



# Semantic Kernel

Where simplicity meets productive AI

Unleashing the Power of AI  
Beginner to Intermediate



**Roger Barreto**  
Sr. Software Engineer  
CoreAI



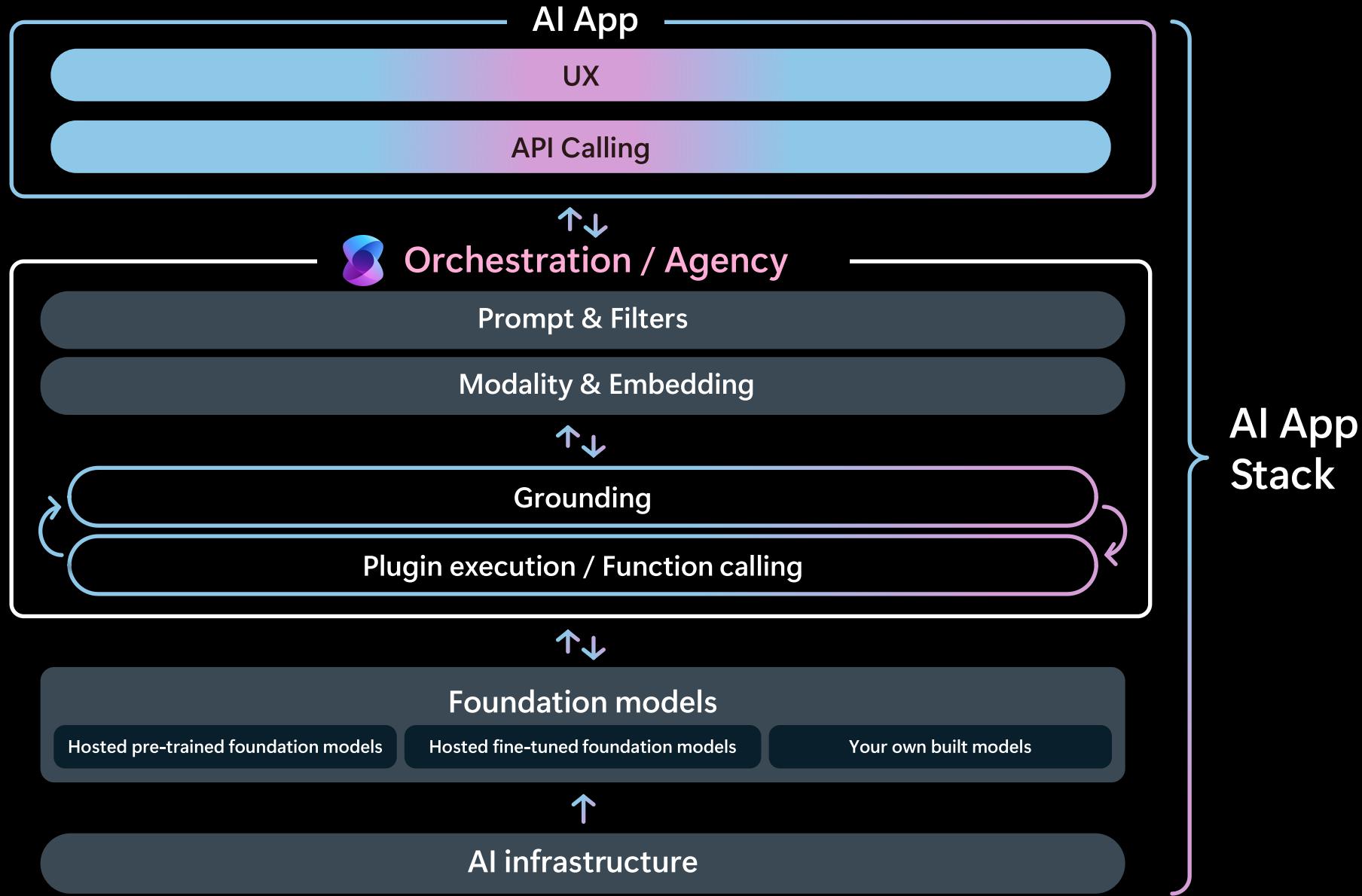
RogerBarreto



RogerBarreto

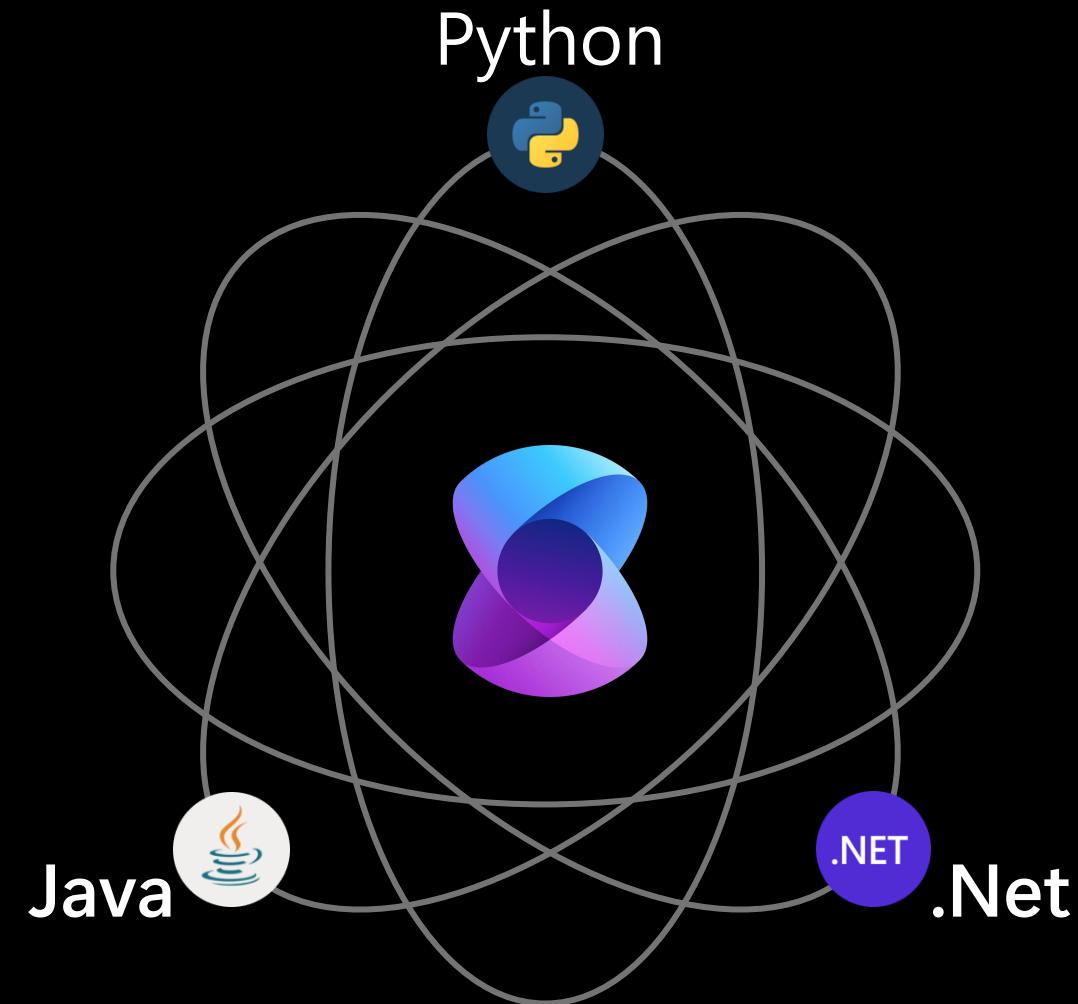


RogerBarretto





Semantic Kernel is a lightweight open-source orchestration SDK middleware that lets you easily integrate AI in your apps.





Our Product Pillars

**Open source  
Reliability  
Performance  
Trustworthy  
Extensible**



Our Product Pillars

# Open source

- **MIT License**
- **Visibility & Transparency**
- **Open for contribution**
- **Quick learning curve**
- **Forks & Mods**



# GitHub Repository

microsoft / semantic-kernel

Type ⌘ to search

Code Issues 400 Pull requests 38 Discussions Actions Projects 1 Security Insights

semantic-kernel Public

Edit Pins Unwatch 285 Fork 3.6k Starred 23.6k

main Branches Tags Go to file Add file Code

markwallace-microsoft .Net: Encode Kusto keys that contain a' (#11114) e96c693 · yesterday 4,175 Commits

.devcontainer Change version of .Net in dev container to 8 so it can run Po... 2 years ago

.github Use checkout with persist-credentials: false (#11072) 3 days ago

.vscode .Net: Add Calendar write for Copilot Agent Plugin Calendar (...) 3 weeks ago

docs .Net: Update Azure and OpenAI to latest beta.4 (#11073) 2 days ago

dotnet .Net: Encode Kusto keys that contain a' (#11114) yesterday

java Java: Migrate java code to new repository (#7460) 8 months ago

prompt\_template\_samples .Net: New Azure AI Inference Connector (#7963) 6 months ago

python Python: Add Copilot Studio Agents and Copilot Studio Skill d... 2 days ago

.editorconfig .Net: Google Gemini - Adding response schema (Structured ...) 2 months ago

.gitattributes .Net: Fix build batch to align with Bash script (#2813) 2 years ago

.gitignore .Net: chore/updates plugins (#9843) 3 months ago

CODE\_OF\_CONDUCT.md First commit, let the journey begin! 2 years ago

About

Integrate cutting-edge LLM technology quickly and easily into your apps

aka.ms/semantic-kernel

sdk ai artificial-intelligence openai llm

Readme MIT license Code of conduct Security policy Activity Custom properties

23.6k stars 285 watching 3.6k forks Report repository

Releases 190 python-1.25.0 Latest

Packages 95

- Microsoft.SemanticKernel.Core
- Microsoft.SemanticKernel.Abstractions
- Microsoft.SemanticKernel

+ 92 packages

Used by 1.2k

C + 1,146

Contributors 377

23.6k stars 285 watching 3.6k forks Report repository

Releases 190 python-1.25.0 Latest

Deployments 500+

integration 1 hour ago

+ more deployments



# GitHub Project

microsoft / Projects / Semantic Kernel

Type ⌘ to search

Semantic Kernel Increased items preview Feedback Add status update

Current Sprint @Me @Me Build Unassigned Build Current Backlog Backlog by Bucket Triage: No Bucket Please Close View 71 Bugs Community PRs Full Backlog Build by Assignee

Assignees

- ArieSLV 1
- EriksonBahr 1
- RogerBarreto 20
- SergeyMenshykh 16
- TaoChenOSU 3
- adamsitnik 8
- crickman 1
- dmytrostruk 13
- eavanvalkenburg 5
- esttenorio 1
- markwallace-microsoft 10
- moonbox3 12
- roji 3
- strdusty 1

-status:"No Status",Released,"Backlog",Epic,"V1 Drafts","Sprint: In Design" sprint:@current 102 Discard Save

**Sprint: In Progress** 15

This is actively being worked on

- semantic-kernel #10730 .Net: Support M.E.AI ChatClients with Kernel Filters .NET msft.ext.ai sk team issue
- semantic-kernel #10100 .Net: Memory for Agents .NET Build memory memory connector sk team issue SK-H2-Planning
- semantic-kernel #10896 .Net: Add hybrid search samples .NET Build memory connector sk team issue
- semantic-kernel #10149 .Net: Az Template Support .NET Build sk team issue SK-H2-Planning
- semantic-kernel #10887 + Add item

**Sprint: In Review** 8

SLA 3 Days. PRs coming from the core SK team

- semantic-kernel #10320 .Net: Microsoft.Extensions.AI Audio-to-Text Abstractions Work .NET Build sk team issue
- semantic-kernel #10262 .Net Bug: Adding a plugin to a Kernel instance does not retrieve dependencies from Kernel.Services by default. .NET bug function\_calling question sk team issue
- semantic-kernel #10729 .Net: Support M.E.AI Abstractions - Enable Agents to use IChatClient's natively from Kernel and DI. .NET msft.ext.ai sk team issue
- semantic-kernel #11019 Bug: Some SK Process unittest cause intermittent build failure bug processes #11008 + Add item

**Community PRs** 1

PRs coming from the public

- semantic-kernel #10021 .Net: add amazon nova support (text) only (WIP) .NET kernel PR: feedback to address 2 reviewers

**Sprint: Done** 37

This has been completed

- semantic-kernel #10950 .Net: Fix CodeQL Errors .NET #10937
- semantic-kernel #10933 .Net: Question: Is it possible to have ChatCompletionAgent connected to two or more data sources .NET ai connector
- semantic-kernel #10767 .Net: Possible Bug: context.Terminate Ignored When Using JSON Schema in .NET Semantic Kernel (1.39.0) .NET bug
- semantic-kernel #10940 .Net - Dependency Injection with Plugins that depend on other service .NET
- semantic-kernel #10069 + Add item



Our Product Pillars

# Performance

- **Fast & Lightweight**
- **Simple Async APIs**
- **Optimized execution paths**
- **Scales to millions of users**
- **Production-ready**



Our Product Pillars

# Reliability

- **Modular packages**
- **Telemetry & Logging enabled**
- **Dependency Injection enabled**
- **Break-glass flexibility**
- **Versioning stability**



# Packages Usage & Adoption

.Net: <https://www.nuget.org/packages?q=Microsoft.SemanticKernel>

**101 packages returned for Microsoft.SemanticKernel**

Sort by Relevance

**Microsoft.SemanticKernel** by Microsoft SemanticKernel

.NET 5.0 .NET Core 2.0 .NET Standard 2.0 .NET Framework 4.6.1

↓ 3,143,964 total downloads ⓘ last updated 6 days ago ⚡ Latest version: 1.42.0

◇ AI Artificial Intelligence SDK

Semantic Kernel common package collection, including SK Core, OpenAI, Azure OpenAI, DALL-E 2. Empowers app owners to integrate cutting-edge LLM technology quickly and easily into their apps.

**Microsoft.SemanticKernel.Abstractions** by Microsoft SemanticKernel

.NET 5.0 .NET Core 2.0 .NET Standard 2.0 .NET Framework 4.6.1

↓ 4,678,115 total downloads ⓘ last updated 6 days ago ⚡ Latest version: 1.42.0

◇ AI Artificial Intelligence SDK

Semantic Kernel interfaces and abstractions. This package is automatically installed by Semantic Kernel packages if needed.

**Microsoft.SemanticKernel.Core** by Microsoft SemanticKernel

.NET 5.0 .NET Core 2.0 .NET Standard 2.0 .NET Framework 4.6.1

↓ 4,453,420 total downloads ⓘ last updated 6 days ago ⚡ Latest version: 1.42.0

◇ AI Artificial Intelligence SDK

Semantic Kernel core orchestration, runtime and functions. This package is automatically installed by 'Microsoft.SemanticKernel' package with other useful packages. Install this package manually... [More information](#)

**Microsoft.SemanticKernel.Connectors.OpenAI** by Microsoft SemanticKernel

.NET 5.0 .NET Core 2.0 .NET Standard 2.0 .NET Framework 4.6.1

↓ 3,058,262 total downloads ⓘ last updated 6 days ago ⚡ Latest version: 1.42.0

◇ AI Artificial Intelligence SDK

Semantic Kernel connectors for OpenAI. Contains clients for chat completion, embedding and DALL-E text to image.

