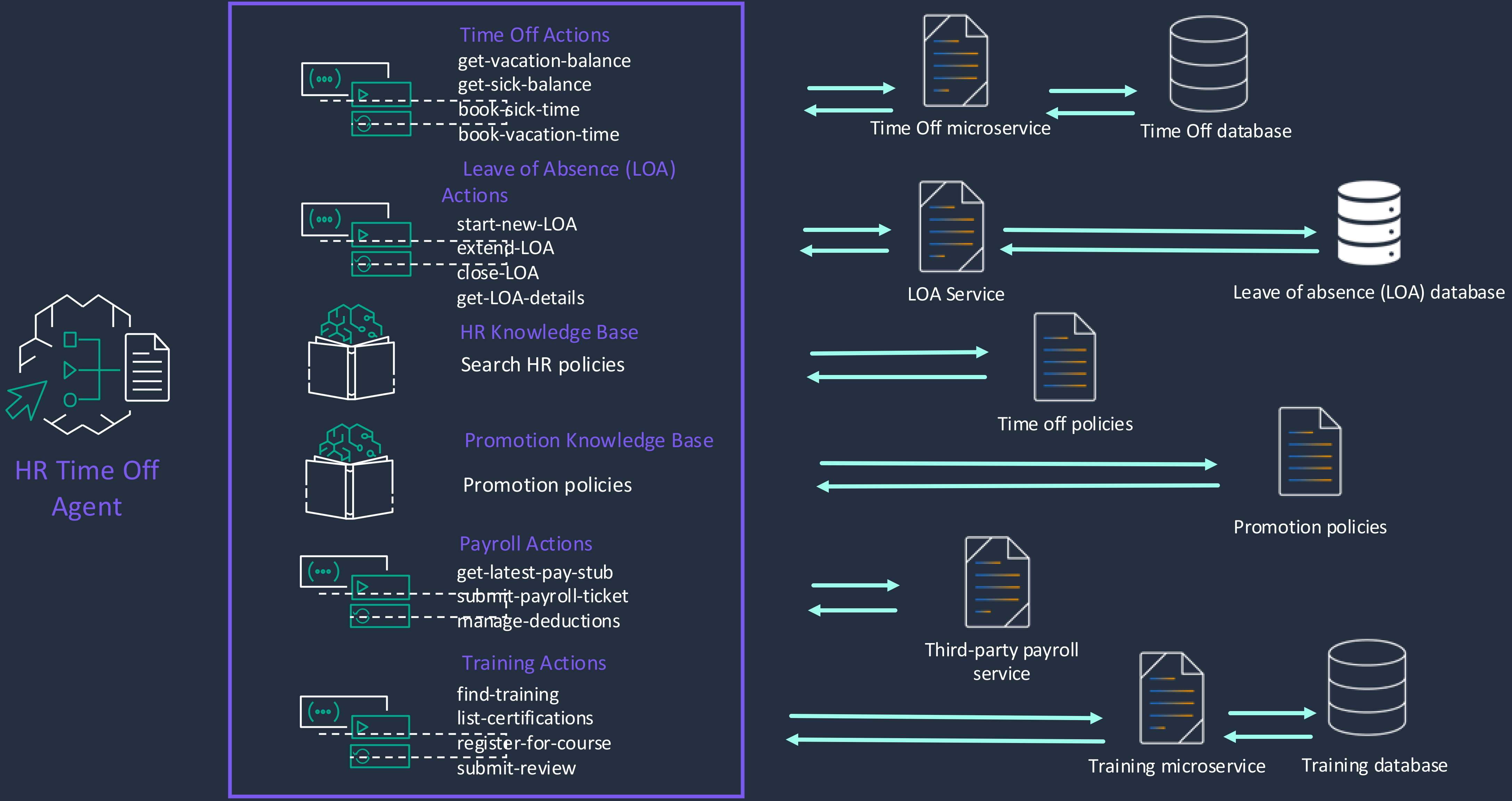
Filter hallucinations by detecting grounded-ness and relevance of model responses based on context

