Microsoft

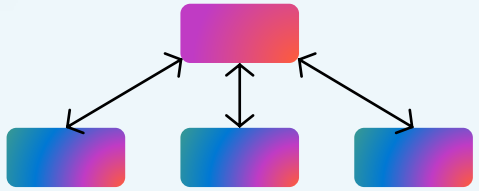
03 Multi Agenten Orchestrierung und offene Standards

Classified as Microsoft Confidential

Considerations for building Multi-Agent systems with AI Foundry

1. **Managed Platform for Fast Time to market**
   Foundry's connected agents offer a managed, low-code environment ideal for quickly building agentic systems of medium complexity

2. **Simplified Development Experience**
   While the UI and SDK make it easy to get started, the abstraction limits fine-grained control over orchestration, memory, and tool invocation



3. **No Support for Open Protocols (Yet)**
   Protocols like MCP and A2A are not yet supported in connected agents, limiting interoperability with broader ecosystems. Support is planned but not yet available

4. **Prompt-Only Orchestration is Costly**
   Complex orchestration must be encoded in prompts, which increases token usage, latency, and cost. There's no native support for conditional logic, retries, or fallback strategies

5. **Flat Orchestration Model**
   Connected agents support only a single layer of delegation. Agent A can call Agent B, but Agent B cannot call Agent C. This tree-based model restricts the design of deeply collaborative agent systems

Classified as Microsoft Confidential

Recap: Freedom to create agents your way

Platform Integrations, ease-of-use, and development speed

IaaS
Infrastructure-as-a-service

Code handling every detail

Azure AI Infrastructure

Bring your own frameworks

e.g. Semantic Kernel

PaaS
Platform-as-a-service

Code with managed services

Azure AI Foundry

Agent Service

SaaS
Software-as-a-service

Drag and drop UI building

Copilot Studio

Instant agent runtime

Control, visibility, and customization

Classified as Microsoft Confidential

Converging the two popular OSS frameworks

Microsoft Agent Framework (coming soon)

Semantic Kernel

Full SDK designed to build AI agents with ease, excellent for single agents and can be extended for multi-agents with integrations to AutoGen

AutoGen

Powerful multi-agent research framework with prebuild conversation orchestration patterns for handling complex agent systems

