

EVOLUTION OF .NET AI: FROM REQUESTS TO AGENTS

Josh Morales

Page Init Solutions LLC

<https://pageinit.net>





Quick Introduction

- .NET Enthusiast
- 20+ Years Experience
- Own Page Init Solutions LLC
- Admin RGV Devs Dot Net User Group



Find Me Online



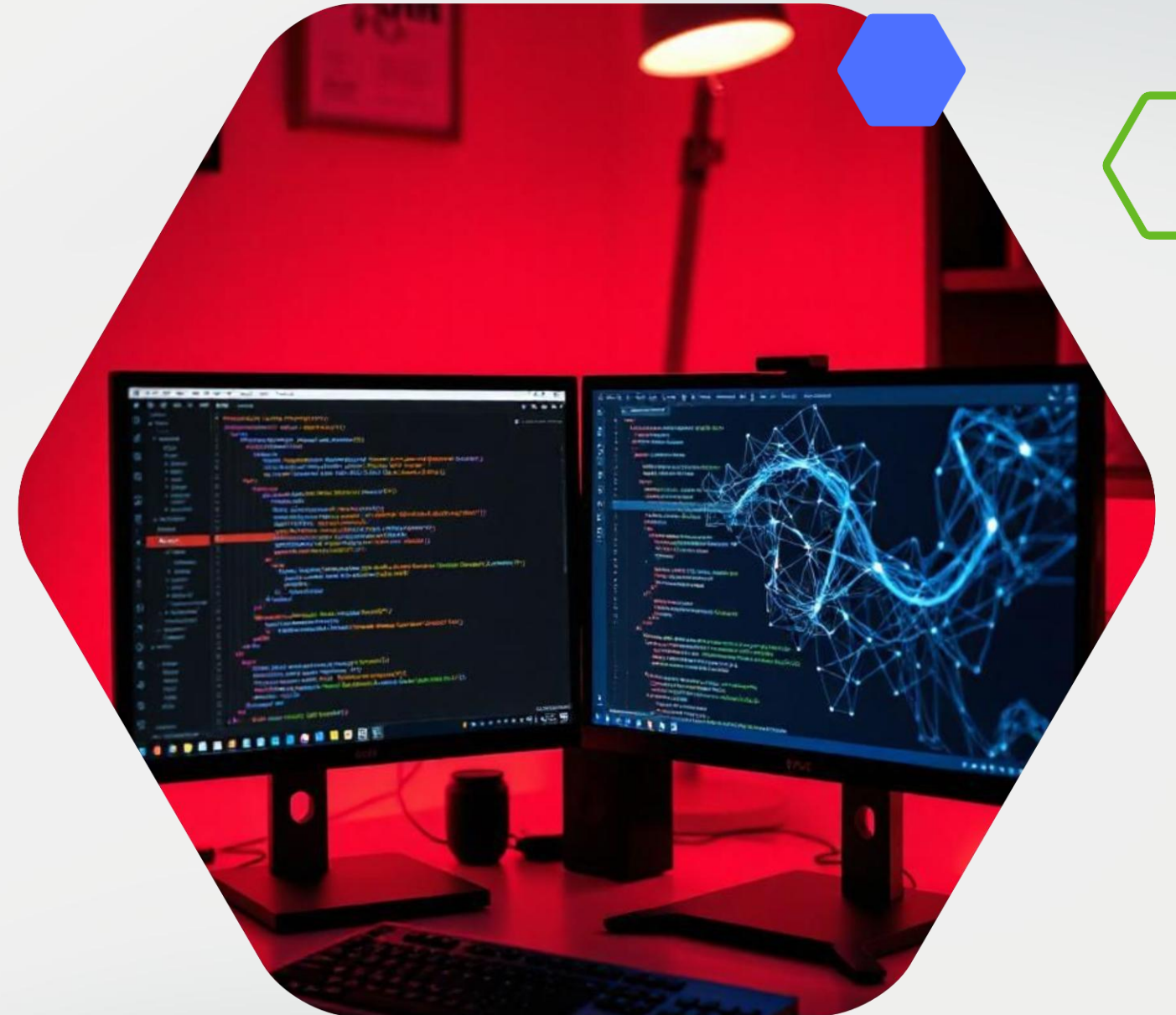
Table of Contents

- Python vs C#
- AI Access Without Abstraction (REST/HTTP)
- Introducing Semantic Kernel
- Standardization With Microsoft.Extensions.AI
- Orchestration with AutoGen
- Agent Framework: Putting it All Together
- Wrapping Up and Getting Involved
- Q&A

Python vs C#?

