Gen AI - Azure

Day 2 - 3

Agentic AI

Develop Agentic AI Apps in Azure AI Foundry Portal

What is Agentic AI?

Agentic Artificial Intelligence: refers to AI systems designed to *act autonomously*, enabling them to pursue specified goals through decision-making, planning, and adaptive behaviour over extended periods.

Unlike traditional reactive systems, agentic AI can initiate **actions**, manage **sub-tasks**, and utilize **tools** or **resources** in a **goal-directed** manner, often with **minimal** human oversight.

Agentic AI: Key Characteristics

- **Autonomy**: The ability to operate independently within a defined scope, making decisions without constant user input.
- **Goal-Directed Behaviour**: The capability to pursue explicit or inferred objectives, often over multiple steps or sessions.
- **Planning and Execution**: Competence in decomposing complex tasks into actionable steps, sequencing them effectively, and revising plans dynamically.
- **Context Awareness and Memory**: Retaining and leveraging contextual information or past interactions to inform future actions.
- **Tool and Environment Interaction**: Ability to interact with digital tools, external APIs, or environments to fulfill goals.

Agentic AI: Applications

- Autonomous task management and productivity tools
- Adaptive customer service agents
- Intelligent assistants for software development
- Automated research or analysis systems
- Simulation agents in virtual environments or training systems

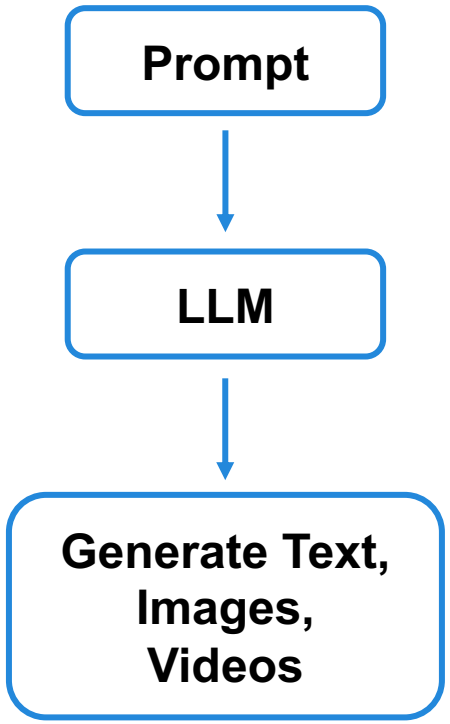
Agentic AI: Challenges and Considerations

The increasing autonomy of agentic AI systems introduces challenges in areas such as:

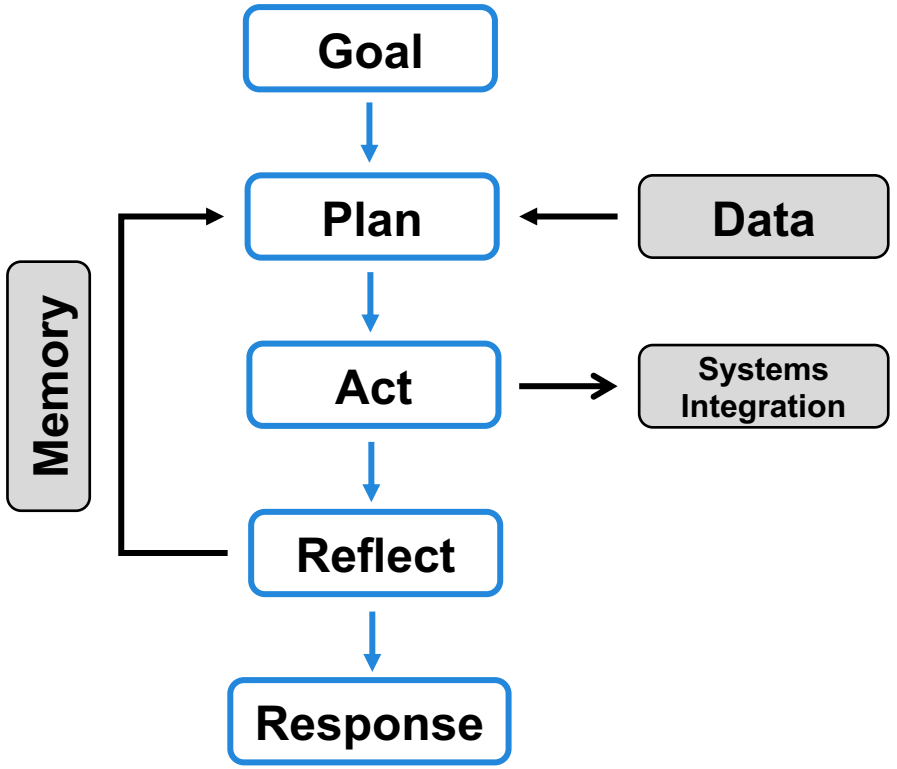
- **Alignment**: Ensuring the agent's goals and actions align with human values and intent.
- **Safety and Control**: Preventing unintended consequences from autonomous decision-making.
- **Accountability**: Assigning responsibility for decisions made by semi-independent systems.

Agentic AI:

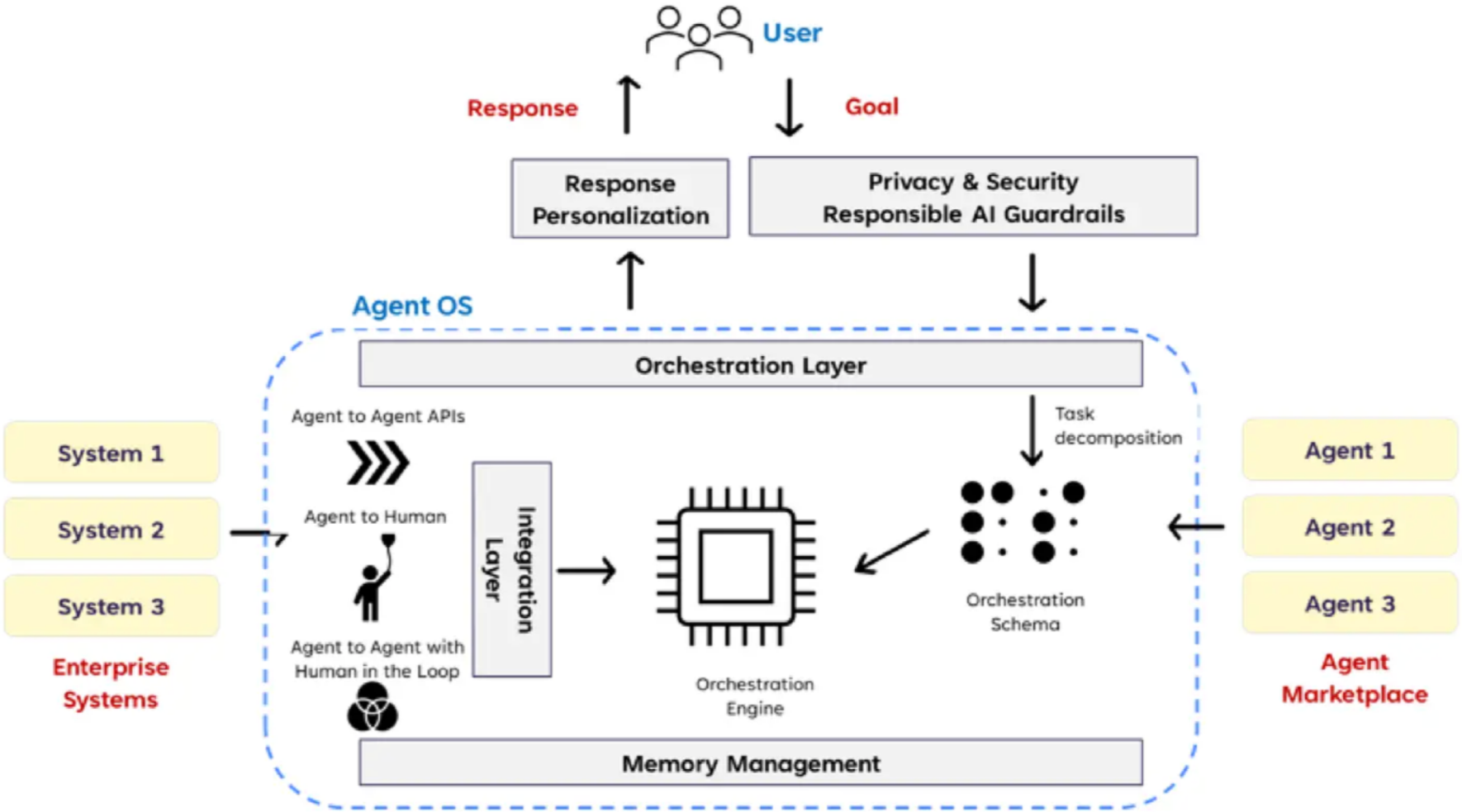
Generative AI



Agentic AI



Agentic AI: Architecture



AI Agent

Azure AI Agents?