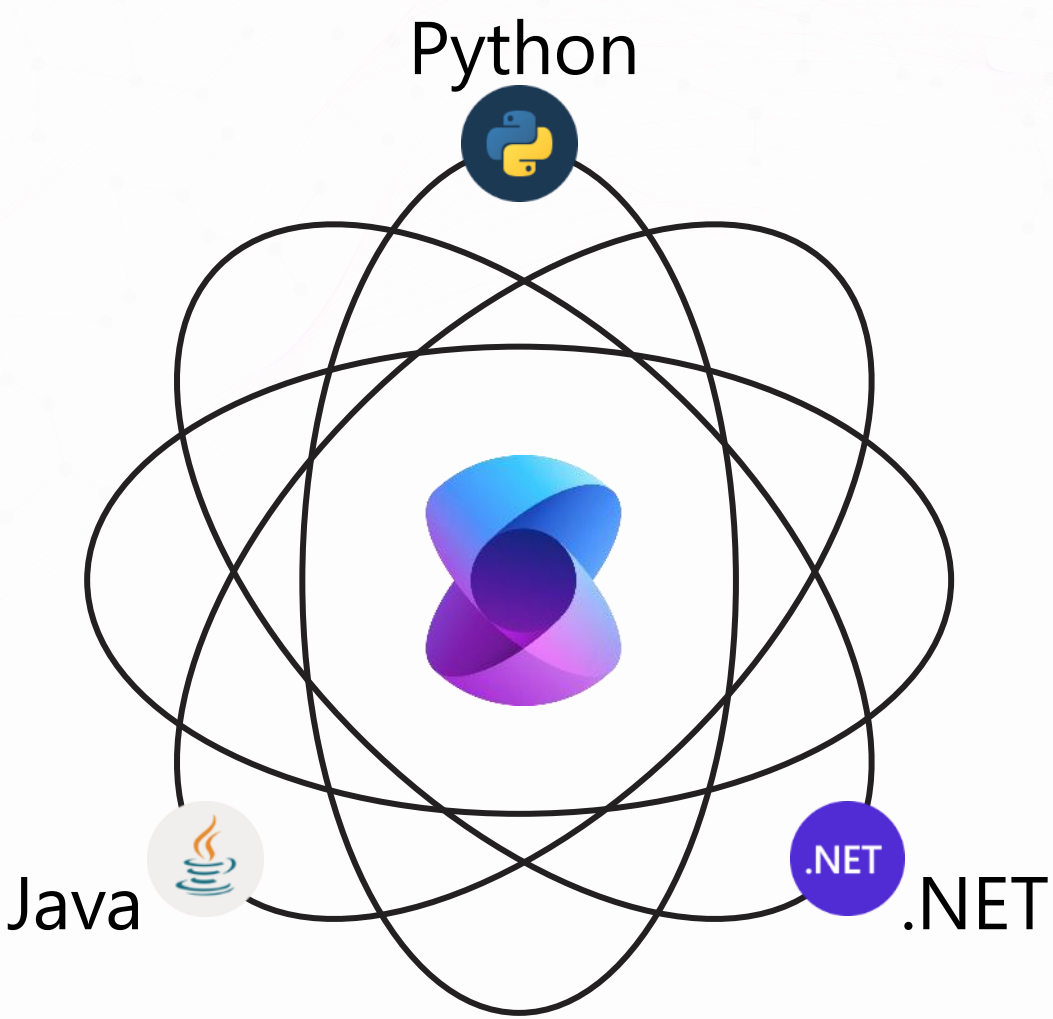
Classified as Microsoft Confidential

What is Semantic Kernel ?

Semantic Kernel is lightweight, **open-source**, production-ready, orchestration middleware that lets you easily add AI to your apps.

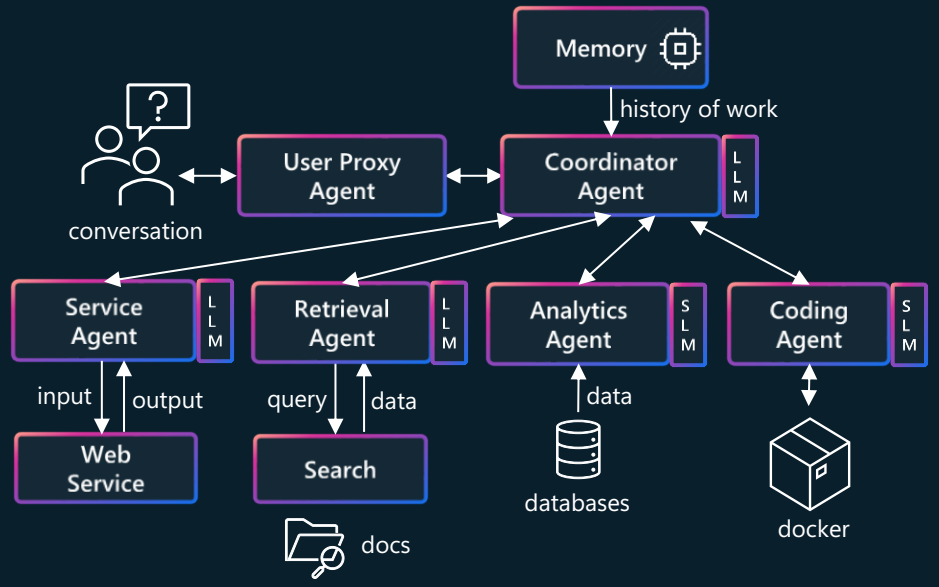
Includes 2 key Frameworks:

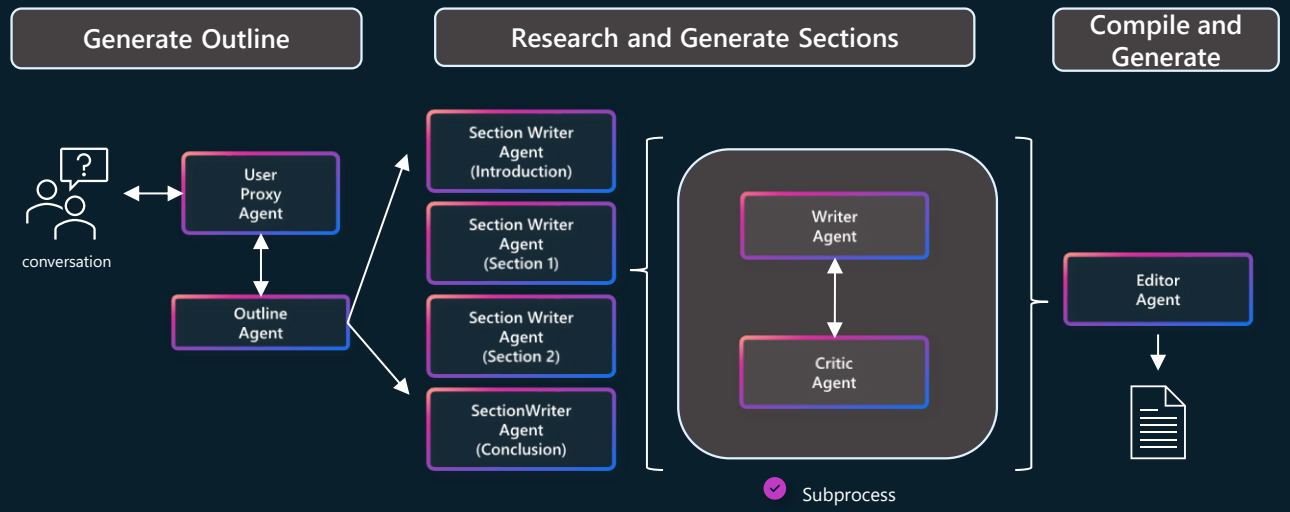
- Agents Framework
- Process Framework



Classified as Microsoft Confidential

Semantic Kernel contains two key Frameworks





Agent Framework

LLM-driven orchestration

Creative reasoning and decision-making

Ex: Propose 2 Instagram marketing campaigns including assets that would leverage the top 2 recent trends in our past quarter US Sales to boost our mailing list user base and predict the impact of each campaign

Process Framework

Workflow-driven orchestration

Deterministic and predictable execution pattern

Ex: Execution the Deep Research process to generate a document outline, write and refine section and synthesize into final document

Classified as Microsoft Confidential

This Sessions Focus: Building Advanced Agentic Systems with Microsoft's Semantic Kernel Agent Framework

Semantic Kernel Agents Framework

The Semantic Kernel Agent Framework, provides a **platform** within the Semantic Kernel eco-system that allow for the creation of AI agents and the ability to incorporate **agentic patterns** into any application based on the same patterns.

Benefits of AI agents:

1. **Modular Components**
2. **Collaboration**
3. **Human-Agent Collaboration**
4. **Process Orchestration**

Overview of other Orchestration Frameworks

- OpenAI Swarm
- Crew AI
- LangGraph
- Agno

Classified as Microsoft Confidential

Classified as Microsoft Confidential

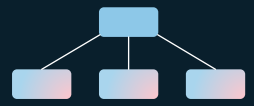
Semantic Kernel Agent Framework – Core Concepts