Python is a natural start for learning and using AI, but what if you prefer C# and .NET? It's not only possible, it's getting easier with every release.

With new tools and frameworks in .NET, developers can build smart AI apps that feel natural and fit right into their existing workflows.



AI Access Without Abstraction (REST/HTTP)



Multiple providers, multiple languages

Each AI service spoke its own language with different auth methods, JSON formats, and response structure.



Brittle and hard to maintain code

Managing raw HTTP calls led to tangled codebases that were tough to scale or update as new models appeared.

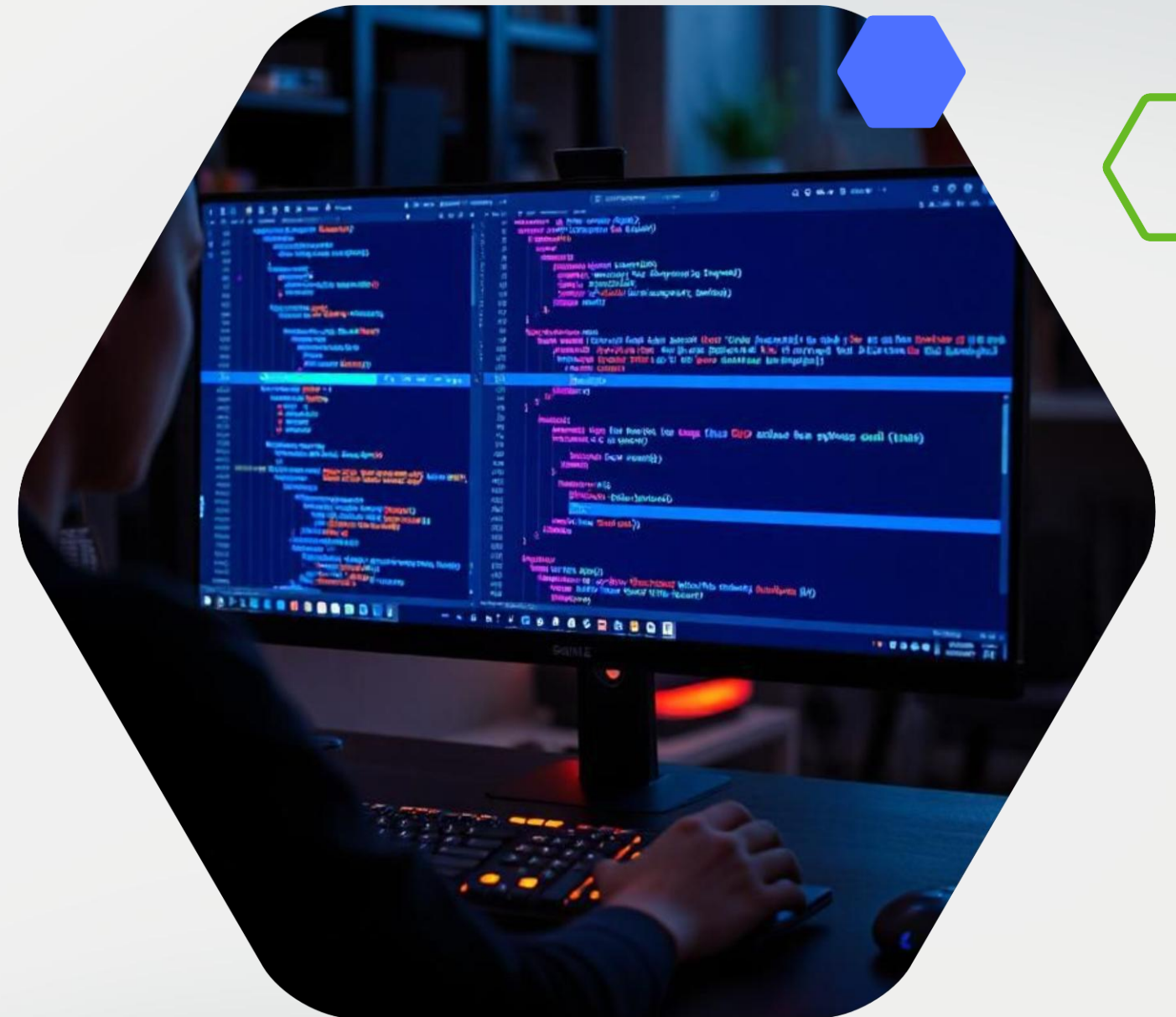
Demo 1:

Brute Force with HttpClient

Low-level interaction with AI models using raw JSON requests and manual response parsing.