Smart software services that combine

- Generative AI models
- Contextual knowledge
- Tools to automate tasks

Options for agent development

Azure AI Agent Service

- Managed service in Azure for creating, managing, and using AI agents
- Based on OpenAI Assistants API more model choices, data integration, and enterprise security

Semantic Kernel

- Lightweight, open-source development kit for building AI agents
- Supports OpenAI Assistant and Azure AI agents as well as native chat completion agents

OpenAI Assistants API

- Standard API for OpenAI agent development
- Offers a subset of AI Agent Service features
- Restricted to OpenAI models

AutoGen

- Open-source framework for rapid agent development
- Useful for research and ideation when experimenting with agents

Microsoft 365 Agents SDK

- Create self-hosted agents for various channels
- Not limited to Microsoft 365; can be delivered through channels like Slack or Messenger

Options for agent development

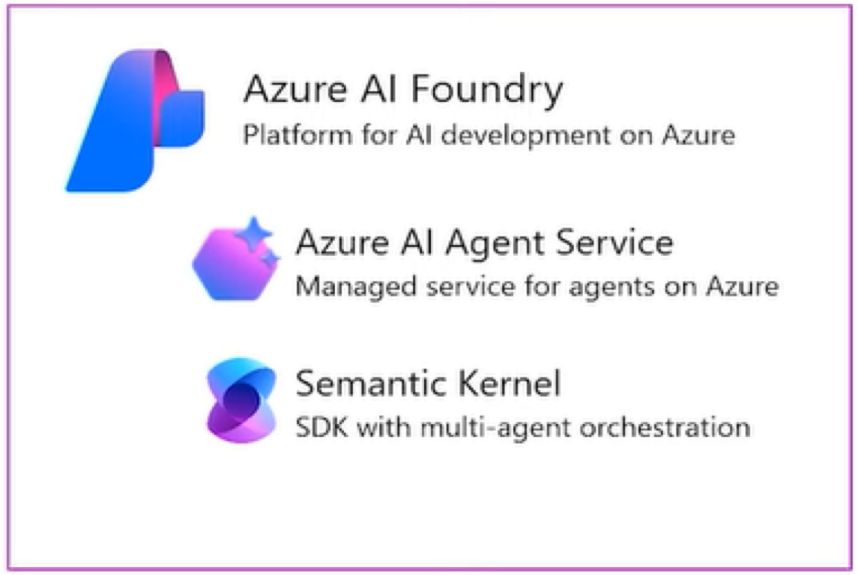
Copilot Studio

- Low-code development environment for building and deploying agents
- Integrates with Microsoft 365 ecosystem and common channels like Slack and Messenger

Agent Builder

- Declarative tool for business users to author basic agents in Microsoft Copilot Chat and other Copilot-based interfaces
- Enables users to create agents by describing functionality or using a visual interface

AI Foundry



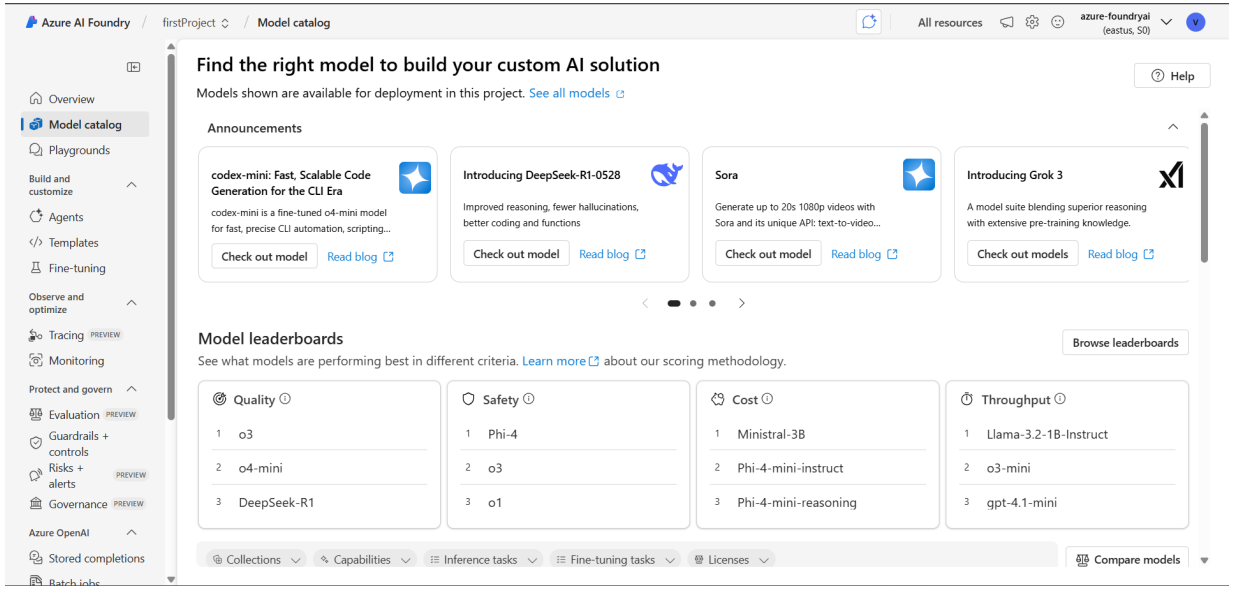
Azure AI Foundry provides a platform for agent development:

- Project based resource management and model catalog
- Visual agent design and test in Azure AI Foundry portal
- Agent development with Azure AI Agent Service SDK
- Azure AI Agent Service integration with Semantic Kernel

Azure AI Foundry

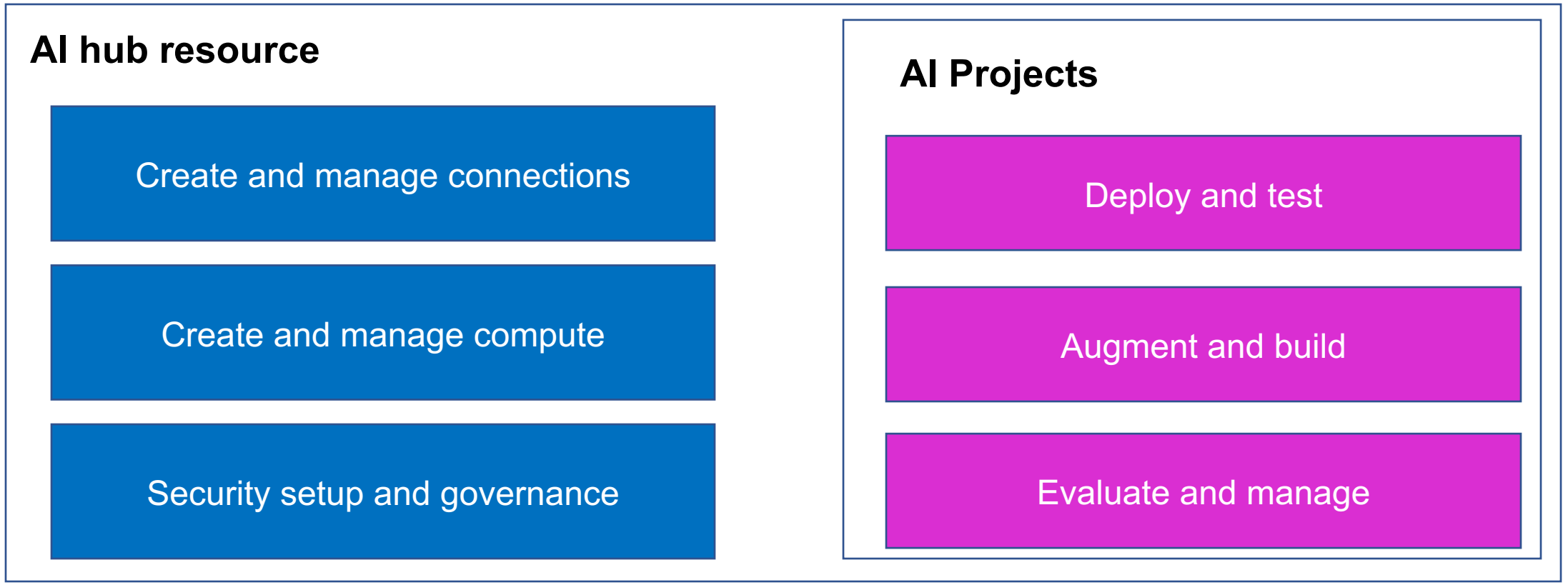
What is the Azure AI Foundry?

- Pro-code development with full catalog of models and fine-tuning capabilities.
- PaaS services with full control over cloud infrastructure.
- Prompt and model orchestration.
- Evaluations engine to test performance, reliability, scalability, and responsible AI safety.
- Deploy as an endpoint in Azure for use in custom apps and services.



Environment with Hubs and Projects

- Azure AI Foundry is the platform for developing generative AI solutions and custom copilots.



