

# MCP ile Yapay Zeka İşgörenlerinin Araç Kullanımı

Statik Bilgiden Dinamik Kararlara ve Eylemlere

Murat Karakaya  
Prof. Dr.  
TED Üniversitesi

# Gündem

## 1. Giriş: Akıllı İşgörenler Çağı

- Yapay Zeka (YZ) İşgöreni (AI Agent) Nedir?
- Neden Geleneksel Sohbet Robotlarından Farklıdırlar?

## 2. Büyük Dil Modellerinin (BDM) Kısıtlılıkları

- Halüsinasyon (Hallucination), Güncellik ve Bağlam Penceresi (Context Window) Sorunları

## 3. Çözüm Arayışı: Araç Kullanımı (Tool/Function Calling)

- Tanım, İş Akışı ve Avantajları
- Temel Kısıtlılığı: "N x M Problemi"

## 4. Standartlaşma İhtiyacı: Model Bağlam Protokolü (MCP)

- MCP Nedir? "YZ için USB-C Portu"
- Geliştiriciler ve Destekçiler
- Mimari: Ana Bilgisayar (Host), İstemci (Client) ve Sunucu (Server)

## 5. Karşılaştırmalı Analiz

- Araç Kullanımı ve MCP Arasındaki Temel Farklar

## 6. MCP: Fırsatlar, Zorluklar ve Gelecek

## 7. Sonuç ve Soru-Cevap



# MURAT KARAKAYA AKADEMI

DEEP LEARNING TUTORIALS  
WITH PYTHON

AI  
DEEP LEARNING  
MACHINE LEARNING  
PYTHON  
KERAS  
TENSORFLOW  
SCIKIT LEARN



## Murat Karakaya Akademi

@MuratKarakayaAkademi • 9.79K subscribers • 380 videos

Hello there! ...more

[youtube.com/channel/UCrCxCxTFL2ytaDrDYrN4\\_eA?sub\\_confirmation=1](https://youtube.com/channel/UCrCxCxTFL2ytaDrDYrN4_eA?sub_confirmation=1) and 7 more links

[Customize channel](#)[Manage videos](#)

Home Videos Shorts Live **Playlists** Community Membership

Created playlists

Sort by

### Developing Applications with AI Agents



**Automatic Report  
Generation  
using Multi Agents**  
2 videos

News from Murat Karakaya  
Akademi

[View full playlist](#)



Kanaldan Haberler  
[View full playlist](#)

### Kanal Üyeligi

**Üyelik  
Ayrıcalıkları  
Nelerdir?**  
1 video



AI Agents and Multi Agent  
Frameworks

[View full playlist](#)

### AI Embedded Software Development

**Automate  
Your FAQ Page  
Creation**  
2 videos



Yapay Zeka Destekli Yazılım  
Geliştirme

[View full playlist](#)

### Yapay Zeka ve Yazılım Geliştirme

**-YZ Yazılımları Geliştirme  
-YZ ile Yazılım Geliştirme  
-YZ Gözetim ve Kontrol**  
1 video



AI Embedded Software  
Development

[View full playlist](#)

### AI Embedded Software Development

**Automate  
Your FAQ Page  
Creation**  
3 videos



Yapay Zeka Ajanları

[View full playlist](#)

### Yapay Zeka Gömülü Yazılım Geliştirme

**Fonksiyon Çağırma  
Kodlamada Nasıl  
Kullanılır?**  
7 videos

# Akıllı İşgörenler Çağı

## Muhakemeden Eyleme

Bir YZ işgöreni (AI Agent), yalnızca girdi-çıkı üreten sistemlerin ötesinde; muhakeme (reasoning) yeteneği ile otonom (autonomous) kararlar alıp eyleme geçebilen sistemlerdir.

### Temel Yetenekleri:

- **Niyet Anlama:** Kullanıcının karmaşık taleplerini anlar.
- **Planlama:** Niyete ulaşmak için bir eylem planı oluşturur.
- **Araç Kullanımı:** Planı uygulamak için dış dünyadaki araçları (API'ler, veritabanları vb.) kullanır.

### Örnek Görev:

- **ASYU Konferansı bu yıl nerede ne zaman yapılacak?**

# BDM Tabanlı İşgörenlerin Temel Kısıtlılıkları

## Güçlü Muhakeme, Ancak Kapalı Bir Dünya

BDM'ler, geniş veri kümeleri üzerinde eğitilerek güçlü genel muhakeme yetenekleri kazansalar da doğaları gereği temel kısıtlılıklara sahiptirler.

### 1. Halüsinasyon (Hallucination):

- Eğitim verilerinde olmayan veya tutarsız bilgilere dayanarak, kulağa mantıklı gelen ancak tamamen yanlış yanıtlar üretebilirler.
- Bu durum, finans, hukuk ve tıp gibi hassas alanlarda kritik bir risktir.

### 2. Statik ve Güncel Olmayan Bilgi:

- BDM'ler eğitildikten sonra bilgileri statikleşir.
- Yeni olaylar veya bilgilerle güncellenmeleri çok maliyetli ve verimsizdir. Bu da modellerin hızla güncelliğini yitirmesine neden olur.

### 3. Sınırlı Bağlam Penceresi (Context Window):

- Tek seferde işleyebilecekleri metin miktarı sınırlıdır. Bu, uzun diyaloglarda veya doküman analizlerinde bağlam kaybına yol açar.

### 4. Diğer Sorunlar:

- Eğitim verilerindeki önyargıları (bias) yansıtma ve gizlilik (privacy) endişeleri.

# İlk Çözüm: Araç Kullanımı (Tool/Function Calling)

## YZ'nin Dış Dünyaya Açılan Penceresi

Araç kullanımı, BDM'lerin **harici** sistemlerle arayüz kurarak eğitim verileri dışında kalan **bilgilere erişmesini** ve **eylemler gerçekleştirmesini** sağlayan bir mekanizmadır.

BDM'leri pasif bilgi işlemcilerden, dış dünyayla etkileşime girebilen **aktif** katılımcılara dönüştürür.

