



Yapay Zeka Ajanı

Prof.Dr.Hüseyin ÇAKIR
Gazi Üniversitesi

Yapay Zekâ Ajanları (AI Agents)

Yapay zekâ ajanları (AI Agents), belirli bir **hedefi** gerçekleştirmek için çevresini **algılayabilen, bilgi toplayabilen, karar verebilen, araç kullanabilen** ve gerektiğinde **kendi eylemlerini planlayarak** uygulayabilen akıllı yazılım sistemleridir.



Çevresini Algılar



Bilgi Toplar



Karar Verir



Araç Kullanır



Planlayıp Uygular

Yapay Zeka ve Uygulamaları - Copyright © 2026

Geleneksel Yapay Zekâ



"Bana literatür taraması nasıl yapılır, açıkla."

Genellikle kendisine verilen sorulara **cevap üretir.**

Yapay Zekâ Ajanları



"**Scopus** ve **Web of Science** verilerini incele, ilgili yayınları sınıflandır, bibliyometrik analizleri gerçekleştir ve akademik rapor oluştur."

Bir hedef doğrultusunda **birden fazla işlemi otonom** olarak gerçekleştirebilir.

Örnek:

"Scopus ve Web of Science verilerini incele, ilgili yayınları sınıflandır, bibliyometrik analizleri gerçekleştir ve akademik rapor oluştur."

→ Verileri İncele (Scopus & WoS) → Yayınları Sınıflandır → Analizleri Gerçekleştir → Rapor Oluştur → **Hedef Başarıyla Tamamlandı!**

Yapay Zeka ve Uygulamaları - Copyright © 2026



Yapay Zeka ve Uygulamaları - Copyright © 2026



Yapay Zeka ve Uygulamaları - Copyright © 2026

Yapay Zekâ Ajanı Türleri

Farklı ihtiyaçlara ve görev türlerine göre yapay zekâ ajanları çeşitli şekillerde sınıflandırılabilir.

Yapay Zeka ve Uygulamaları

1 Basit Refleks Ajanı



Önceden tanımlanmış kurallara göre hareket eder.

2 Model Tabanlı Ajan



Çevresi hakkında tuttuğu bilgiyi kullanır.

3 Hedef Tabanlı Ajan



Belirli bir hedefe ulaşmaya çalışır.

4 Fayda Tabanlı Ajan



Alternatifler arasından en yararlı olanı seçer.

5 Öğrenen Ajan



Deneyimlerden ve geri bildirimlerden öğrenir.

6 Otonom Ajan



İnsan müdahalesi olmadan çok aşamalı görevler gerçekleştirebilir.

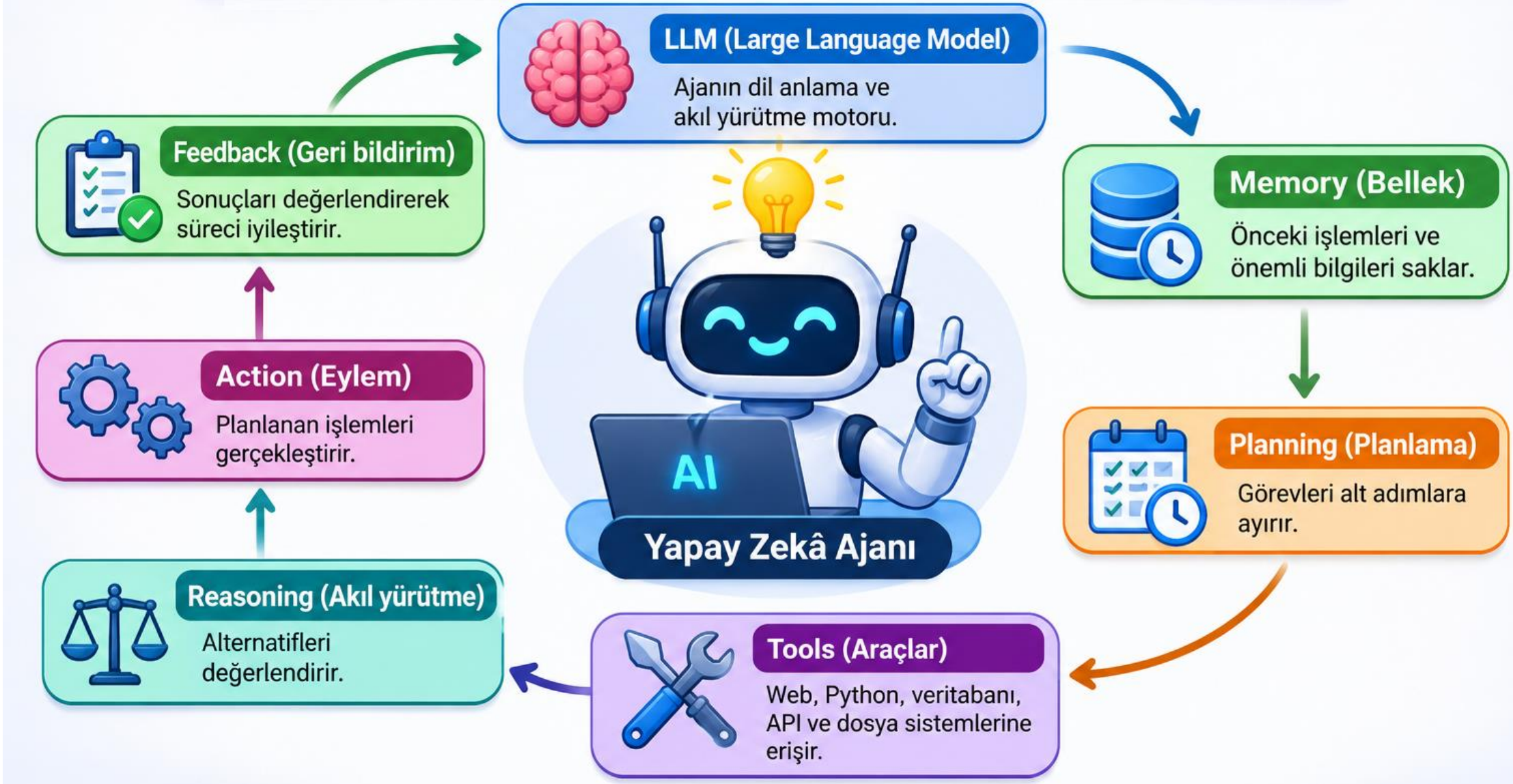
7 Çoklu Ajan Sistemi



Birden fazla ajanın iş birliğiyle çalışmasını sağlar.