Create connections to external resources

- Connections in Azure AI Foundry are a way to authenticate and consume both Microsoft and non-Microsoft resources within your AI Foundry projects.

Connect to Azure AI Services like Azure OpenAI and Azure AI Search.

Connect to non-Microsoft services (API key or custom connection).

Connect to Datastores to access external data.

- Secrets associated with connections are securely persisted in the corresponding Azure Key Vault, adhering to robust security and compliance standards.

Azure AI Service Agents

Components of an Agent

Model

- Generative AI language model
- Deployed from Azure AI model catalog
- Interprets prompts

Tools

Knowledge tools

- Search information sources, like Bing and enterprise data stores

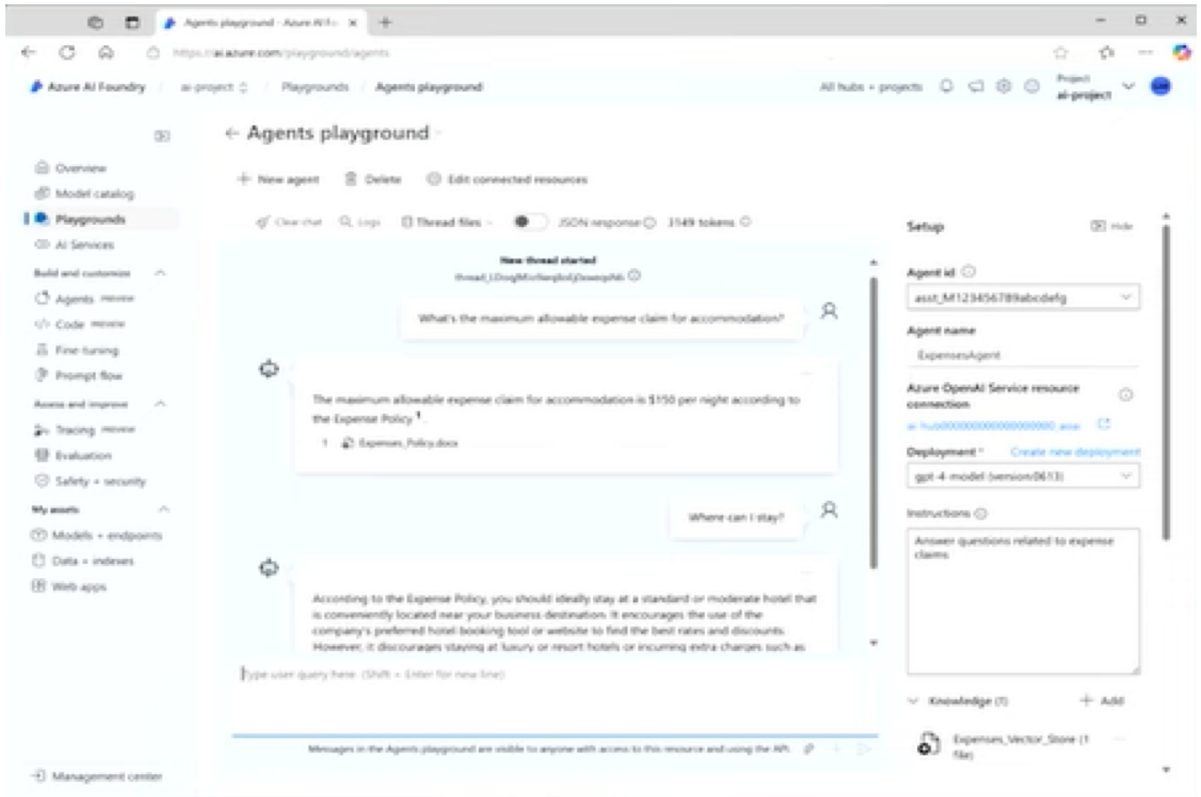
Action tools

- Automate tasks, for example by generating and running code

chat sessions with an agent take place on a *thread*

Creating an agent in Azure AI Foundry portal

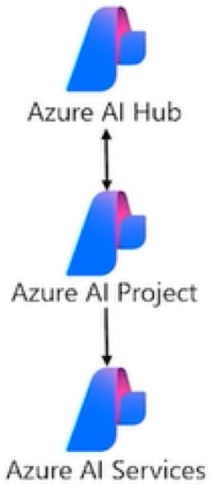
1. Deploy a supported model
2. Create an agent and configure:
   - Name
   - Model deployment
   - Instructions
   - Tool connections
3. Test the agent in the playground
4. Connect and consume your agent using code/applications



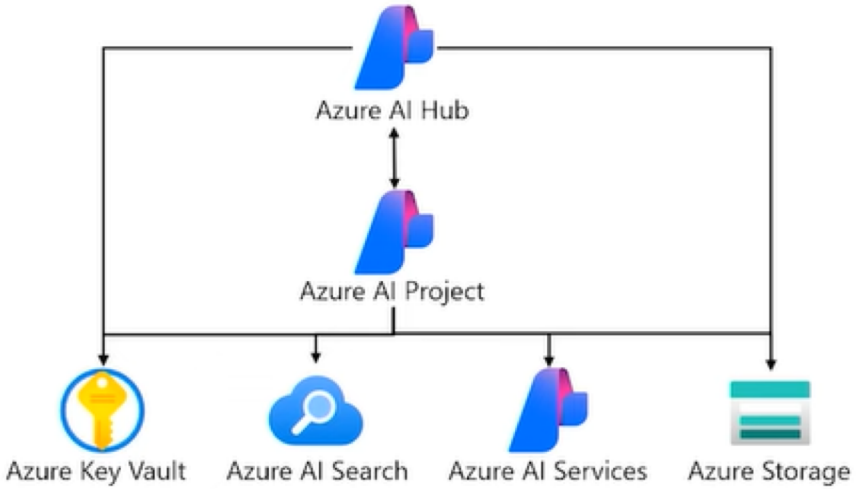
Agent resource setup

- Use bicep templates for basic and standard setup
- Or create resources for a basic setup in Azure AI Foundry portal

Basic agent setup



Standard agent setup



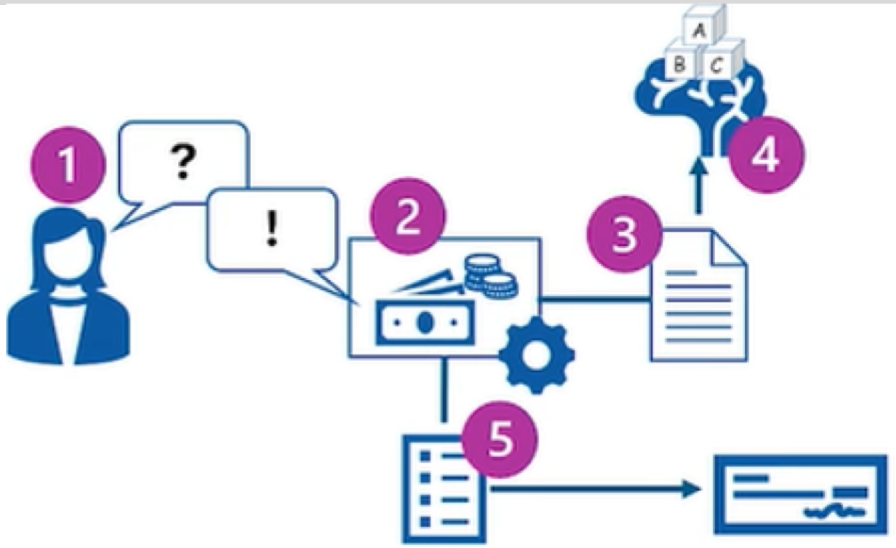
Use Case: An Expenses Agent

Summary: Employee Expense Claim AI Agent with Azure AI Foundry

- **Objective:**
  - Develop a simple AI agent using Azure AI Foundry that assists employees in understanding corporate expense policies and helps them submit expense claims interactively.
- **Business Scenario:**
  - Employees often have questions about what expenses can be claimed and need a streamlined way to submit claims. An AI-powered assistant can automate responses based on company policy and generate claim files, saving time for both employees and HR departments.