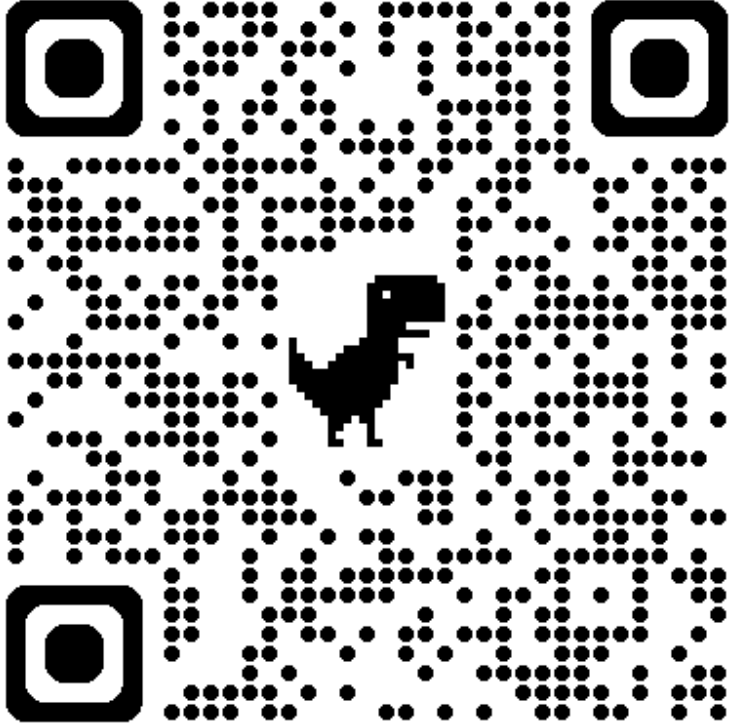
# İlk Çözüm: Araç Kullanımı (Tool/Function Calling)

## YZ'nin Dış Dünyaya Açılan Penceresi

### Avantajları:

- **Dinamik ve Güncel Bilgi Erişimi:** Gerçek zamanlı verilere (hava durumu, borsa, müşteri kayıtları) erişim sağlar.
- **Eylem Gerçekleştirme:** Bir CRM güncelleme, e-posta gönderme gibi gerçek dünya eylemlerini mümkün kılar.
- **Halüsinasyon Azaltma:** Yanıtları harici ve doğrulanmış kaynaklardan gelen verilerle üreterek halüsinasyon riskini azaltır.

Sunumdaki tüm kodlara ve sunumun kendisine [https://github.com/kmkarakaya/mcp\\_tutorial](https://github.com/kmkarakaya/mcp_tutorial) ulaşabilirsiniz.



github.com/kmkarakaya/mcp\_tutorial

kmkarakaya / mcp\_tutorial

<> Code Issues Pull requests Actions Projects Wiki Security Insights

mcp\_tutorial Public Pin Watch 0 Fork 0 Star 0

main Go to file + <> Code

kmkarakaya Initial commit bdaf646 · 8 minutes ago

1_gemini_agent_without_to...	Initial commit	5 minutes ago
2_gemini_agent_with_tools_...	Initial commit	5 minutes ago
3_gemini_agent_with_tools_...	Initial commit	5 minutes ago
4_mcp_server.py	Initial commit	5 minutes ago
5_mcp_client.py	Initial commit	5 minutes ago
6_mcp_gemini_agent.py	Initial commit	5 minutes ago
7_how_to_convert_docker.md	Initial commit	5 minutes ago
8_mcp_docker_server.py	Initial commit	5 minutes ago
9_mcp_docker_gemini agen...	Initial commit	5 minutes ago
Dockerfile	Initial commit	5 minutes ago
LICENSE	Initial commit	8 minutes ago
requirements.txt	Initial commit	5 minutes ago

About

MCP Tutorial

MIT license

Activity

0 stars

0 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages

Python 98.3%

Dockerfile 1.7%



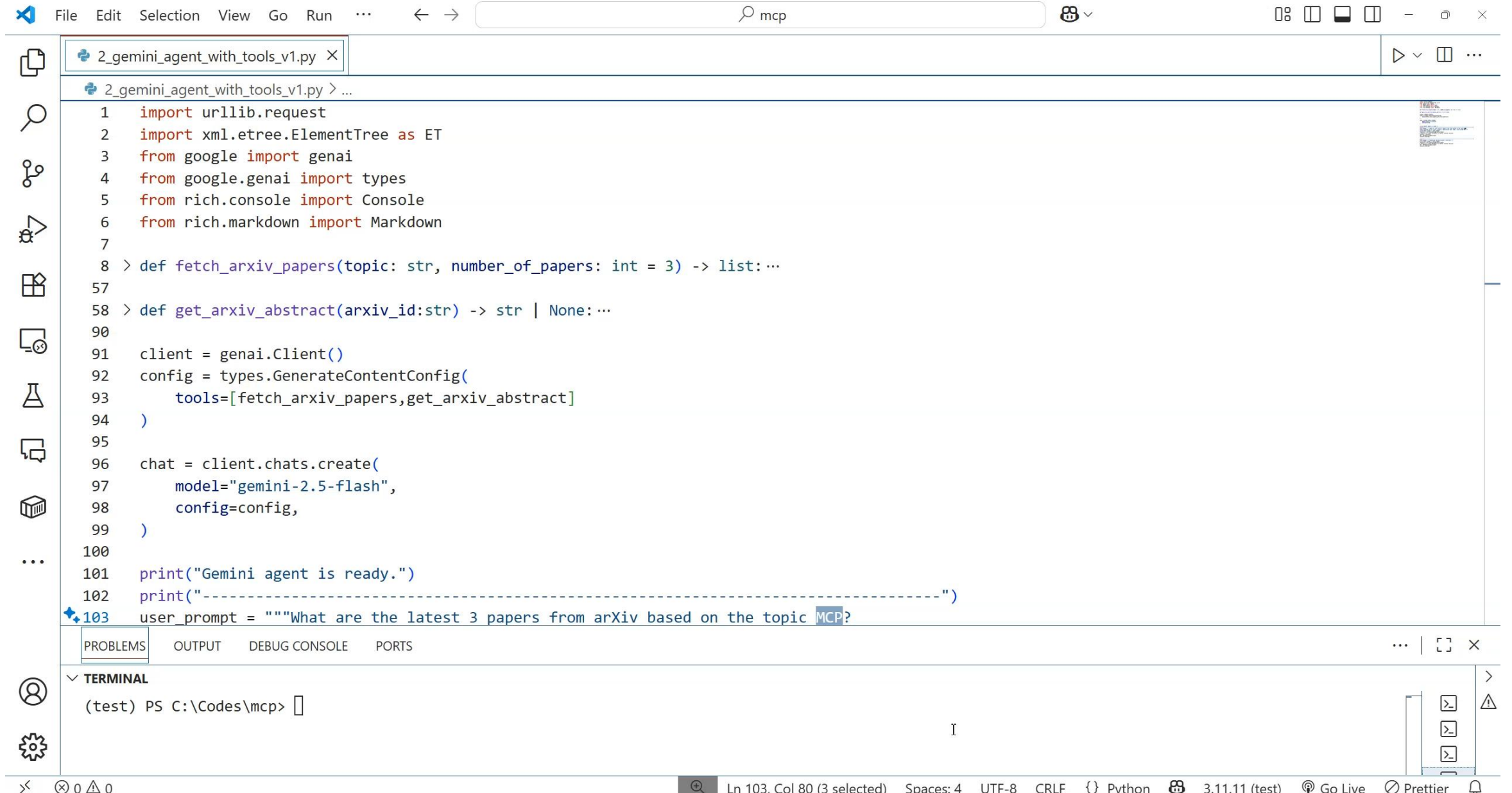
# Araç Kullanmayan YZ İlgöreni

The screenshot displays the Visual Studio Code interface with the following components:

- Menu Bar:** File, Edit, Selection, View, Go, Run, and a search bar containing 'mcp'.
- Explorer Panel:** Shows a project structure with a folder named 'MCP' containing subfolders '\_\_pycache\_\_' and 'reports'. The file '1\_gemini\_agent\_without\_tools.py' is selected and highlighted in blue.
- Code Editor:** Displays the content of '1\_gemini\_agent\_without\_tools.py'. The code is as follows:

```
1 from google import genai
2 from google.genai import types
3 from rich.console import Console
4 from rich.markdown import Markdown
5
6 client = genai.Client()
7 chat = client.chats.create(
8     model="gemini-2.5-flash",
9 )
10
11 user_prompt = """What are the latest 3 papers from arXiv based on the topic MCP?
12 Please provide the title, authors, publication date, and a link to the PDF."""
13 response = chat.send_message(user_prompt)
14 # Use Rich to render Markdown for better terminal display
15 console = Console()
16 md = Markdown(response.text)
17 console.print(md)
```
- Terminal Panel:** Shows a PowerShell prompt '(test) PS C:\Codes\mcp>' with a cursor.
- Bottom Bar:** Includes a status bar with 'Ln 17, Col 18', 'Spaces: 4', 'UTF-8', 'CRLF', '{ } Python', '3.11.11 (test)', 'Go Live', 'Prettier', and a bell icon.

# Araç Kullanan YZ İşgöreni: Basit Görevler



The image shows a Visual Studio Code editor window with a Python file named `2_gemini_agent_with_tools_v1.py`. The script defines two functions: `fetch_arxiv_papers` and `get_arxiv_abstract`, and sets up a Gemini agent with these tools. The terminal window at the bottom shows the command prompt in a PowerShell session.

```
File Edit Selection View Go Run ... ← → mcp
```

```
2_gemini_agent_with_tools_v1.py X
```

```
2_gemini_agent_with_tools_v1.py > ...
```

```
1 import urllib.request
2 import xml.etree.ElementTree as ET
3 from google import genai
4 from google.genai import types
5 from rich.console import Console
6 from rich.markdown import Markdown
7
8 > def fetch_arxiv_papers(topic: str, number_of_papers: int = 3) -> list: ...
57
58 > def get_arxiv_abstract(arxiv_id: str) -> str | None: ...
90
91 client = genai.Client()
92 config = types.GenerateContentConfig(
93     tools=[fetch_arxiv_papers, get_arxiv_abstract]
94 )
95
96 chat = client.chats.create(
97     model="gemini-2.5-flash",
98     config=config,
99 )
100
101 print("Gemini agent is ready.")
102 print("-----")
103 user_prompt = """What are the latest 3 papers from arXiv based on the topic MCP?
```

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS
```

```
TERMINAL
```

```
(test) PS C:\Codes\mcp> 
```

Ln 103, Col 80 (3 selected) Spaces: 4 UTF-8 CRLF {} Python 3.11.11 (test) Go Live Prettier

# Araç Kullanan YZ İşgöreni: Karmaşık Görevler

File Edit Selection View Go Run ... ← → mcp

EXPLORER

- 3\_gemini\_agent\_with\_tools\_v2.py
- MCP
  - > \_\_pycache\_\_
  - > reports
  - 1\_gemini\_agent\_without\_tools.py
  - 2\_gemini\_agent\_with\_tools\_v1.py
  - 3\_gemini\_agent\_with\_tools\_v2.py
  - 4\_mcp\_server.py
  - 5\_mcp\_client.py
  - 6\_mcp\_gemini\_agent.py
  - 7\_how\_to\_convert\_docker.md
  - 8\_mcp\_docker\_server.py
  - 9\_mcp\_docker\_gemini\_agent.py
  - Dockerfile
  - requirements.txt

3\_gemini\_agent\_with\_tools\_v2.py > ...

```
1 import urllib.request
2 import xml.etree.ElementTree as ET
3 from google import genai
4 from google.genai import types
5 from rich.console import Console
6 from rich.markdown import Markdown
7 import os
8 import re
9
10
11 > def fetch_arxiv_papers(topic: str, number_of_papers: int = 3) -> list: ...
60
61 > def get_arxiv_abstract(arxiv_id: str) -> str | None: ...
93
94 > def save_md_to_file(text: str, filename: str) -> None: ...
138
139 client = genai.Client()
140 config = types.GenerateContentConfig(
141     tools=[fetch_arxiv_papers, get_arxiv_abstract, save_md_to_file]
142 )
143
144 chat = client.chats.create(
145     model="gemini-2.5-flash",
146     config=config,
147 )
148
149 user_prompt = """Prepare a report summarizing the research problems of the latest 3 papers
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS

OUTLINE

TERMINAL

(test) PS C:\Codes\mcp>

Ln 156, Col 18 Spaces: 4 UTF-8 CRLF {} Python 3.11.11 (test) Go Live Prettier

# Araç Kullanımının Temel Kısıtlılığı

## Standart Değil, Bir Entegrasyon Soyutlaması

Araç kullanımı, modelin çıktısını belirli bir **formatta** (genellikle JSON) olmaya "**zorlayan**" bir mühendislik yaklaşımıdır.

### Sorun: Sağlayıcıya Özgü Formatlar

- Her BDM sağlayıcısı (OpenAI, Google, Mistral vb.) kendi araç çağırma formatını ve mimarisini geliştirmiştir.
- Bu durum, geliştiricilerin her model için ayrı entegrasyon kodu yazmasını gerektirir.

### "N x M Problemi":

- N sayıda farklı BDM'yi, M sayıda farklı araca entegre etmek, N x M adet özel bağlantı katmanı (custom connector) oluşturmayı gerektirir.
- Bu, geliştirme maliyetlerini, karmaşıklığı ve bakım yükünü ciddi şekilde artırmaktadır.
- Bu parçalı ekosistem, ölçeklenebilir ve sürdürülebilir bir çözüm değildir. Bu durum, standart ve güvenli bir protokole olan ihtiyacı doğurmuştur.

# Çözüm: Model Bağlam Protokolü (MCP)

## Entegrasyonun Geleceği: "YZ için USB-C Portu"

MCP, Kasım 2024'te Anthropic tarafından tanıtılan, BDM'lerin harici araçlar, veriler ve hizmetlerle entegrasyonunu standartlaştırmak için tasarlanmış açık kaynaklı (open-source) bir protokoldür.