Yapay zekâ ajanlarının temel bileşenleri

Yapay Zeka ve Uygulamaları - Copyright © 2026

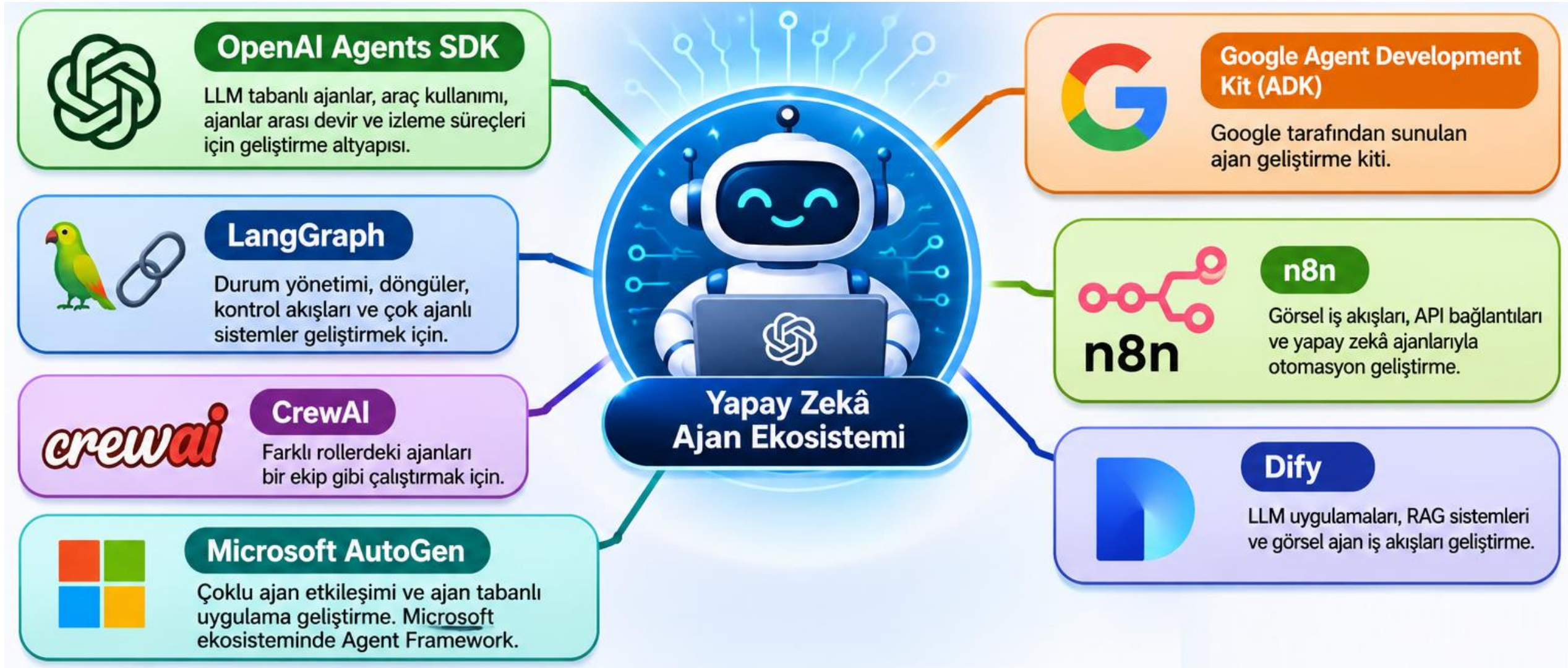


Yapay Zeka ve Uygulamaları - Copyright © 2026

Önemli Yapay Zekâ Ajan Platformları ve Araçları

Ajan geliştirme, yönetme veya kullanma amacıyla öne çıkan araçlar şunlardır.

Yapay Zeka ve Uygulamaları - Copyright © 2026



Yapay Zeka ve Uygulamaları - Copyright © 2026

Bu Araçlarla Neler Yapılabilir?



