



Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

Cooking with the Semantic Kernel

Recipes for Building Chatbots, Agents, and more with LLMs

Alex Chao

FOR 3P DEV + PRODUCT LEADERS



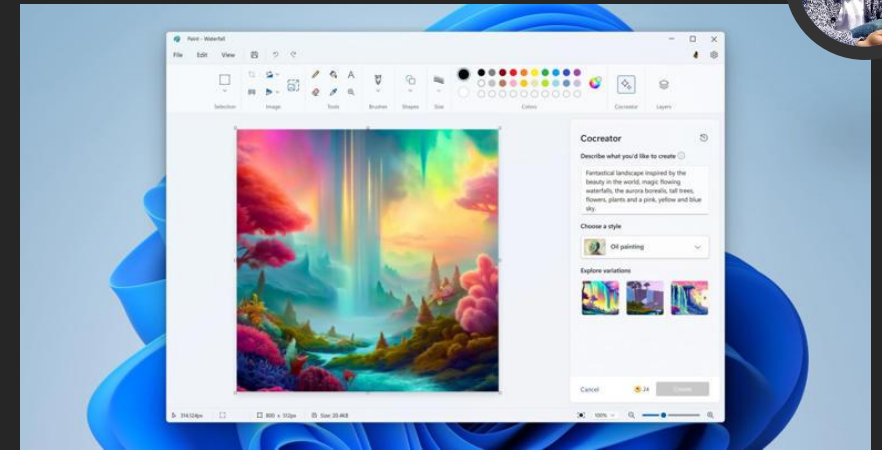
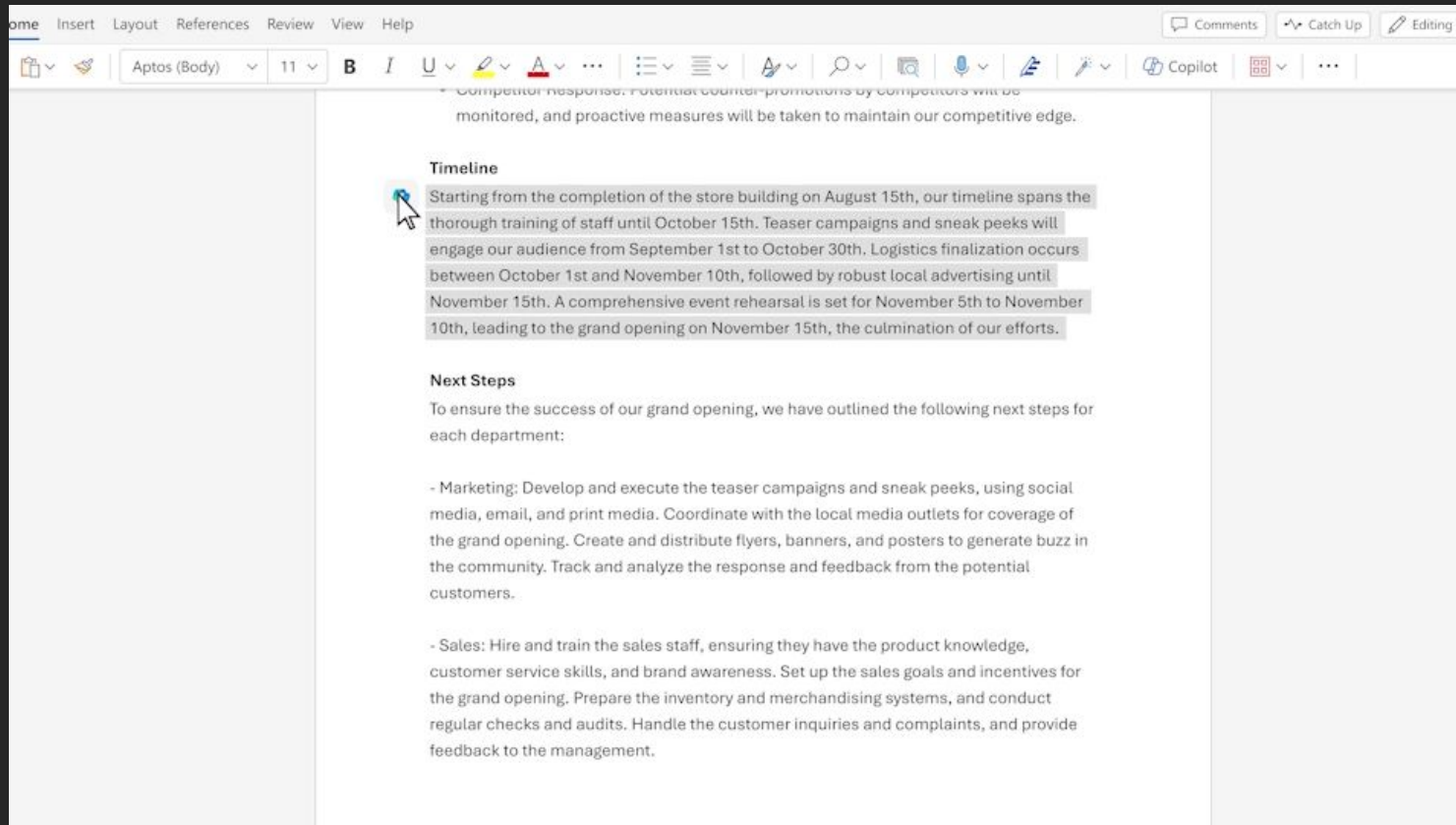
aka.ms/semantic-kernel



Copilot

Your everyday AI companion

Copilots everywhere!