**Temel Amaç:** Araç kullanımının yarattığı "N x M" entegrasyon problemini ortadan kaldırarak, yapay zeka ekosistemi için "tak-çalıştır" (plug-and-play) bir mimari oluşturmak.

# Çözüm: Model Bağlam Protokolü (MCP)

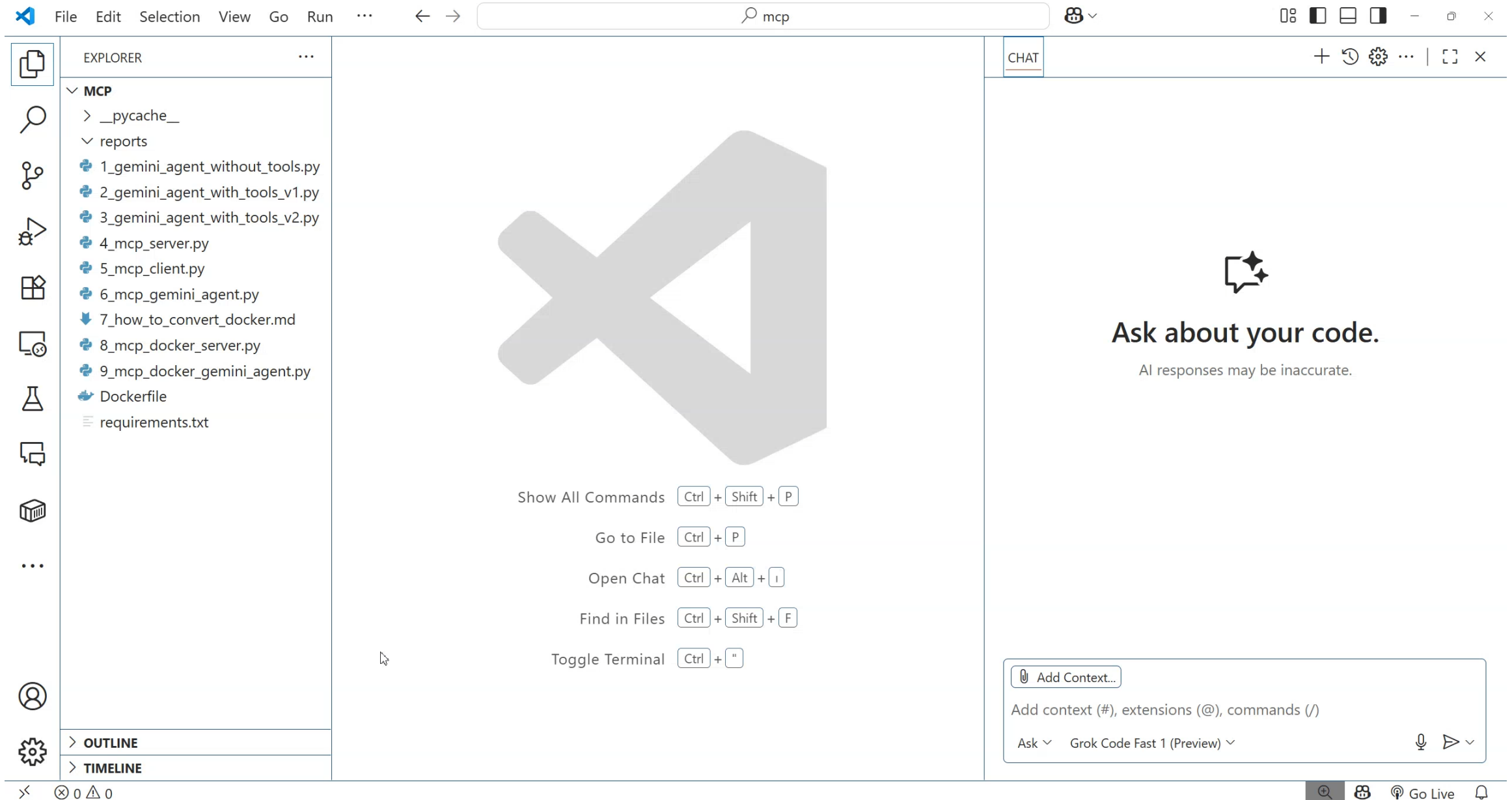
## Entegrasyonun Geleceği: "YZ için USB-C Portu"

### Geliştirici ve Destekçiler:

- **Geliştiren:** Anthropic.
- **Benimseyenler:** OpenAI Google, Claude gibi büyük BDM sağlayıcıları tarafından hızla benimsenmiştir.
- **Ekosistem:** VS Code, Cursor, Zed, Replit, Sourcegraph gibi IDE'ler ve kodlama platformları da desteklemektedir.

Bu geniş endüstri desteği, MCP'nin standart olma potansiyelini güçlendirmektedir.

# VS CODE için MCP Sunucuları



# MCP Mimarisi ve Bileşenleri

## Dağıtık ve Modüler Bir Mimari

MCP, üç ana bileşenden oluşan bir istemci-sunucu (client-server) mimarisine dayanır. Bu yapı, ana uygulamayı ve araçları birbirinden ayırır.

**1. Ana Bilgisayar (Host):**

**2. İstemci (Client):**

**3. Sunucu (Server):**



# MCP Mimarisi ve Bileşenleri

## Dağıtık ve Modüler Bir Mimari

### 1. Ana Bilgisayar (Host):

- Kullanıcının etkileşimde bulunduğu ana YZ uygulamasıdır (örn. Claude Desktop, VS Code, ya da Uygulamamızın çalıştığı sunucu).
- BDM'yi içerir, istemcileri yönetir ve güvenlik politikalarını uygular.