Python: <https://pypistats.org/packages/semantic-kernel>

**PyPI Stats semantic-kernel**

Search

All packages

Top packages

Track packages

PyPI page

Home page

Author: None

Summary: Semantic Kernel Python SDK

Latest version: 1.25.0

Required dependencies: aiohttp | aiortc | azure-identity | cloudevents | defusedxml | jinja2 | nest-asyncio | numpy | openai | openapi\_core | opentelemetry-api | opentelemetry-sdk | prance | pybars4 | pydantic | pydantic-settings | scipy | websockets

Optional dependencies: aiortc | anthropic | autogen-agentchat | azure-ai-inference | azure-ai-projects | azure-core-tracing-opentelemetry | azure-cosmos | azure-identity | azure-search-documents | boto3 | chromadb | dapr | dapr-ext-fastapi | faiss-cpu | flask-dapr | google-cloud-aiplatform | google-generativeai | ipykernel | milvus | mistralai | motor | ollama | onnxruntime-genai | pandas | pinecone | psycopg | pyarrow | pymilvus | pymongo | qrant-client | redis | redisv | sentence-transformers | torch | transformers | types-redis | usearch | weaviate-client | websockets

Downloads last day: 8,300

Downloads last week: 56,669

Downloads last month: 548,286



Our Product Pillars

# Trustworthy

- **Security checks**
- **Code coverage checks**
- **Code review and quality**
- **Rigor at Testing**
- **Feature discussion and open architecture records (ADRs)**
- **Minimum of 2 core team approvals**



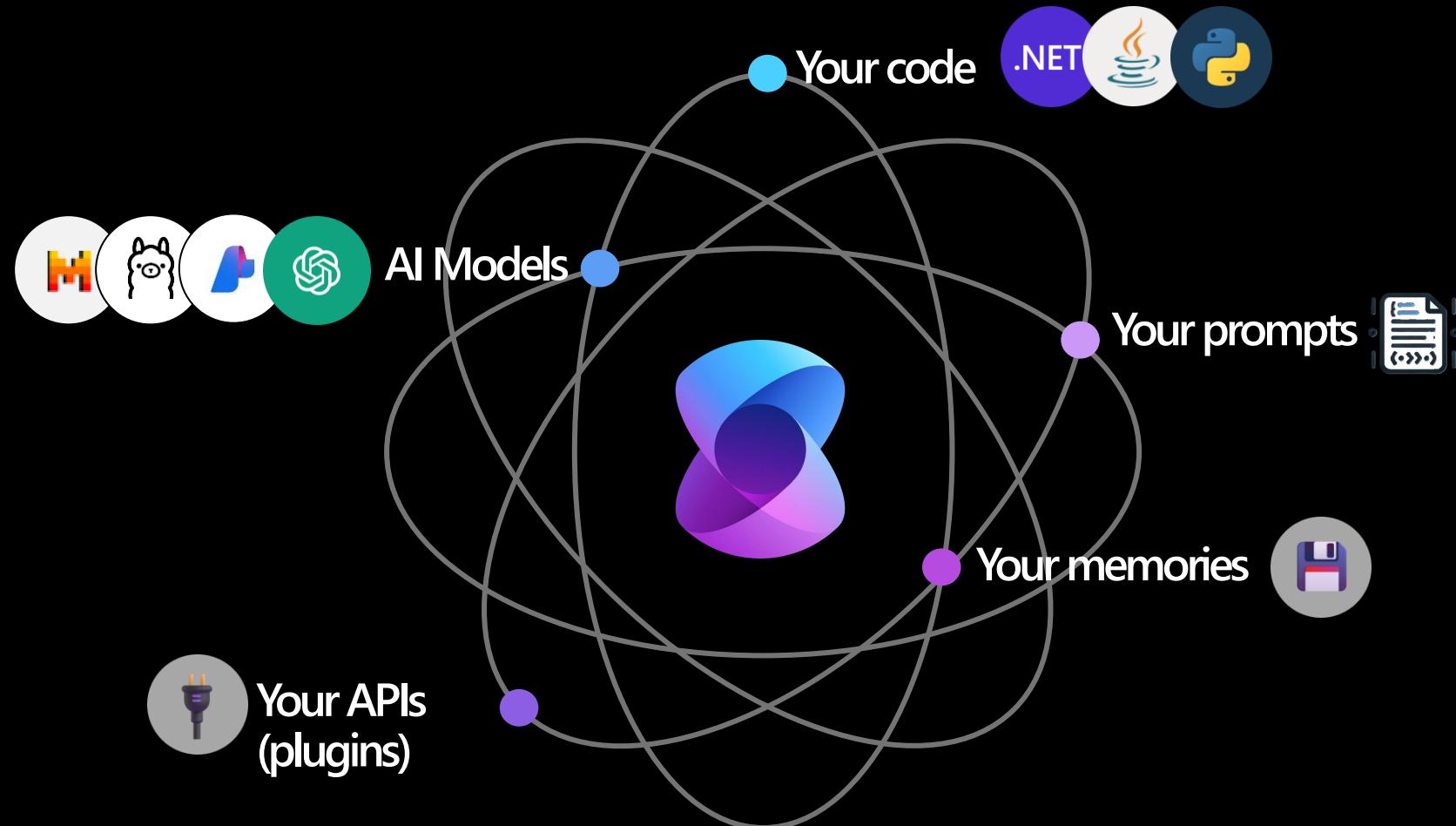
Our Product Pillars

# Extensibility

- **Built on top of abstractions**
- **Create your own customized SK component**
- **Multiple providers (and growing)**



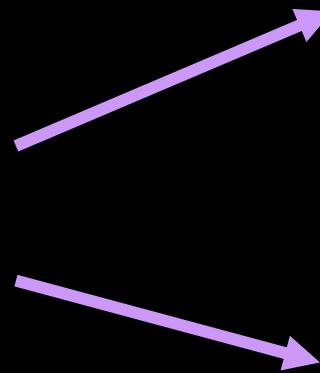
# Designed for modular extensibility



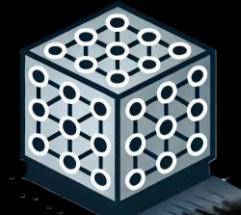


# Main components (AI Services)

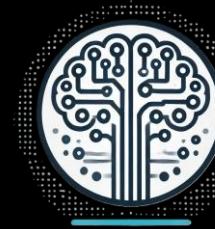
AI Services  
(Connectors)



Vector  
Databases



AI Models





# Main components (Plugins)

Your C# Classes



Open API Spec



Disk Files



Native  
Code



Plugins  
(Functions)