Example - HR time off agent



. . . but if you take a SINGLE agent too far

Version N



HR Time Off Agent

Time Off Actions

get-vacation-balance
get-sick-balance
book-sick-time
book-vacation-time

Leave of Absence (LOA) Actions

start-new-LOA
extend-LOA
close-LOA
get-LOA-details

HR Knowledge Base

Search HR policies

Promotion Knowledge Base

Promotion policies

Payroll Actions

get-latest-pay-stub
submit-payroll-ticket
manage-deductions

Training Actions

find-training
list-certifications
register-for-course
submit-review

Time Off microservice

Time Off database

LOA Service

Leave of absence (LOA) database

Time off policies

Promotion policies

Third-party payroll service

Training microservice

Training database

. . . it leads to challenges

Coding gets complicated

- Complex prompts to limit hallucinations
- Fragile, hard to maintain

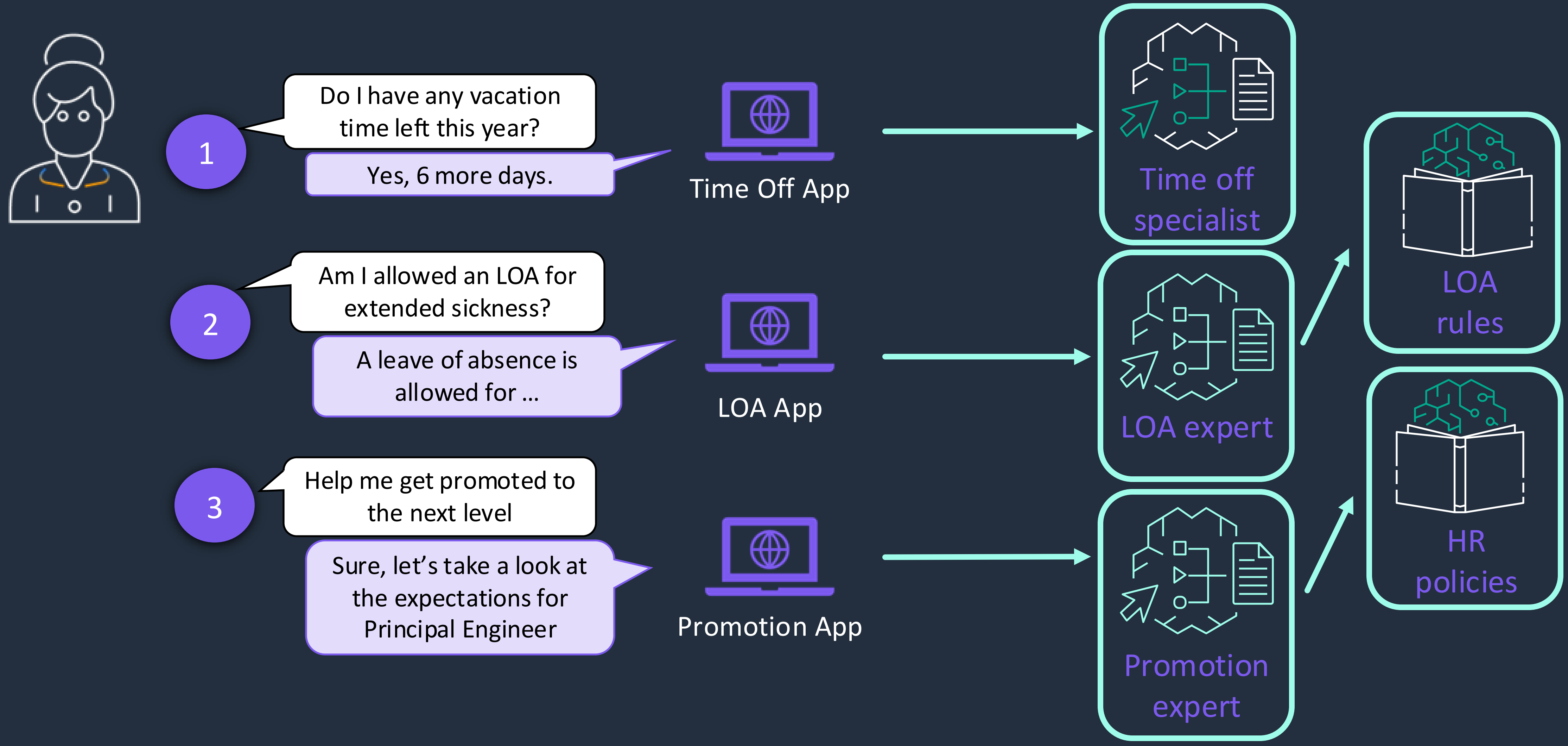
Agent gets confused

- Calling wrong tools
- Passing wrong arguments
- Inconsistent responses

Agent gets slower and more expensive

- Frontier models needed
- Prompt sizes grow
- Agents retry steps

Using multiple agents helps . . .