1

Built-in advanced orchestration patterns

Sequential

Concurrent

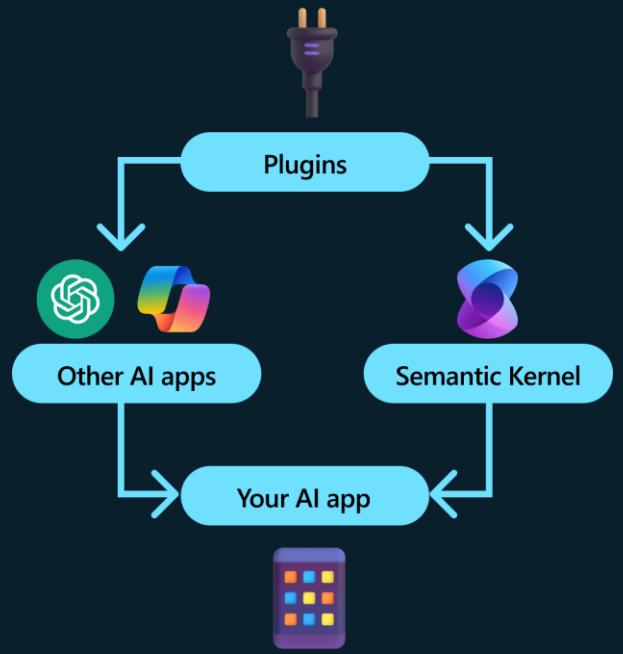


Group Chat

Magentic

2

Plugins for Function Calling, OpenAPI & MCP



3

Native Support of several Agent Types

Azure AI Agents Service

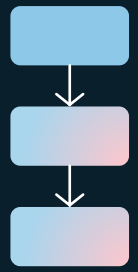
Bedrock AI

Classified as Microsoft Confidential

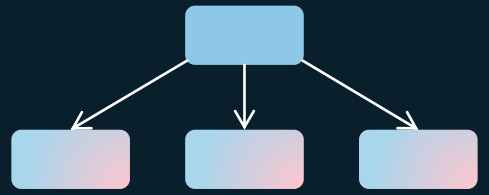
1 Orchestration Patterns

Multi-Agent Orchestration Patterns in Semantic Kernel

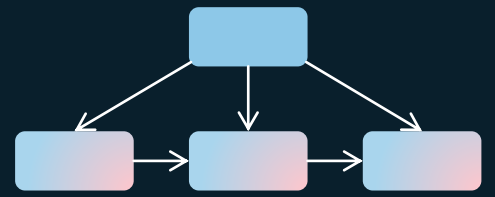
Sequential



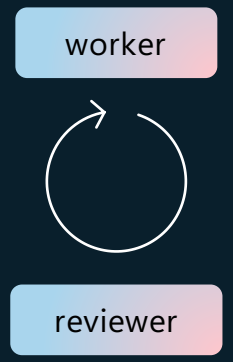
Concurrent



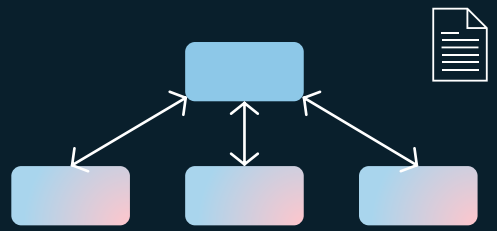
Handoff



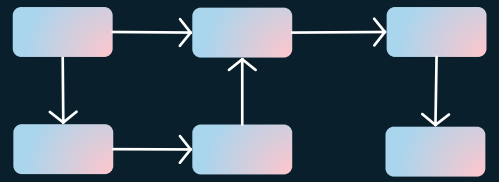
Group Chat



Magentic



Workflow Process



Full Documentation

Classified as Microsoft Confidential

Invocation of Orchestration Frameworks

All orchestration patterns share a unified interface for construction and invocation. No matter which orchestration you choose, you:

1. Define your agents and their capabilities
2. Create an orchestration by passing the agents (and, if needed, a manager).
3. Optionally provide callbacks or transforms for custom input/output handling.
4. Start a runtime and invoke the orchestration with a task.
5. Await the result in a consistent, asynchronous manner.

```
# Create the Agents
agent_a = ChatCompletionAgent(
        name="agent_a",
        instructions="You are agent a",
        service=AzureChatCompletion(),
    )

(…)

# Choose an orchestration pattern with your agents
orchestration = SequentialOrchestration(members=[agent_a,
agent_b])
# or ConcurrentOrchestration, GroupChatOrchestration,
HandoffOrchestration, MagenticOrchestration, ...

# Start the runtime
runtime = InProcessRuntime()
runtime.start()

# Invoke the orchestration
result = await orchestration.invoke(task="Your task here",
runtime=runtime)

# Get the result
final_output = await result.get()

await runtime.stop_when_idle()
```

Full Documentation

Classified as Microsoft Confidential

Plugins are a key component of Semantic Kernel

Native Code Plugins

```python
agent = ChatCompletionAgent(
        service=AzureChatCompletion(),
        name="Host",
        instructions="Answer questions about the menu.",
        plugins=[MenuPlugin()],
    )


class MenuPlugin:
    @kernel_function
    def get_specials(self):
        return """
        Special Soup: Clam Chowder
        Special Salad: Cobb Salad
        Special Drink: Chai Tea
        """

    @kernel_function
    def get_item_price(self, menu_item):
        return "$9.99"
```

External Connectors

Logic Apps

MCP Server

OpenAPI Spec.