AI Models

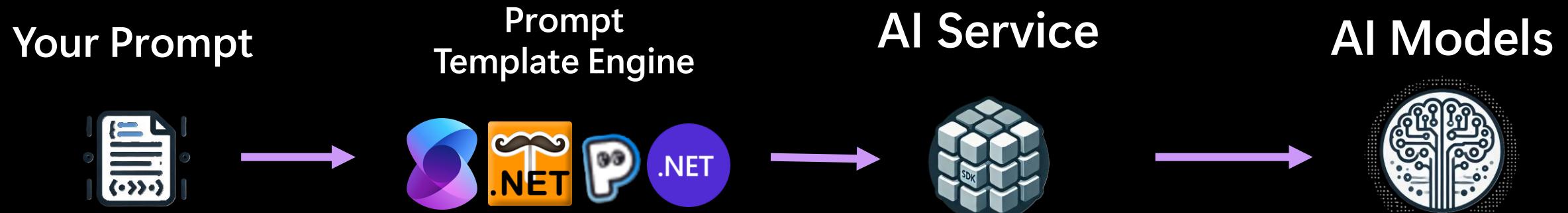


Prompt  
Template  
Engines



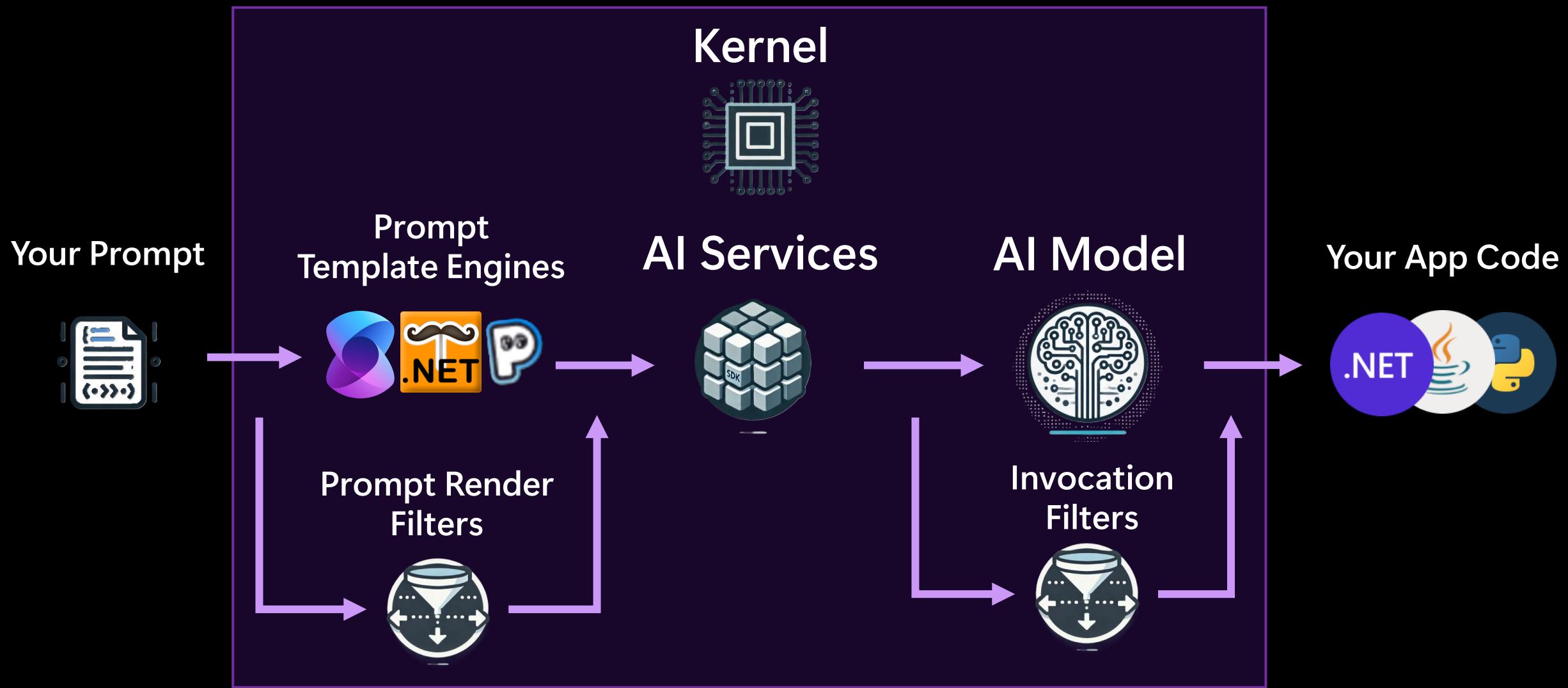


# Main components (Prompt Templates)



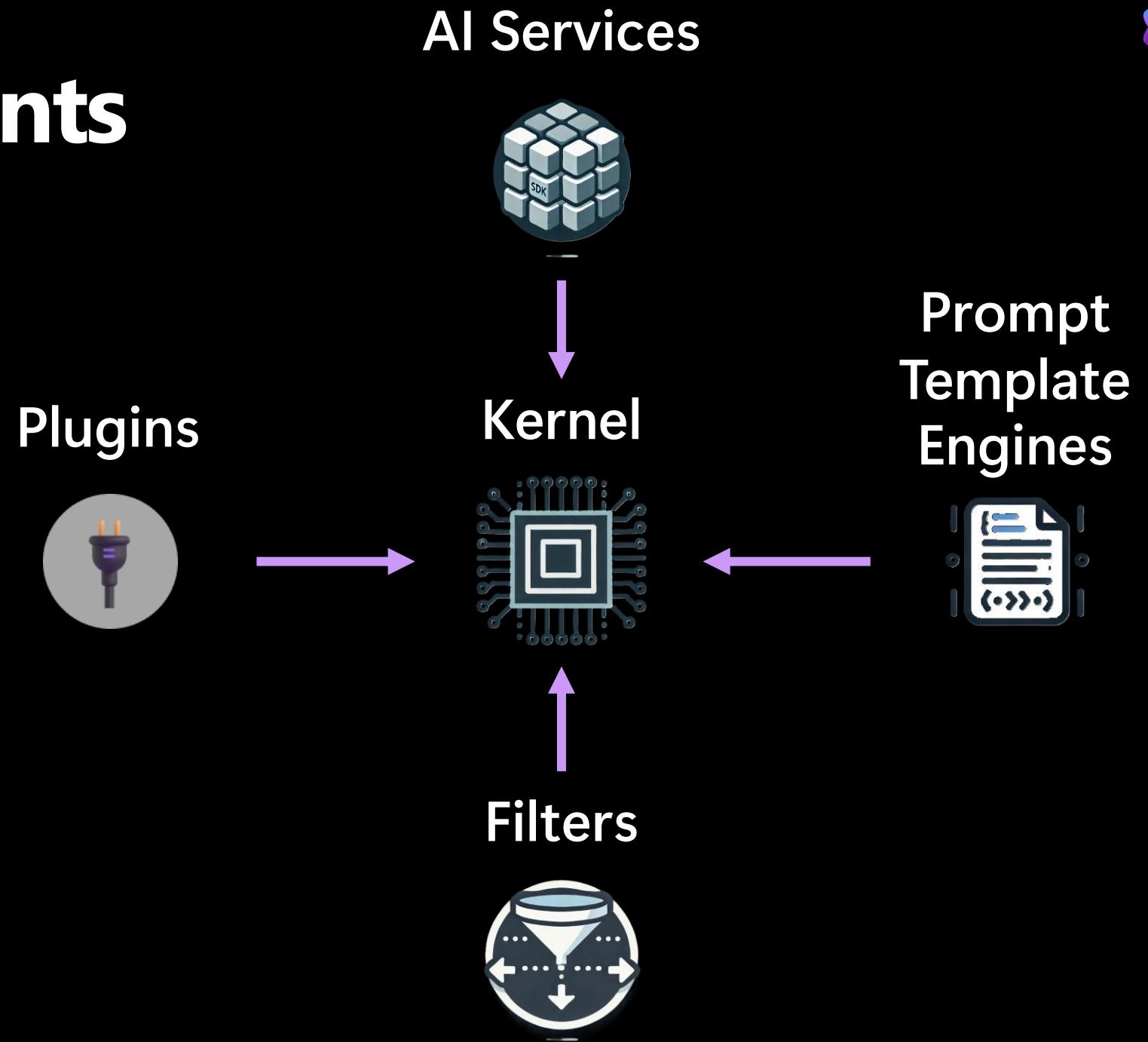


# Main components (Kernel Filters)





# Main components (Kernel)





# Additional components

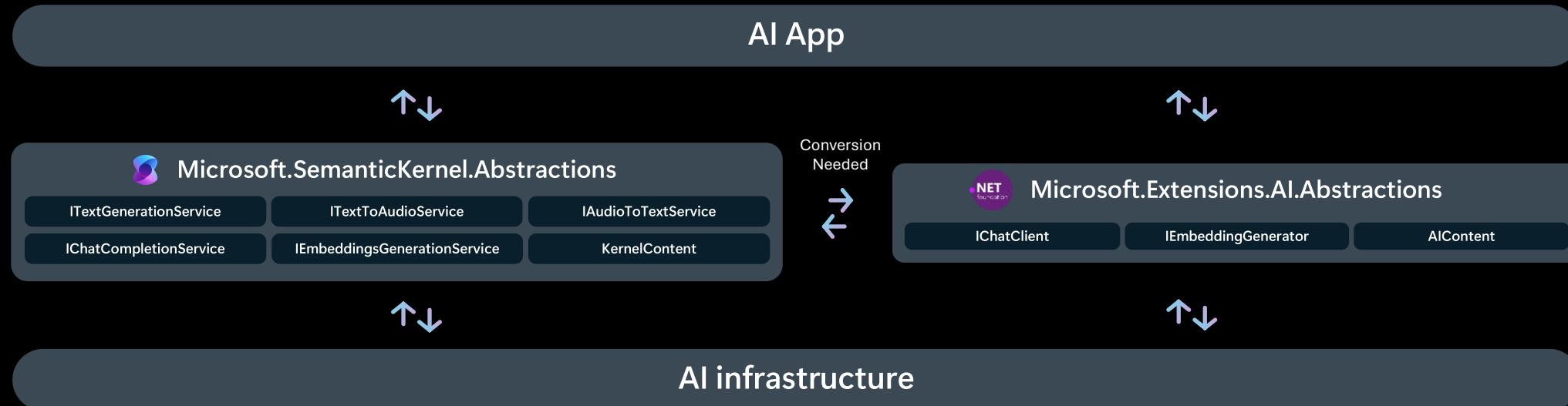
- AI Service Selector
- Prompt Template Engine
- Content Abstractions (Chat Messages, Image, Audio, Binary)
- Service Modalities (Text Generation, Chat Completion, Text-to-Image, etc)
- Vector Data & Text Search (Embeddings + Search)
- Microsoft Extensions AI Types



# Semantic Kernel + Microsoft.Extensions.AI (Preview)

Extensions project was released in last October as the foundation abstraction for AI types and services.

In case you already have M.E.AI code you can use it with Semantic Kernel simply thru our conversion extensions and vice-versa.



<https://devblogs.microsoft.com/dotnet/introducing-microsoft-extensions-ai-preview/>



# Semantic Kernel + Microsoft.Extensions.AI (GA)

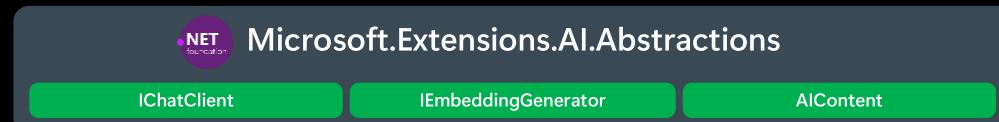
AI App



Near term, when M.E.AI becomes GA

SK will have native Support for M.E.AI Types

This change moves the M.E.AI abstractions as the foundation abstraction layer that semantic kernel can rely and use natively.



AI infrastructure



# Coding time!

[github.com/RogerBarreto/semantic-kernel-demos](https://github.com/RogerBarreto/semantic-kernel-demos)

README MIT license

## Semantic Kernel Demonstrations

This repository contains a collection of Semantic Kernel demonstration projects. Semantic Kernel is a powerful framework for building AI applications using Large Language Models (LLMs) and other AI models. These demonstrations showcase various use cases and features of Semantic Kernel, providing insights into its capabilities and potential applications.

### Beginner Hands-On Semantic Kernel Demos

A collection of .NET samples demonstrating various Semantic Kernel features:

1. [Kernel Prompting](#): Basic prompt handling and streaming
2. [Kernel Dependency Injection](#): Using DI with Semantic Kernel
3. [Kernel Prompt Template](#): Working with prompt templates and variables
4. [Chat Service with Chat History](#): Managing conversation history
5. [Prompt Execution Settings](#): Configuring prompt execution
6. [Grounding Prompts with Plugins](#): Using plugins for context
7. [Plugin Function Calling](#): Implementing and using plugins
8. [Using OpenAPI Plugins](#): Integrating external APIs via OpenAPI
9. [Using Kernel Filters](#): Implementing kernel filters
10. [Aspire Dashboard + OpenTelemetry](#): Monitoring and telemetry
11. [Multiple AI Models](#): Working with different AI providers
12. [Multi-Modality](#): Handling text, images, and audio



# 1. Kernel Prompting (Simplicity)

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel;
3
4  Console.WriteLine("==> Kernel prompting - Non-Streaming ==>\n");
5
6  var apiKey = new ConfigurationBuilder()
7      .AddUserSecrets<Program>()
8      .Build()["OpenAI:ApiKey"]!;
9
10 var kernel = Kernel.CreateBuilder()
11     .AddOpenAIChatCompletion("gpt-4o", apiKey)
12     .Build();
13 var prompt = "Why is Neptune blue?";
14
15 Console.WriteLine(await kernel.InvokePromptAsync(prompt));
16
17 Console.WriteLine("\n==> Kernel prompting - Streaming ==>\n");
18 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
19 {
20     Console.Write(token);
21 }
```

## Our Product Pillars

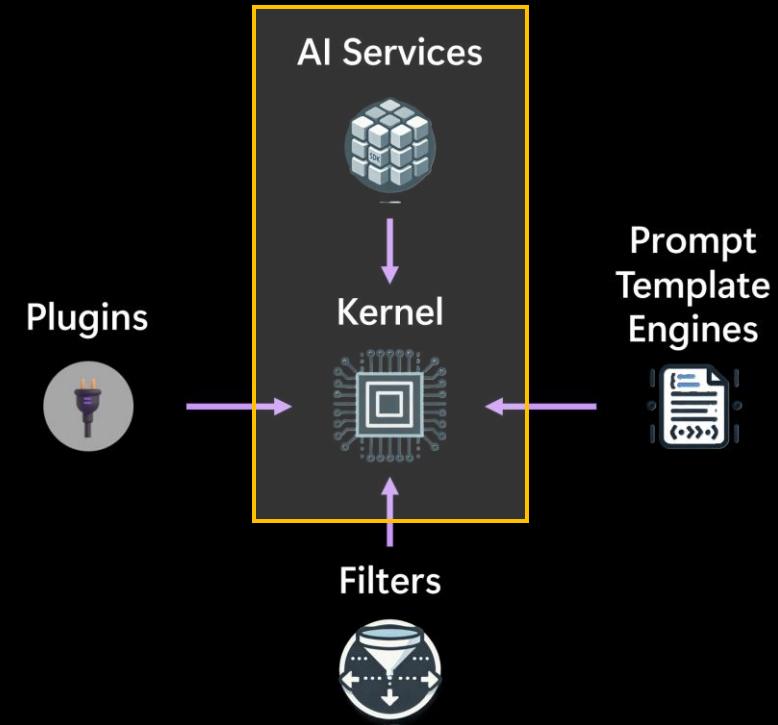
### Performance

- **Fast & Lightweight**
- **Simple Async APIs**
- **Optimized execution paths**
- **Scales to millions of users**
- **Production-ready**



# 1. Kernel Prompting (AI Service)

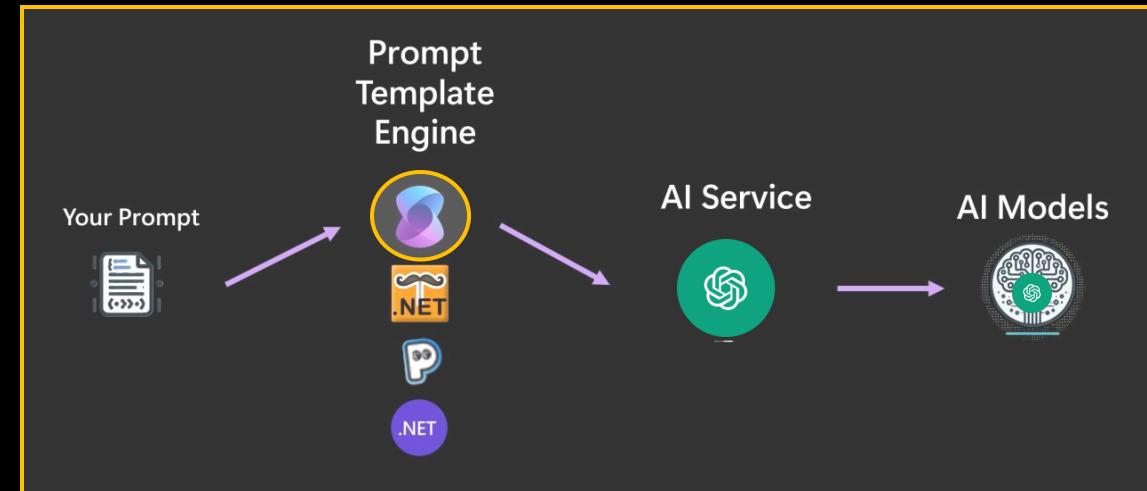
```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel;
3
4  Console.WriteLine("==> Kernel prompting - Non-Streaming ==>\n");
5
6  var apiKey = new ConfigurationBuilder()
7      .AddUserSecrets<Program>()
8      .Build()["OpenAI:ApiKey"]!;
9
10 var kernel = Kernel.CreateBuilder()
11     .AddOpenAIChatCompletion("gpt-4o", apiKey)
12     .Build();
13 var prompt = "Why is Neptune blue?";
14
15 Console.WriteLine(await kernel.InvokePromptAsync(prompt));
16
17 Console.WriteLine("\n==> Kernel prompting - Streaming ==>\n");
18 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
19 {
20     Console.Write(token);
21 }
```





# 1. Kernel Prompting (Invocation)

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel;
3
4  Console.WriteLine("==> Kernel prompting - Non-Streaming ==>\n");
5
6  var apiKey = new ConfigurationBuilder()
7      .AddUserSecrets<Program>()
8      .Build()["OpenAI:ApiKey"]!;
9
10 var kernel = Kernel.CreateBuilder()
11     .AddOpenAIChatCompletion("gpt-4o", apiKey)
12     .Build();
13 var prompt = "Why is Neptune blue?";
14
15 Console.WriteLine(await kernel.InvokePromptAsync(prompt));
16
17 Console.WriteLine("\n==> Kernel prompting - Streaming ==>\n");
18 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
19 {
20     Console.Write(token);
21 }
```





# 1. Kernel Prompting (Execution)

```
1  ✓  using Microsoft.Extensions.Configuration;
2    |  using Microsoft.SemanticKernel;
3
4    Console.WriteLine("==> Kernel prompting - Non-Streaming ==\n");
5
6    var apiKey = new ConfigurationBuilder()
7        .AddUserSecrets<Program>()
8        .Build()["OpenAI:ApiKey"]!;
9
10   var kernel = Kernel.CreateBuilder()
11       .AddOpenAIChatCompletion("gpt-4o", apiKey)
12       .Build();
13   var prompt = "Why is Neptune blue?";
14
15   Console.WriteLine(await kernel.InvokePromptAsync(prompt));
16
17   Console.WriteLine("\n==> Kernel prompting - Streaming ==\n");
18   await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
19   {
20     |   Console.Write(token);
21 }
```

```
D:\repo\work\conferences\se
```

```
==> Kernel prompting - Non-Streaming ==
Why is Neptune blue?

==> Kernel prompting - Streaming ==
Why is Neptune blue?
```



# 2. Kernel Dependency Injection

```
1  using Microsoft.Extensions.DependencyInjection;
2  using Microsoft.SemanticKernel;
3
4  #pragma warning disable SKEXP0070 // Type is for evaluation purposes on
5
6  Console.WriteLine("==> Kernel Dependency Injection ==\n");
7
8  var modelId = "llama3.2";
9  var endpoint = new Uri("http://localhost:11434");
10
11 var services = new ServiceCollection();
12 services.AddOllamaChatCompletion(modelId, endpoint);
13 services.AddKernel();
14
15 var serviceProvider = services.BuildServiceProvider();
16 var kernel = serviceProvider.GetRequiredService<Kernel>();
17
18 var prompt = "Why is Neptune blue?";
19 Console.WriteLine($"User > {prompt}");
20 Console.Write("Assistant > ");
21 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
22 {
23     Console.Write(token);
24 }
```

## Our Product Pillars

### Reliability

- Modular packages
- Telemetry & Logging enabled
- Dependency Injection enabled
- Break-glass flexibility
- Versioning stability



# 2. Kernel Dependency Injection

```
1  ↘  using Microsoft.Extensions.DependencyInjection;
2   |  using Microsoft.SemanticKernel;
3
4   #pragma warning disable SKEXP0070 // Type is for evaluation purposes only
5
6   Console.WriteLine("==> Kernel Dependency Injection ==\n");
7
8   var modelId = "llama3.2";
9   var endpoint = new Uri("http://localhost:11434");
10
11  var services = new ServiceCollection();
12  services.AddOllamaChatCompletion(modelId, endpoint);
13  services.AddKernel();
14
15  var serviceProvider = services.BuildServiceProvider();
16  var kernel = serviceProvider.GetRequiredService<Kernel>();
17
18  var prompt = "Why is Neptune blue?";
19  Console.WriteLine($"User > {prompt}");
20  Console.Write("Assistant > ");
21  await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
22  {
23   |  Console.Write(token);
24 }
```

## Our Product Pillars

### Reliability

- Modular packages
- Telemetry & Logging enabled
- **Dependency Injection enabled**
- Break-glass flexibility
- Versioning stability



# 2. Kernel Dependency Injection (Execution)

```
1  using Microsoft.Extensions.DependencyInjection;
2  using Microsoft.SemanticKernel;
3
4  #pragma warning disable SKEXP0070 // Type is for evaluation purposes only
5
6  Console.WriteLine("==> Kernel Dependency Injection ==\n");
7
8  var modelId = "llama3.2";
9  var endpoint = new Uri("http://localhost:11434");
10
11 var services = new ServiceCollection();
12 services.AddOllamaChatCompletion(modelId, endpoint);
13 services.AddKernel();
14
15 var serviceProvider = services.BuildServiceProvider();
16 var kernel = serviceProvider.GetRequiredService<Kernel>();
17
18 var prompt = "Why is Neptune blue?";
19 Console.WriteLine($"User > {prompt}");
20 Console.Write("Assistant > ");
21 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt))
22 {
23     Console.Write(token);
24 }
```

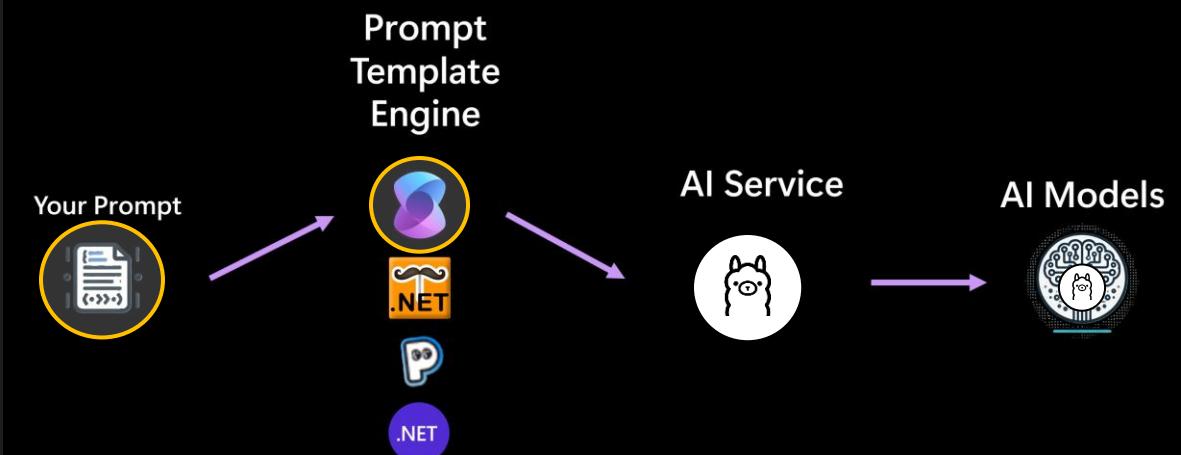
The screenshot shows a code editor window with the following details:

- Title Bar:** D:\repo\work\conferences\se
- Code Area:** Displays the C# code provided in the left panel.
- Output Area:** Below the code area, there is a large, empty dark gray box representing the terminal or output window where the application's response would be displayed.