Ajan Geliştirme



İş Akışı Otomasyonu



Çoklu Ajan Yönetimi



Harici Araç ve API Entegrasyonu



Akıllı Uygulamalar Geliştirme

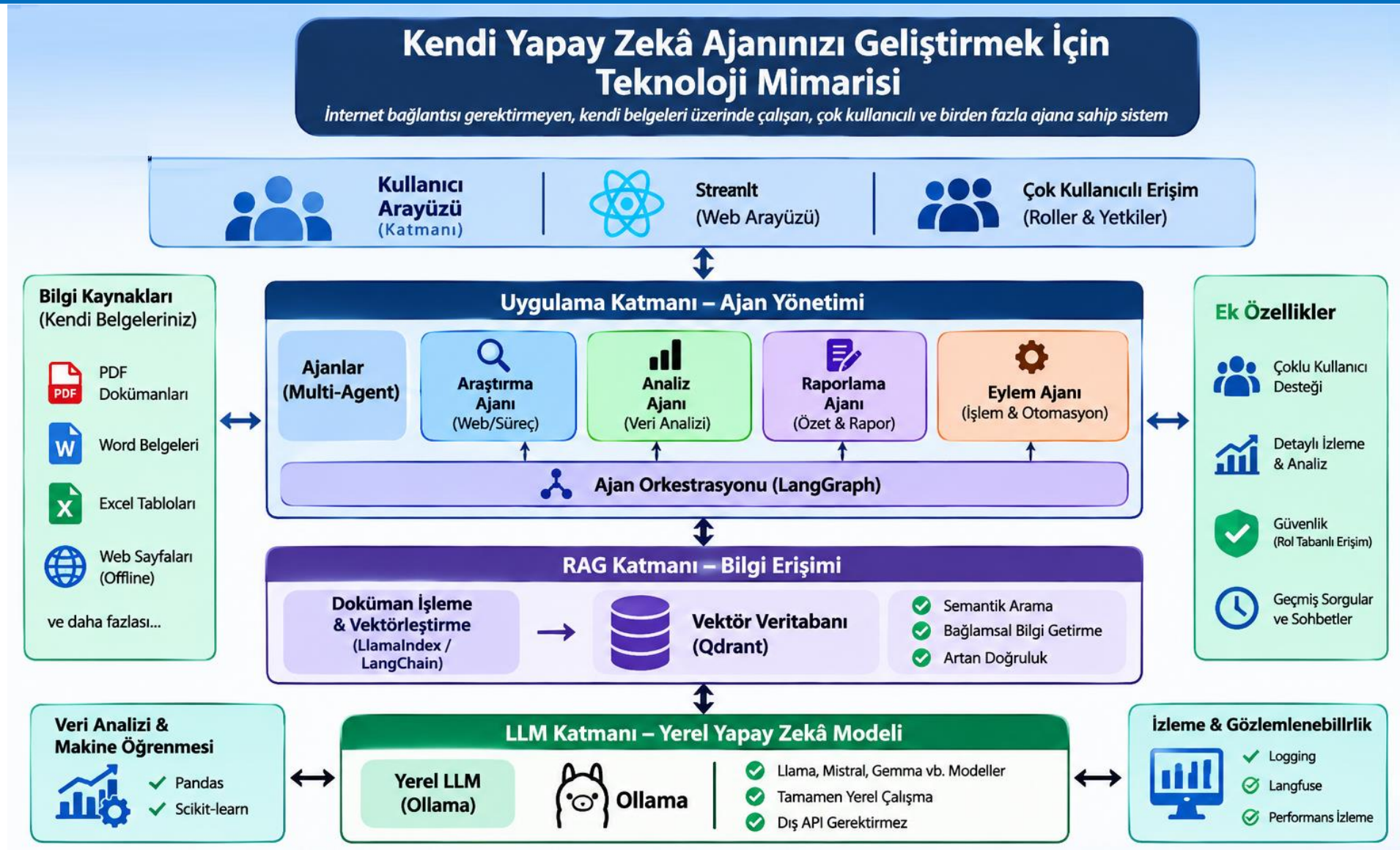
Çoklu Ajan Sistemleri (Multi-Agent Systems)

Çoklu ajan sistemlerinde her ajan belirli bir uzmanlık veya sorumluluk üstlenir.



Yapay Zeka ve Uygulamaları - Copyright © 2026


Akademik Araştırma Süreci Örneği
Araştırma Konusunun Belirlenmesi
Literatür Taraması
Veri Analizi
Bibliyometrik Analiz
Raporlama ve Yazım
Kontrol ve Doğrulama
Sonuçların Sunulması



Yapay Zeka ve Uygulamaları - Copyright © 2026

Bir Yapay Zekâ Ajanının Çalışma Örneği

Senaryo: Bir öğrenci için kişiselleştirilmiş sınav hazırlık programı oluşturulması.

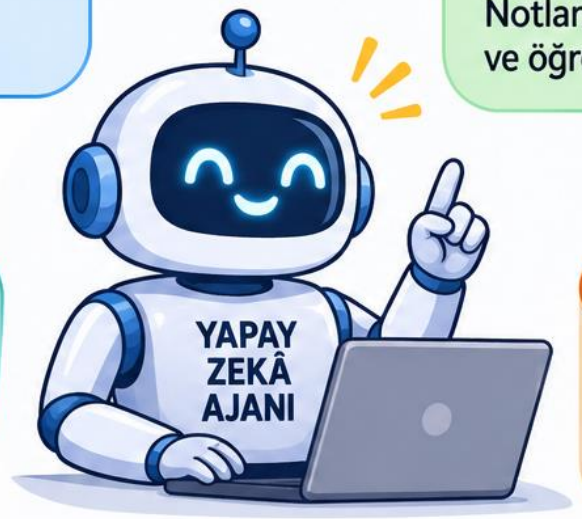
1. Hedef Belirleme

Öğrenciyi sınava hazırlamak

2. Veri Toplama

Notlar, sınav sonuçları ve öğrenme geçmişi





3. Analiz ve Karar Verme

Eksik konuları ve öğrenme seviyesini belirleme



4. Eylem Gerçekleştirme

Kişiselleştirilmiş ders ve soru planı oluşturma



5. Sonuçları Değerlendirme

Başarıyı ölçme ve öğrenme planını güncelleme



Doğru yanıt oranı: %92

Yapay Zeka ve Uygulamaları - Copyright © 2026



Yapay Zekâ Ajanı Geliştirmek İçin Kullanılabilecek Teknolojiler

Teknoloji	Kullanım Amacı
Python	Ajan geliştirme
LangChain	LLM ve araç entegrasyonu
LangGraph	Ajan iş akışları ve durum yönetimi
CrewAI	Çoklu ajan koordiinasyonu
Ollama	Yerel LLM çalıştırma
Qdrant	Vektör veritabanı
FastAPI	API ve servis geliştirme
MCP	Model ve dış araç entegrasyonu

Yapay Zeka ve Uygulamaları - Copyright © 2026

Akıllı Üniversite Asistanı

Gerçek Bir Yapay Zekâ Ajanı Örneği

Üniversite öğrencilerinin akademik, idari ve öğrenme süreçlerinde kişisel bir yapay zekâ asistanı.
Sadece sohbet değil, planlama, eylem ve çözüm sunar.