1
Do I have any vacation time left this year?
Yes, 6 more days.
Time Off App
Time off specialist
2
Am I allowed an LOA for extended sickness?
A leave of absence is allowed for …
LOA App
LOA expert
LOA rules
3
Help me get promoted to the next level
Sure, let’s take a look at the expectations for Principal Engineer
Promotion App
Promotion expert
HR policies

Amazon Bedrock Agents multi-agent collaboration

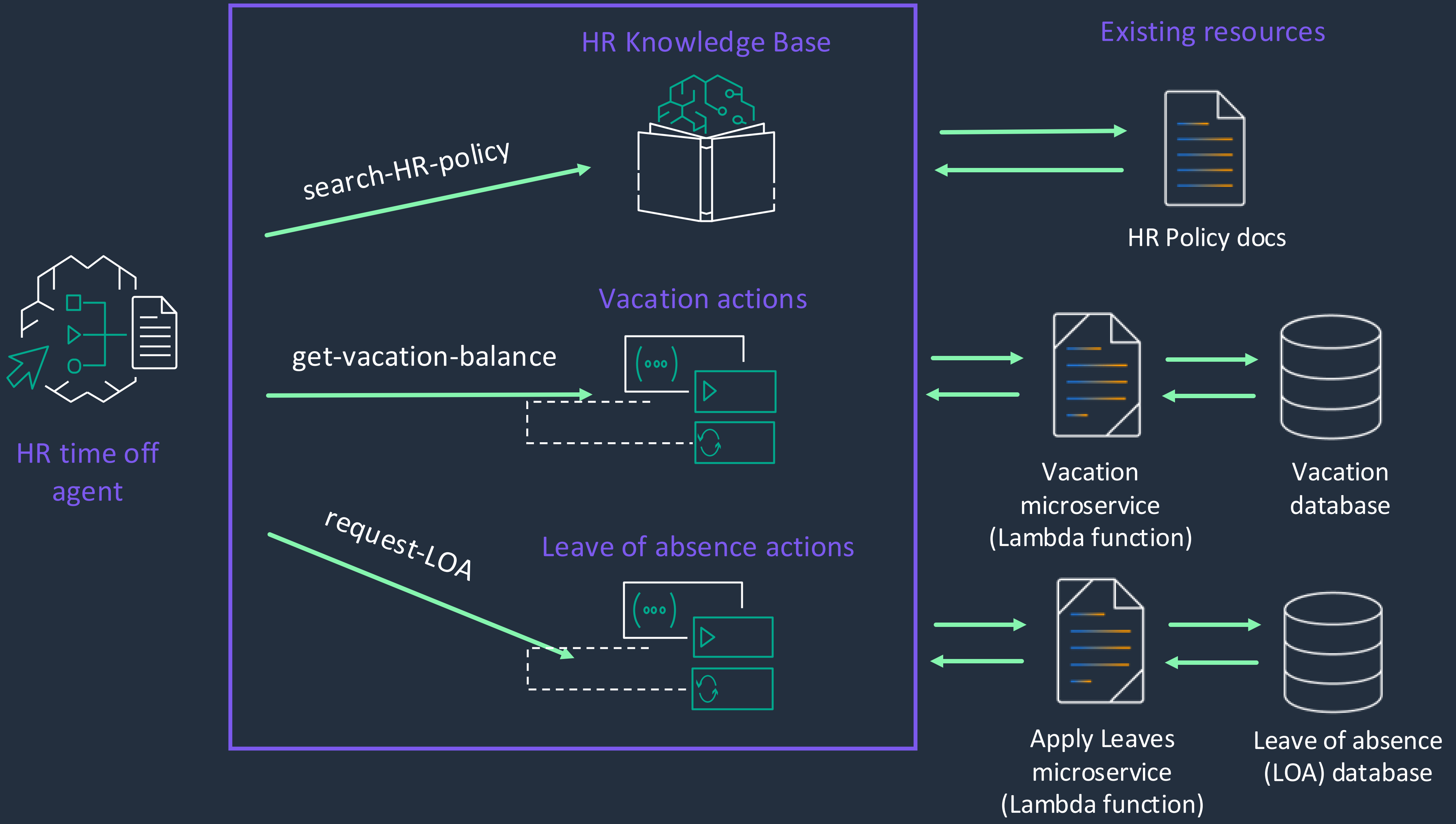
Scaling agentic experiences

Preview

- Easily assemble agents and knowledge bases
- Plan and execute complex tasks across agents
- Unify conversations across agents with built-in intent classification
- Observability across multi-agent flows
- Guardrails, security, and privacy

Demo

Demo - HR time off agent



Recap and summary

AGENTS ON AMAZON BEDROCK

- Amazon Bedrock and agents
- Agent fundamentals
- Building Amazon Bedrock Agents
- Core capabilities:like custom orchestration, code interpretation, memory retention, guardrails, inline agents, etc.
- Multi-agent collaboration on Amazon Bedrock Agents
- Benefits on building agents with Amazon Bedrock

Resources

Multi-agent collaboration workshop

Amazon Bedrock Agents training

Bedrock skill training

Thank you!

Sahil Verma

Senior GTM Specialist Solutions Architect,
Generative AI
AWS India