# 3. Kernel Prompt Template (Basic)

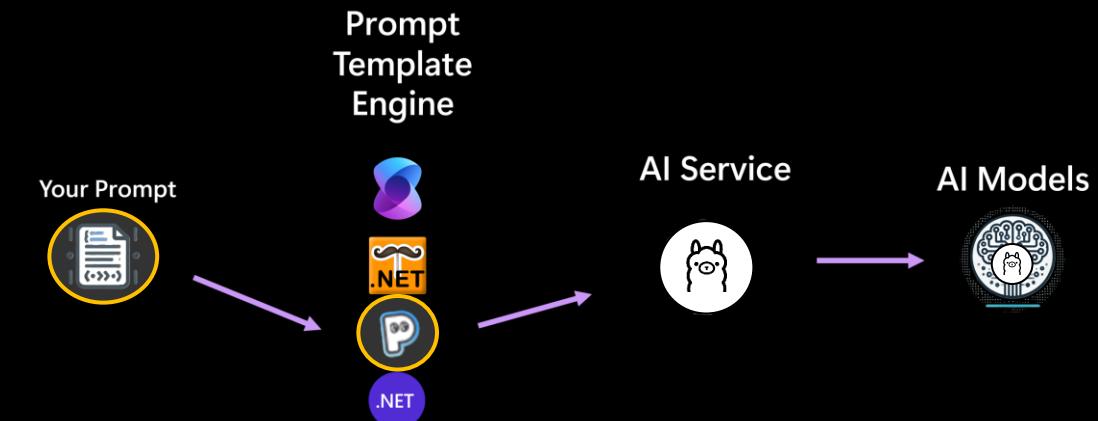
```
1  using Microsoft.SemanticKernel;
2
3  Console.WriteLine("== Kernel Basic Template & Argument variables ==\n");
4
5  var modelId = "phi4";
6  var endpoint = new Uri("http://localhost:11434");
7
8  var kernel = Kernel.CreateBuilder()
9      .AddOllamaChatCompletion(modelId, endpoint)
10     .Build();
11
12  var prompt = """
13      Hello, today is {{$date}}, I'm {{$name}}, and I have {{$age}}.
14      What you thin about my name and what technological advancement happened the year I was born?
15      """;
16
17  var arguments = new KernelArguments()
18  {
19      ["date"] = DateTime.Now.ToString("yyyy-MM-dd"),
20      ["name"] = "Roger",
21      ["age"] = 42
22  };
23
24  Console.WriteLine($"User Template\n---\n{prompt}\n---\n");
25  Console.Write("Assistant > ");
26  await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, arguments))
27  {
28      Console.Write(token);
29  }
```





# 3. Kernel Prompt Template (Prompty)

```
17     var arguments = new KernelArguments()
18     {
19         ["date"] = DateTime.Now.ToString("yyyy-MM-dd"),
20         ["name"] = "Roger",
21         ["age"] = 42
22     };
23
24
25
26
27
28
29
30
31
32
33
34
35     Console.WriteLine("==> Kernel Prompty Template & Argument variables ===\n");
36
37     var promptyTemplate = """
38     ---
39     name: PromptySample
40     ---
41     Hello, today is {{date}}, I'm {{name}}, and I have {{age}}.
42     What you thin about my name and what technological advancement happened the year I was born?
43     """;
44
45     #pragma warning disable SKEXP0040 // Type is for evaluation purposes only and is subject to change
46     var promptyFunction = kernel.CreateFunctionFromPrompty(promptyTemplate);
47
48     Console.WriteLine($"User Template\n{promptyTemplate}\n---\n");
49     Console.Write("Assistant > ");
50     await foreach (var token in kernel.InvokeStreamingAsync(promptyFunction, arguments))
51     {
52         Console.Write(token);
53     }
```





# 3. Kernel Prompt Template (Basic Exec.)

```
1  using Microsoft.SemanticKernel;
2
3  Console.WriteLine("==> Kernel Basic Template & Argument variables ==\n");
4
5  var modelId = "phi4";
6  var endpoint = new Uri("http://localhost:11434");
7
8  var kernel = Kernel.CreateBuilder()
9    .AddOllamaChatCompletion(modelId, endpoint)
10   .Build();
11
12  var prompt = """
13    Hello, today is {{$date}}, I'm {{$name}}, and I have {{$age}}.
14    What you thin about my name and what technological advancement happened the year I was born?
15  """;
16
17  var arguments = new KernelArguments()
18  {
19    ["date"] = DateTime.Now.ToString("yyyy-MM-dd"),
20    ["name"] = "Roger",
21    ["age"] = 42
22  };
23
24  Console.WriteLine($"User Template\n---\n{prompt}\n---\n");
25  Console.Write("Assistant > ");
26  await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, arguments))
27  {
28    Console.Write(token);
29 }
```

```
D:\repo\work\conferences\se
```



# 3. Kernel Prompt Template (Prompty Exec.)

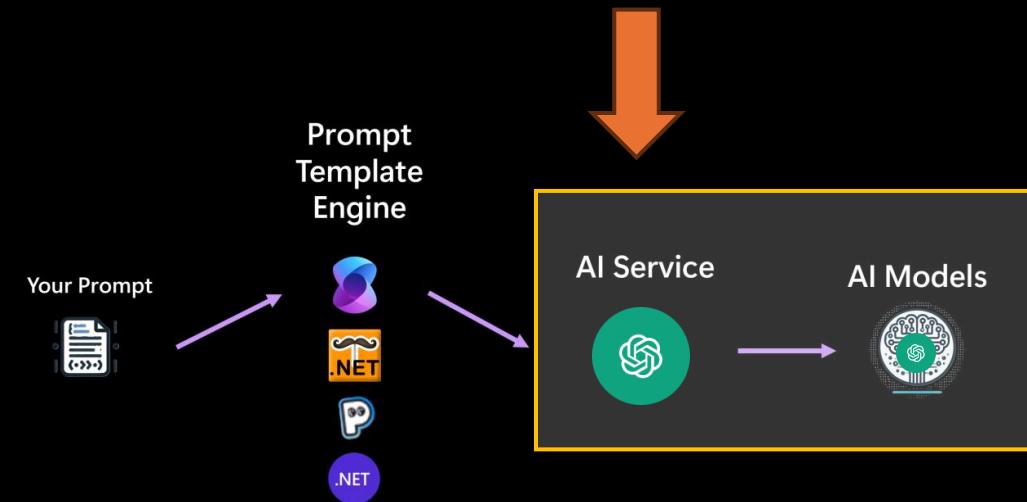
```
17     var arguments = new KernelArguments()
18     {
19         ["date"] = DateTime.Now.ToString("yyyy-MM-dd"),
20         ["name"] = "Roger",
21         ["age"] = 42
22     };
23
24
25
26
27
28
29
30
31
32
33
34
35     Console.WriteLine("==> Kernel Prompty Template & Argument variables ==\n");
36
37     var promptyTemplate = """
38     ---
39     name: PromptySample
40     ---
41     Hello, today is {{date}}, I'm {{name}}, and I have {{age}}.
42     What you thin about my name and what technological advancement happened the year I was born?
43     """;
44
45 #pragma warning disable SKEXP0040 // Type is for evaluation purposes only and is subject to change
46 var promptyFunction = kernel.CreateFunctionFromPrompty(promptyTemplate);
47
48 Console.WriteLine($"User Template\n{promptyTemplate}\n---\n");
49 Console.Write("Assistant > ");
50 await foreach (var token in kernel.InvokeStreamingAsync(promptyFunction, arguments))
51 {
52     Console.Write(token);
53 }
```

```
D:\repo\work\conferences\se
==> Kernel Prompty Template & Argument variables ==
User Template
---
name: PromptySample
---
Hello, today is {{date}}, I'm {{name}}, and I have {{age}}.
What you thin about my name and what technological advancement happened the year I was born?
---
Assistant > Hi Roger!
```



# 4. Chat Service with Chat History

```
1  ✓  using Microsoft.Extensions.Configuration;
2    |  using Microsoft.SemanticKernel.ChatCompletion;
3    |  using Microsoft.SemanticKernel.Connectors.OpenAI;
4
5    Console.WriteLine("==> Chat Service Streaming with Chat History ==\n");
6
7    var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
8    var modelId = "o1-mini";
9    var chatService = new OpenAIChatCompletionService(modelId, apiKey);
10
11   var userPrompt = "Why is Neptune blue?";
12
13   ChatHistory chatHistory = [];
14   chatHistory.AddSystemMessage("You are presenting in a conference");
15   chatHistory.AddUserMessage(userPrompt);
16
17   Console.WriteLine($"User > {userPrompt}");
18   Console.Write("Assistant > ");
19   await foreach (var token in chatService.GetStreamingChatMessageContentsAsync(chatHistory))
20   {
21     |  Console.Write(token);
22   }
```





# 4. Chat Service with Chat History (Execution)

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel.ChatCompletion;
3  using Microsoft.SemanticKernel.Connectors.OpenAI;
4
5  Console.WriteLine("== Chat Service Streaming with Chat History ==\n");
6
7  var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
8  var modelId = "o1-mini";
9  var chatService = new OpenAIChatCompletionService(modelId, apiKey);
10
11 var userPrompt = "Why is Neptune blue?";
12
13 ChatHistory chatHistory = [];
14 chatHistory.AddSystemMessage("You are presenting in a conference");
15 chatHistory.AddUserMessage(userPrompt);
16
17 Console.WriteLine($"User > {userPrompt}");
18 Console.Write("Assistant > ");
19 await foreach (var token in chatService.GetStreamingChatMessageContentsAsync(chatHistory))
20 {
21     Console.Write(token);
22 }
```

```
D:\repo\work\conferences\se  X + ▾
== Chat Service Streaming with Chat History ==

User > Why is Neptune blue?
Assistant >
```



# 5. Prompt Execution Settings

```
1  ↘  using Microsoft.SemanticKernel;
2    |  using Microsoft.SemanticKernel.ChatCompletion;
3    |  using Microsoft.SemanticKernel.Connectors.Ollama;
4
5    Console.WriteLine("==> Using prompt execution settings ==\n");
6
7    var modelId = "llama3.2";
8    var endpoint = new Uri("http://localhost:11434");
9    var prompt = "Write a 5-word sentence describing a sunny day.";
10
11   // At a low temperature, you'd expect a straightforward, predictable response like "Warm sunshine fills the air."
12   var lowTempSettings = new OllamaPromptExecutionSettings { Temperature = 0 };
13
14   // At a high temperature, you might get something more unusual or poetic, like "Warmth filled the bright sky."
15   var highTempSettings = new OllamaPromptExecutionSettings { Temperature = 1.0f };
16
17   var kernel = Kernel.CreateBuilder()
18     .AddOllamaChatCompletion(modelId, endpoint)
19     .Build();
20
21   // Get the service from the kernel
22   var service = kernel.GetRequiredService<IChatCompletionService>();
23
24   Console.WriteLine("\nKernel (low temperature):");
25   await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, new(lowTempSettings)))
26   {
27     |  Console.Write(token);
28   }
29
30   Console.WriteLine("\n\nChat Service (high temperature):");
31   await foreach (var token in service.GetStreamingChatMessageContentsAsync(prompt, highTempSettings))
32   {
33     |  Console.Write(token);
34   }
```



# 5. Prompt Execution Settings

```
1  using Microsoft.SemanticKernel;
2  using Microsoft.SemanticKernel.ChatCompletion;
3  using Microsoft.SemanticKernel.Connectors.Ollama;
4
5  Console.WriteLine("==> Using prompt execution settings ==\n");
6
7  var modelId = "llama3.2";
8  var endpoint = new Uri("http://localhost:11434");
9  var prompt = "Write a 5-word sentence describing a sunny day.";
10
11 // At a low temperature, you'd expect a straightforward, predictable response like "Warm sunshine fills the air."
12 var lowTempSettings = new OllamaPromptExecutionSettings { Temperature = 0 };
13
14 // At a high temperature, you might get something more unusual or poetic, like "Warmth filled the bright sky."
15 var highTempSettings = new OllamaPromptExecutionSettings { Temperature = 1.0f };
16
17 var kernel = Kernel.CreateBuilder()
18     .AddOllamaChatCompletion(modelId, endpoint)
19     .Build();
20
21 // Get the service from the kernel
22 var service = kernel.GetRequiredService<IChatCompletionService>();
23
24 Console.WriteLine("\nKernel (low temperature):");
25 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, new(lowTempSettings)))
26 {
27     Console.Write(token);
28 }
29
30 Console.WriteLine("\n\nChat Service (high temperature):");
31 await foreach (var token in service.GetStreamingChatMessageContentsAsync(prompt, highTempSettings))
32 {
33     Console.Write(token);
34 }
```

```
D:\repo\work\conferences>
```

```
==> Using prompt execution settings ==
At a low temperature, you'd expect a straightforward, predictable response like "Warm sunshine fills the air."
Warm sunshine fills the air.