A43 C1033

	A	B	C	D	E	F	G	H
1	Store Code	Store Name	Location	Year	Quarter	Sales		
2	CHI031	ContosoHI 31	Phoenix, Arizona	2019	Q3	\$ 7,003,071		
3	CHI031	ContosoHI 31	Phoenix, Arizona	2019	Q4	\$ 8,966,978		
4	CHI031	ContosoHI 31	Phoenix, Arizona	2020	Q1	\$ 6,263,955		
5	CHI031	ContosoHI 31	Phoenix, Arizona	2020	Q2	\$ 5,374,864		
6	CHI031	ContosoHI 31	Phoenix, Arizona	2020	Q3	\$ 7,435,062		
7	CHI031	ContosoHI 31	Phoenix, Arizona	2020	Q4	\$ 10,319,964		
8	CHI031	ContosoHI 31	Phoenix, Arizona	2021	Q1	\$ 6,580,746		
9	CHI031	ContosoHI 31	Phoenix, Arizona	2021	Q2	\$ 5,695,138		
10	CHI031	ContosoHI 31	Phoenix, Arizona	2021	Q3	\$ 9,132,134		
11	CHI031	ContosoHI 31	Phoenix, Arizona	2021	Q4	\$ 11,844,765		
12	CHI031	ContosoHI 31	Phoenix, Arizona	2022	Q1	\$ 8,723,031		
13	CHI031	ContosoHI 31	Phoenix, Arizona	2022	Q2	\$ 7,519,256		
14	CHI031	ContosoHI 31	Phoenix, Arizona	2022	Q3	\$ 10,648,465		
15	CHI031	ContosoHI 31	Phoenix, Arizona	2022	Q4	\$ 14,087,186		
16	CHI031	ContosoHI 31	Phoenix, Arizona	2023	Q1	\$ 9,726,792		
17	CHI031	ContosoHI 31	Phoenix, Arizona	2023	Q2	\$ 8,198,962		
18	CHI032	ContosoHI 32	Tucson, Arizona	2019	Q3	\$ 7,017,496		
19	CHI032	ContosoHI 32	Tucson, Arizona	2019	Q4	\$ 8,814,793		

Copilot PREVIEW

Forecast sales for the next 4 quarters

COPILOT IN EXCEL

How do you
build copilots?





Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App



What's the TL;DR of this burning moment in technology?

FROM

"AI" seems interesting but it's just a research project

TO

I want ChatGPT on my own enterprise data!



Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App



You need more than just a model API

LLM AI requires a new way of writing code that's unfamiliar.

Tokens? Temperature? Dealing with non-determinism

You soon run into the limits of what it can't do out-of-the-box.

No memory built-in, has a "knowledge-cutoff", can often hallucinate

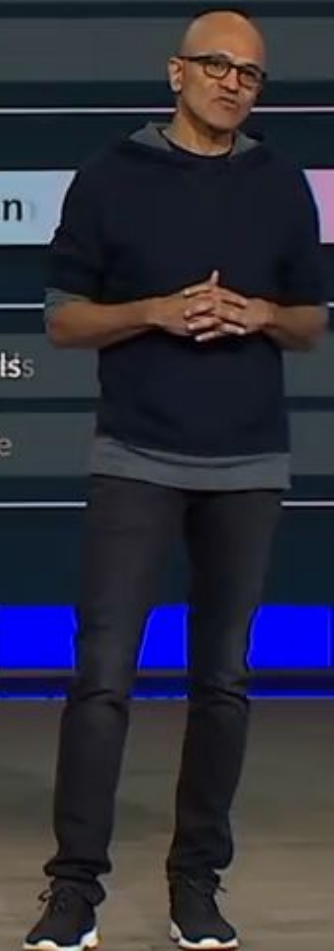
You need new tools that 10X your value and velocity with AI.

Picture infinite chats, long-running tasks, 1400+ connectors.



Announcing

Copilot stack



Satya Nadella
CEO
Microsoft



What is Semantic Kernel?

Semantic Kernel is a lightweight open-source orchestration SDK that lets you integrate Large Language Models (LLMs) with native code in languages like C#, Python, and Java.

Semantic Kernel



Fully open-source!

Join the community

We welcome your contributions and suggestions to SK community! One of the easiest ways to participate is to engage in discussions in the GitHub repository. Bug reports and fixes are welcome!

For new features, components, or extensions, please open an issue and discuss with us before sending a PR. This is to avoid rejection as we might be taking the core in a different direction, but also to consider the impact on the larger ecosystem.

To learn more and get started:

- Read the [documentation](#)
- Learn how to [contribute](#) to the project
- Join the [Discord community](#)
- Attend [regular office hours and SK community events](#)
- Follow the team on our [blog](#)

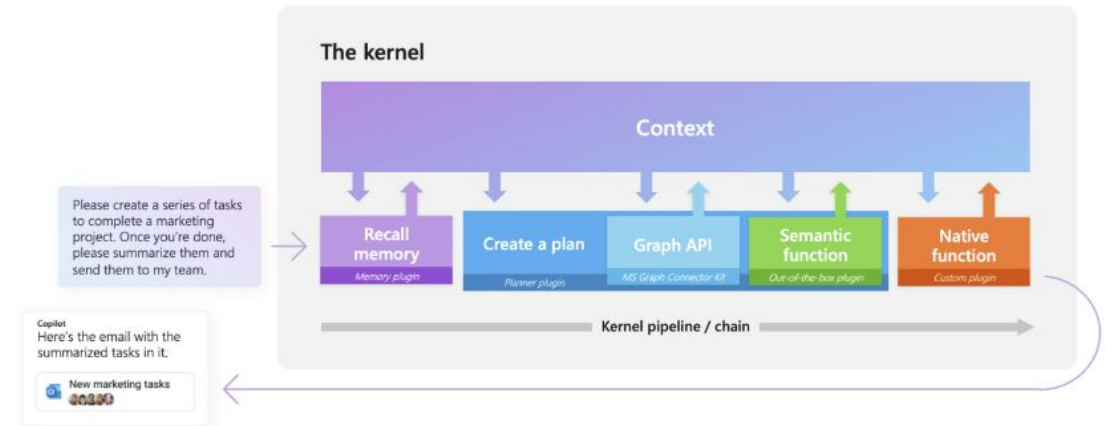


Semantic Kernel

`pip` v0.3.8.dev0 `nuget` v0.20.230821.4-preview `dotnet-ci-docker` passing `dotnet-ci-windows` passing `license` MIT
`Discord` 423 online

Semantic Kernel is an SDK that integrates Large Language Models (LLMs) like OpenAI, Azure OpenAI, and Hugging Face with conventional programming languages like C#, Python, and Java. Semantic Kernel achieves this by allowing you to define **plugins** that can be chained together in just a few lines of code.

What makes Semantic Kernel *special*, however, is its ability to *automatically* orchestrate plugins with AI. With Semantic Kernel **planners**, you can ask an LLM to generate a plan that achieves a user's unique goal. Afterwards, Semantic Kernel will execute the plan for the user.



Getting started with Semantic Kernel

The Semantic Kernel SDK is available in C#, Python, and Java. To get started, choose your preferred language below. See the [Feature Matrix](#) to see a breakdown of feature parity between our currently supported languages.