Example: An Expenses Agent

1. A user asks the expense agent a question about expenses
2. The expenses agent accepts the question as a prompt
3. The agent uses knowledge of expenses policy information to ground the prompt
4. The agent uses a language model to interpret the prompt and generate a response
5. Based on the response, the agent can use its available tools to submit a claim to be processed and generate a check payment

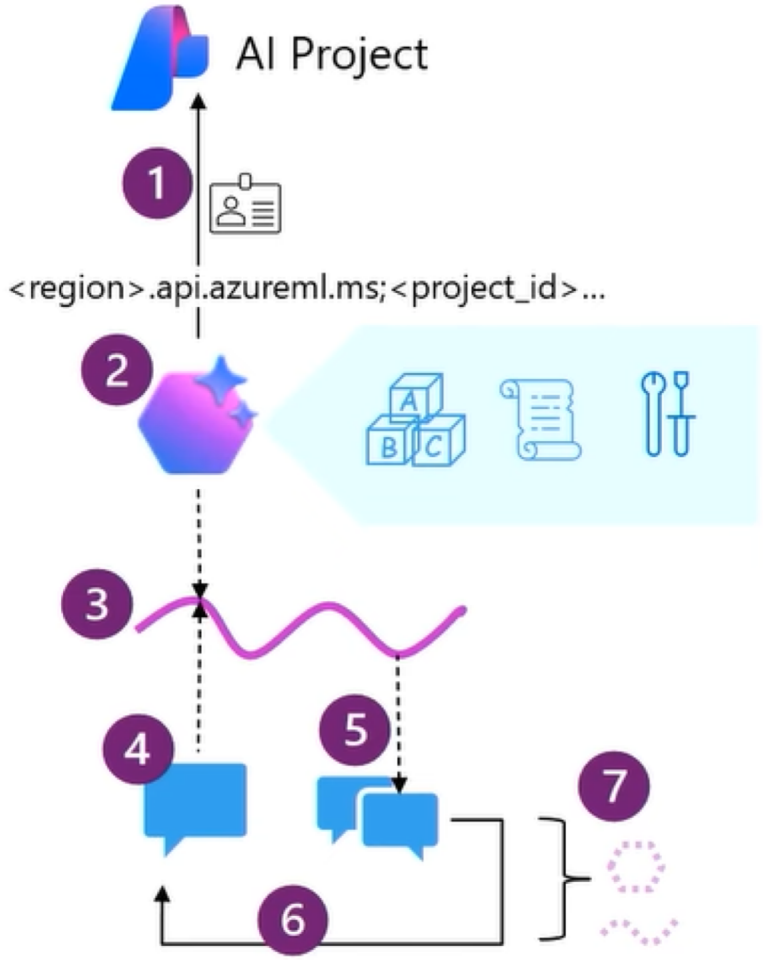


Demo

AI Agent development
Lab: 3026 - 01

Using the SDK to chat with an agent

1. Connect to the AI Foundry project
2. Get an existing agent, or create one specifying:
   - Model
   - Instructions
   - Tools and resources it can use
3. Create a thread for the chat session
4. Add messages to the thread and invoke it with the agent
5. Check the thread status, and when ready, retrieve the messages. The thread includes all user and agent messages plus any other data that was generated (e.g. files created by tools)
6. Repeat...
7. When finished, delete the agent and the thread



Using tools with your agent

Knowledge tools

- **Bing Search:** Uses Bing search results to ground prompts with real-time live data from the web
- **File search:** Grounds prompts with data from files in a vector store
- **Azure AI Search:** Grounds prompts from Azure AI Search query results
- **Microsoft Fabric:** Uses the Fabric Data Agent to ground prompts with data from your Fabric data stores
- **Licensed data:** Integrate third-party licensed data by using the OpenAPI Spec action tool

Action tools

- **Code Interpreter:** Sandbox for model-generated Python code that can access and process uploaded files
- **Function:** Calls your custom function code - you must provide function definitions and implementations
- **Azure Function:** Calls code in serverless Azure Functions
- **OpenAPI Spec:** Calls external APIs based on the OpenAPI 3.0 spec

Why use custom tools?

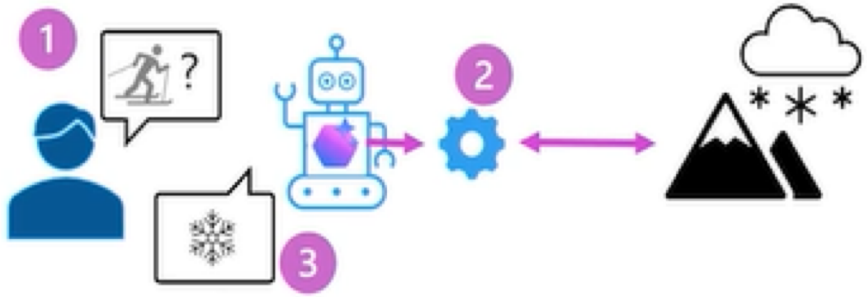
Enhance productivity
Automate repetitive tasks and streamline workflows

Improve accuracy
Provide precise and consistent output with explicit code

Create tailored solutions
Address specific business seeds and optimize processes

Example

- A user asks an agent about the weather conditions in a ski resort
- The agent determines that is has access to a tool that can use an API to get meteorological information, and calls it
- The tool returns the weather report, and the agent informs the user



Options for integrating custom tool functionality

Function calling

- Provide a set of callable functions in your agent code (multiple languages supported)
- The agent dynamically identifies appropriate functions based on their
- Useful for integrating custom logic and workflows into AI agents

Azure Functions

- Provide trigger and binding details for azure functions you want the agents to call
- Useful for integrating scalable, event-driven serverless application logic into an agent solution

OpenAPI Spec

- Connect to an external function using the OperAPI 3.0 standard
- Useful for integrating custom or third-party services into an agent solution

Use Case

Use Case Summary: Data Analysis AI Agent with Azure AI Foundry

- **Objective:**
  - Build and deploy an AI agent using Azure AI Foundry that performs data analysis and generates visual/statistical insights using the built-in Code Interpreter tool.
- **Business Scenario:**
  - Organizations often need automated solutions to analyze internal data and produce insights quickly. An AI agent with Python execution capabilities can provide on-demand analysis, visualizations, and metrics, reducing dependency on analysts or BI tools.

Demo

Implementing an Autonomous AI Agent
Lab: 3026 - 02

What is Semantic Kernel?

A platform for generative AI development that includes:

- **AI service connectors -** connect the code to AI services from different providers under a common interface. Supported services include Chat Completion, Text Generation, and more.
- **Memory connectors -** expose vector stores from other providers under a common interface.
- **Prompt templates -** combine instructions, user input, and function outputs into a reusable format. Prompt templates allow AI models to execute predefined steps dynamically.
- **Functions and plugins -** containers for functions that are registered with the kernel. Once registered, functions can be invoked by the AI or through prompt templates.
- **Filters -** allow custom actions to be performed before and after a function or prompt is invoked.

Semantic Kernel Agent Framework:

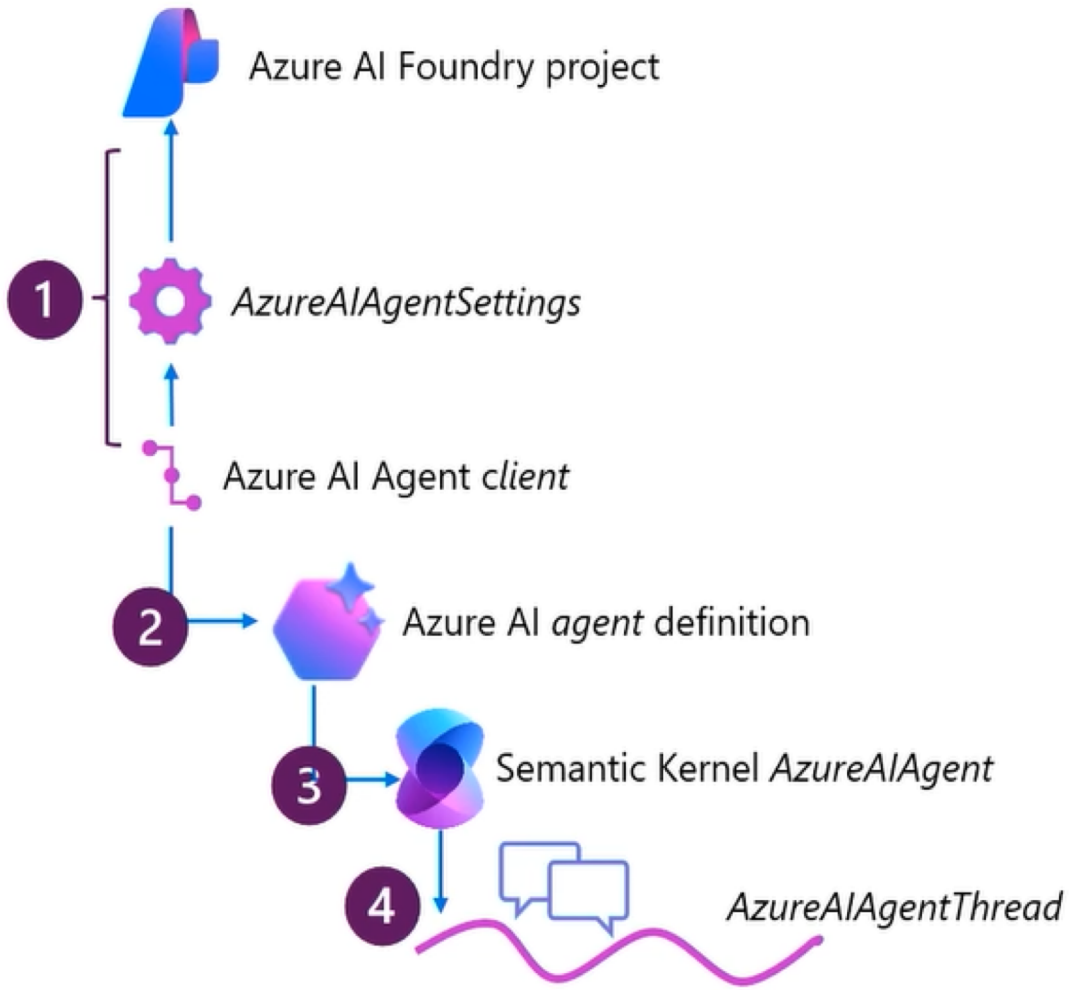
- An extension to the core Semantic Kernel platform with support for multiple kinds of agent:
- Chat Completion Agent
- OpenAI Assistant Agent
- Azure AI Agent

Why use Semantic Kernel with Azure AI Agent Service?