At a high temperature, you might get something more unusual or poetic, like "Warmth filled the bright sky."
Warmth filled the bright sky.
```



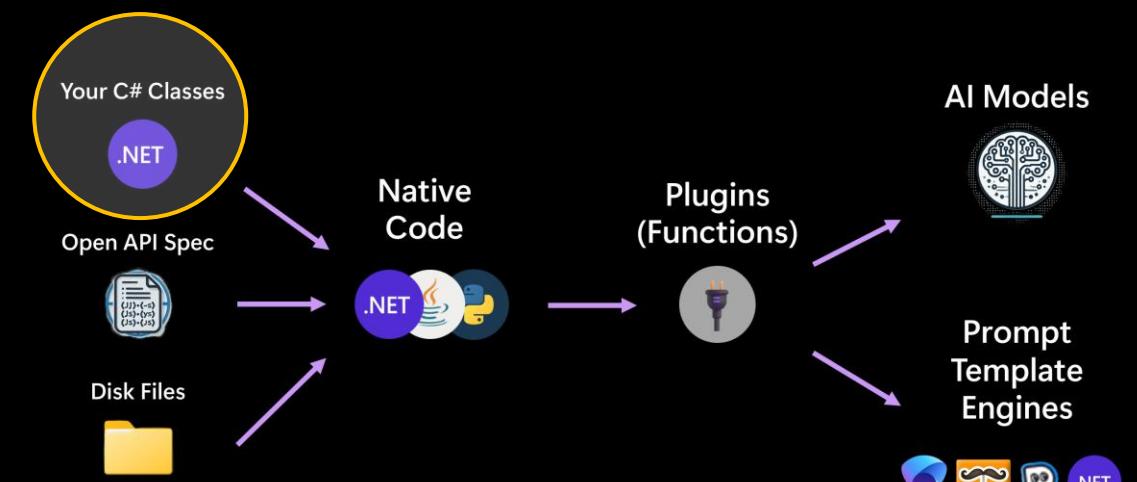
# 6. Plugins – Template Grounding

## Light Bulb Plugin (Stateful)

```
6   /// <summary>
7   /// Stateful plugin (It keeps the current bulb state)
8   /// </summary>
9   /// <param name="isOn">At the initialization, defines if the bulb is on or off</param>
10  public class LightBulbPlugin(bool isOn = false)
11  {
12      const string foregroundYellow = "\u001b[93m";
13      const string foregroundReset = "\u001b[0m";
14
15      private bool _isOn = isOn;
16
17      [KernelFunction, Description("Checks if the light bulb status is on or off")]
18      3 references | Roger Barreto, 23 hours ago | 1 author, 1 change
19      public string GetStatus()
20      {
21          Console.WriteLine($"{foregroundYellow}Plugin Triggered: {nameof(LightBulbPlugin)}");
22
23          return this._isOn ? "on" : "off";
24      }
25
26      [KernelFunction, Description("Switches the light bulb status")]
27      3 references | Roger Barreto, 23 hours ago | 1 author, 1 change
28      public string Switch()
29      {
30          Console.WriteLine($"{foregroundYellow}Plugin Triggered: {nameof(LightBulbPlugin)}");
31
32          this._isOn = !this._isOn;
33
34          return "switches";
35      }
36  }
```

## Time Plugin (Stateless)

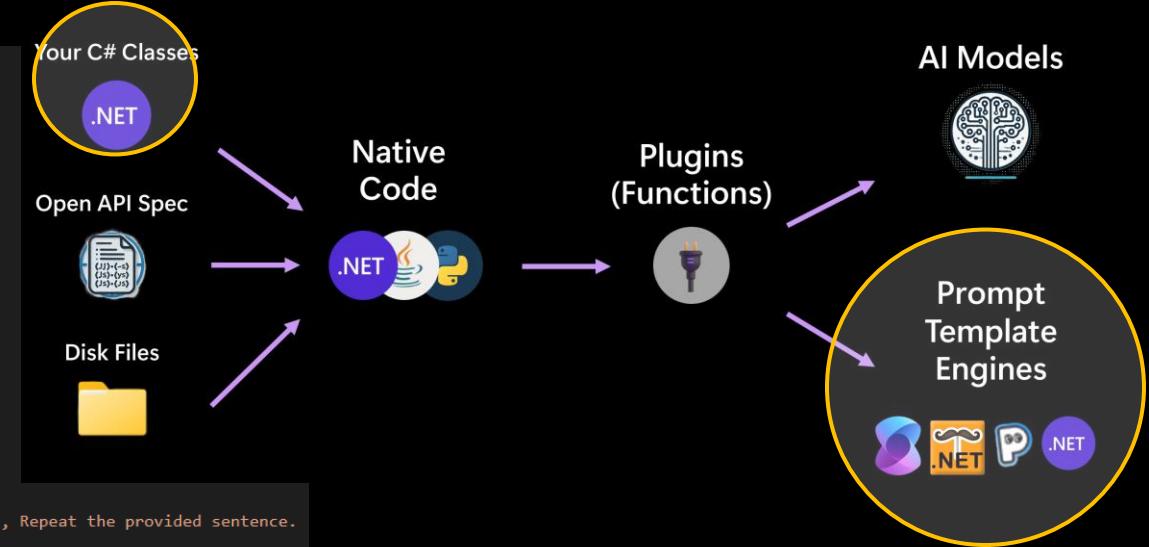
```
5   /// <summary>
6   /// Stateless plugin (It does not keep state)
7   /// </summary>
8   2 references | Roger Barreto, 23 hours ago | 1 author, 1 change
9   public class TimePlugin
10  {
11      const string foregroundYellow = "\u001b[93m";
12      const string foregroundReset = "\u001b[0m";
13
14      [KernelFunction]
15      0 references | Roger Barreto, 23 hours ago | 1 author, 1 change
16      public static string GetDateTime()
17      {
18          Console.WriteLine($"{foregroundYellow}Plugin Triggered: {nameof(TimePlugin)}");
19
20          var currentTime = DateTime.UtcNow.ToString("R");
21
22          return currentTime;
23      }
24  }
```





# 6. Plugins – Template Grounding (Functions)

```
1  using Microsoft.SemanticKernel;
2  using Sample;
3
4  Console.WriteLine("==> Grounding Prompts with Stateful and Stateless Plugins ==>\n");
5
6  var modelId = "llama3.2";
7  var endpoint = new Uri("http://localhost:11434");
8  var kernel = Kernel.CreateBuilder()
9    .AddOllamaChatCompletion(modelId, endpoint)
10   .Build();
11
12  var myLightBulbPlugin = new LightBulbPlugin(isOn: true);
13  kernel.Plugins.AddFromObject(myLightBulbPlugin);
14  kernel.Plugins.AddFromType<TimePlugin>();
15
16  var arguments = new KernelArguments
17  {
18    ["name"] = "Roger"
19  };
20
21  var prompt = """
22    | At {{GetDateTime}}. {{$name}} looked at the light bulb and see that the light is {{GetStatus}} and then {{Switch}} it., Repeat the provided sentence.
23  """
24  Console.WriteLine("Take 1: ");
25  await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, arguments))
26  {
27    Console.Write(token);
28  }
29
30  Console.WriteLine("\n---");
31
32  prompt """
33    | At {{GetDateTime}}. {{$name}} looked at the light bulb and see that the light is {{GetStatus}}., Repeat the provided sentence.
34  """
35  Console.WriteLine("Take 2: ");
36  await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, arguments))
37  {
38    Console.Write(token);
39  }
```





# 6. Plugins – Template Grounding (Execution)

```
1  using Microsoft.SemanticKernel;
2  using Sample;
3
4  Console.WriteLine("==> Grounding Prompts with Stateful and Stateless Plugins ==\n");
5
6  var modelId = "llama3_2";
7  var endpoint = new Uri("http://localhost:11434");
8  var kernel = Kernel.CreateBuilder()
9      .AddDefaultChatCompletion(modelId, endpoint)
10     .Build();
11
12 var myLightBulbPlugin = new LightBulbPlugin(isOn: true);
13 kernel.Services.AddScoped<ILightBulbPlugin>(myLightBulbPlugin);
14 kernel.Plugins.Add(kernelType: KernelType.TimerOrLogin);
15
16 var arguments = new KernelArguments
17 {
18     [{"name": "Roger"}];
19 }
20
21 var prompt = """
22 |At {{GetDateTime}}: {{$name}} looked at the light bulb and see that the light is {{GetStatus}} and then {{Switch}} it., Repeat the provided sentence.
23 |""";
24 Console.WriteLine("Take 1:");
25 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, arguments))
26 {
27     Console.Write(token);
28 }
29
30 Console.WriteLine("\n----");
31
32 prompt = """
33 |At {{GetDateTime}}: {{$name}} looked at the light bulb and see that the light is {{GetStatus}}., Repeat the provided sentence.
34 |""";
35 Console.WriteLine("Take 2:");
36 await foreach (var token in kernel.InvokePromptStreamingAsync(prompt, arguments))
37 {
38     Console.Write(token);
39 }
```

```
D:\repo\work\conferences\se
```

```
==> Grounding Prompts with Stateful and Stateless Plugins ==

At 2024-02-27T14:45:23.123Z: Roger looked at the light bulb and see that the light is On and then Turn it., Repeat the provided sentence.

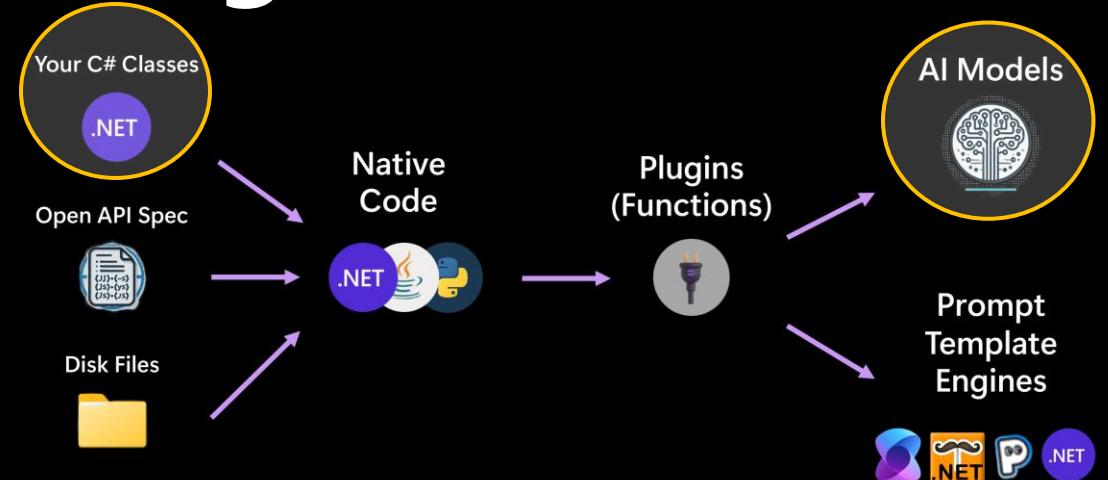
Take 1:
At 2024-02-27T14:45:23.123Z: Roger looked at the light bulb and see that the light is On., Repeat the provided sentence.

Take 2:
```



# 7. Plugins – AI Function Calling

```
1  using Microsoft.SemanticKernel;
2  using Microsoft.SemanticKernel.ChatCompletion;
3  using Microsoft.SemanticKernel.Connectors.Ollama;
4  using Sample;
5
6  Console.WriteLine("== Plugin Function Calling with Stateful and Stateless Plugins ==");
7
8  var modelId = "llama3.2";
9  var endpoint = new Uri("http://localhost:11434");
10 var kernel = Kernel.CreateBuilder()
11     .AddOllamaChatCompletion(modelId, endpoint)
12     .Build();
13
14 var lightBulbPlugin = new LightBulbPlugin(isOn: true);
15 kernel.Plugins.AddFromObject(lightBulbPlugin);
16 kernel.Plugins.AddFromType<TimePlugin>();
17
18 var service = kernel.GetRequiredService<IChatCompletionService>();
19
20 var settings = new OllamaPromptExecutionSettings { FunctionChoiceBehavior = FunctionChoiceBehavior.Auto() };
21
22 var prompt = "What is the light status and what time of the day is it?";
23 Console.WriteLine($"\\nUser: {prompt}");
24
25 var result = await kernel.InvokePromptAsync(prompt, new(settings));
26 Console.WriteLine($"Assistant: {result}");
27
28 prompt = "Please switch the light.";
29 Console.WriteLine($"\\nUser: {prompt}");
30
31 result = await kernel.InvokePromptAsync(prompt, new(settings));
32 Console.WriteLine($"Assistant: {result}");
33
34 prompt = "What is the light bulb status and what time of the day is it?";
35 Console.WriteLine($"\\nUser: {prompt}");
36
37 var contentResult = await service.GetChatMessageContentAsync(prompt, settings, kernel);
38 Console.WriteLine($"Assistant: {contentResult}");
39
```



## Light Bulb Plugin (Stateful)

3 references | Roger Barreto, 1 day ago | 1 author, 1 change  
public class LightBulbPlugin(bool isOn = false)

```
[  
    [KernelFunction, Description("Checks if the light bulb status is on or off")]  
    0 references | Roger Barreto, 1 day ago | 1 author, 1 change  
    public string GetStatus()...  
  