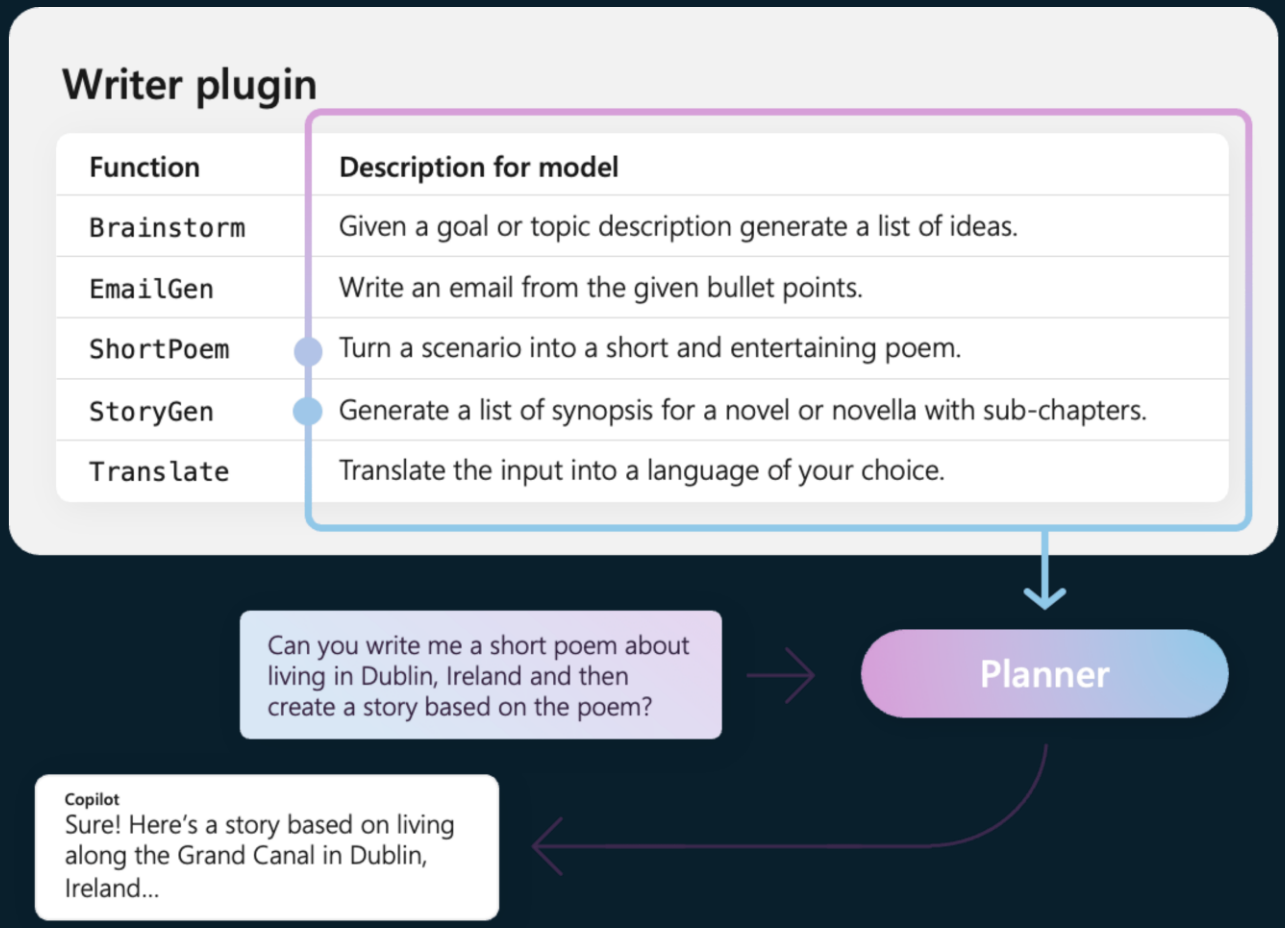
A2A Protocol

Classified as Microsoft Confidential

Ensure LLM readable function Specification

Plugins need to provide details that **semantically describe** how their **functions** behave.

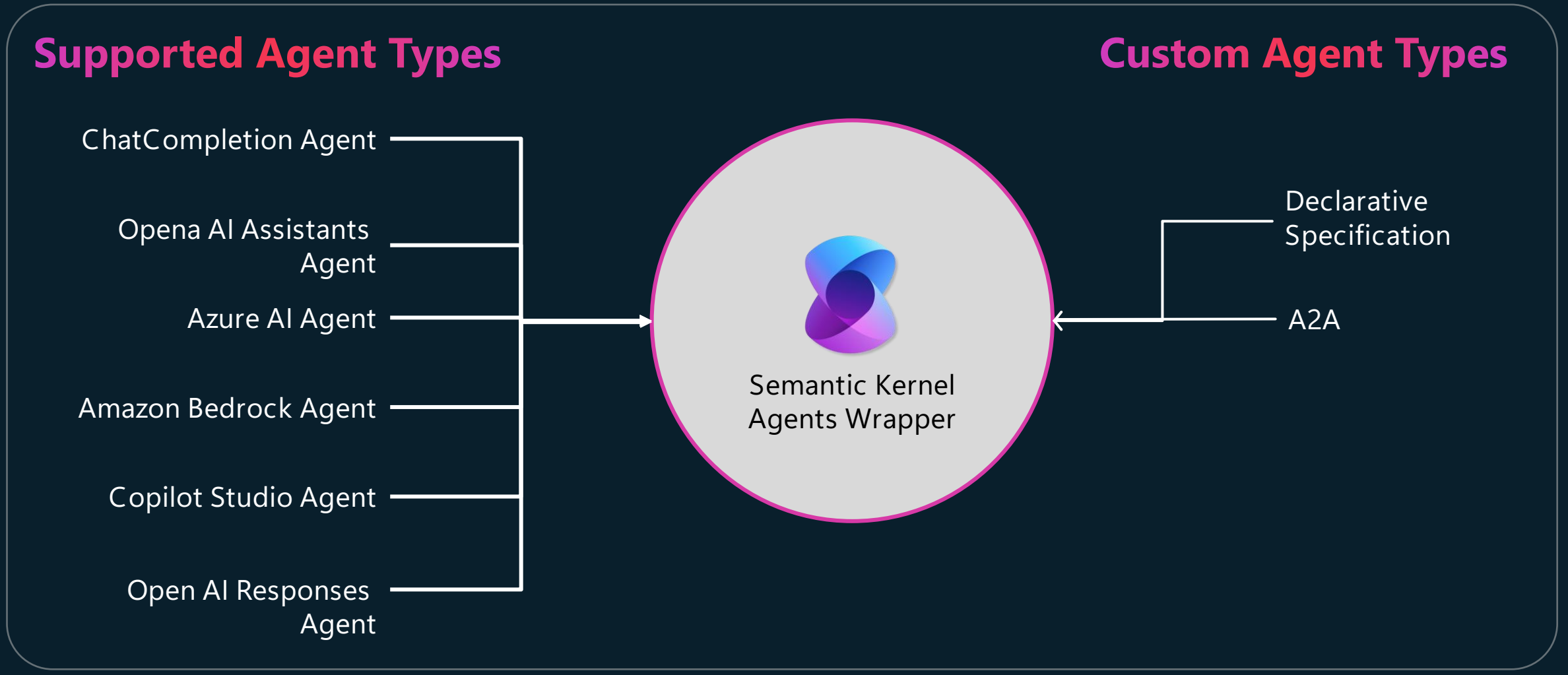
Everything from the function's **input, output**, and **side effects** need to be described in a way that the **AI can understand,** otherwise, the AI will not correctly call the function