Using Semantic Kernel in C#



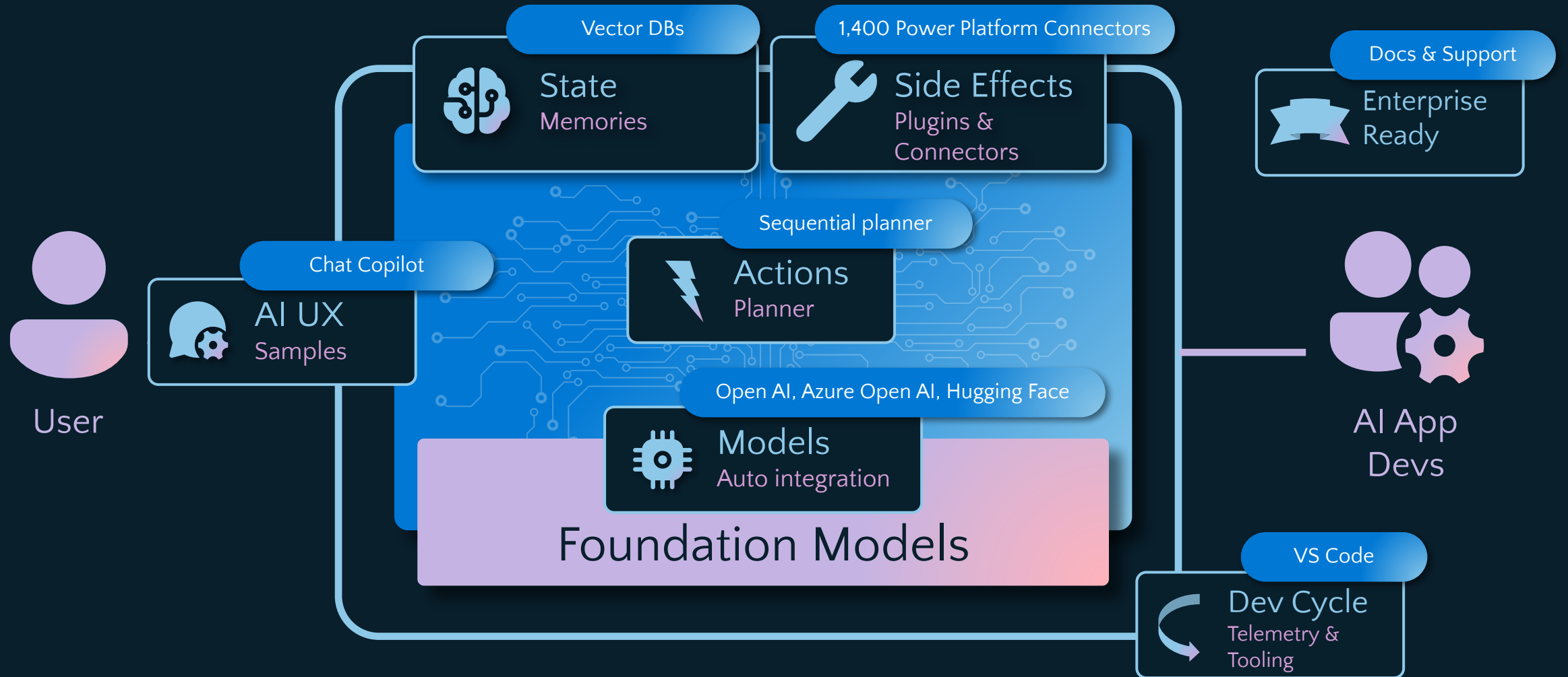
Using Semantic Kernel in Python



Using Semantic Kernel in Java



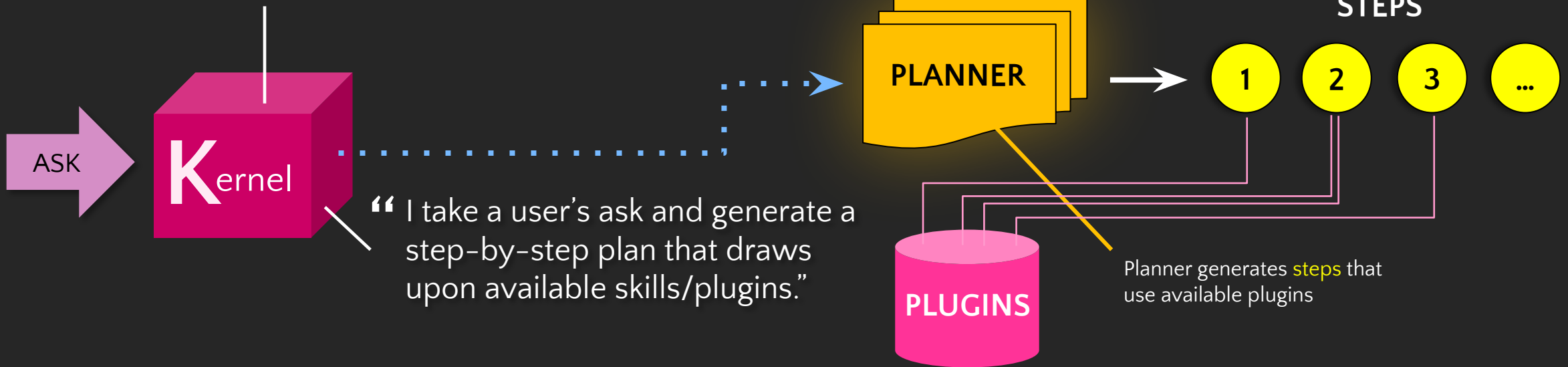
Semantic Kernel makes it easier for AI App Devs to get going.





Meet the lightweight Kernel of Semantic Kernel

“I’ve been designed to orchestrate complicated LLM AI prompts combined with native code, use multiple AI models, and ... I have a special skill to **PLAN**.”