    [KernelFunction, Description("Switches the light bulb status")]  
    0 references | Roger Barreto, 1 day ago | 1 author, 1 change  
    public string Switch()...
```



# 7. Plugins – AI Function Calling (Execution)

```
1  ✓ using Microsoft.SemanticKernel;
2  using Microsoft.SemanticKernel.ChatCompletion;
3  using Microsoft.SemanticKernel.Connectors.Ollama;
4  using Sample;
5
6  Console.WriteLine("==> Plugin Function Calling with Stateful and Stateless Plugins ==>");
7
8  var modelId = "llama3.2";
9  var endpoint = new Uri("http://localhost:11434");
10 var kernel = Kernel.CreateBuilder()
11     .AddOllamaChatCompletion(modelId, endpoint)
12     .Build();
13
14 var lightBulbPlugin = new LightBulbPlugin(isOn: true);
15 kernel.Plugins.AddFromObject(lightBulbPlugin);
16 kernel.Plugins.AddFromType<TimePlugin>();
17
18 var service = kernel.GetRequiredService<IChatCompletionService>();
19
20 var settings = new OllamaPromptExecutionSettings { FunctionChoiceBehavior = FunctionChoiceBehavior.Auto() };
21
22 var prompt = "What is the light status and what time of the day is it?";
23 Console.WriteLine($"<User: {prompt}>");
24
25 var result = await kernel.InvokePromptAsync(prompt, new(settings));
26 Console.WriteLine($"Assistant: {result}");
27
28 prompt = "Please switch the light.";
29 Console.WriteLine($"<User: {prompt}>");
30
31 result = await kernel.InvokePromptAsync(prompt, new(settings));
32 Console.WriteLine($"Assistant: {result}");
33
34 prompt = "What is the light bulb status and what time of the day is it?";
35 Console.WriteLine($"<User: {prompt}>");
36
37 var contentResult = await service.GetChatMessageContentAsync(prompt, settings, kernel);
38 Console.WriteLine($"Assistant: {contentResult}");
```

```
D:\repo\work\conferences\se
```

```
==> Plugin Function Calling with Stateful and Stateless Plugins ==>
[1]: var modelId = "llama3.2";
[1]: var endpoint = new Uri("http://localhost:11434");
[1]: var kernel = Kernel.CreateBuilder()
[1]:     .AddOllamaChatCompletion(modelId, endpoint)
[1]:     .Build();
[1]: 
[1]: var lightBulbPlugin = new LightBulbPlugin(isOn: true);
[1]: kernel.Plugins.AddFromObject(lightBulbPlugin);
[1]: kernel.Plugins.AddFromType<TimePlugin>();
[1]: 
[1]: var service = kernel.GetRequiredService<IChatCompletionService>();
[1]: 
[1]: var settings = new OllamaPromptExecutionSettings { FunctionChoiceBehavior = FunctionChoiceBehavior.Auto() };
[1]: 
[1]: var prompt = "What is the light status and what time of the day is it?";
[1]: Console.WriteLine($"<User: {prompt}>");
[1]: 
[1]: var result = await kernel.InvokePromptAsync(prompt, new(settings));
[1]: Console.WriteLine($"Assistant: {result}");
[1]: 
[1]: prompt = "Please switch the light.";
[1]: Console.WriteLine($"<User: {prompt}>");
[1]: 
[1]: result = await kernel.InvokePromptAsync(prompt, new(settings));
[1]: Console.WriteLine($"Assistant: {result}");
[1]: 
[1]: prompt = "What is the light bulb status and what time of the day is it?";
[1]: Console.WriteLine($"<User: {prompt}>");
[1]: 
[1]: var contentResult = await service.GetChatMessageContentAsync(prompt, settings, kernel);
[1]: Console.WriteLine($"Assistant: {contentResult}");
```



# 8. OpenAPI Plugins – AI Function Calling

https://localhost:7299/swagger/index.html

Select a definition Weather API v1

## Weather API v1 OAS 3.0

A simple weather forecast API

### WebApiWeather

GET /weatherforecast

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7299/weatherforecast' \
  -H 'accept: application/json'
```

Request URL

https://localhost:7299/weatherforecast

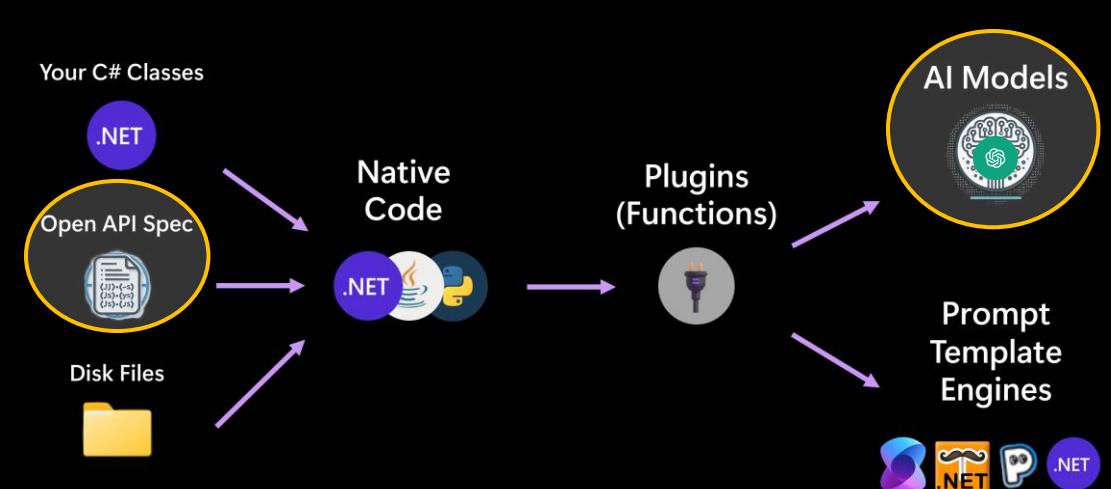
Server response

Code Details

200 Response body

```
[{"date": "2022-03-26", "temperatureC": 26, "temperatureF": 78, "summary": "Mild"}]
```

```
1  using Microsoft.SemanticKernel;
2  using Microsoft.SemanticKernel.Plugins.OpenApi;
3  using Microsoft.Extensions.Configuration;
4  using Sample;
5
6  var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
7
8  var kernel = Kernel.CreateBuilder()
9      .AddOpenAIChatCompletion("gpt-4o-mini", apiKey)
10     .Build();
11
12 var weatherOpenApiUri = new Uri("https://localhost:7299/swagger/v1/swagger.json");
13 var weatherPlugin = await OpenApiKernelPluginFactory.CreateFromOpenApiAsync("WeatherService", weatherOpenApiUri);
14
15 kernel.Plugins.Add(weatherPlugin);
16 kernel.Plugins.AddFromType<TimePlugin>(); // Enable AI to know which day is today
17
18 var settings = new PromptExecutionSettings { FunctionChoiceBehavior = FunctionChoiceBehavior.Auto() };
19
20 var prompt = "What is the weather forecast for tomorrow? Ensure you check which day is today.";
21 Console.WriteLine($"\\nUser > {prompt}");
22 var result = await kernel.InvokePromptAsync(prompt, new(settings));
23
24 Console.WriteLine($"Assistant > {result}");
```





# 8. OpenApi Plugins – AI Function Calling (Exec)

The image shows two terminal windows side-by-side. Both windows have a dark theme and are titled "semantic-kernel-demos [main ↑1]".

The left terminal window shows the command: `dotnet run --framework net8.0`. The right terminal window shows the command: `dotnet run --framework net8.0`.

The code being run is as follows:

```
1  using Microsoft.SemanticKernel;
2  using Microsoft.SemanticKernel.Plugins.OpenApi;
3  using Microsoft.Extensions.Configuration;
4  using Sample;
5
6  var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
7
8  var kernel = Kernel.CreateBuilder()
9      .AddOpenAIChatCompletion("gpt-4o-min", apiKey)
10     .Build();
11
12  var weatherOpenApiUri = new Uri("https://localhost:7299/swagger/v1/swagger.json");
13  var weatherPlugin = await OpenApiKernelPluginFactory.CreateFromOpenApiAsync("WeatherService", weatherOpenApiUri);
14
15  kernel.Plugins.Add(weatherPlugin);
16  kernel.Plugins.AddFromType<TimePlugin>(); // Enable AI to know which day is today
17
18  var settings = new PromptExecutionSettings { FunctionChoiceBehavior = FunctionChoiceBehavior.Auto() };
19
20  var prompt = "What is the weather forecast for tomorrow? Ensure you check which day is today.";
21  Console.WriteLine($"\\nUser > {prompt}");
22  var result = await kernel.InvokePromptAsync(prompt, new(settings));
23
24  Console.WriteLine($"Assistant > {result}");
```

A yellow box highlights the last few lines of the code, specifically the prompt definition and the final output line.



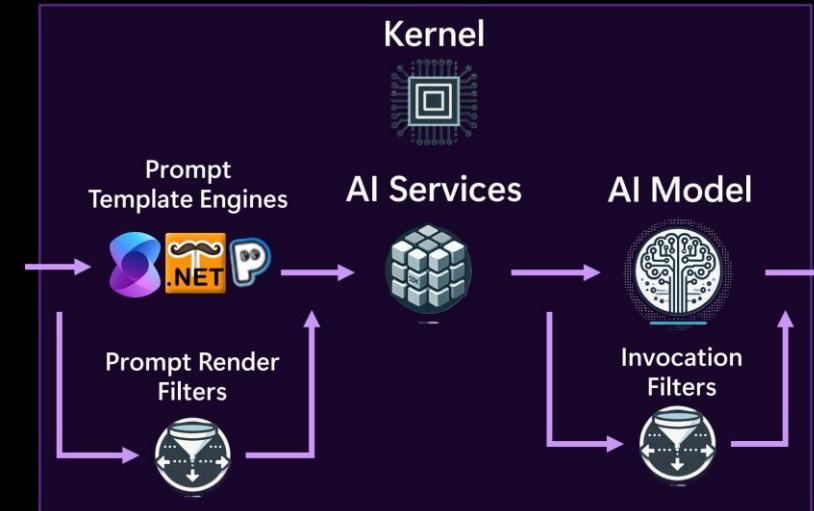
# 9. Kernel Filters – Render & Function Invocation

## Prompt Rendering

```
5  /// <summary>
6  /// This filter is used to intercept before and after a kernel prompt is rendered and write to the console its name and result.
7  /// </summary>
8  1 reference | Roger Barreto, 1 day ago | 1 author, 1 change
9   public class ConsoleWritePromptRenderingFilter : IPromptRenderFilter
10  {
11      0 references | Roger Barreto, 1 day ago | 1 author, 1 change
12      public async Task OnPromptRenderAsync(PromptRenderContext context, Func<PromptRenderContext, Task> next)
13      {
14          // Before prompt was rendered
15          await next.Invoke(context);
16
17          // After prompt was rendered
18          Console.ForegroundColor = ConsoleColor.Green;
19          Console.WriteLine($"Prompt Rendered: '{context.RenderedPrompt}'");
20          Console.ResetColor();
21
22          // Write the prompt to the console as is with no color formatting (Similar to assistant output)
23          Console.WriteLine($"{context.RenderedPrompt}");
24      }
25  }
```

## Function Invocation

```
5  /// <summary>
6  /// This filter is used to intercept before and after a kernel function is invoked and write to the console its name and result.
7  /// </summary>
8  1 reference | Roger Barreto, 1 day ago | 1 author, 1 change
9   public class ConsoleWriteFunctionInvocationFilter : IFunctionInvocationFilter
10  {
11      0 references | Roger Barreto, 1 day ago | 1 author, 1 change
12      public async Task OnFunctionInvocationAsync(FunctionInvocationContext context, Func<FunctionInvocationContext, Task> next)
13      {
14          string writeMessage = $"Function Invoked: '{context.Function.Name}'";
15          if (!string.IsNullOrEmpty(context.Function.PluginName))
16          {
17              writeMessage += $" - Plugin: '{context.Function.PluginName}'";
18          }
19
20          Console.ForegroundColor = ConsoleColor.Cyan;
21          Console.WriteLine(writeMessage);
22          Console.ResetColor();
23
24          await next.Invoke(context);
25
26          // On this sample all functions without a plugin name are AI model responses
27          if (string.IsNullOrEmpty(context.Function.PluginName))
28          {
29              Console.WriteLine($"Assistant: {context.Result}");
29          }
29  }
```



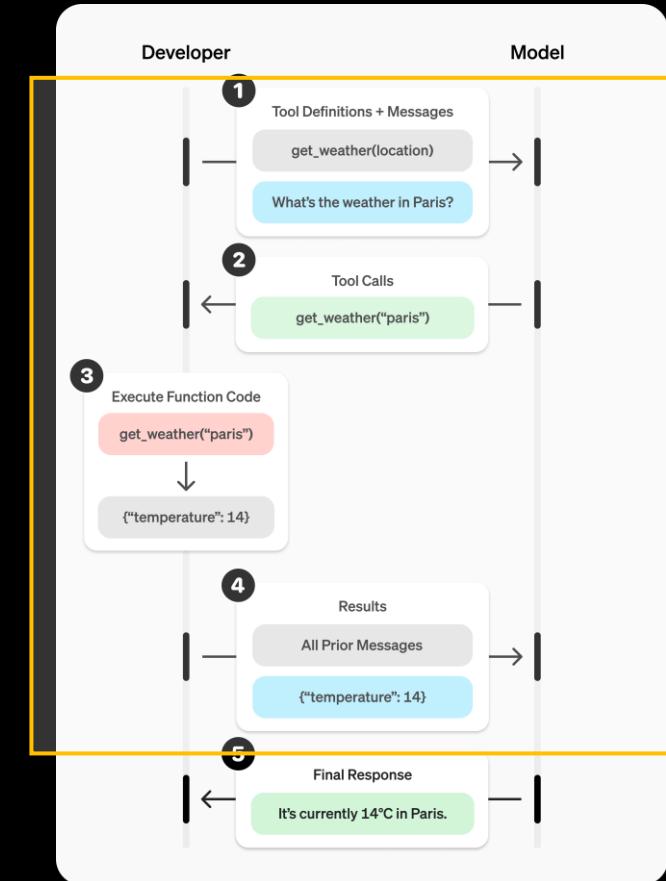


# 9. Kernel Filters – Auto Function Invocation

## Auto Function Invocation (AI Function Calling Loop)

```
6  //<summary>
7  // This filter is used specifically for function calling models where you want to intercept
8  // AI Model function calls and trace (Console.WriteLine) or change the behavior.
9  //</summary>
10 // 1 reference | Roger Barreto, 1 day ago | 1 author, 1 change
11 public class ConsoleWriteAutoFunctionInvocationFilter : IAutoFunctionInvocationFilter
12 {
13     // 0 references | Roger Barreto, 1 day ago | 1 author, 1 change
14     public async Task OnAutoFunctionInvocationAsync(AutoFunctionInvocationContext context, Func<AutoFunctionInvocationContext, Task> next)
15     {
16         // Request sequence index identifies what and how many requests
17         // $ - Request sequence index: {context.RequestSequenceIndex},
18
19         // Function sequence index identifies what and which is the current function
20         // Note: Multiple functions can be called in a single request (e.g. get_weather("paris"), get_forecast("paris"))
21         // $ - Function sequence index: {context.FunctionSequenceIndex},
22
23         // Total number of functions requested to be called by the AI Model
24         // $ - Total number of functions: {context.FunctionCount});
25
26         // Reset color after each function call
27         Console.ForegroundColor = ConsoleColor.DarkYellow;
28         Console.WriteLine(string.Concat($"Auto Function Invoked: '{context.Function.Name}'",
29
30             // Request sequence index identifies what and how many requests
31             // $ - Request sequence index: {context.RequestSequenceIndex},
32
33             // Function sequence index identifies what and which is the current function
34             // Note: Multiple functions can be called in a single request (e.g. get_weather("paris"), get_forecast("paris"))
35             // $ - Function sequence index: {context.FunctionSequenceIndex},
36
37             // Total number of functions requested to be called by the AI Model
38             // $ - Total number of functions: {context.FunctionCount})));
39
40         // Reset color after each function call
41         Console.ResetColor();
42
43         await next.Invoke(context);
44     }
45 }
```