Classified as Microsoft Confidential

Integrating various Agent Types into Semantic Kernel

Multiple agents of different types can collaborate within a **single conversation**, be integrated in **orchestration frameworks** and utilize **kernel plugins**



Classified as Microsoft Confidential

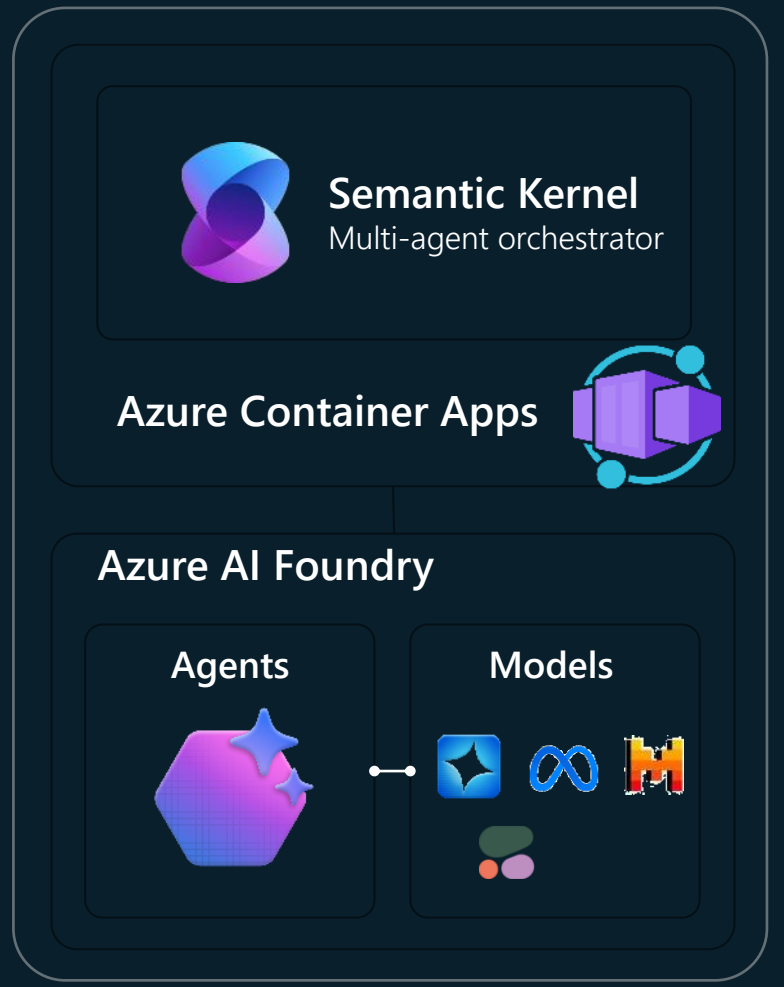
Summary: Building Agents with Semantic Kernel

```python
agent = ChatCompletionAgent(
        service=AzureChatCompletion(),
        name="Host",
        instructions="Answer questions
about the menu.",
        plugins=[MenuPlugin()],
    )

class MenuPlugin:
    @kernel_function
    def get_specials(self):
        return """
        Special Soup: Clam Chowder
        Special Salad: Cobb Salad
        Special Drink: Chai Tea
        """

    @kernel_function
    def get_item_price(self,
menu_item):
        return "$9.99"
```