Kullanıcı



Öğrenci

"Bu hafta hangi derslerim var? Sınav için nasıl çalışmalıyım?"

Ajan Mimarisi

Kullanıcı Sorgusu → Anlama (LLM + RAG) → Planlama ve Karar → Eylem (Araçlar) → Sonuç ve Öneri

Bilgi Kaynakları:
- Ders İçerikleri
- Akademik Takvim
- Sınav Bilgileri
- Üniversite Sistemleri (API)
- Güncel Web Kaynakları

Kullanıcı Arayüzü



Somut Yetenekler

- Akademik bilgiye anında erişim (RAG)
- Kişisel öğrenme planı oluşturma
- Sınav hatırlatıcıları gönderme
- Üniversite sistemleriyle işlem yapma (ders kaydı, not sorgulama)
- Tekrar ve test önerme

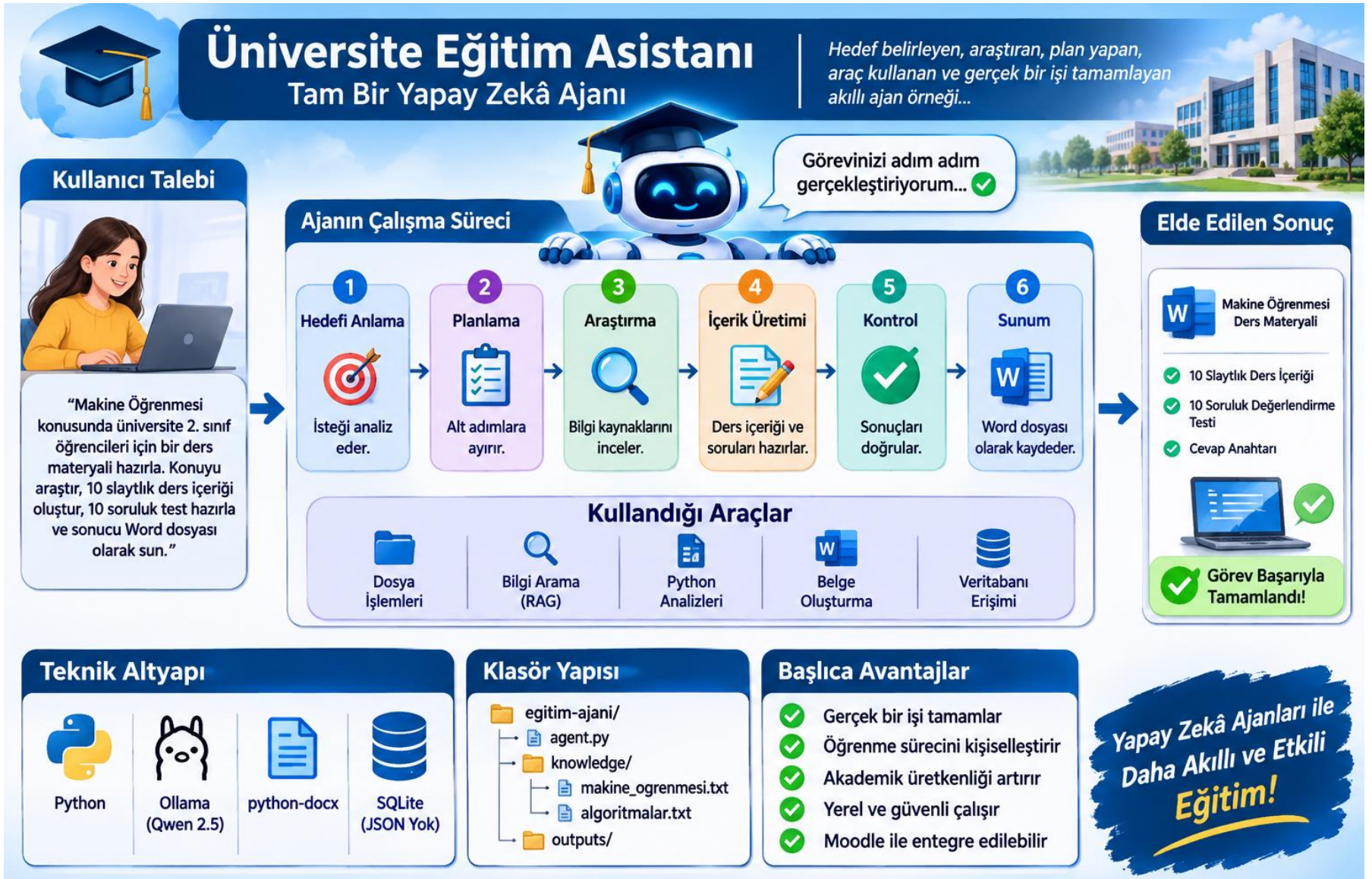
Örnek Eylem Akışı

1. Soru Al (Kullanıcı)
2. Bilgi Ara (RAG + API)
3. Plan Oluştur (Ajan Kararı)
4. Hatırlatma Yap (Eylem)
5. Sonucu Sun (Kullanıcıya)

Sağladığı Faydalar

- Zaman Tasarrufu
- Daha Etkili Öğrenme
- Kişiselleştirilmiş Destek
- Daha Başarılı Öğrenci