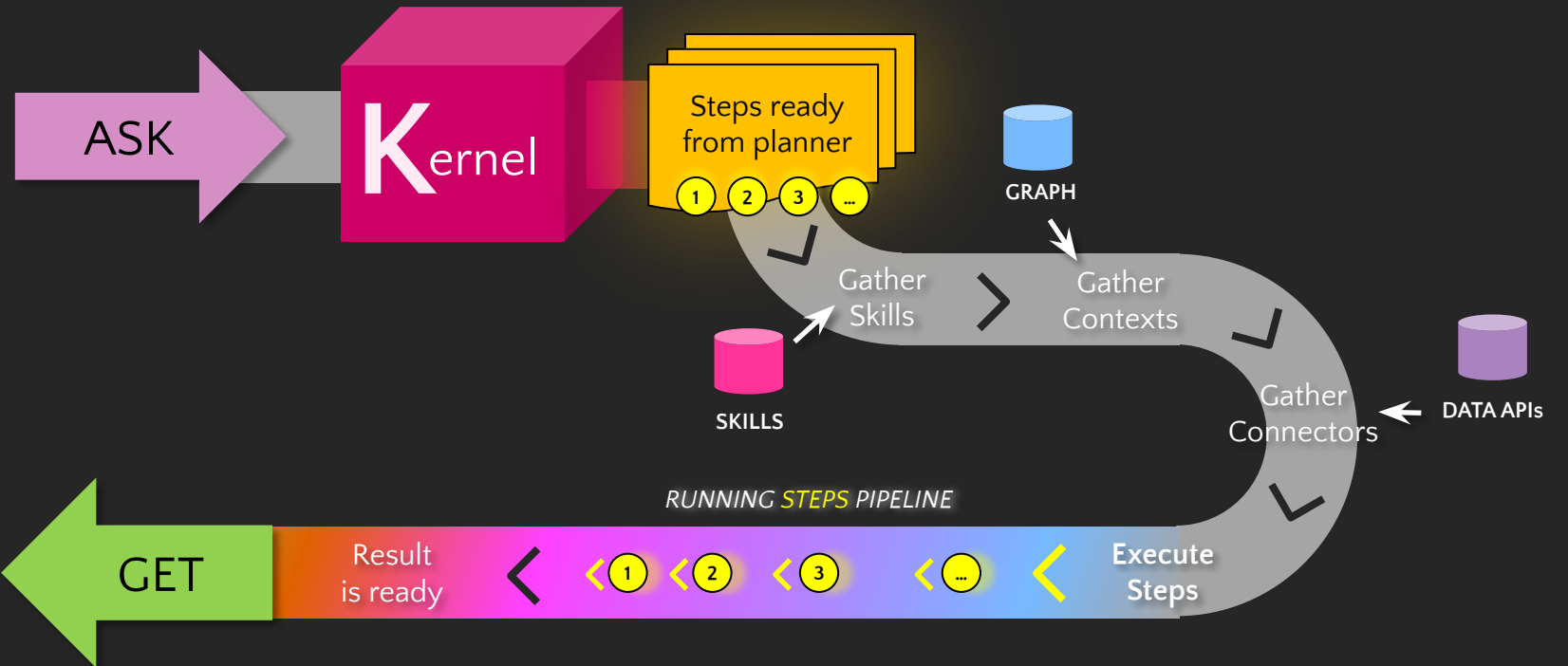
“I take a user’s ask and generate a step-by-step plan that draws upon available skills/plugins.”



SK Tour

Available NOW

It all starts with a user's AI ask ...



... resulting in new productivity



Why SK? Build with common tools to avoid future tech debt.