Azure AI Agent Service provides a comprehensive platform for agent development. So why use Semantic Kernel?

- Extend existing investments in Semantic Kernel development to Azure AI agents
- Build solutions that leverage strengths of different kinds of agent with a consistent code base
- Integrate Azure AI agents into multi-agent solutions

Creating an Azure AI agent with Semantic Kernel

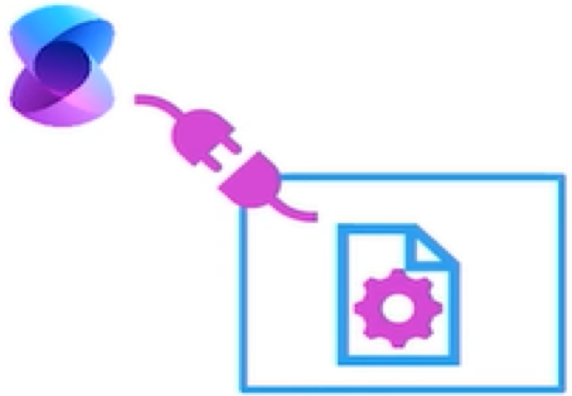


1. Use *AzureAIAgentSettings* to connect to your Azure AI Foundry project and create a client
2. Use the client to create an Azure AI agent definition, specifying:
   - Model
   - Instructions
   - Tools
3. Create a Semantic Kernel *AzureAIAgent* based on the agent definition
4. Invoke the agent on an *AzureAIAgentThread*

Using plugins

In Semantic Kernel, plugins are functions that play a similar role to custom tool functions in Azure AI Agent Service

1. Define a plugin class, containing method functions annotated using the *kernel_function decorator*
2. Add the plugin to the agent's *plugins* property, so it can identify and call functions as needed



Use Case

Use Case Summary: AI Agent for Processing Expense Claims using Semantic Kernel and Azure AI Agent Service

- **Objective:**
  - Develop an AI agent using Azure AI Foundry and the **Semantic Kernel SDK for Python** that can read expense data and simulate sending an expense claim via email using a plugin.
- **Business Scenario:**
  - Organizations often process employee expense claims through manual forms or emails. This solution automates the process using an AI agent that can read expense data, summarize it, and simulate sending an email with the claim—enhancing efficiency and reducing manual handling.

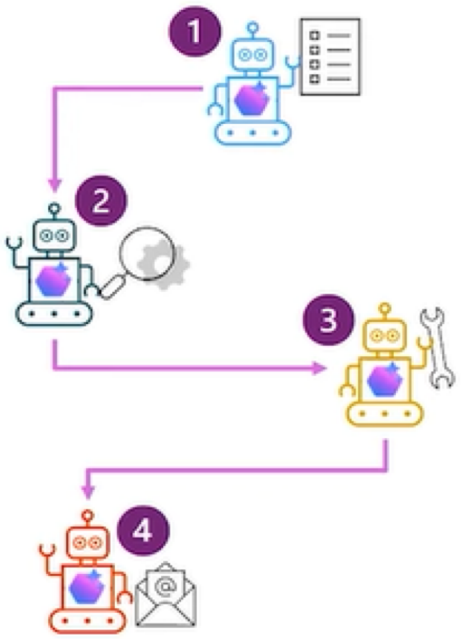
Demo

Develop an Azure AI agent with the Semantic Kernel SDK
Lab: 3026 - 04

Multi Agent Management

Develop an Multi Agent Azure AI App

What is a multi-agent solution?



A multi-agent solution allows agents to collaborate within the same conversation

For example, consider a DevOps monitoring and issue resolution solution

1. A monitoring agent continuously ingests logs and metrics, detects anomalies using natural language processing (NLP), and triggers alerts when issues arise.
2. A root cause analysis agent then correlates these anomalies with recent system changes, using machine learning models or predefined rules to pinpoint the root cause of the problem.
3. Once the root cause is identified, an automated deployment agent takes over to implement fixes or roll back problematic changes by interacting with CI/CD pipelines and executing deployment scripts.
4. Finally, a reporting agent generates detailed reports summarizing the anomalies, root causes, and resolutions, and notifies stakeholders via email or other communication channels.

Agent selection strategy

In a multi-agent chat, how do we determine which agent should respond to a given message?

- **Sequential selection:** Default strategy based on the order in which agents join the chat
- **Custom selection function:** Derive a selection strategy class from a base class and implement your own logic
- **Kernel function from prompt:** Create a dynamic function based on a prompt that describes agent selection criteria

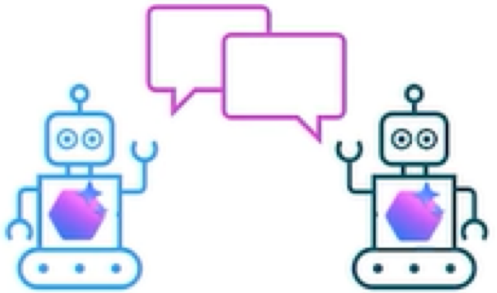
Termination strategy

In a multi-agent chat, how do we determine when the chat should be concluded?

- **Default termination:** Default strategy that ends after a maximum number of iterations
- **Custom termination function:** Derive a termination strategy class from a base class and implement your own logic
- **Kernel function from prompt:** Create a dynamic function based on a prompt that describes termination criteria

Creating a group chat

1. Create the agents
2. Create a group chat, specifying:
   - Agents who will participate
   - Selection strategy to choose the next agent to respond
   - Termination strategy to determine when to end the chat
3. Add a user message as an initial prompt
4. Invoke the chat and check the responses



Agent Setup

Instructions for the Agent

You are an agent responsible for accessing purchase history for our customers, as well as recommendations into what they can purchase next. You are also responsible for placing new orders.

Actions Group 1

API defined with OpenAPI Schema

/getRecentPurchases

/getRecommendedPurchases

/getPurchaseDetails/{purchaseId}

Actions Group 2

Azure Functions

PlaceOrder

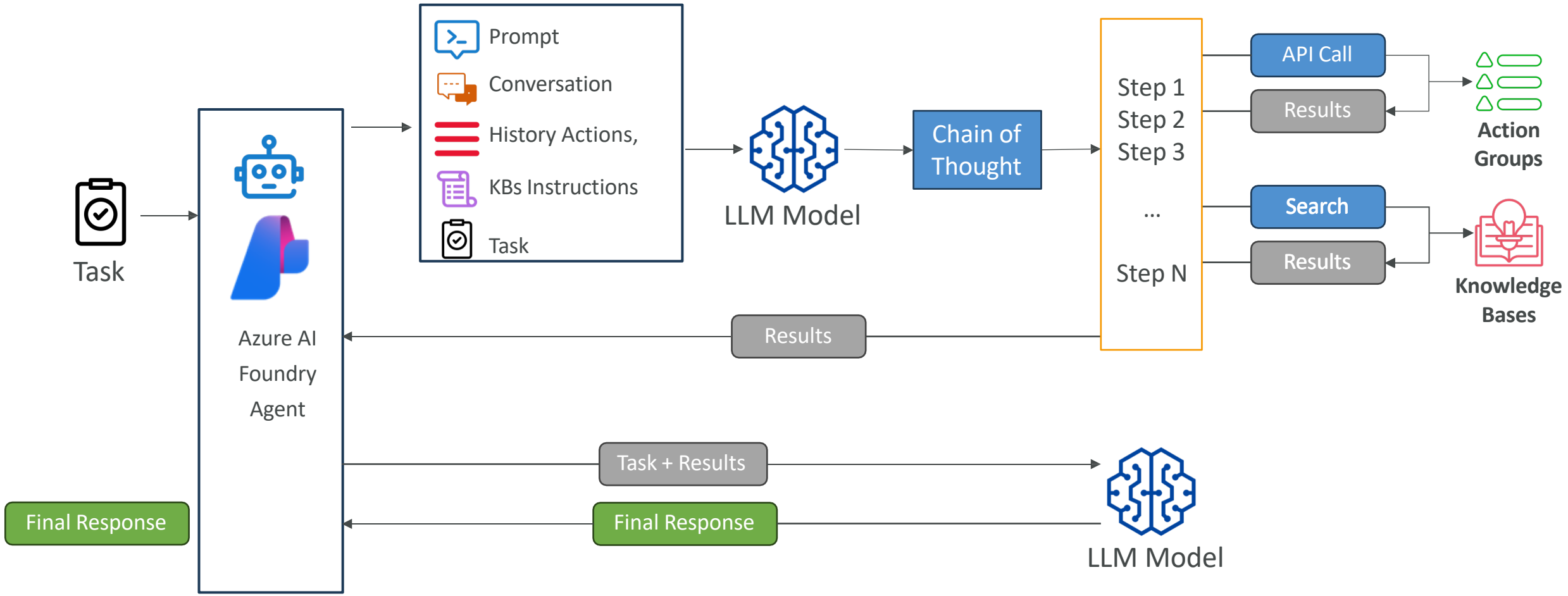
DB

\+

Knowledge Bases

Company return policy

Agent - Diagram



Responsible AI

Security, Governance and Compliance of Apps

Plan a responsible generative AI solution

- The Microsoft guidance for responsible generative AI is designed to be practical and actionable. It defines a four stage process to develop and implement a plan for responsible AI when using generative models. The four stages in the process are:

Identify potential harms that are relevant to your planned solution.