## AI Function Calling Loop





# 9. Kernel Filters – Setup & Execution

## Adding directly to a Kernel instance

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel;
3  using Microsoft.SemanticKernel.ChatCompletion;
4  using Microsoft.SemanticKernel.Connectors.OpenAI;
5  using Sample;
6
7  var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
8
9  Console.WriteLine("== Using Kernel Filters ==");
10
11 var lightBulbPlugin = new LightBulbPlugin(isOn: true);
12
13 var modelId = "gpt-4o-mini";
14 var kernel = Kernel.CreateBuilder()
15     .AddOpenAIChatCompletion(modelId, apiKey)
16     .Build();
17
18 // Add the custom function invocation filter
19 kernel.FunctionInvocationFilters.Add(new ConsoleWriteFunctionInvocationFilter());
20
21 // Add the custom prompt rendering filter
22 kernel.PromptRenderFilters.Add(new ConsoleWritePromptRenderingFilter());
23
24 // Add the custom auto function invocation filter
25 kernel.AutoFunctionInvocationFilters.Add(new ConsoleWriteAutoFunctionInvocationFilter());
26
27 kernel.Plugins.AddFromObject(lightBulbPlugin);
28 kernel.Plugins.AddFromType<TimePlugin>();
29
30 var service = kernel.GetRequiredService<IChatCompletionService>();
31
32 var settings = new OpenAIPromptExecutionSettings { FunctionChoiceBehavior = FunctionChoiceBehavior.Auto() };
33 var arguments = new KernelArguments(settings) { ["name"] = "Roger" };
34
35 var prompt = "{$name}: What is the light status and what time of the day is it?";
36 await kernel.InvokePromptAsync(prompt, arguments);
37
38 prompt = "{$name}: Please switch the light.";
39 var promptFunction = kernel.CreateFunctionFromPrompt(prompt, functionName: "FunctionFromPrompt");
40 await kernel.InvokeAsync(promptFunction, arguments);
41
42 // Direct service calls don't trigger prompt rendering
43 prompt = $"{arguments["name"]}: What is the light bulb status and what time of the day is it?";
44 Console.WriteLine($"{prompt}");
45 var kernelContent = await service.GetChatMessageContentAsync(prompt, settings, kernel);
46 Console.WriteLine($"Assistant: {kernelContent}");
47
```

## Adding via Dependency Injection

```
var builder = Kernel.CreateBuilder()
    .AddOpenAIChatCompletion(modelId, apiKey);
builder.Services.AddSingleton<IFunctionInvocationFilter, ConsoleWriteFunctionInvocationFilter>();
builder.Services.AddSingleton<IPromptRenderFilter, ConsoleWritePromptRenderingFilter>();
builder.Services.AddSingleton<IAutoFunctionInvocationFilter, ConsoleWriteAutoFunctionInvocationFilter>();
var kernel = builder.Build();
```

## Execution

```
D:\repo\work\conferences\se
```



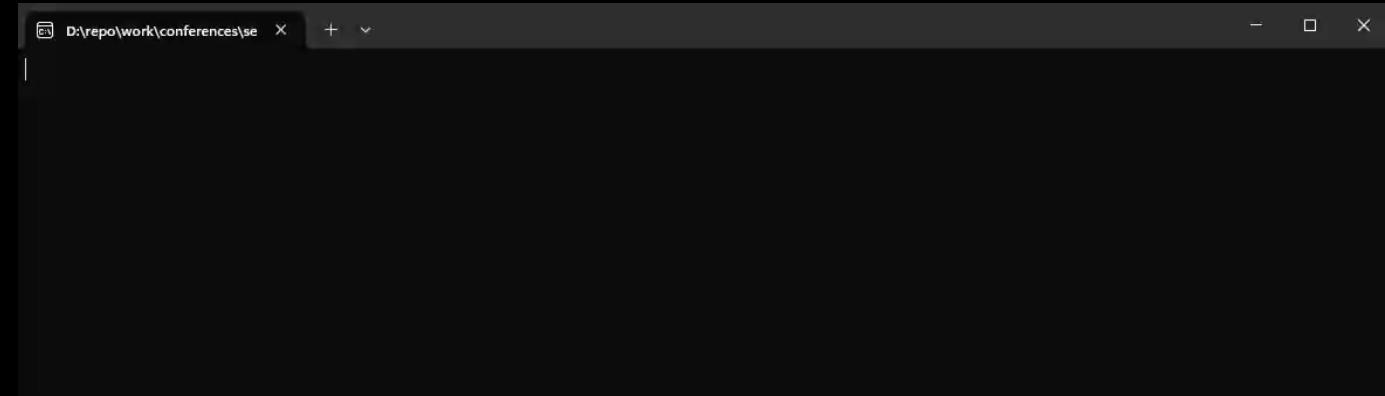
# 10. Open Telemetry + Aspire Dashboard

## Setup: Resource + Tracer + Meter + Logger Factory

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.Extensions.DependencyInjection;
3  using Microsoft.Extensions.Logging;
4  using Microsoft.SemanticKernel;
5  using OpenTelemetry;
6  using OpenTelemetry.Logs;
7  using OpenTelemetry.Metrics;
8  using OpenTelemetry.Resources;
9  using OpenTelemetry.Trace;
10 using Sample;
11
12 Console.WriteLine("==== Open Telemetry Aspire Dashboard ====\n\n");
13
14 var builder = Kernel.CreateBuilder();
15 var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
16 var modelId = "gpt-4o-mini";
17 var oTelExporterEndpoint = "http://localhost:4317";
18
19 var resourceBuilder = ResourceBuilder
20     .CreateDefault()
21     .AddService("SemanticKernel-Telemetry");
22
23 // Enable model diagnostics with sensitive data.
24 ApplicationContext.SetSwitch("Microsoft.SemanticKernel.Experimental.GenAI.EnableOtelDiagnosticsSensitive", true);
25
26 // Configuring the tracer provider
27 using var traceProvider = Sdk.CreateTracerProviderBuilder()
28     .SetResourceBuilder(resourceBuilder)
29     .AddSource("Microsoft.SemanticKernel*")
30     .AddOtlpExporter(options => options.Endpoint = new Uri(oTelExporterEndpoint))
31     .Build();
32
33 // Configuring the meter provider
34 using var meterProvider = Sdk.CreateMeterProviderBuilder()
35     .SetResourceBuilder(resourceBuilder)
36     .AddMeter("Microsoft.SemanticKernel*")
37     .AddOtlpExporter(options => options.Endpoint = new Uri(oTelExporterEndpoint))
38     .Build();
39
40 // Configuring the logger factory
41 using var loggerFactory = LoggerFactory.Create(builder =>
42 {
43     // Add OpenTelemetry as a logging provider
44     builder.AddOpenTelemetry(options =>
45     {
46         options.SetResourceBuilder(resourceBuilder);
47         options.AddOtlpExporter(options => options.Endpoint = new Uri(oTelExporterEndpoint));
48         // Format log messages. This is default to false.
49         options.IncludeFormattedMessage = true;
50         options.IncludeScopes = true;
51     });
52     builder.SetMinimumLevel(LogLevel.Information);
53 });


```

## Execution



## Aspire Dashboard

SemanticKernel-Telemetry: InvokePromptAsync\_76c6c94183394105af829fab51ba12be

Trace detail 3/25/2025 10:23:23.570 PM Duration 2.63s Resources 1 Depth 2 Total spans 4

Name	0ms	656.5ms	1.31s	1.97s	2.63s	Actions
SemanticKernel-Telemetry-a36b3109	0ms	656.5ms	1.31s	1.97s	2.63s	...
SemanticKernel-Telemetry-a36b3109	0ms	656.5ms	1.31s	1.97s	2.63s	...
SemanticKernel-Telemetry-a36b3109	0ms	656.5ms	1.31s	1.97s	2.63s	...
SemanticKernel-Telemetry-a36b3109	0ms	656.5ms	1.31s	1.97s	2.63s	...

SemanticKernel-Telemetry: TurnOn 3144c86

Resource SemanticKernel-Telemetry Duration 13.26ms Start time 1.71s

Span

Name	Value
SpanId	3144c863137113df
Name	TurnOn

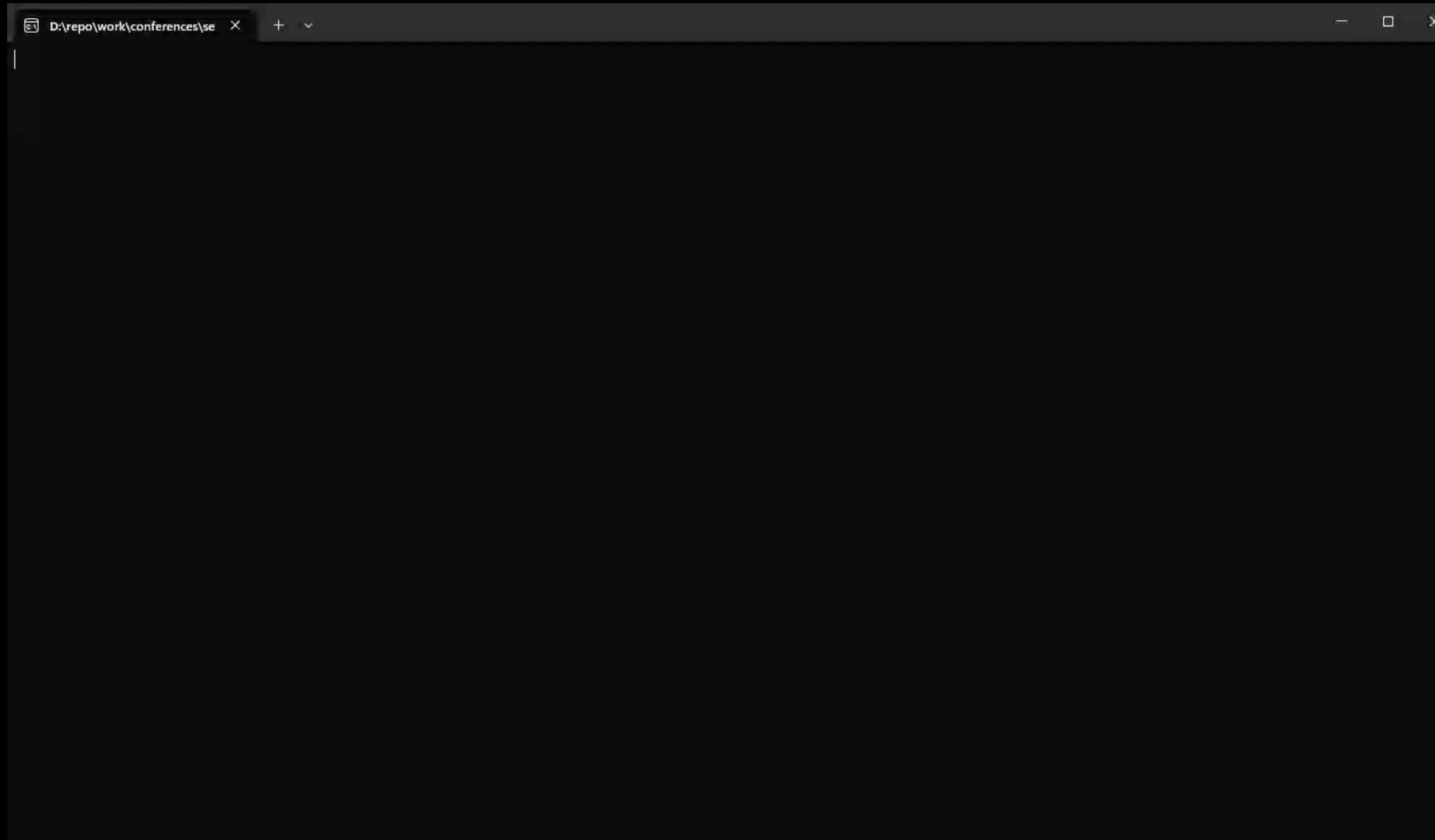


# 11. Multi AI Models - Setup

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel;
3  using Microsoft.SemanticKernel.ChatCompletion;
4  using Microsoft.SemanticKernel.Services;
5  using Sample;
6
7  #pragma warning disable SKEXP0070 // Type is for evaluation purposes only and is subject to change or removal in future update
8
9  Console.WriteLine("==> Kernel AI Model Selection ==>");
10
11 // 1st GPU Inference AI Model (Ollama - llama3.2)
12 var ollamaModelId = "llama3.2";
13 var ollamaEndpoint = new Uri("http://localhost:11434");
14
15 // 2nd CPU Inference AI Model (ONNX - phi3)
16 var onnxModelId = "phi3";
17 var onnxModelPath = "D:\\repo\\huggingface-models\\Phi-3-mini-4k-instruct-onnx\\cpu_and_mobile\\cpu-int4-rtn-block-32";
18
19 // 3rd Cloud Inference AI Model (OpenAI - gpt-4o-mini)
20 var openAIModelId = "gpt-4o-mini";
21 var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
22
23 var kernelBuilder = Kernel.CreateBuilder();
24
25 kernelBuilder
26     .AddOpenAIChatCompletion(openAIModelId, apiKey)
27     .AddOnnxRuntimeGenAIChatCompletion(onnxModelId, onnxModelPath)
28     .AddOllamaChatCompletion(ollamaModelId, ollamaEndpoint);
29 // Last service will be the default service
30
31 var kernel = kernelBuilder.Build();
32
33 kernel.PromptRenderFilters.Add(new SelectedModelRenderFilter());
34
35 var modelNames = kernel.GetAllServices<IChatCompletionService>().Select(s => s.GetModelId());
36
```



# 11. Multi AI Models - Execution





# 12. Multi Modalities - Setup

```
1  using Microsoft.Extensions.Configuration;
2  using Microsoft.SemanticKernel;
3  using Microsoft.SemanticKernel.AudioToText;
4  using Microsoft.SemanticKernel.ChatCompletion;
5  using Microsoft.SemanticKernel.Connectors.OpenAI;
6
7  #pragma warning disable SKEXP0001 // Type is for evaluation purposes only and is subject to change
8  #pragma warning disable SKEXP0010 // Type is for evaluation purposes only and is subject to change
9
10 var apiKey = new ConfigurationBuilder().AddUserSecrets<Program>().Build()["OpenAI:ApiKey"]!;
11
12 // Used for audio speech transcription
13 var audioToTextService = new OpenAIAudioToTextService("whisper-1", apiKey);
14
15 // Used for image generation
16 var textToImageService = new OpenAITextToImageService(modelId: "dall-e-3", apiKey);
17
18 // Used for image description
19 var chatService = new OpenAIChatCompletionService("gpt-4o-mini", apiKey);
20
21 // Used for audio speech generation
22 var textToAudioService = new OpenAITextToAudioService("tts-1", apiKey);
23
24 Console.WriteLine("""
25 === Multi Modality Sample ===
26 Pipeline:
27 1. Load the input audio request
28 2. Transcribe the audio speech into text
29 3. Use the transcription as the prompt to generate an image
30 4. Capture a detailed text description from the generated image
31 5. Generate and audio speech from the detailed image description
32
33 Starting ...
34 """);
```



# 12. Multi Modalities – Audio-to-Text



Audio-to-Text

“Hey, AI can you generate for me a futuristic metropolis city in 2050, please”