### 2. MCP İstemci (Client):

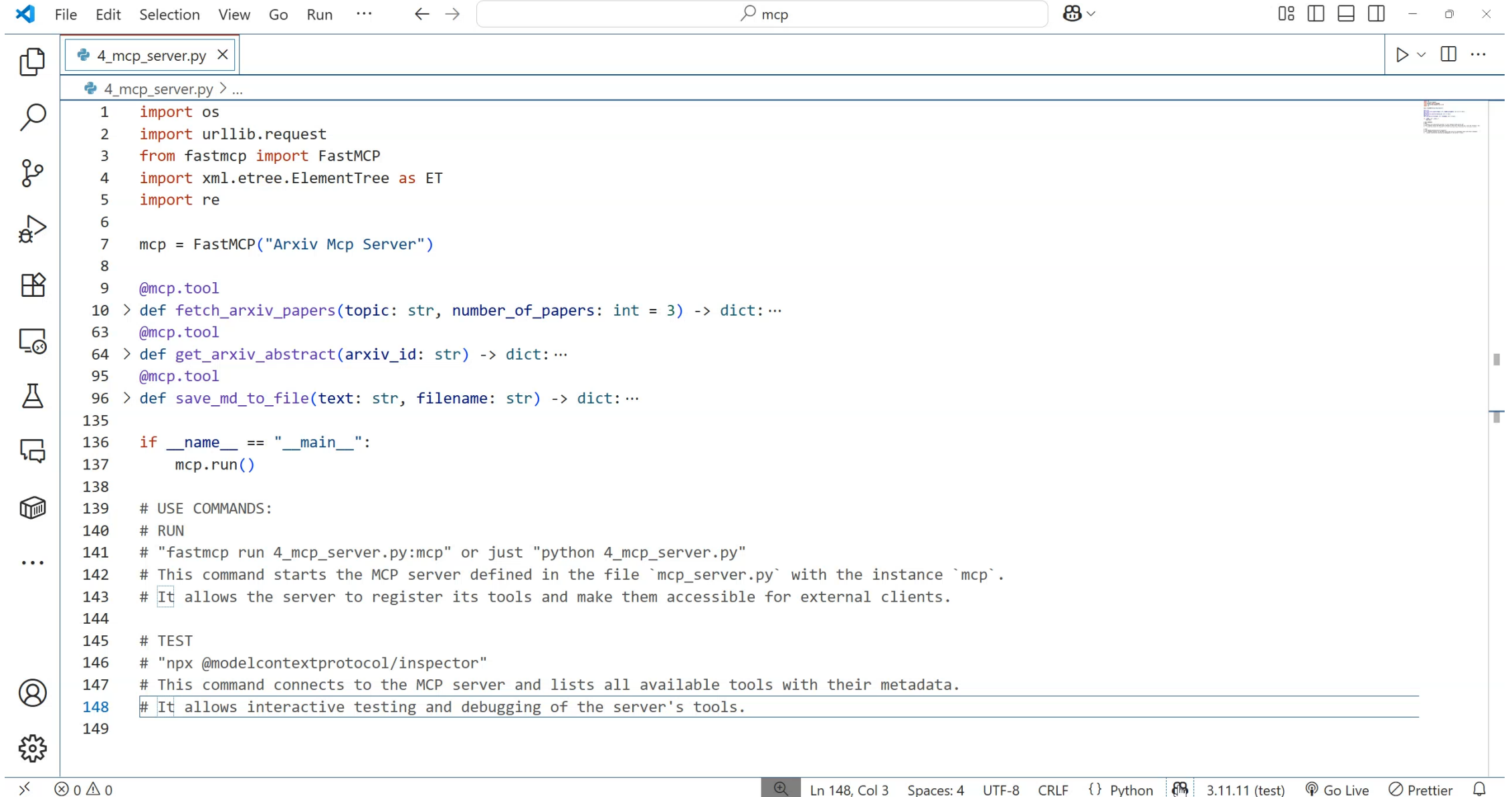
- Ana bilgisayar içinde yer alır ve her bir sunucuya bire bir (1:1) bağlantı kurar.
- BDM'den gelen istekleri protokole çevirir ve sunucudan gelen yanıtları BDM'ye iletir.

### 3. MCP Sunucu (Server):

- Dış dünyadaki veri ve araçları sağlayan harici hizmettir.
- Yerel bir süreç (dosya sistemi) veya uzak bir API olabilir.
- Her sunucu tek bir alana odaklanır (örn. TODO listesi, dosya sistemi), bu da sistemin karmaşıklığını azaltır (Separation of Concerns).

# MCP Mimarisi ve Bileşenleri

## MCP Sunucusu Oluşturma ve Testi



The screenshot shows a code editor with a file named `4_mcp_server.py`. The code implements an MCP server using the `fastmcp` library. It includes imports for `os`, `urllib.request`, `FastMCP`, `ElementTree`, and `re`. The server is initialized with the name "Arxiv Mcp Server". Three tools are registered: `fetch_arxiv_papers`, `get_arxiv_abstract`, and `save_md_to_file`. The `__main__` block contains instructions on how to run the server and test it using the `modelcontextprotocol/inspector` command.

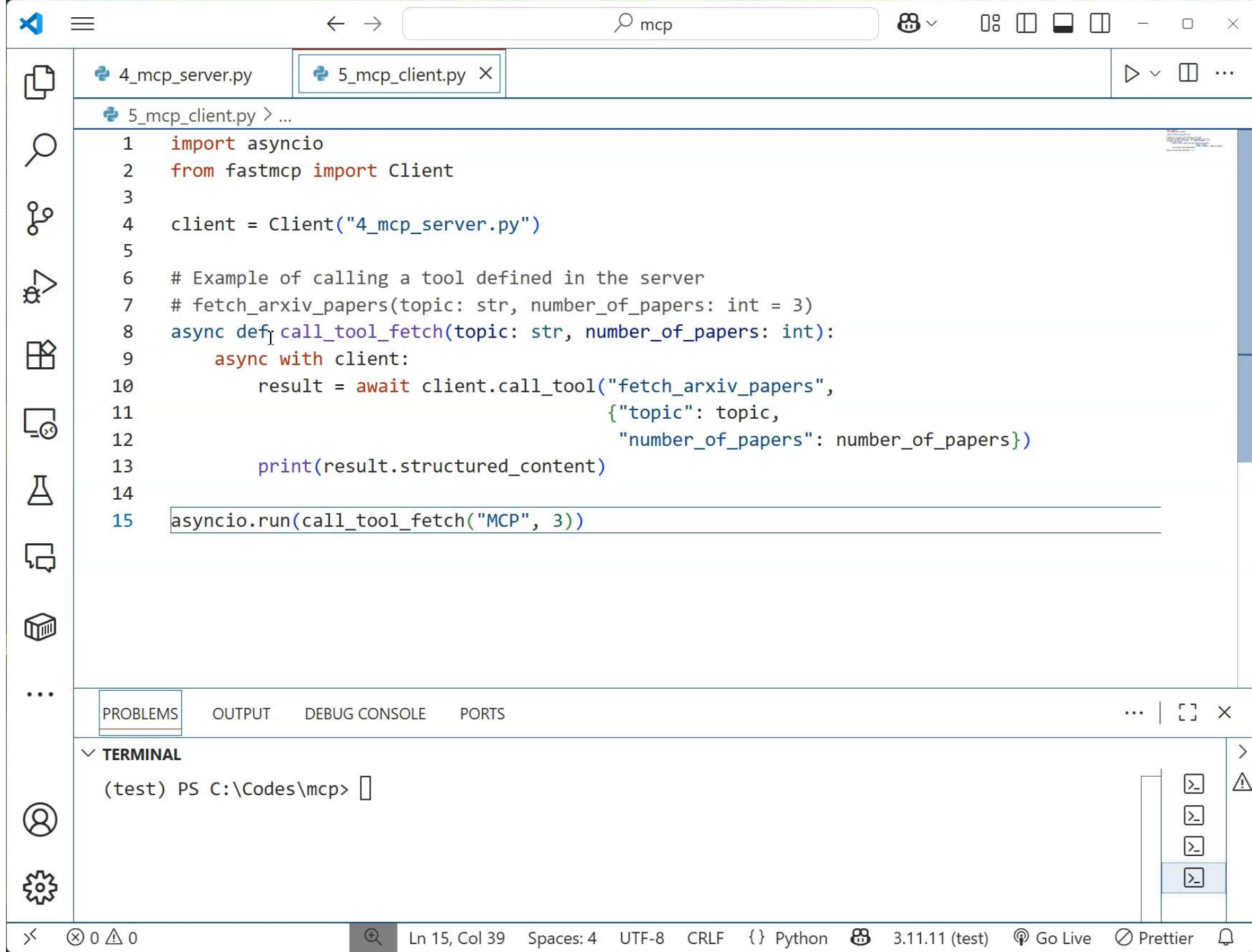
```
1 import os
2 import urllib.request
3 from fastmcp import FastMCP
4 import xml.etree.ElementTree as ET
5 import re
6
7 mcp = FastMCP("Arxiv Mcp Server")
8
9 @mcp.tool
10 > def fetch_arxiv_papers(topic: str, number_of_papers: int = 3) -> dict:...
63 @mcp.tool
64 > def get_arxiv_abstract(arxiv_id: str) -> dict:...
95 @mcp.tool
96 > def save_md_to_file(text: str, filename: str) -> dict:...
135
136 if __name__ == "__main__":
137     mcp.run()
138
139 # USE COMMANDS:
140 # RUN
141 # "fastmcp run 4_mcp_server.py:mcp" or just "python 4_mcp_server.py"
142 # This command starts the MCP server defined in the file `mcp_server.py` with the instance `mcp`.
143 # It allows the server to register its tools and make them accessible for external clients.
144
145 # TEST
146 # "npx @modelcontextprotocol/inspector"
147 # This command connects to the MCP server and lists all available tools with their metadata.
148 # It allows interactive testing and debugging of the server's tools.
149
```