Introducing Semantic Kernel

A unified, higher-level framework for integrating AI models and orchestration capabilities across various providers within .NET, specifically designed to streamline the integration process and move beyond disparate, model-specific libraries





Semantic Kernel: Features and Benefits

- **AI Orchestration Layer:** Combines AI services like OpenAI, Azure OpenAI, and Hugging Face with conventional programming languages. This layer allows developers to orchestrate execution plans and deploy features like **plugins**, **planners**, and **memory**
- **Minimized Learning Curve:** The SDK provides abstractions designed to reduce the complexity of working directly with different AI models or services
- **Improved Reliability:** It offers mechanisms to fine-tune prompts and plan tasks, helping to reduce the unpredictable behavior often seen in raw prompts and responses from AI models
- **Foundation for the Copilot Stack:** SK was released to help developers build their own "Copilot experiences" using the same AI orchestration patterns that power Microsoft 365 Copilot and Bing

.

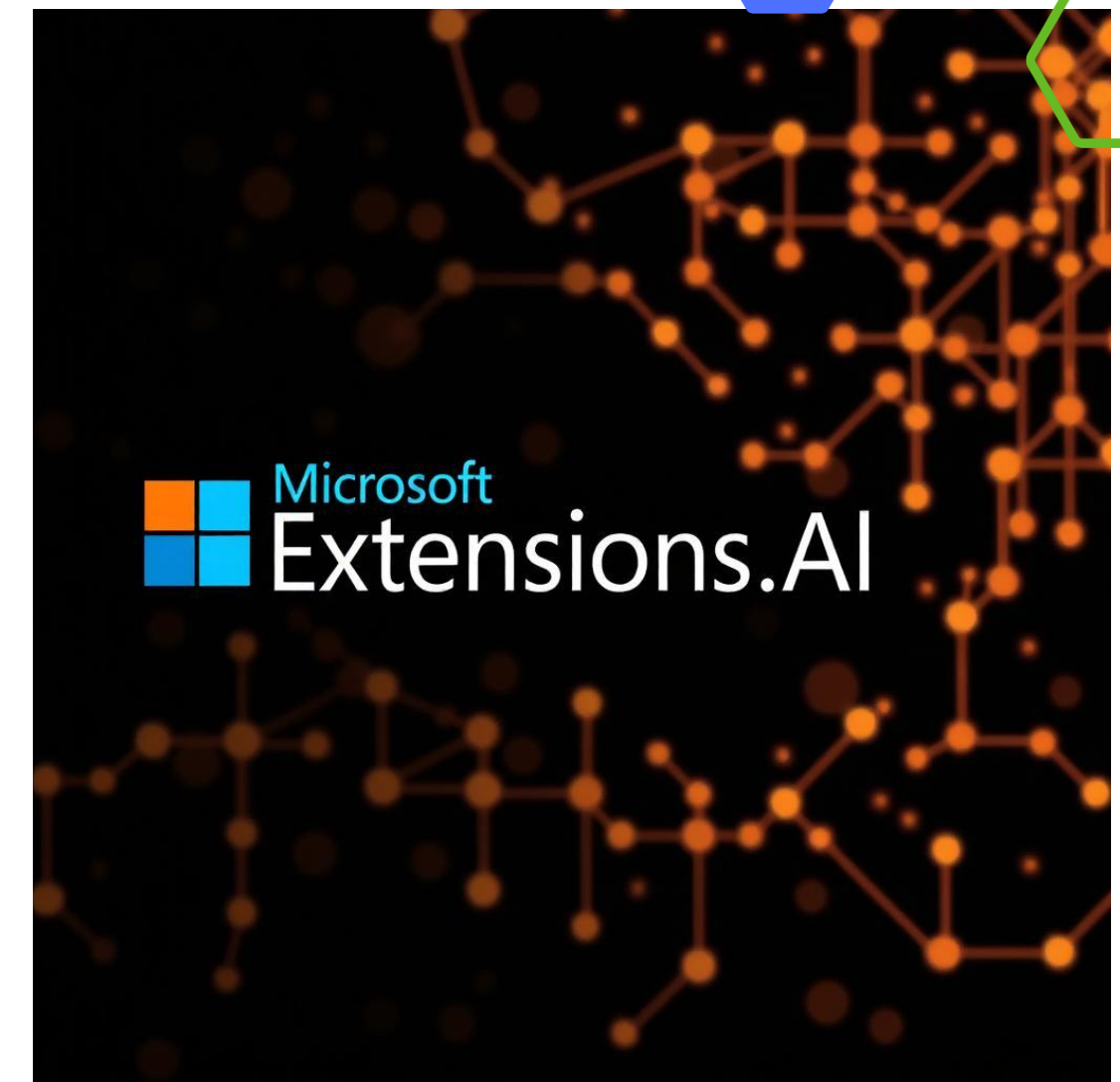
Demo 2:

Orchestration via Abstraction

Using Semantic Kernel to interact with AI
using natural language and reusable functions.

Standardization With Microsoft.Extensions.AI

A unified, low-level framework for integrating AI models and orchestration capabilities across various providers within .NET, specifically designed to streamline the integration process and move beyond disparate, model-specific libraries





Semantic Kernel and Extensions.AI: A Winning Combination

1. **Semantic Kernel (SK) as the Prototype Orchestrator (Higher-Level):** SK was initially designed with abstractions for building robust AI applications (like the `ITextCompletionService`). It acts as a specialized, opinionated SDK focused on orchestration, planning, memory, and agents.
2. **Microsoft.Extensions.AI (ME.AI) as the Standardized Abstraction Layer (Lower-Level):** The low-level APIs used within SK, such as `IChatClient` and `IEmbeddingGenerator<TInput, TEmbedding>`, were later extracted from Semantic Kernel and moved into the dedicated `Microsoft.Extensions.AI` namespace.
3. **The True Unifying Layer:** ME.AI explicitly acts as the unifying layer for generative AI in .NET, providing core C# abstractions that are deliberately not tailored to any specific provider's services. It leverages established .NET patterns like dependency injection and middleware for logging, caching, and telemetry.

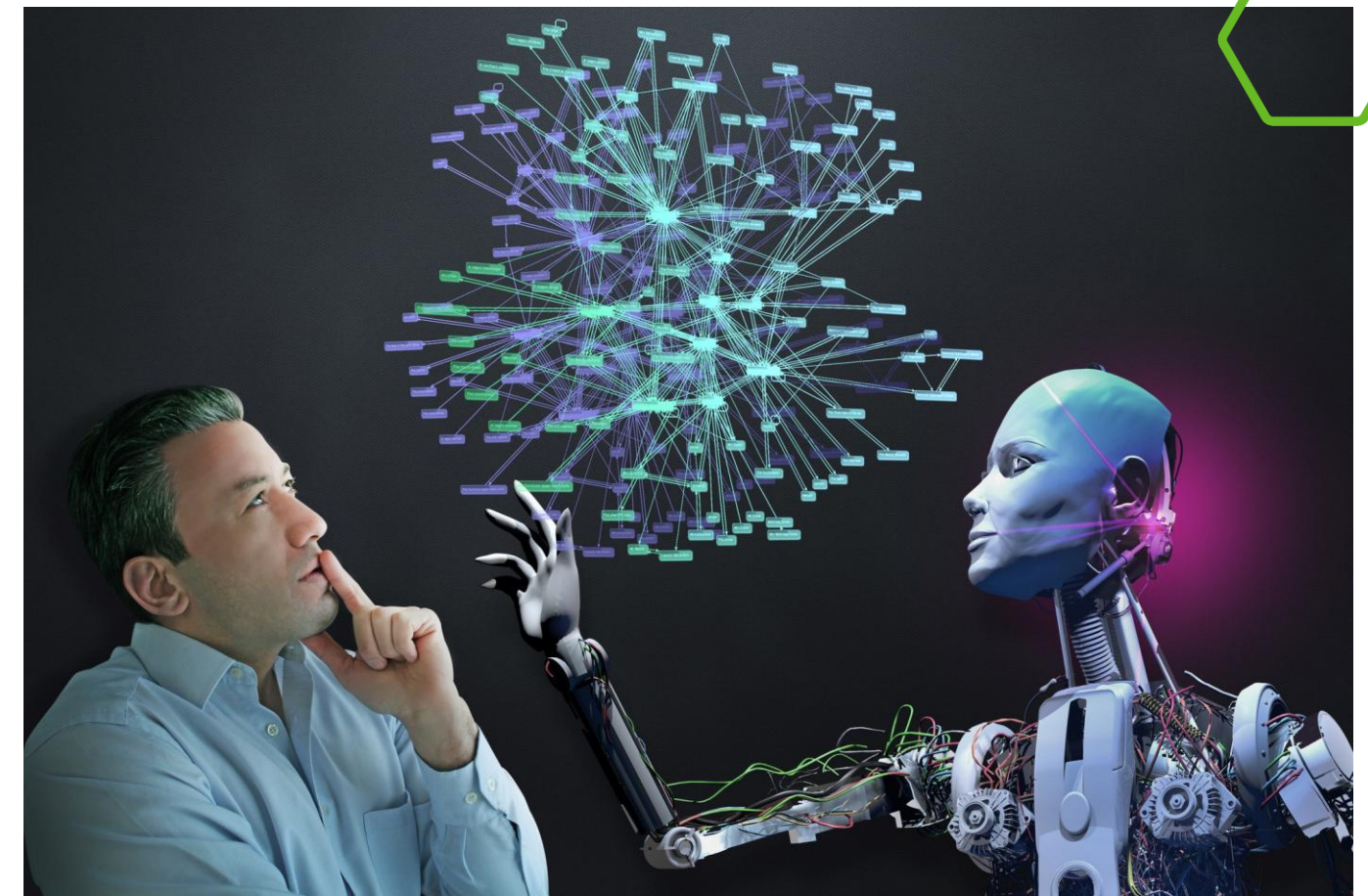
Demo 3:

Framework Integration with Abstractions

Demonstrate integration with Dependency Injection using
Microsoft.Extensions.AI

Orchestration with AutoGen

With AutoGen.NET, you can build AI agents that talk to each other and tackle complex tasks together, making your .NET apps smarter and more capable.





AutoGen: Features and Benefits

- **Primary Design Goal:** AutoGen was built to facilitate **cooperation among multiple agents to solve tasks**
- **Enhanced LLM Inference and Optimization:** Ensuring agentic applications are efficient and cost-effective
- **Teachability and Personalization:** Enabling customizable, personalized agents for intuitive applications
- **Advanced Orchestration:** Pioneering experimental multi-agent orchestration patterns like debate, reflection, facilitator/worker, and group chat, which inspired the community
- **Complex Workflows:** Supporting scenarios where multiple agents interact to complete complex tasks autonomously or with human oversight, particularly suited for **long-running autonomous agents**
- .

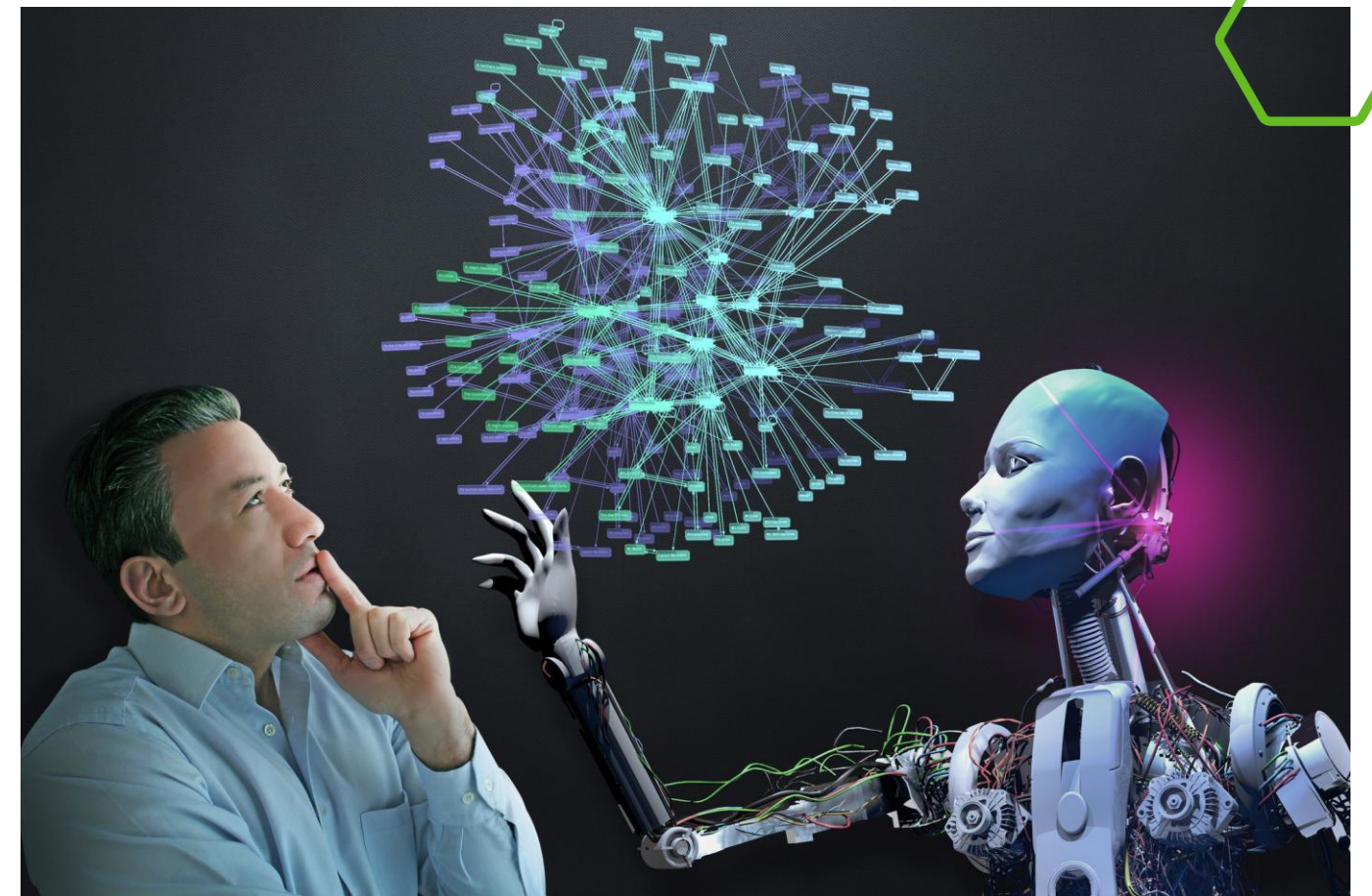
Demo 4:

Orchestrating Multiple Agents Together

Demonstrate how multiple agents can communicate to create a complete picture to complete a given task.

Agent Framework: Putting it All Together

The **Microsoft Agent Framework (MAF)** combines AutoGen's dynamic orchestration with Semantic Kernel's durable foundations to create fully stateful, autonomous agents.





MAF: Features and Benefits

Robust State Management: MAF provides a **robust state management system** explicitly built for **long-running and human-in-the-loop scenarios**

Checkpointing: MAF's graph-based workflows offer **checkpointing** capabilities to save workflow states, enabling the server-side **recovery and resumption** of long-running processes. This provides the **stronger composability and durability** sought after during multi-agent migrations

Agent Thread for State: MAF provides an **agent thread for state management**. When migrating from SK, agents now manage threads natively within MAF, simplifying invocation.

Multiple Language Support: Consistent APIs for both C# and Python

Demo 5:


Stateful Agents with MAF

Demonstrate how multiple agents can not only communicate but also store and recall state for long-running tasks.



Wrapping Up & Getting Involved

A quick recap of what we've learned and how to dive deeper with fellow .NET developers.



Reviewing the Journey



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

- **Raw HTTP calls and proprietary SDKs**
Starting with basic API calls that were brittle and hard to maintain.
- **Smart abstractions**
Introducing Semantic Kernel and Extensions.AI for clean, reusable AI workflows.
- **Fully Autonomous agents**
Building stateful, collaborative agents with AutoGen.NET and the Agent Framework.

Helpful Resources

Semantic Kernel

- Official Documentation: [Semantic Kernel docs \(Microsoft Learn\)](#)
- Getting Started: [Semantic Kernel Cookbook \(DeepWiki\)](#)

Microsoft.Extensions.AI

- Official Documentation: [Microsoft.Extensions.AI libraries \(Microsoft Learn\)](#)
- Getting Started: [Microsoft.Extensions.AI: Integrating AI into your .NET applications \(TechCommunity blog\)](#)

AutoGen

- Official Documentation: [AutoGen Quick Start \(Microsoft GitHub Pages\)](#)
- Getting Started: [AutoGen GitHub Repository](#)

Microsoft Agent Framework (MAF)

- Official Documentation: [Agent Framework docs \(Microsoft Learn\)](#)
- Getting Started: [Agent Framework GitHub Repository](#)



<https://rgvdevs.net>

Join RGV Devs: A community for .NET developers in the Rio Grande Valley

RGV Devs Dot Net is your local spot to meet other .NET developers, share knowledge, and explore new tech together. Whether you're a seasoned pro or just starting out, this community welcomes you to join the conversation and build something amazing.

THANK YOU



<https://rgvdevs.net>



Presentation Resources