Yapay Zeka ve Uygulamaları - Copyright © 2026


Üniversite Eğitim Asistanı
Tam Bir Yapay Zekâ Ajanı
Hedef belirleyen, araştıran, plan yapan, araç kullanan ve gerçek bir işi tamamlayan akıllı ajan örneği...
Kullanıcı Talebi
"Makine Öğrenmesi konusunda üniversite 2. sınıf öğrencileri için bir ders materyali hazırla. Konuyu araştır, 10 slaytlık ders içeriği oluştur, 10 soruluk test hazırla ve sonucu Word dosyası olarak sun."
Görevinizi adım adım gerçekleştiriyorum...
Ajanın Çalışma Süreci
1 Hedefi Anlama
İsteği analiz eder.
2 Planlama
Alt adımlara ayırır.
3 Araştırma
Bilgi kaynaklarını inceler.
4 İçerik Üretimi
Ders içeriği ve soruları hazırlar.
5 Kontrol
Sonuçları doğrular.
6 Sunum
Word dosyası olarak kaydeder.
Kullandığı Araçlar
Dosya İşlemleri
Bilgi Arama (RAG)
Python Analizleri
Belge Oluşturma
Veritabanı Erişimi
Elde Edilen Sonuç
Makine Öğrenmesi Ders Materyali
10 Slaytlık Ders İçeriği
10 Soruluk Değerlendirme Testi
Cevap Anahtarı
Görev Başarıyla Tamamlandı!
Teknik Altyapı
Python
Ollama (Qwen 2.5)
python-docx
SQLite (JSON Yok)
Klasör Yapısı
egitim-ajani/
agent.py
knowledge/
makine_ogrenmesi.txt
algoritmalar.txt
outputs/
Başlıca Avantajlar
Gerçek bir işi tamamlar
Öğrenme sürecini kişiselleştirir
Akademik üretkenliği artırır
Yerel ve güvenli çalışır
Moodle ile entegre edilebilir
Yapay Zekâ Ajanları ile Daha Akıllı ve Etkili Eğitim!



Yapay Zeka ve Uygulamaları - Copyright © 2026

Claude | Ajan Ekosistemi

Daha akıllı, daha üretken, daha fazla iş birliği!

Claude; kod yazan, birlikte çalışan ve özel ajanlar oluşturan güçlü bir yapay zekâ platformudur.



Claude Code Ajanı

Geliştiriciler için güçlü kod asistanı

- Terminalde doğal dil ile kod yazma
- Kod tabanını anlama ve düzenleme
- Test, hata düzeltme ve uygulama geliştirme

Örnek Komutlar:

```
/init       # Projeyi başlat
/plan       # Plan oluştur
/implement  # Uygulamayı geliştir
/test       # Testleri çalıştır
```

Claude Cowork Ajanı

Ortak çalışma alanı, birlikte üretim

- Projeler üzerinde ekip çalışması
- Plan yapma ve görev dağılımı
- Dosya paylaşımı ve fikir üretimi

Örnek Komutlar:

```
@ekip     Proje ekibi oluştur
/plan     Yol haritası hazırla
/assign   Görev ata
/share    Dosya paylaş
```

Claude Agent SDK

Kendi özel ajanlarınızı geliştirin

- Özel ajanlar oluşturma
- MCP entegrasyonu
- Araç ve API erişimi

Örnek Komutlar:

```
create agent  analizci
add tool      (mcp)
run agent     veri-analizi
```

Claude Ajan Komutları

Slash Komutları

```
/help    : Komutları gösterir
/memory  : Belleği yönetir
/mcp     : MCP araçlarını yönetir
```

Terminal Komutları (Claude Code)

```
claude              : Claude Code başlatır
claude --add-dir    : Projeye klasör ekle
claude --resume     : Önceki oturuma dön
```

Ajan Oluşturma

```
/create-agent       : Yeni ajan oluştur
```

Ajan Nasıl Oluşturulur?

1. /create-agent komutunu kullanın
2. Ajanın adını ve amacını belirtin
3. İhtiyaç duyduğu araçları seçin (dosya, web, MCP vb.)
4. Gerekirse özel talimatlar ekleyin
5. Ajanı kaydedin ve kullanın

Örnek:

```
/create-agent veri-analizci
```

Bu ajan, CSV dosyalarını analiz eder, görselleştirir ve raporlar hazırlar.
Gerekli araçlar:
Python, Matplotlib, Pandas.

Artık kendi ajanın hazır!

Örnek Ajan: Veri Analizci

Neler Yapabilir?

- Veri analizi ve görselleştirme
- Rapor oluşturma
- İstatistiksel analiz

Nasıl Kullanılır?

"Son 3 aylık satış verilerini analiz et ve grafikle göster."

Diğer Özellikler

- MCP Entegrasyonu
- Araç Kullanımı (Web, API, Dosya)
- Bellek ve Bağlam Yönetimi
- Güvenli ve Kontrollü Erişim

Kullanım Senaryoları

- Yazılım Geliştirme
- Veri Analizi ve Raporlama
- Eğitim ve Araştırma
- Kurumsal Verimlilik