Ln 148, Col 3 Spaces: 4 UTF-8 CRLF {} Python 3.11.11 (test) Go Live Prettier

# MCP

## Mimarisi ve Bileşenleri

## MCP İstemcisi Oluşturma



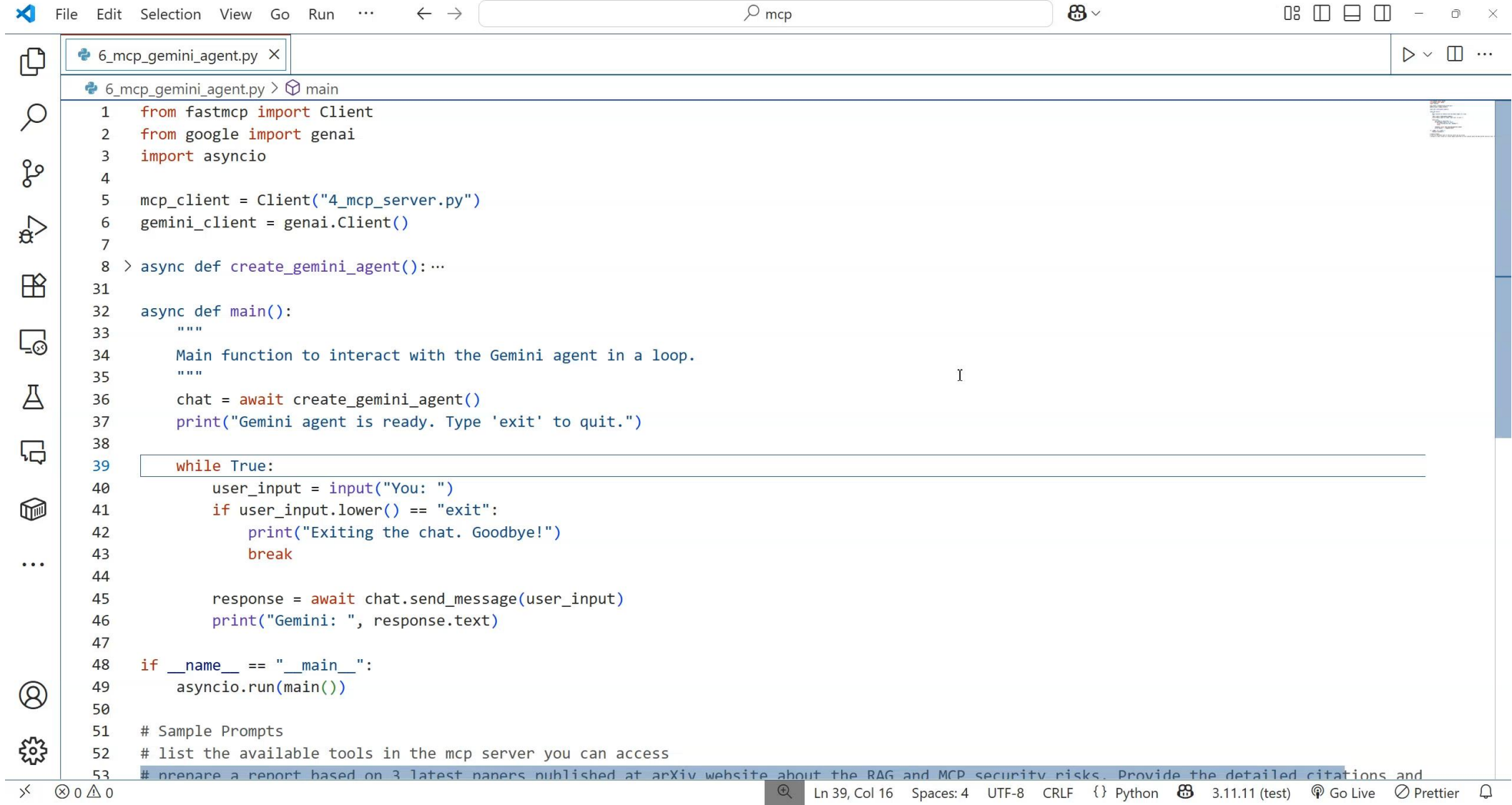
The screenshot shows a code editor with two tabs: `4_mcp_server.py` and `5_mcp_client.py`. The `5_mcp_client.py` tab is active, displaying the following Python code:

```
1 import asyncio
2 from fastmcp import Client
3
4 client = Client("4_mcp_server.py")
5
6 # Example of calling a tool defined in the server
7 # fetch_arxiv_papers(topic: str, number_of_papers: int = 3)
8 async def call_tool_fetch(topic: str, number_of_papers: int):
9     async with client:
10         result = await client.call_tool("fetch_arxiv_papers",
11                                         {"topic": topic,
12                                          "number_of_papers": number_of_papers})
13         print(result.structured_content)
14
15 asyncio.run(call_tool_fetch("MCP", 3))
```

The editor's interface includes a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, Extensions, Testing, Run and Test, Accounts, and Settings. The bottom panel shows the **PROBLEMS** tab, which is currently empty, and the **TERMINAL** tab, which displays the command prompt: `(test) PS C:\Codes\mcp>`. The status bar at the bottom indicates the current position is **Ln 15, Col 39**, with **Spaces: 4**, **UTF-8** encoding, **CRLF** line endings, **Python** language, **3.11.11 (test)** version, and **Go Live** and **Prettier** extensions.