FROM

TO

~~Shipping a one-off, iffy GPT feature~~

Take a **paved** path for GPT deploys

~~Solve problems just with prompts~~

Mix prompts with native code

~~Struggle to build prompt contexts~~

Retrieve memory for prompts++

~~Build prompt orchestration yourself~~

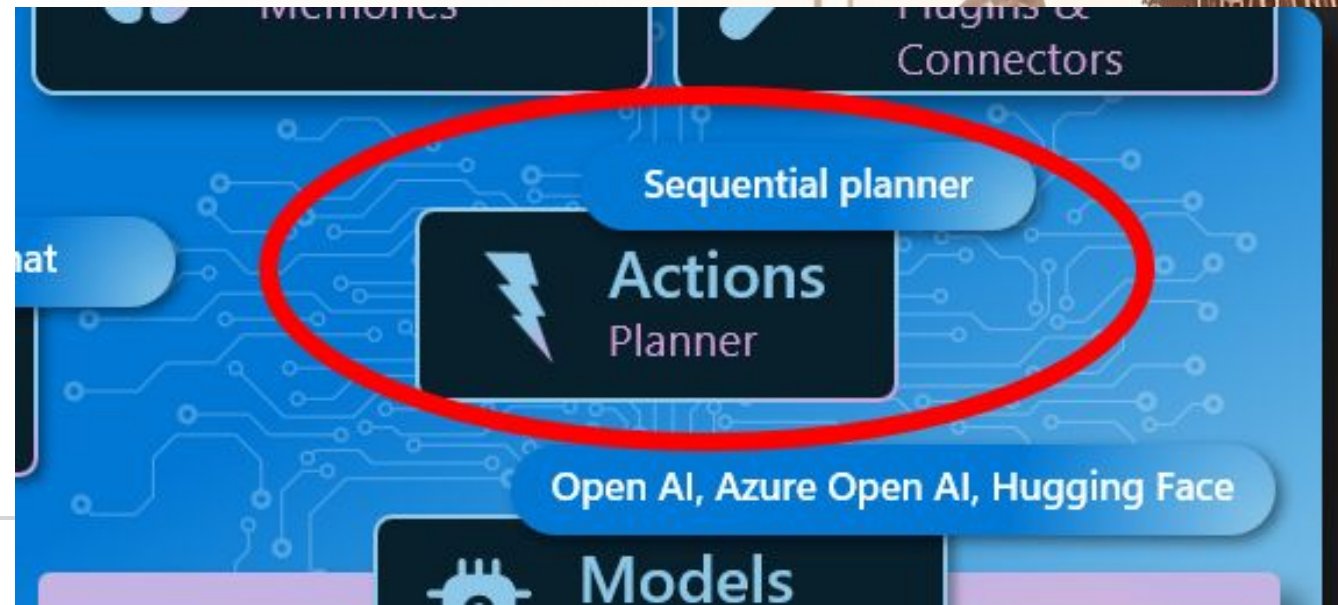
Use an emerging, common approach

~~Make prompt libraries on your own~~

Access stock, ready-to-go prompts



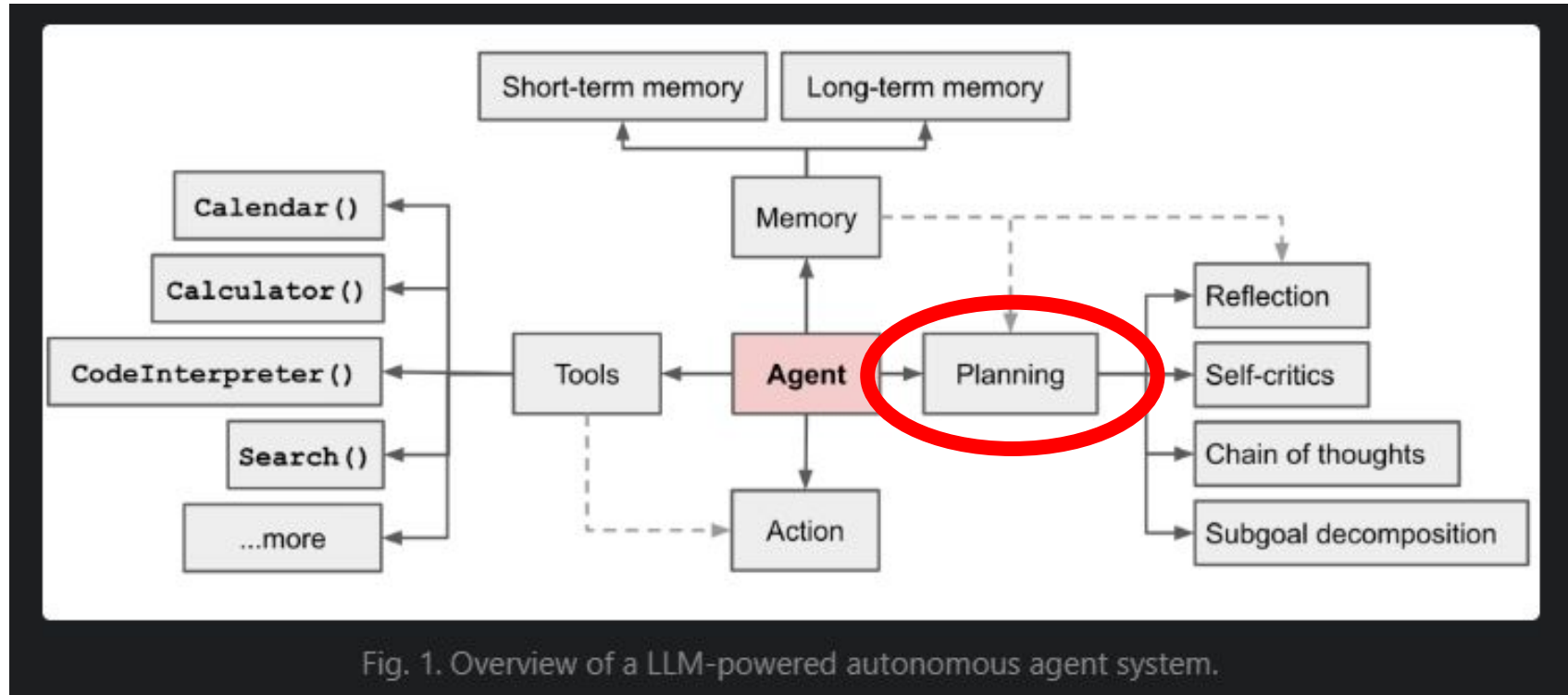
Let's talk planning





“LLM Powered Autonomous Agents”

From Lilian Weng (OpenAI)





Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

Planner Types

Action Planner

0-1 Skill to achieve a goal



Sequential Planner

Stitch available skills to achieve a goal



Stepwise Planner

Execute a plan step by step, changing the next step based on the outcome of the previous



Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

Planner DEMO

Introduction to the Planner

The Planner is one of the fundamental concepts of the Semantic Kernel.

It makes use of the collection of native and semantic functions that have been registered to the kernel and using AI, will formulate a plan to execute the given ask.

From our own testing, planner works best with more powerful models like `gpt4` but sometimes you might get working plans with cheaper models like `gpt-35-turbo`. We encourage you to implement your own versions of the planner and use different models that fit your user needs.

Read more about planner [here](#)

```
In [ ]: !python -m pip install semantic-kernel==0.3.12.dev0
```



It all begins with an ask

In []:

```
ask = """
Tomorrow is Valentine's day. I need to come up with a few date ideas. She speaks French so write it in French.
Convert the text to uppercase"""
```

Providing skills to the planner

The planner needs to know what skills are available to it. Here we'll give it access to the `SummarizeSkill` and `WriterSkill` we have defined on disk. This will include many semantic functions, of which the planner will intelligently choose a subset.

You can also include native functions as well. Here we'll add the `TextSkill`.

In []:

```
from semantic_kernel.core_skills.text_skill import TextSkill

skills_directory = "../../samples/skills/"
summarize_skill = kernel.import_semantic_skill_from_directory(skills_directory, "SummarizeSkill")
writer_skill = kernel.import_semantic_skill_from_directory(skills_directory, "WriterSkill")
text_skill = kernel.import_skill(TextSkill(), "TextSkill")
```

Define your ASK. What do you want the Kernel to do?



Basic Planner

Let's start by taking a look at a basic planner. The `BasicPlanner` produces a JSON-based plan that aims to solve the provided ask sequentially and evaluated in order.

```
In [5]: ▶ from semantic_kernel.planning.basic_planner import BasicPlanner
        ▶ planner = BasicPlanner()
```

```
In [6]: ▶ basic_plan = await planner.create_plan_async(ask, kernel)
```

```
In [7]: ▶ print(basic_plan.generated_plan)
```

```
{
  "input": "Valentine's Day Date Ideas",
  "subtasks": [
    {"function": "WriterSkill.Brainstorm"},
    {"function": "WriterSkill.EmailTo", "args": {"recipient": "significant_other"}},
    {"function": "WriterSkill.Translate", "args": {"language": "French"}},
    {"function": "TextSkill.uppercase"}
  ]
}
```




You can see that the Planner took my ask and converted it into an JSON-based plan detailing how the AI would go about solving this task, making use of the skills that the Kernel has available to it.

As you can see in the above plan, the AI has determined which functions to call in order to fulfill the user ask. The output of each step of the plan becomes the input to the next function.