Claude Ajanları ile

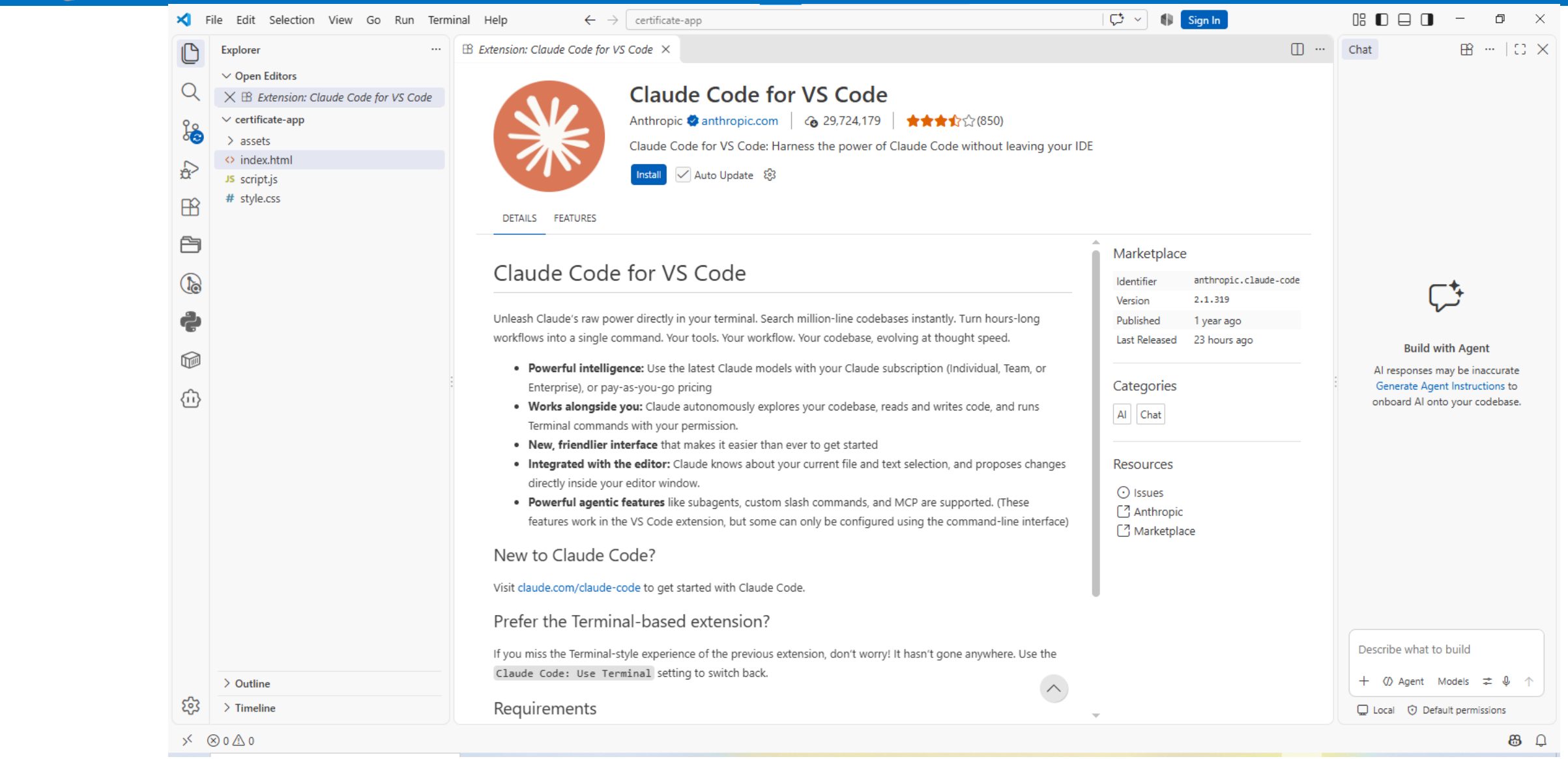
Fikirlerinizi Gerçeğe Dönüştürün!

Daha üretken bir çalışma deneyimi sizi bekliyor.

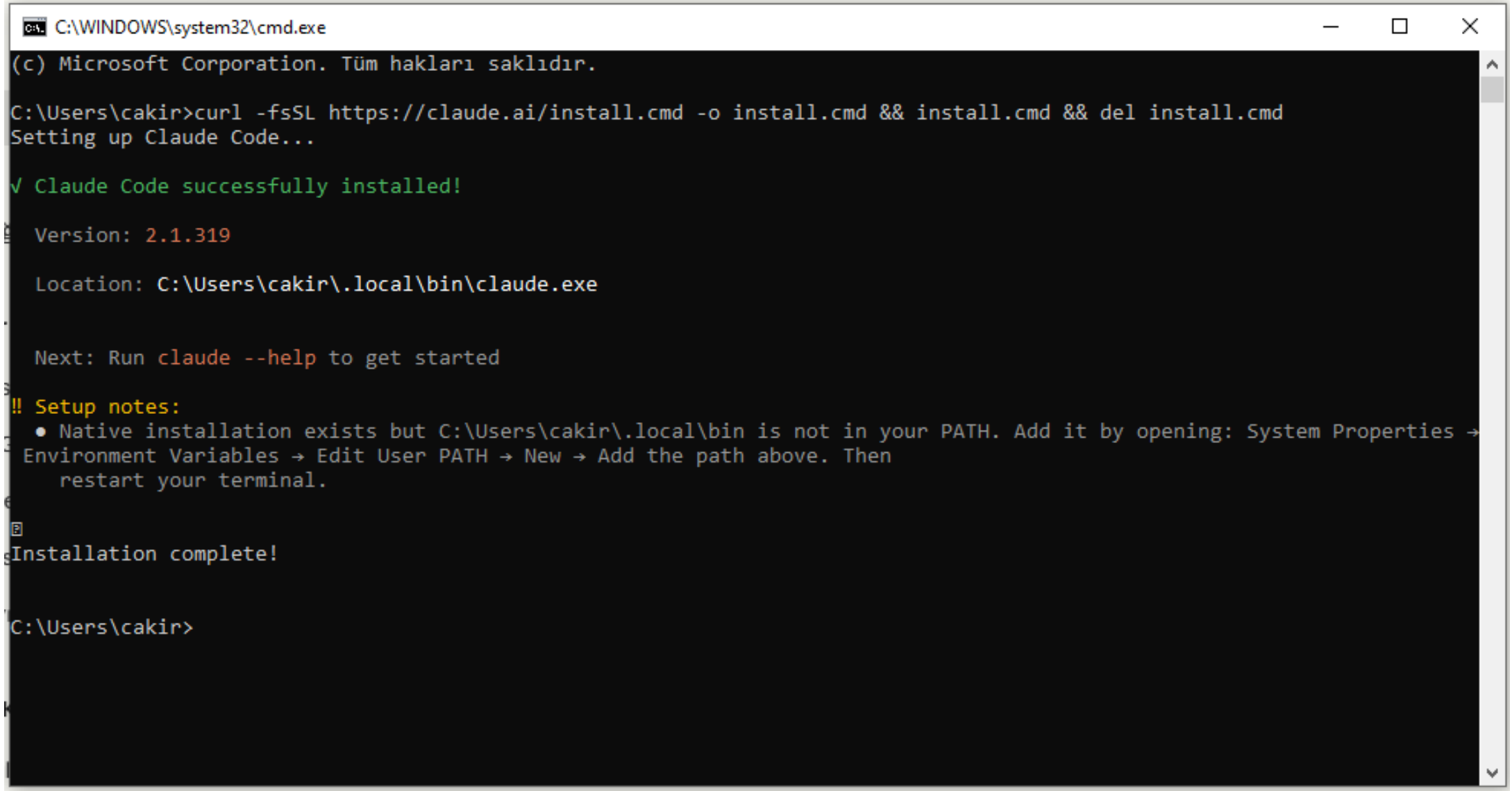
Windows CMD:

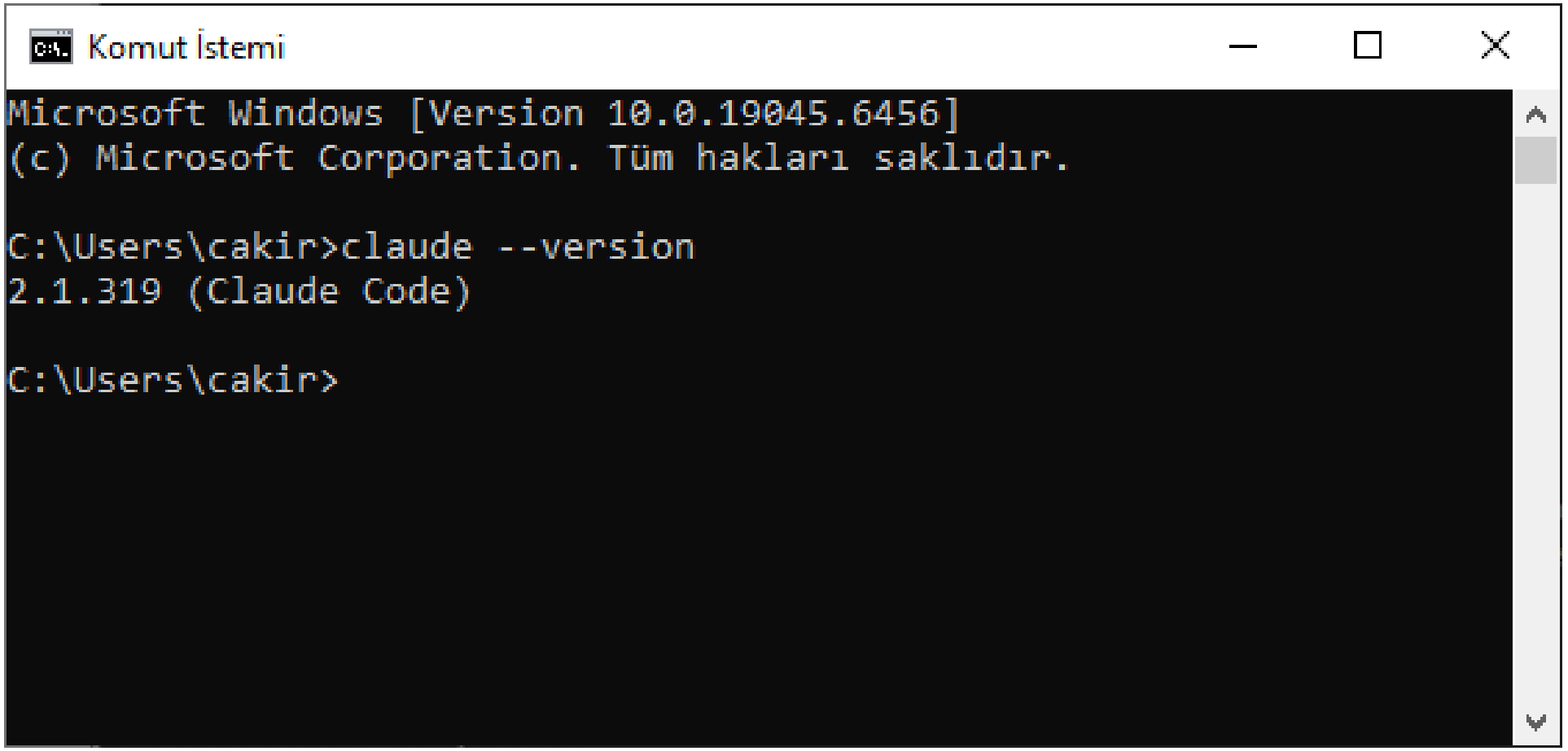
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del install.cmd

Kurulum komutu, Claude Code indirilirken herhangi bir ilerleme göstermiyor. Kurulum tamamlandığında, yeni bir terminal penceresi açın ve komutunu çalıştırın claude --version. Başarılı bir kurulum, sürüm numarasını yazdırır. Kabuğunuz " claudebulunamadı" veya "tanınmadı" diyorsa, kurulum dizini henüz PATH'inizde değil demektir: bkz. **PATH'inizi düzeltin** .Eğer işaretini görürseniz The token '&&' is not a valid statement separator, CMD'de değil, PowerShell'desiniz demektir. Eğer işaretini görürseniz 'irm' is not recognized as an internal or external command, PowerShell'de değil, CMD'desiniz demektir.Yükleme komutu syntax error near unexpected token '<', 403veya başka bir hatayla başarısız olursa, hatayı gidermek ve alternatif yükleme yöntemleri için **Yükleme Sorun Giderme bölümüne bakın.Windows işletim sisteminde, Claude Code'un Bash aracını kullanabilmesi için Git for** Windows'un yüklü olması önerilir. Git for Windows yüklü değilse, Claude Code bunun yerine PowerShell'i kabuk aracı olarak kullanır. WSL kurulumlarında Git for Windows'a gerek yoktur.