# MCP ile BDM Entegrasyonu

## YZ İlgörenlerinin MCP Sunucusunu Kullanması



```
File Edit Selection View Go Run ... ← → mcp
6_mcp_gemini_agent.py X
6_mcp_gemini_agent.py > main
1 from fastmcp import Client
2 from google import genai
3 import asyncio
4
5 mcp_client = Client("4_mcp_server.py")
6 gemini_client = genai.Client()
7
8 > async def create_gemini_agent():...
31
32 async def main():
33     """
34     Main function to interact with the Gemini agent in a loop.
35     """
36     chat = await create_gemini_agent()
37     print("Gemini agent is ready. Type 'exit' to quit.")
38
39     while True:
40         user_input = input("You: ")
41         if user_input.lower() == "exit":
42             print("Exiting the chat. Goodbye!")
43             break
44
45         response = await chat.send_message(user_input)
46         print("Gemini: ", response.text)
47
48 if __name__ == "__main__":
49     asyncio.run(main())
50
51 # Sample Prompts
52 # list the available tools in the mcp server you can access
53 # prepare a report based on 3 latest papers published at arXiv website about the RAG and MCP security risks. Provide the detailed citations and
```

Ln 39, Col 16 Spaces: 4 UTF-8 CRLF {} Python 3.11.11 (test) Go Live Prettier

# Araç Kullanımı x MCP

## İki Farklı Yaklaşım, Tek Bir Hedef

Özellik	Araç Kullanımı (Function Calling)	Model Bağlam Protokolü (MCP)
Mimari	Monolitik (monolithic) - Araçlar uygulama içine gömülüdür.	Dağıtık (distributed) - Araçlar harici sunucularda çalışır.
Standartlaşma	Sağlayıcıya özgü (vendor-specific) - Her BDM'nin kendi formatı vardır.	Açık Standart (open standard) - Her model ve araç uyumlu olabilir.
Bağlam Yönetimi	Tek seferlik istek-yanıt döngüsü - Bağlamın devamlılığı zordur.	Durumlu (stateful) oturum - Uzun diyaloglarda bağlamı korur.
Geliştirme Karmaşıklığı	Uygulama bazında düşük, ancak her entegrasyon için yüksek.	Başlangıçta daha karmaşık, ancak ölçeklendikçe kolaylaşır.
Güvenlik	Uygulama bağımlı - Her geliştirici kendi önlemini almalıdır.	Protokol odaklı - Ana bilgisayar (host) izinleri merkezden yönetir.
Yeniden Kullanılabilirlik	Düşük - Araçlar tek bir uygulamaya sıkı sıkıya bağlıdır.	Yüksek - Bir MCP sunucusu birçok farklı istemci tarafından kullanılabilir.

# Karşılaştırmalı Analiz: Araç Kullanımı vs. MCP

## İki Farklı Yaklaşım, Tek Bir Hedef

Özellik	Araç Kullanımı (Function Calling)	Model Bağlam Protokolü (MCP)
Mimari	Monolitik (monolithic) - Araçlar uygulama içine gömülüdür.	Dağıtık (distributed) - Araçlar harici sunucularda çalışır.
Standartlaşma	Sağlayıcıya özgü (vendor-specific) - Her BDM'nin kendi formatı vardır.	Açık Standart (open standard) - Her model ve araç uyumlu olabilir.
Bağlam Yönetimi	Tek seferlik istek-yanıt döngüsü - Bağlamın devamlılığı zordur.	Durumlu (stateful) oturum - Uzun diyaloglarda bağlamı korur.
Geliştirme Karmaşıklığı	Uygulama bazında düşük, ancak her entegrasyon için yüksek.	Başlangıçta daha karmaşık, ancak ölçeklendikçe kolaylaşır.
Güvenlik	Uygulama bağımlı - Her geliştirici kendi önlemini almalıdır.	Protokol odaklı - Ana bilgisayar (host) izinleri merkezden yönetir.
Yeniden Kullanılabilirlik	Düşük - Araçlar tek bir uygulamaya sıkı sıkıya bağlıdır.	Yüksek - Bir MCP sunucusu birçok farklı istemci tarafından kullanılabilir.

**Temel Çıkarım:** Araç kullanımı, BDM'nin "*ne yapacağına karar vermesini*" sağlarken, MCP "*bu kararı nasıl standart ve güvenli bir şekilde uygulayacağını*" yöneten bir çerçeve sunar

# MCP'nin Avantajları ve Fırsatları

## Standardizasyon, Otonomi ve Güvenlik

### 1. Entegrasyon Probleminin Çözümü:

- "N x M" problemini ortadan kaldırarak YZ entegrasyonlarını standartlaştırır. Bir kez yazılan bir MCP sunucusu, tüm uyumlu istemcilerle çalışabilir. Bu, geliştirme maliyetlerini ve süresini ciddi oranda azaltır.

### 2. Artırılmış Otonomi ve Yetenek:

- BDM'lerin CRM güncelleme, gerçek zamanlı veri analizi veya karmaşık hesaplamalar gibi görevleri otonom olarak gerçekleştirmesini sağlar.

# MCP'nin Avantajları ve Fırsatları

## Standardizasyon, Otonomi ve Güvenlik

### **3. Gelişmiş Güvenlik Modeli:**

- Geleneksel araç kullanımına kıyasla daha güvenli bir model sunar. Ana bilgisayar (host), izin denetimini merkezi olarak yöneterek hassas verilere erişimi ve komut çalıştırmayı kontrol altında tutar.

### **4. Yeniden Kullanılabilirlik ve Bakım Kolaylığı:**

- Dağıtık mimarisi sayesinde, araçlar (sunucular) farklı uygulamalarda yeniden kullanılabilir ve bağımsız olarak güncellenebilir.



# Docker üzerinden MCP Sunucusunu Kullanma

File Edit Selection View Go Run ...

← →

🔍 mcp

👤

🔑 📄 📄 📄 - 🗑️ ✕

📄

EXPLORER

...