Let's also define an inline skill and have it be available to the Planner. Be sure to give it a function name and skill name.

```
In [8]: ▶ sk_prompt = """
        {{${input}}

        Rewrite the above in the style of Shakespeare.
        """
        shakespeareFunction = kernel.create_semantic_function(sk_prompt, "shakespeare", "ShakespeareSkill",
                                                             max_tokens=2000, temperature=0.8)
```

Let's update our ask using this new skill

```
In [9]: ▶ ask = """
        Tomorrow is Valentine's day. I need to come up with a few date ideas.
        She likes Shakespeare so write using his style. She speaks French so write it in French.
        Convert the text to uppercase."""

        new_plan = await planner.create_plan_async(ask, kernel)
```

```
In [10]: ▶ print(new_plan.generated_plan)

{
  "input": "Valentine's Day Date Ideas",
  "subtasks": [
    {"function": "WriterSkill.Brainstorm"},
    {"function": "ShakespeareSkill.shakespeare"},
    {"function": "WriterSkill.EmailGen", "args": {"to": "significant_other"}},
    {"function": "WriterSkill.Translate", "args": {"language": "French"}},
    {"function": "TextSkill.uppercase"}
  ]
}
```



Executing the plan

Now that we have a plan, let's try to execute it! The Planner has a function called `execute_plan`.

```
In [11]: ▶ results = await planner.execute_plan_async(new_plan, kernel)
```

```
In [12]: ▶ print(results)
```

1. NOUS POUVONS PROFITER D'UNE PROMENADE ROMANTIQUE À TRAVERS LE PARC LUXURIANT ENSEMBLE.
2. TRAVAILLONS ENSEMBLE POUR CRÉER UN REPAS DÉLICIEUX ET LUXUEUX.
3. NOUS POUVONS VISITER UN VIGNOBLE À PROXIMITÉ ET PARTICIPER À UNE DÉGUSTATION DE VIN.
4. NOUS POUVONS NOUS BLOTTIR CHEZ NOUS ET REGARDER UN FILM ROMANTIQUE ENSEMBLE.
5. PRENONS UNE LEÇON DE DANSE ENSEMBLE ET APPRÉCIONS LA COMPAGNIE DE L'AUTRE.
6. NOUS POUVONS FAIRE DU PATIN À GLACE DANS UNE PATINOIRE LOCALE ET NOUS AMUSER ENSEMBLE.
7. PRENONS UN VOL EN MONTGOLFIÈRE ENSEMBLE ET APPRÉCIONS LA VUE.
8. NOUS POUVONS VISITER UNE GALERIE OU UN MUSÉE LOCAL POUR APPRÉCIER L'ART ET LA CULTURE.
9. NOUS POUVONS FAIRE UNE PROMENADE TRANQUILLE AU CRÉPUSCULE POUR ADMIRER LE MAGNIFIQUE COUCHER DE SOLEIL.
10. RÉSERVONS UN MASSAGE EN COUPLE DANS UN SPA RELAXANT.



Sequential Planner

The sequential planner is an XML-based step-by-step planner. You can see the prompt used for it here

([https://github.com/microsoft/semantic-](https://github.com/microsoft/semantic-kernel/blob/main/python/semantic_kernel/planning/sequential_planner/Skills/SequentialPlanning/skprompt.txt)

[kernel/blob/main/python/semantic_kernel/planning/sequential_planner/Skills/SequentialPlanning/skprompt.txt](https://github.com/microsoft/semantic-kernel/blob/main/python/semantic_kernel/planning/sequential_planner/Skills/SequentialPlanning/skprompt.txt))

```
In [ ]: from semantic_kernel.planning import SequentialPlanner
        planner = SequentialPlanner(kernel)
```

```
In [ ]: sequential_plan = await planner.create_plan_async(goal=ask)
```

To see the steps that the Sequential Planner will take, we can iterate over them and print their descriptions

```
In [ ]: for step in sequential_plan._steps:
        print(step.description, ":", step._state.__dict__)
```

Let's ask the sequential planner to execute the plan.

```
In [ ]: result = await sequential_plan.invoke_async()
```

```
In [ ]: print(result)
```




Action Planner

The action planner takes in a list of functions and the goal, and outputs a **single** function to use that is appropriate to meet that goal.

```
[6]: from semantic_kernel.planning import ActionPlanner
      planner = ActionPlanner(kernel)
```

Let's add more skills to the kernel

```
[7]: from semantic_kernel.core_skills import FileIOSkill, MathSkill, TextSkill, TimeSkill
      kernel.import_skill(MathSkill(), "math")
      kernel.import_skill(FileIOSkill(), "fileIO")
      kernel.import_skill(TimeSkill(), "time")
      kernel.import_skill(TextSkill(), "text")
```

```
[7]: {'lowercase': <semantic_kernel.orchestration.sk_function.SKFunction at 0x1b3ea1b72b0>,
      'trim': <semantic_kernel.orchestration.sk_function.SKFunction at 0x1b3ea1b6560>,
      'trim_end': <semantic_kernel.orchestration.sk_function.SKFunction at 0x1b3ea1b79d0>,
      'trim_start': <semantic_kernel.orchestration.sk_function.SKFunction at 0x1b3ea1b6da0>,
      'uppercase': <semantic_kernel.orchestration.sk_function.SKFunction at 0x1b3ea04a350>}
```

```
[8]: ask = "What is the sum of 110 and 990?"
```

```
[9]: plan = await planner.create_plan_async(goal=ask)
```

```
[10]: result = await plan.invoke_async()
```

```
[11]: print(result)
```

```
1100
```



Stepwise Planner

Stepwise Planner is based off the paper from MRKL (Modular Reasoning, Knowledge and Language) and is similar to other papers like ReACT (Reasoning and Acting in Language Models). At the core, the stepwise planner allows for the AI to form "thoughts" and "observations" and execute actions based off those to achieve a user's goal. This continues until all required functions are complete and a final output is generated.

See a video walkthrough of Stepwise Planner [here](#).

```
[ ]: from semantic_kernel.planning import StepwisePlanner
      from semantic_kernel.planning.stepwise_planner.stepwise_planner_config import (
          StepwisePlannerConfig,
      )
```



Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

Let's create a Bing Search native skill that we can pass in to the Kernel.

Make sure you have a Bing Search API key in your `.env` file

(<https://www.microsoft.com/en-us/bing/apis/bing-web-search-api>)

```
[ ]: class WebSearchEngineSkill:
    """
    A search engine skill.
    """
    from semantic_kernel.orchestration.sk_context import SKContext
    from semantic_kernel.skill_definition import sk_function, sk_function_context_parameter

    def __init__(self, connector) -> None:
        self._connector = connector

    @sk_function(
        description="Performs a web search for a given query", name="searchAsync"
    )
    @sk_function_context_parameter(
        name="query",
        description="The search query",
    )
    async def search_async(self, query: str, context: SKContext) -> str:
        query = query or context.variables.get("query")[1]
        result = await self._connector.search_async(query, num_results=5, offset=0)
        return str(result)
```

```
[ ]: from semantic_kernel.connectors.search_engine import BingConnector

BING_API_KEY = sk.bing_search_settings_from_dot_env()
connector = BingConnector(BING_API_KEY)
kernel.import_skill(WebSearchEngineSkill(connector), skill_name="WebSearch")
```



Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

Let's also add a couple more skills

```
[ ]: from semantic_kernel.core_skills.math_skill import MathSkill
     from semantic_kernel.core_skills.time_skill import TimeSkill

     kernel.import_skill(TimeSkill(), "time")
     kernel.import_skill(MathSkill(), "math")

[ ]: planner = StepwisePlanner(
     kernel, StepwisePlannerConfig(max_iterations=10, min_iteration_time_ms=1000)
     )
```




Now let's do a more complicated ask that will require planner to make a call to Bing to get the latest information.

```
[37]: ask = """How many total championships combined do the top 5 teams in the NBA have?"""
```

```
plan = planner.create_plan(goal=ask)
```

```
[38]: result = await plan.invoke_async()
```

```
[39]: print(result)
```

The top 5 teams in the NBA have a combined total of 49 championships.