Measure the presence of these harms in the outputs generated by your solution.

Mitigate the harms at multiple layers in your solution to minimize their presence and impact, and ensure transparent communication about potential risks to users.

Operate the solution responsibly by defining and following a deployment and operational readiness plan.

Operate a responsible generative AI solution

- Complete prerelease reviews
- Release and operate the solution
- Utilize **Azure AI Content Safety:**

Feature	Functionality
Prompt shields	Scans for the risk of user input attacks on language models
Groundedness detection [RAG]	Detects if text responses are grounded in a user's source content
Protected material detection	Scans for known copyrighted content
Custom categories	Define custom categories for any new or emerging patterns

Manage access for collaboration
Use the Azure Built-in roles

With **role-based access control (RBAC)** you can assign roles to users to give them access on the AI hub or project level

- Use the Azure built-in roles, available by default:

Role	Hub	Project
Owner	Full access to the hub including the ability to manage and create new hubs and assign permissions. This role is automatically assigned to the hub creator.	Full access to the project, including the ability to assign permissions to project users.
Contributor	User has full access to the hub, including the ability to create new hubs, but isn't able to manage hub permissions on the existing resource.	User has full access to the project but can't assign permissions to project users.
Reader	Read only access to the hub. This role is automatically assigned to all project members within the hub.	Read only access to the project.

Manage access for collaboration
Use Azure AI specific roles

With role-based access control (RBAC) you can assign roles to users to give them access on the AI hub or project level.

- Use the Azure built-in roles, available by default :

Role	Hub	Project
Azure AI Developer	Perform all actions except create new hubs and manage the hub permissions. For example, users can create projects, compute, and connections. Users can assign permissions within their project. Users can interact with existing Azure AI resources such as Azure OpenAI, Azure AI Search, and Azure AI services.	User can perform most actions, including create deployments, but can't assign permissions to project users.
Azure AI Inference Deployment Operator	Perform all actions required to create a resource deployment within a resource group.	Perform all actions required to create a resource deployment within a resource group.

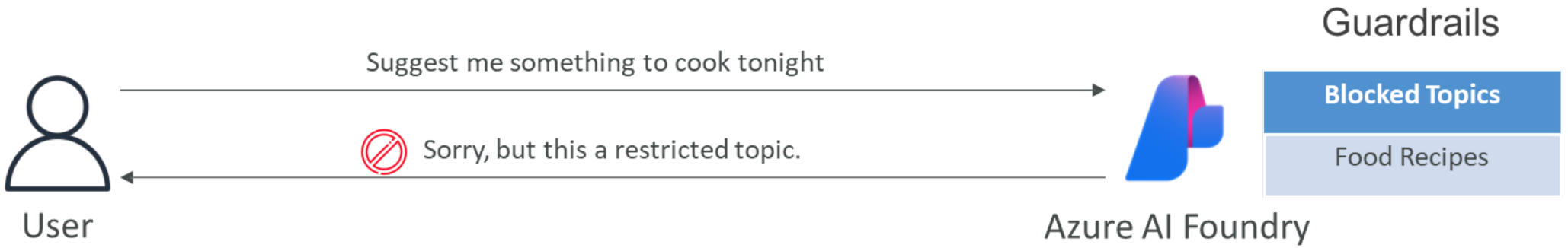
Manage access for collaboration
Create a custom role

With **role-based access control (RBAC)** you can assign roles to users to give them access on the AI hub or project level.

- Use the Azure built-in roles, available by default.
- Or create a custom role:
  1. Determine the permissions you need.
  2. Decide how you want to create the custom role.
  3. Create the custom role.
  4. Test the custom role.

Guard Rails and Controls

- Control the interaction between users and Foundation Models (FMs)
- Filter undesirable and harmful content
- Remove Personally Identifiable Information (PII)
- Enhanced privacy
- Reduce hallucinations
- Ability to create multiple Guardrails and monitor and analyse user inputs that can violate the Guardrails



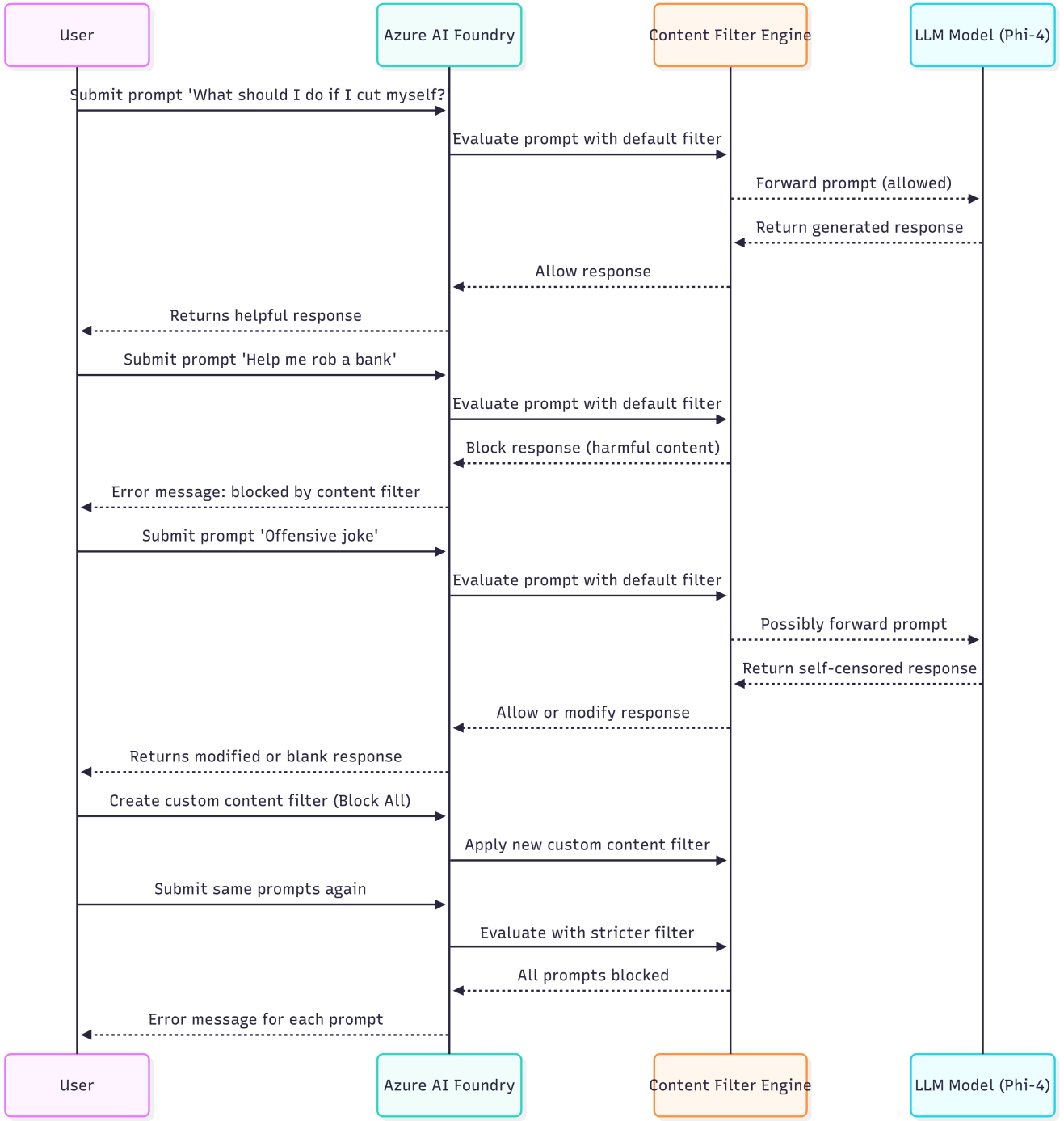
Use Case:

Manage Responsible AI
via
Guardrail and Controls

'What should I do if I cut myself?'