🔍

🔗

🔧

📦

🖨️

🧪

💬

📦

...

👤

⚙️

> OUTLINE

> TIMELINE

Preview 7\_how\_to\_convert\_docker.md

8\_mcp\_docker\_server.py

7\_how\_to\_convert\_docker.md ✕

...

7\_how\_to\_convert\_docker.md > # How to Run MCP Server in Docker and Gemini Agent Locally > ## 6. Test the MCP Server > ### Using npx @modelcontextprotocol

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

# How to Run MCP Server in Docker and Gemini Agent Locally

This document explains how to set up and run the MCP server in a Docker container, while running the Gemini agent locally on your host machine. This approach keeps the server containerized for portability, while allowing the agent to run natively for easier interaction.

## Architecture Overview

This setup uses a **hybrid architecture**:

1. **Docker Container**: Runs the MCP server (`8_mcp_docker_server.py`) with HTTP transport on port 1923

2. **Local Host**: Runs the Gemini agent (`6_mcp_gemini_agent.py`) which connects to the Dockerized server

3. **Volume Mount**: The `./reports` directory is shared between container and host for file persistence

### Why Two Server Files?

- `4_mcp_server.py`: For development and testing locally with stdio transport (faster, simpler)

- `8_mcp_docker_server.py`: For production Docker deployment with HTTP transport (network-accessible, containerized)

This approach provides the best of both worlds: easy local development and robust containerized deployment.

## Quick Start Commands

### For Local Development:

```
```bash
# Test locally with stdio transport
fastmcp run 4_mcp_server.py:mcp
npx @modelcontextprotocol/inspector
```
```

Ln 190, Col 1

Spaces: 2

UTF-8

CRLF

{ } Markdown

🔗 Go Live

🔧 Prettier

🔔

# MCP'nin Kısıtlılıkları ve Zorlukları

## Dikkate Alınması Gerekenler

### 1. Başlangıç Karmaşıklığı:

- Geleneksel (monolitik) araç kullanımına göre, ayrı sunucu süreçleri çalıştırmayı gerektirdiği için daha karmaşık bir başlangıç ve yönetim süreci vardır. Özellikle basit uygulamalar için ek bir yük oluşturabilir.

### 2. Potansiyel Gecikme (Latency):

- Protokolün getirdiği ek soyutlama katmanı ve ağ iletişimi, potansiyel bir gecikme ek yüküne (overhead) neden olabilir. Düşük gecikme gerektiren gerçek zamanlı uygulamalar için bu bir endişe kaynağı olabilir.

# MCP'nin Kısıtlılıkları ve Zorlukları

## Dikkate Alınması Gerekenler

### 3. Yeni Güvenlik Riskleri:

- Harici kod çalıştırma yeteneği doğası gereği risklidir.
- Kötü niyetli sunucular aracılığıyla yetkisiz komut enjeksiyonu (command injection) veya "kafası karışmış vekil" (confused deputy) gibi yeni saldırı vektörleri ortaya çıkabilir.
- Protokol güvenlik prensipleri sunsa da, bunların doğru bir şekilde uygulanması geliştiricinin sorumluluğundadır.

# Sonuç ve Gelecek Perspektifi

## YZ Entegrasyonunda Yeni Bir Dönem

### Ana Çıkarımlar:

- BDM'lerin doğal kısıtlılıkları (halüsinasyon, güncellik eksikliği), onları dış dünyaya bağlama ihtiyacını doğurmuştur.
- Araç kullanımı bu ihtiyaca ilk yanıt olmuş, ancak standartlaşma eksikliği "N x M" problemini yaratmıştır.
- MCP, bu problemi çözmek için tasarlanmış, dağıtık mimariye sahip açık kaynaklı bir standarttır.
- Araç kullanımı "**ne yapılacağını**", MCP ise "**bunun nasıl yapılacağını**" tanımlayan, birbirini tamamlayan katmanlardır.

# Sonuç ve Gelecek Perspektifi

## YZ Entegrasyonunda Yeni Bir Dönem

### **MCP'nin Geleceęi:**

- Anthropic, OpenAI ve Google DeepMind gibi endüstri devlerinin desteęiyle YZ için evrensel bir standart olma potansiyeli taşımaktadır.
- Kullanım alanlarının, basit sohbet arayüzlerinin ötesine geçerek entegre geliştirme ortamları (IDE'ler), robotik ve ev otomasyonu gibi alanlara yayılması beklenmektedir.
- MCP, daha akıllı, otonom ve entegre YZ işğörenlerinin gelişimini hızlandıracaktır.

Sunum boyunca gösterdiğiniz ilgi ve sabır için teşekkür ederim.

Fikirlerinizi, yorumlarınızı ve sorularınızı dinlemek için sabırsızlanıyorum.



*Tesekkürler*

# İrtibat için



<https://www.muratkarakaya.net/>



[murat.karakaya@tedu.edu.tr](mailto:murat.karakaya@tedu.edu.tr)



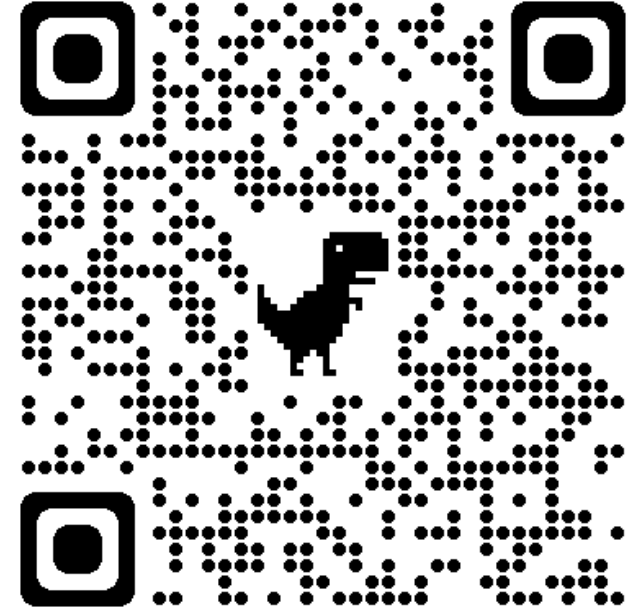
[kmkarakaya@gmail.com](mailto:kmkarakaya@gmail.com)



<https://www.youtube.com/c/muratkarakayaakademi>



<https://www.linkedin.com/in/muratkarakaya/>







# Soru Ve Yorumlar

 <https://www.muratkarakaya.net/>

 [murat.karakaya@tedu.edu.tr](mailto:murat.karakaya@tedu.edu.tr)

 [kmkarakaya@gmail.com](mailto:kmkarakaya@gmail.com)

 <https://www.youtube.com/c/muratkarakayaakademi>

 <https://www.linkedin.com/in/muratkarakaya/>