```
35
36 Console.WriteLine("\n1. Loading audio file ... ");
37 var audioContent = new AudioContent(File.ReadAllBytes("Audios/input.m4a"), "audio/m4a");
38 #pragma warning restore SKEP0001 // Type is for evaluation purposes only and is subject to change or removal in future updates. Suppress this diagnostic to proceed.
39 Console.WriteLine("Done");
40
41 Console.WriteLine("\n2. Transcribing audio speech into text ... ");
42 var audioTranscriptionText = await audioToTextService.GetTextContentAsync(audioContent);
43 Console.WriteLine("Done\n");
44
45 Console.WriteLine($"Transcription: {audioTranscriptionText}");
46
47 Console.WriteLine("\n3. Generating image from transcription ... ");
48 var generatedImage = (await textToImageService.GetImageContentsAsync(audioTranscriptionText, new OpenAITextToImageExecutionSettings { ResponseFormat = "bytes" })).First();
49 Console.WriteLine("Done\n");
50
51 ChatHistory chatHistory = [];
52 var imageContent = new ImageContent(generatedImage.Data!.Value, "image/jpeg");
53
54 // Save the image to disk
55 var outputImagePath = "Images/output.jpg";
56 if (!Directory.Exists("Images")) { Directory.CreateDirectory("Images"); }
57 if (File.Exists(outputImagePath)) { File.Delete(outputImagePath); }
58 await File.WriteAllBytesAsync(outputImagePath, generatedImage.Data!.Value.ToArray());
59
60 var currentDirectory = Directory.GetCurrentDirectory();
61 Console.WriteLine($"Image Generated. Ctrl + Click to view: {new Uri(Path.Combine(currentDirectory, outputImagePath)).AbsoluteUri}");
62
63 chatHistory.AddUserMessage(
64   [
65     new TextContent("Describe the image in detail"),
66     imageContent
67   ]);
68
69 Console.WriteLine("\n4. Generating image description ... ");
70 var imageDescriptionText = await chatService.GetChatMessageContentAsync(chatHistory);
71 Console.WriteLine("Done\n");
72
73 Console.WriteLine($"Image Description: {imageDescriptionText}");
74
75 Console.WriteLine("\n5. Generating audio speech from image description ... ");
76 var generatedAudio = (await textToAudioService.GetAudioContentsAsync(imageDescriptionText.ToString())).First();
77 Console.WriteLine("Done\n");
78
79 // Save the audio to disk
80 var outputAudioPath = "Audios/output.mp3";
81 if (File.Exists(outputAudioPath)) { File.Delete(outputAudioPath); }
82 await File.WriteAllBytesAsync(outputAudioPath, generatedAudio.Data!.Value.ToArray());
83
84 Console.WriteLine($"Audio speech from image description: (Ctrl + Click to view) {new Uri(Path.Combine(currentDirectory, outputAudioPath)).AbsoluteUri}");
```



# 12. Multi Modalities – Text-to-Image

```
35
36     Console.WriteLine($"\\n1. Loading audio file ... ");
37     var audioContent = new AudioContent(File.ReadAllBytes("Audios/input.m4a"), "audio/m4a");
38     #pragma warning restore SKEP0001 // Type is for evaluation purposes only and is subject to change or removal in future updates. Suppress this diagnostic to proceed.
39     Console.WriteLine("Done");
40
41     Console.WriteLine($"\\n2. Transcribing audio speech into text ... ");
42     var audioTranscriptionText = await audioToTextService.GetTextContentAsync(audioContent);
43     Console.WriteLine("Done\\n");
44
45     Console.WriteLine($"Transcription: {audioTranscriptionText}");
46
47     Console.WriteLine($"\\n3. Generating image from transcription ... ");
48     var generatedImage = (await textToImageService.GetImageContentsAsync(audioTranscriptionText, new OpenAITextToImageExecutionSettings { ResponseFormat = "bytes" })).First();
49     Console.WriteLine("Done\\n");
50
51     ChatHistory chatHistory = [];
52     var imageContent = new ImageContent(generatedImage.Data!.Value, "image/jpeg");
53
54     // Save the image to disk
55     var outputImagePath = "Images/output.jpg";
56     if (!Directory.Exists("Images")) { Directory.CreateDirectory("Images"); }
57     if (File.Exists(outputImagePath)) { File.Delete(outputImagePath); }
58     await File.WriteAllBytesAsync(outputImagePath, generatedImage.Data!.Value.ToArray());
59
60     var currentDirectory = Directory.GetCurrentDirectory();
61     Console.WriteLine($"Image Generated. Ctrl + Click to view: {new Uri(Path.Combine(currentDirectory, outputImagePath)).AbsoluteUri}");
62
63     chatHistory.AddUserMessage(
64         [
65             new TextContent("Describe the image in detail"),
66             imageContent
67         ]
68     );
69
70     Console.WriteLine($"\\n4. Generating image description ... ");
71     var imageDescriptionText = await chatService.GetChatMessageContentAsync(chatHistory);
72     Console.WriteLine("Done\\n");
73
74     Console.WriteLine($"Image Description: {imageDescriptionText}");
75
76     Console.WriteLine($"\\n5. Generating audio speech from image description ... ");
77     var generatedAudio = (await textToAudioService.GetAudioContentsAsync(imageDescriptionText.ToString())).First();
78     Console.WriteLine("Done\\n");
79
80     // Save the audio to disk
81     var outputAudioPath = "Audios/output.mp3";
82     if (File.Exists(outputAudioPath)) { File.Delete(outputAudioPath); }
83     await File.WriteAllBytesAsync(outputAudioPath, generatedAudio.Data!.Value.ToArray());
84
85     Console.WriteLine($"Audio speech from image description: (Ctrl + Click to view) {new Uri(Path.Combine(currentDirectory, outputAudioPath)).AbsoluteUri}");
```



Audio-to-Text

"Hey, AI can you generate for me a futuristic metropolis city in 2050, please"





# 12. Multi Modalities – Chat-Image-to-Text



Audio-to-Text

“Hey, AI can you generate for me a futuristic metropolis city in 2050, please”



“The image depicts a futuristic cityscape...  
... that enhance connectivity and livability.”

```
35
36     Console.WriteLine("\n1. Loading audio file ... ");
37     var audioContent = new AudioContent(File.ReadAllBytes("Audios/input.m4a"), "audio/m4a");
38     #pragma warning restore SKEW0001 // Type is for evaluation purposes only and is subject to change or removal in future updates. Suppress this diagnostic to proceed.
39     Console.WriteLine("Done");
40
41     Console.WriteLine("\n2. Transcribing audio speech into text ... ");
42     var audioTranscriptionText = await audioToTextService.GetTextContentAsync(audioContent);
43     Console.WriteLine("Done\n");
44
45     Console.WriteLine($"Transcription: {audioTranscriptionText}");
46
47     Console.WriteLine("\n3. Generating image from transcription ... ");
48     var generatedImage = (await textToImageService.GetImageContentsAsync(audioTranscriptionText, new OpenAITextToImageExecutionSettings { ResponseFormat = "bytes" })).First();
49     Console.WriteLine("Done\n");
50
51     ChatHistory chatHistory = [];
52     var imageContent = new ImageContent(generatedImage.Data!.Value, "image/jpeg");
53
54     // Save the image to disk
55     var outputImagePath = "Images/output.jpg";
56     if (!Directory.Exists("Images")) { Directory.CreateDirectory("Images"); }
57     if (File.Exists(outputImagePath)) { File.Delete(outputImagePath); }
58     await File.WriteAllBytesAsync(outputImagePath, generatedImage.Data!.Value.ToArray());
59
60     var currentDirectory = Directory.GetCurrentDirectory();
61     Console.WriteLine($"Image Generated. Ctrl + Click to view: {new Uri(Path.Combine(currentDirectory, outputImagePath)).AbsoluteUri}");
62
63     chatHistory.AddUserMessage(
64         [
65             new TextContent("Describe the image in detail"),
66             imageContent
67         ]);
68
69     Console.WriteLine("\n4. Generating image description ... ");
70     var imageDescriptionText = await chatService.GetChatMessageContentAsync(chatHistory);
71     Console.WriteLine("Done\n");
72
73     Console.WriteLine($"Image Description: {imageDescriptionText}");
74
75     Console.WriteLine("\n5. Generating audio speech from image description ... ");
76     var generatedAudio = (await textToAudioService.GetAudioContentsAsync(imageDescriptionText.ToString())).First();
77     Console.WriteLine("Done\n");
78
79     // Save the audio to disk
80     var outputAudioPath = "Audios/output.mp3";
81     if (File.Exists(outputAudioPath)) { File.Delete(outputAudioPath); }
82     await File.WriteAllBytesAsync(outputAudioPath, generatedAudio.Data!.Value.ToArray());
83
84     Console.WriteLine($"Audio speech from image description: (Ctrl + Click to view) {new Uri(Path.Combine(currentDirectory, outputAudioPath)).AbsoluteUri}");
```



# 12. Multi Modalities – Text-to-Audio

```
35
36     Console.WriteLine($"\\n1. Loading audio file ... ");
37     var audioContent = new AudioContent(File.ReadAllBytes("Audios/input.m4a"), "audio/m4a");
38     #pragma warning restore SKEW0001 // Type is for evaluation purposes only and is subject to change or removal in future updates. Suppress this diagnostic to proceed.
39     Console.WriteLine("Done");
40
41     Console.WriteLine($"\\n2. Transcribing audio speech into text ... ");
42     var audioTranscriptionText = await audioToTextService.GetTextContentAsync(audioContent);
43     Console.WriteLine("Done\\n");
44
45     Console.WriteLine($"Transcription: {audioTranscriptionText}");
46
47     Console.WriteLine($"\\n3. Generating image from transcription ... ");
48     var generatedImage = (await textToImageService.GetImageContentsAsync(audioTranscriptionText, new OpenAITextToImageExecutionSettings { ResponseFormat = "bytes" })).First();
49     Console.WriteLine("Done\\n");
50
51     ChatHistory chatHistory = [];
52     var imageContent = new ImageContent(generatedImage.Data!.Value, "image/jpeg");
53
54     // Save the image to disk
55     var outputImagePath = "Images/output.jpg";
56     if (!Directory.Exists("Images")) { Directory.CreateDirectory("Images"); }
57     if (File.Exists(outputImagePath)) { File.Delete(outputImagePath); }
58     await File.WriteAllBytesAsync(outputImagePath, generatedImage.Data!.Value.ToArray());
59
60     var currentDirectory = Directory.GetCurrentDirectory();
61     Console.WriteLine($"Image Generated. Ctrl + Click to view: {new Uri(Path.Combine(currentDirectory, outputImagePath)).AbsoluteUri}");
62
63     chatHistory.AddUserMessage(
64         [
65             new TextContent("Describe the image in detail"),
66             imageContent
67         ]);
68
69     Console.WriteLine($"\\n4. Generating image description ... ");
70     var imageDescriptionText = await chatService.GetChatMessageContentAsync(chatHistory);
71     Console.WriteLine("Done\\n");
72
73     Console.WriteLine($"Image Description: {imageDescriptionText}");
74
75     Console.WriteLine($"\\n5. Generating audio speech from image description ... ");
76     var generatedAudio = (await textToAudioService.GetAudioContentsAsync(imageDescriptionText.ToString())).First();
77     Console.WriteLine("Done\\n");
78
79     // Save the audio to disk
80     var outputAudioPath = "Audios/output.mp3";
81     if (File.Exists(outputAudioPath)) { File.Delete(outputAudioPath); }
82     await File.WriteAllBytesAsync(outputAudioPath, generatedAudio.Data!.Value.ToArray());
83
84     Console.WriteLine($"Audio speech from image description: (Ctrl + Click to view) {new Uri(Path.Combine(currentDirectory, outputAudioPath)).AbsoluteUri}");
```



Audio-to-Text

"Hey, AI can you generate for me a futuristic metropolis city in 2050, please"



"The image depicts a futuristic cityscape...  
... that enhance connectivity and livability."





# 12. Multi Modalities - Execution

```
File C:\Users\roger\OneDrive\Doc X + ▾  
--- Multi Modality Sample ---  
Pipeline:  
1. Load the input audio request  
2. Transcribe the audio speech into text  
3. Use the transcription as the prompt to generate an image  
4. Capture a detailed text description from the generated image  
5. Generate and audio speech from the detailed image description  
  
Starting ...  
1. Loading audio file ... Done  
2. Transcribing audio speech into text ... Done  
Transcription: Hey, I can generate for me a futuristic metropolis city in 2050, please  
3. Generating image from transcription ...  
Done  
Image Generated. Ctrl + Click to view: file:///D:/repo/work/conferences/semantic-kernel-demos/Demos/Beginner-Hands-On/Samples/MultiModality/Images/output.jpg  
4. Generating image description ... Done  
Image Description: The image depicts a futuristic cityscape brimming with advanced technology and architecture. Towering skyscrapers made of glass and metal dominate the skyline, reflecting a vibrant sunset in the background. The buildings are sleek and modern, featuring various designs, from cylindrical forms to angular structures, many adorned with glowing blue and purple lights.  
  
In the foreground, a circular plaza acts as a focal point, surrounded by lush greenery and modern architecture. High-tech transportation systems are visible, with roadways marked by glowing lines and hexagonal shapes indicating lanes or traffic control elements. Roads are bustling with vehicles, including autonomous cars that seem to glide effortlessly along the streets.  
  
Above the city, drones and flying vehicles navigate through the air, contributing to a sense of dynamic movement and innovation. Holographic displays can be seen projected in various areas, further emphasizing the advanced technological theme of the city.  
  
The overall atmosphere is one of progress and efficiency, showcasing a potential vision of urban life infused with futuristic elements that enhance connectivity and livability.  
5. Generating audio speech from image description ... Done  
Image Description Audio: (Ctrl + Click to view) file:///D:/repo/work/conferences/semantic-kernel-demos/Demos/Beginner-Hands-On/Samples/MultiModality/Audios/output.mp3
```



Audio-to-Text

“Hey, AI can you generate for me a futuristic metropolis city in 2050, please”

Text-to-Image



Chat-Image-to-Text

“The image depicts a futuristic cityscape...  
... that enhance connectivity and livability.”

Text-to-Audio





# Support Channels for Semantic Kernel

1. Discord – Community driven
2. Office Hours – weekly open meeting for anyone to join and ask questions of the product team (General Questions)
3. GitHub Issues – reviewed daily (Feature / Bugs)
4. Microsoft Support – open issue that is routed to the team (Urgent)



## Get Involved



## Join the community



Office hours every Wednesday at 8 AM PST - <https://aka.ms/sk-community-calendar>



# Microsoft Learn



MS Learn  
<https://aka.ms/sk/learn>

# DevBlogs



SK DevBlogs  
<https://aka.ms/sk/devblogs>

Office hours every Wednesday at 8 AM PST - <https://aka.ms/sk-community-calendar>



# Questions?

Office hours every Wednesday at 8 AM PST

<https://aka.ms/sk-community-calendar>