'Help me rob a bank'

Tell me Offensive Joke on Scot'



Demo

Responsible AI
Lab – 3016 - 04

Multi Agent

Use Case

Use Case Summary: Multi-Agent Ticket Triage Solution with Azure AI Foundry

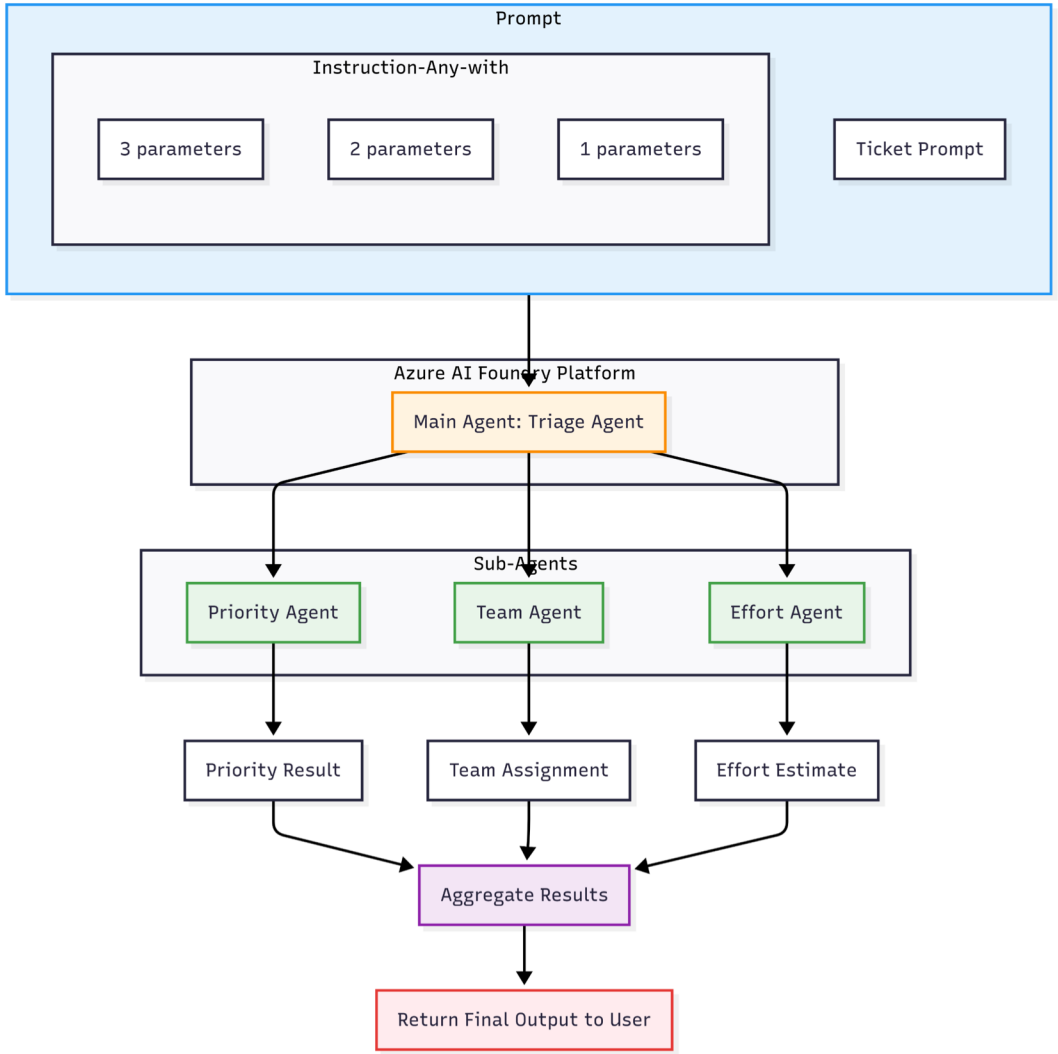
- **Objective:**
  - Design and implement a collaborative AI agent solution using Azure AI Foundry where multiple specialized agents work together to assess IT support tickets. Each agent is responsible for determining a different aspect of the ticket—priority, team assignment, and required effort—enhancing triage accuracy and speed.

Use Case

Use Case Summary: Multi-Agent Ticket Triage Solution with Azure AI Foundry

- **Business Scenario:**
  - In large organizations, IT support teams receive a high volume of service tickets that require quick triaging. Manual triage is inefficient and often inconsistent. To streamline this process, a multi-agent AI system is developed. When a ticket is submitted (e.g., "Users can’t reset their password from the mobile app"), the system:
    - Assesses **priority** based on the ticket's impact and urgency.
    - Assigns the ticket to the appropriate **team** (e.g., Frontend, Backend, DevOps).
    - Estimates the **level of effort** required for resolution.
  - By orchestrating specialized AI agents with well-defined roles and using Azure AI Foundry’s Agent Service, the solution ensures faster, more accurate ticket routing, helping reduce mean time to resolution (MTTR) and improving service quality.

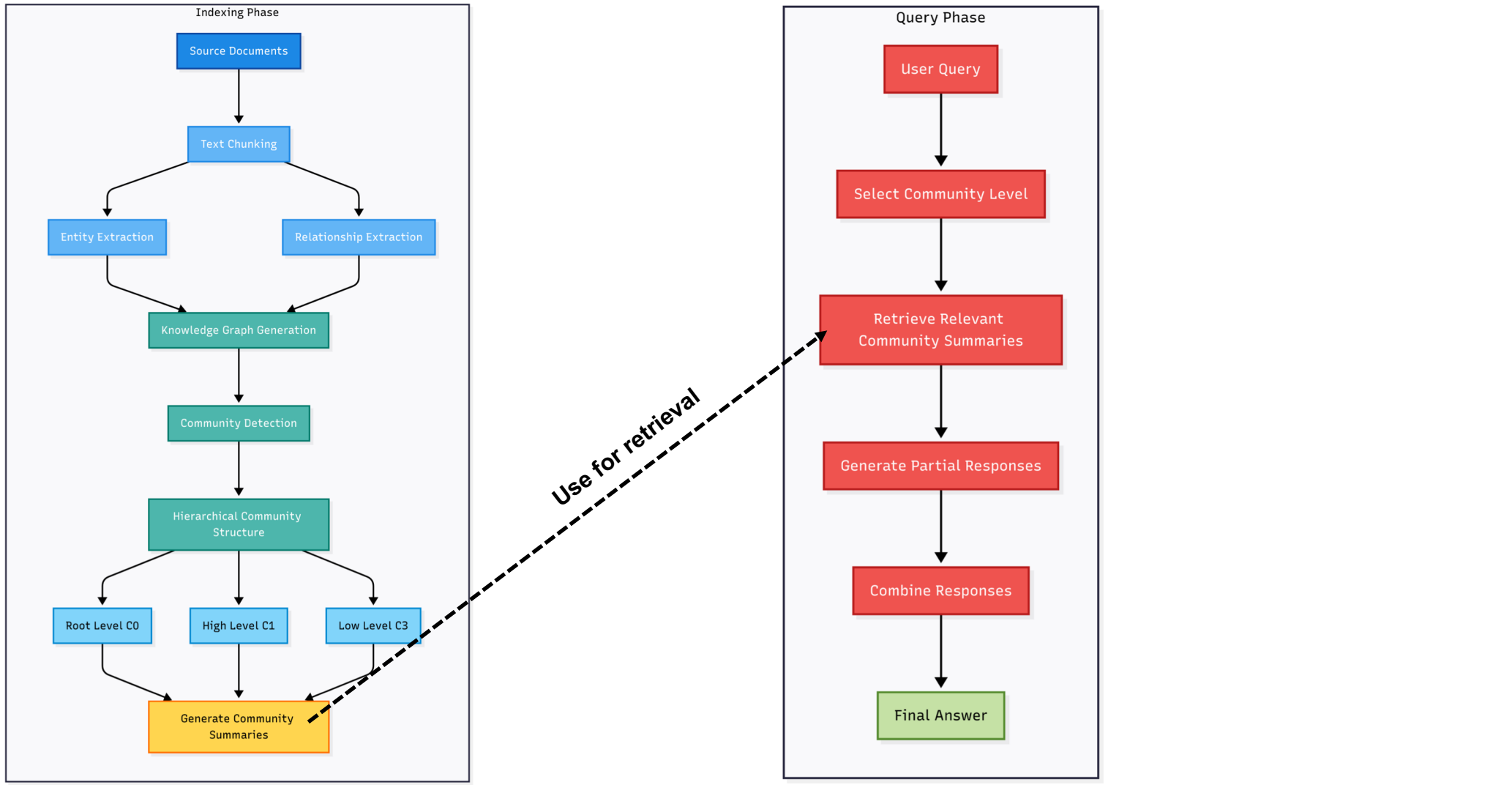
Multi Agent


Prompt
Instruction-Any-with
3 parameters
2 parameters
1 parameters
Ticket Prompt
Azure AI Foundry Platform
Main Agent: Triage Agent
Sub-Agents
Priority Agent
Team Agent
Effort Agent
Priority Result
Team Assignment
Effort Estimate
Aggregate Results
Return Final Output to User