Create

```python
thread: ChatHistoryAgentThread = None

try:
    for user_input in USER_INPUTS:
        print(f"# User: {user_input}")

        response = await
agent.get_response(messages=user_input,
thread=thread)

        print(f"# {response.name}:
{response}")
        thread = response.thread

finally:
    await thread.delete()
```

Run



Deploy

Classified as Microsoft Confidential

Microsoft

04 Hands-On Hack: Semantic Kernel

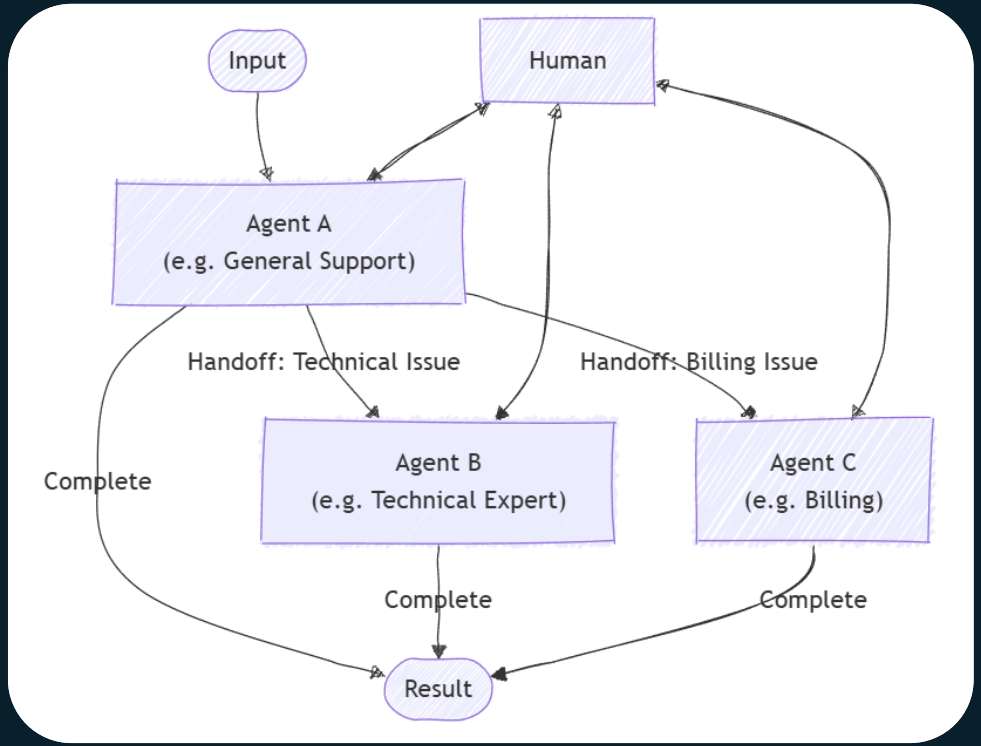
Classified as Microsoft Confidential

Hacking Session 2: Exploring Semantic Kernel

You're role

Given that the Travel Booking Agentic System was such a huge success you are tasked with implementing a **Multi-Agent Support System** for Munich Agent Factory GmbH. This system will automate **customer service requests** regarding purchased orders by orchestrating multiple specialized agents.

What you are going to build



Find all instructions in the Github Repository: [Go to Hacking Session 2]

Classified as Microsoft Confidential

Microsoft

Short Break

Classified as Microsoft Confidential