Let's see the steps that the AI took to get to the answer.

```
[40]: for index, step in enumerate(plan._steps):
    print("Step:", index)
    print("Description:", step.description)
    print("Function:", step.skill_name + "." + step._function.name)
    if len(step._outputs) > 0:
        print( " Output:\n", str.replace(result[step._outputs[0]], "\n", "\n  "))
```



Microsoft Semantic Kernel

The Easy Way To Add LLM AI To Your App

Step: 0

Description: Execute a plan

Function: StepwisePlanner.ExecutePlan

Output:

This was my previous work (but they haven't seen any of it! They only see what I return as final answer):

[THOUGHT]

To find the total championships combined for the top 5 teams in the NBA, I need to first identify the top 5 teams and then find the number of championships each team has won. I can use the WebSearch.searchAsync function to find this information.

[ACTION]

```
{"action": "WebSearch.searchAsync", "action_variables": {"query": "top 5 NBA teams with most championships"}}
```

[OBSERVATION]

["The Celtics and Lakers are tied for the most championships in NBA history with 17 each. Last season, the Warriors won the franchise's seventh title, defeating the Celtics in the 2022 NBA...", 'The Lakers of Kareem Abdul-Jabbar, Magic Johnson, Shaquille O'Neal and Kobe Bryant and the Celtics of Bob Cousy, Bill Russell and Larry Bird are among the best teams ever, and rightfully so. List of NBA franchises with multiple championships NBA Teams that never won a Championship', 'Only five teams have managed to win more than three championships: The Boston Celtics, Los Angeles Lakers, Chicago Bulls, San Antonio Spurs, and Golden State Warriors. In fact, these franchises have alone claimed nearly 70 percent of all titles. Los Angeles Lakers Have Won Last time In 2020: Photo Source: Sportsnews.com', 'Boston Celtics (1959 - 1966) Three consecutive Minneapolis Lakers (1952 - 1954) Chicago Bulls (1991 - 1993) Chicago Bulls (1996 - 1998) Los Angeles Lakers (2000 - 2002) Two consecutive Minneapolis Lakers (1949 - 1950) Boston Celtics (1968 - 1969) Los Angeles Lakers (1987 - 1988) Detroit Pistons (1989 - 1990)']

[THOUGHT]

Based on the search results, the top 5 NBA teams with the most championships are the Boston Celtics, Los Angeles Lakers, Chicago Bulls, San Antonio Spurs, and Golden State Warriors. I will now search for the number of championships each team has won.

[ACTION]

```
{"action": "WebSearch.searchAsync", "action_variables": {"query": "number of championships for Boston Celtics, Los Angeles Lakers, Chicago Bulls, San Antonio Spurs, and Golden State Warriors"}}
```



Microsoft Semantic Kernel

The Easy Way To Add LLM AI To Your App

[THOUGHT]

I found the number of championships for the Chicago Bulls, San Antonio Spurs, and Golden State Warriors. The Chicago Bulls have 6 championships, the San Antonio Spurs have 5 championships, and the Golden State Warriors have 4 championships. Now I can calculate the total number of championships for the top 5 teams.

[ACTION]

```
{"action": "math.Add", "action_variables": {"input": "17", "Amount": "17"}}
```

[OBSERVATION]

34

[THOUGHT]

[ACTION]

```
{"action": "math.Add", "action_variables": {"input": "34", "Amount": "6"}}
```

[ACTION]

```
{"action": "math.Add", "action_variables": {"input": "34", "Amount": "6"}}
```

[OBSERVATION]

40

[THOUGHT]

[ACTION]

```
{"action": "math.Add", "action_variables": {"input": "40", "Amount": "5"}}
```

[OBSERVATION]

45

[THOUGHT]

[ACTION]

```
{"action": "math.Add", "action_variables": {"input": "45", "Amount": "4"}}
```

[OBSERVATION]

49



Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

But what's the answer? (as of Sept 2023)

“Top 5 Teams” could be ambiguous.

Lakers: 17

Celtics : 17

Warriors: 7

Bulls: 6

Spurs: 5

Answer: **52** championships

TLDR: We still have a long way to go...

The screenshot shows a chatbot interface with a blue speech bubble icon on the left and a row of icons (thumbs up, thumbs down, copy, download) on the right. The main text reads: "According to the web search results, the top five teams in the NBA by championships are the Los Angeles Lakers, the Boston Celtics, the Golden State Warriors, the Chicago Bulls, and the San Antonio Spurs ^{1 2 3 4}. These five teams have won a combined total of **42 championships** out of the 76 seasons in NBA history. Here is the breakdown of how many championships each team has won:

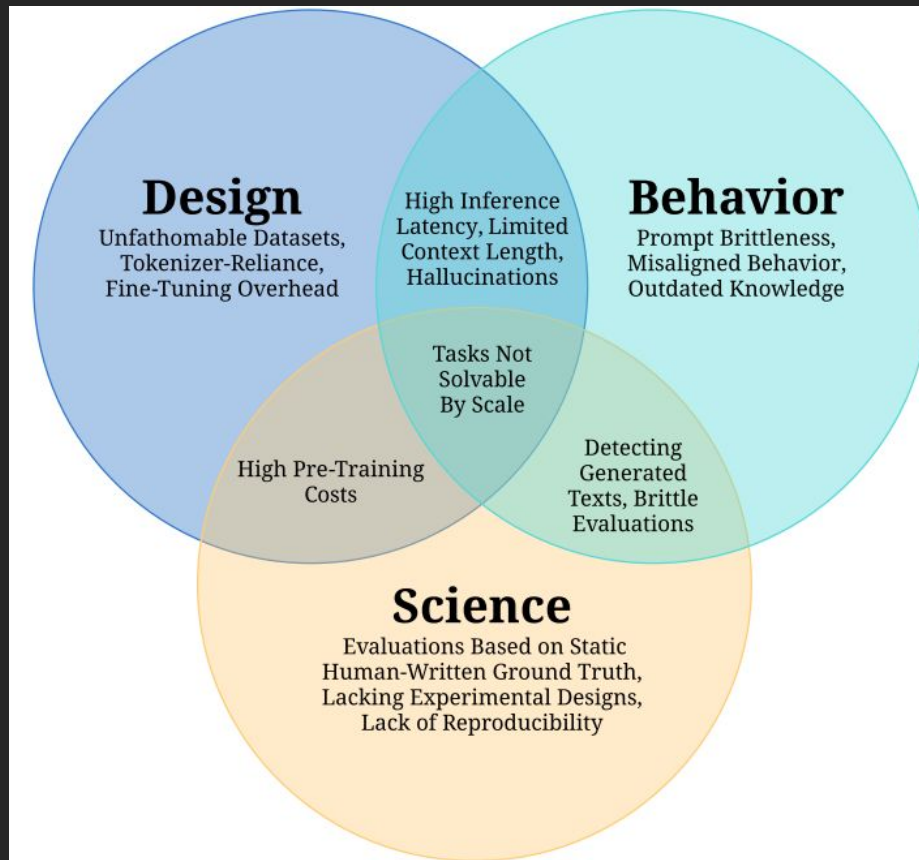
- Los Angeles Lakers: **17 championships** ^{1 5}
- Boston Celtics: **17 championships** ^{1 5}
- Golden State Warriors: **7 championships** ^{1 5}
- Chicago Bulls: **6 championships** ^{1 5}
- San Antonio Spurs: **5 championships** ^{1 5}

I hope this answers your question. 😊

Learn more:

- 1. [sportingnews.com](#)
- 2. [jobsinsports.com](#)
- 3. [sportszpro.com](#)
- 4. [scottfujita.com](#)
- +2 more

Many Open Challenges Remain



Challenges and Applications of Large Language Models
([2307.10169.pdf \(arxiv.org\)](https://arxiv.org/pdf/2307.10169.pdf))



Semantic Kernel Discord

Join in on the conversation and
let's learn together!



dluc. invited you to join

Semantic Kernel

● 417 Online ● 3,385 Members

DISPLAY NAME

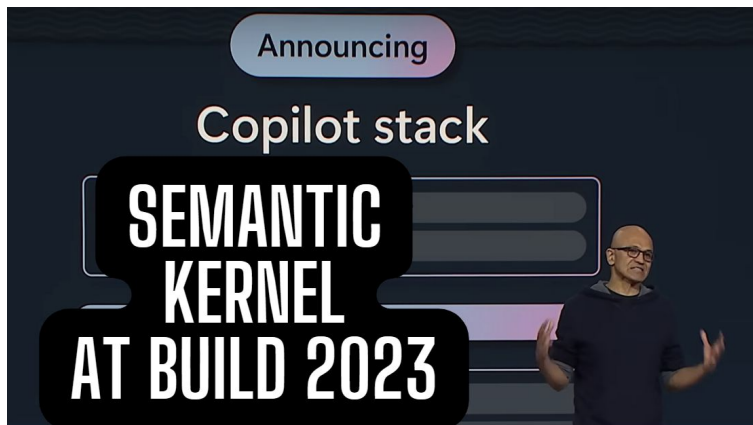
This is how others see you. You can use special characters and emoji.

Continue

By registering, you agree to Discord's [Terms of Service](#) and [Privacy Policy](#).

[Already have an account?](#)

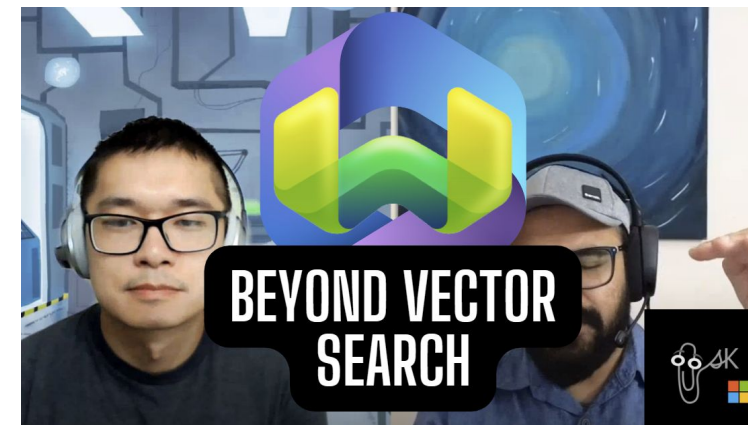
Find SK content (on YouTube!)



Copilot Stack at Build 2023

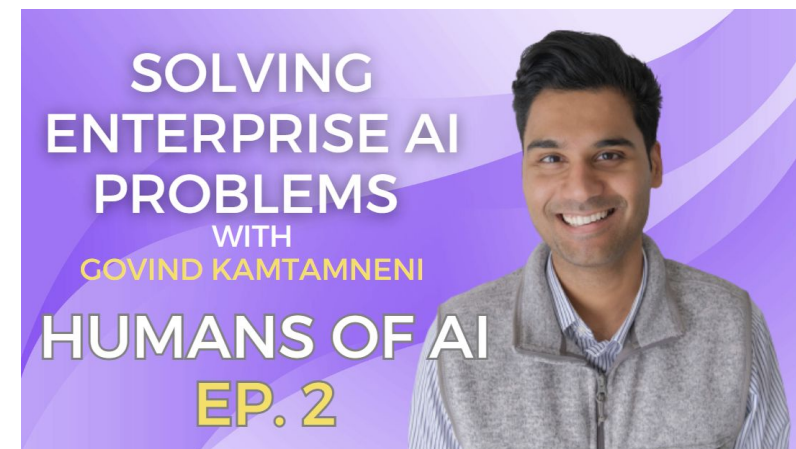


Azure Cognitive Search + SK



3P Vector DBs integrations

08.16.23
OFFICE
HOURS





Let's Connect!

• [Newsletter](#) | [Twitter](#) | [LinkedIn](#) | [YouTube](#)





Microsoft

Microsoft
for Startups

Microsoft for Startups
Founders Hub

Sign up today at
startups.microsoft.com





Microsoft

Semantic Kernel

The Easy Way To Add LLM AI To Your App

Thank you for helping
everyone exceed their goals.

GitHub Repo
Semantic Kernel Docs
Join the community

aka.ms/skrepo
aka.ms/semantic-kernel
aka.ms/sk-community