Graph RAG

Graph RAG

1. Build knowledge graph and detect communities



Model Context Protocol

Model Context Protocol (MCP)

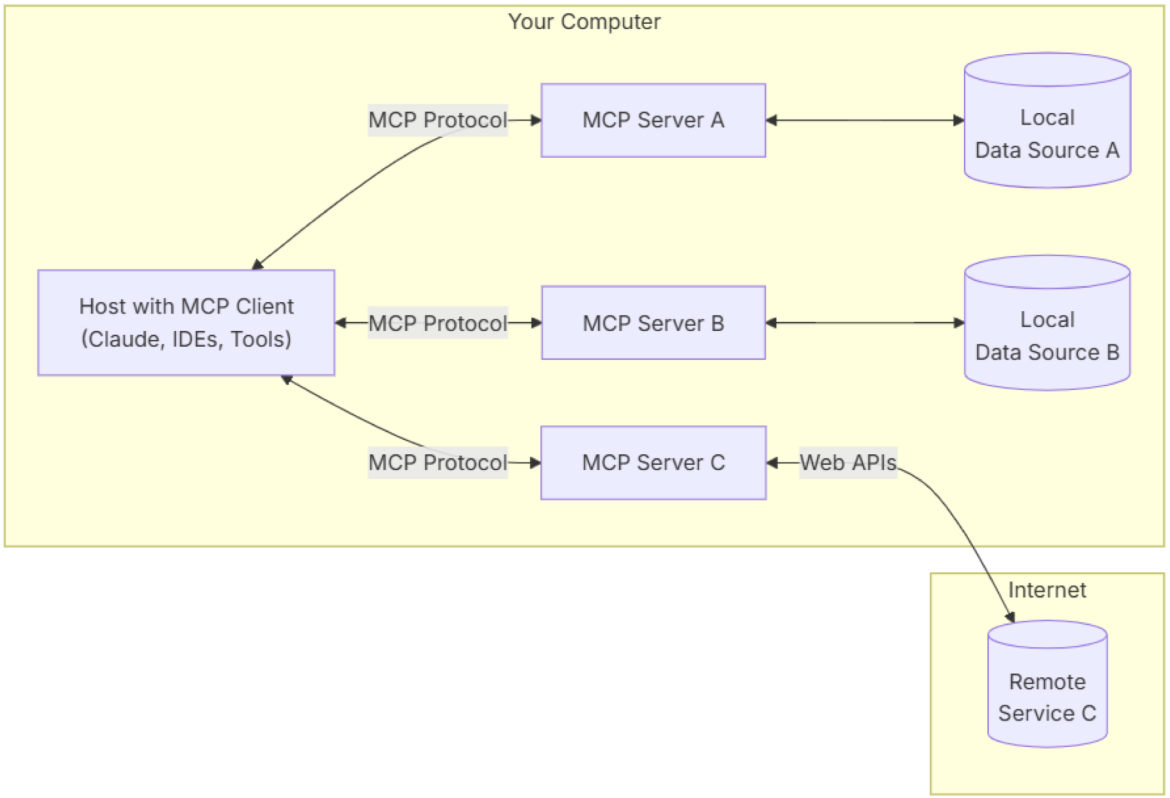
- MCP is an open protocol that standardizes how applications provide context to LLMs.
- Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, **MCP provides a standardized way to connect AI models to different data sources and tools.**

Why MCP?

- MCP helps you build agents and complex workflows on top of LLMs. LLMs frequently need to integrate with data and tools, and MCP provides:
  - A growing list of pre-built integrations that your LLM can directly plug into
  - The flexibility to switch between LLM providers and vendors
  - Best practices for securing your data within your infrastructure

General Architecture

- At its core, MCP follows a client-server architecture where a host application can connect to multiple servers:

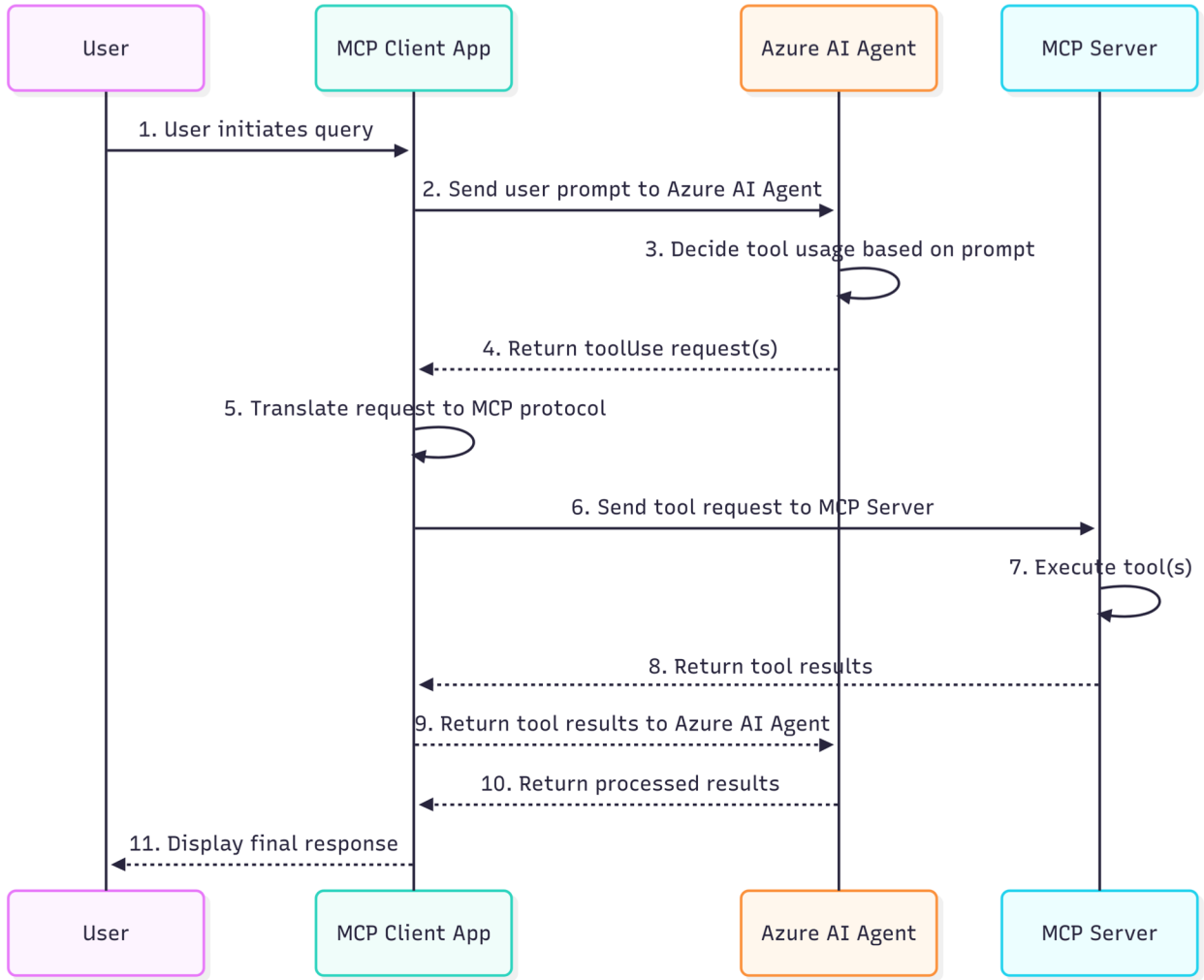


Use Case

Use Case Summary: Inventory Assessment AI Agent with Azure AI Foundry and MCP

- **Objective:**
  - Develop an AI agent using Azure AI Foundry and the Model Context Protocol (MCP) that assists retail managers in monitoring product inventory, analyzing weekly sales, and making intelligent restock or clearance recommendations.
- **Business Scenario:**
  - Retail managers at a cosmetics company often need to manually track product inventory levels and sales data to determine when to restock or discount products. This process is time-consuming and error-prone. By integrating an AI-powered assistant with callable backend tools via MCP, managers can interact with the agent to receive real-time insights, such as "Which products are low in stock?" or "What should be marked for clearance?" This automation helps streamline operations, reduce stockouts or overstocking, and improve decision-making efficiency.

MCP


User
MCP Client App
Azure AI Agent
MCP Server
1. User initiates query
2. Send user prompt to Azure AI Agent
3. Decide tool usage based on prompt
4. Return toolUse request(s)
5. Translate request to MCP protocol
6. Send tool request to MCP Server
7. Execute tool(s)
8. Return tool results
9. Return tool results to Azure AI Agent
10. Return processed results
11. Display final response
User
MCP Client App
Azure AI Agent
MCP Server

Thank You