Yapay Zeka ve Uygulamaları - Copyright © 2026


C:\WINDOWS\system32\cmd.exe
(c) Microsoft Corporation. Tüm hakları saklıdır.
C:\Users\cakir>curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del install.cmd
Setting up Claude Code...
√ Claude Code successfully installed!
Version: 2.1.319
Location: C:\Users\cakir\.local\bin\claude.exe
Next: Run claude --help to get started
!! Setup notes:
• Native installation exists but C:\Users\cakir\.local\bin is not in your PATH. Add it by opening: System Properties → Environment Variables → Edit User PATH → New → Add the path above. Then restart your terminal.
Installation complete!
C:\Users\cakir>


Komut İstemi
Microsoft Windows [Version 10.0.19045.6456]
(c) Microsoft Corporation. Tüm hakları saklıdır.
C:\Users\cakir>claude --version
2.1.319 (Claude Code)
C:\Users\cakir>

Agent SDK'ya genel bakış

Claude Code'u kütüphane olarak kullanarak üretim amaçlı yapay zeka ajanları oluşturun. Bir ajan, kendi adımlarını planlayarak ve dosya okuyan, komut çalıştıran veya kod düzenleyen araçları çağırarak bir görevi tamamlayan bir uygulamadır. Ajan SDK'sı, Claude Code'u destekleyen aynı araçları, **ajan döngüsünü ve bağlam yönetimini Python ve TypeScript ile programlanabilir şekilde sunar.**

Agent SDK'yı diğer Claude araçlarıyla karşılaştırın.

Agent SDK, CLI, Client SDK ve Managed Agents, aracı kimin çalıştırdığı, hangi özelliklerin yerleşik olarak geldiği ve bunlara nasıl erişildiği konusunda farklılık gösterir. Kendi aracınızı nasıl oluşturmak ve çalıştırmak istediğinize uygun satırı bulun.

İstediğiniz	Kullanmak	Ne elde edersiniz?
Claude Code ajanını kendi Python veya TypeScript uygulamanıza, işlettiğiniz bir sürece entegre edin.	Ajan SDK'sı	Claude Code ikili dosyasını çalıştıran ve Claude Code'un yerleşik araçları, izinleri, oturumları ve kancaları gibi özelliklerini içeren bir kütüphane.
Terminalden etkileşimli geliştirme yapın veya tek seferlik görevler çalıştırın.	Claude Code CLI	Günlük etkileşimli kullanım için tasarlanmış terminal arayüzü.
Claude API'sini doğrudan kendi kodunuzdan çağırın.	İstemci SDK'sı	İstemci SDK dillerinin herhangi birinden Claude API'sine doğrudan erişim. Araç döngüsünü kendiniz yazabilir veya istemci SDK'sının beta araç çalıştırıcısının bunu yönetmesine izin verebilirsiniz.
Anthropic'in, Claude API'si aracılığıyla yapılandırılmış olan aracıyı barındırmasını sağlayın.	Yönetilen Acenteler	Anthropic tarafından yönetilen bir bulut sanal ortamında veya kendi altyapınızda barındırdığınız bir sanal ortamda oturumlar çalıştıran, ajan döngüsünü yürüten, barındırılan bir ajan aracı . Bunu, kullandığınız dilin SDK'sından , antCLI'dan veya REST API'dan kullanabilirsiniz.

Aynı ajan döngüsünü Python veya TypeScript dışında bir dilden çalıştırmak için, CLI'yi-p ve bayraklarını kullanarak bir alt işlem olarak çalıştırın --output-format json.

Yetenekler

SDK'da Claude Code'un şu özellikleri mevcuttur:

Yetenek	Ne işe yarar?	Daha fazla bilgi edin
Dahili araçlar	Dosyaları okuyun, yazın, düzenleyin, komutları çalıştırın ve internette arama yapın.	Araçlar referansı
Kancalar	Ajan yaşam döngüsünün kilit noktalarında özel kod çalıştırın.	Kancalar
Alt acenteler	Belirli alt görevler için uzmanlaşmış ajanlar oluşturun.	Alt acenteler
MCP	Model Bağlam Protokolü aracılığıyla harici araçları ve veri kaynaklarını bağlayın.	MCP
İzinler	Hangi araçların otomatik olarak çalışacağını, hangilerinin onay gerektirdiğini kontrol edin.	İzinler
Oturumlar	Paylaşımlar arasında bağlamı koruyun, daha sonra devam edin veya çatallayın.	Oturumlar
Beceriler, komutlar ve hafıza	Projenizden .claude/ve diğer kaynaklardan otomatik olarak yükleyin ~/.claude/, tıpkı Claude Code'da olduğu gibi.	Beceriler , Komutlar , Bellek , Yapılandırma yükleme
Eklentiler	Becerileri, aracıları, kancaları ve MCP sunucularını paketleyin ve bunları yerel yoldan yükleyin.	Eklentiler

Hızlı Başlangıç

Otonom olarak çalışan yapay zeka ajanları oluşturmak için Python veya TypeScript Agent SDK'sını kullanmaya başlayın.

Agent SDK'yı kullanarak, kodunuzu okuyan, hataları bulan ve manuel müdahale olmadan düzelten bir yapay zeka ajanı oluşturabilirsiniz.

Yapacaklarınız:

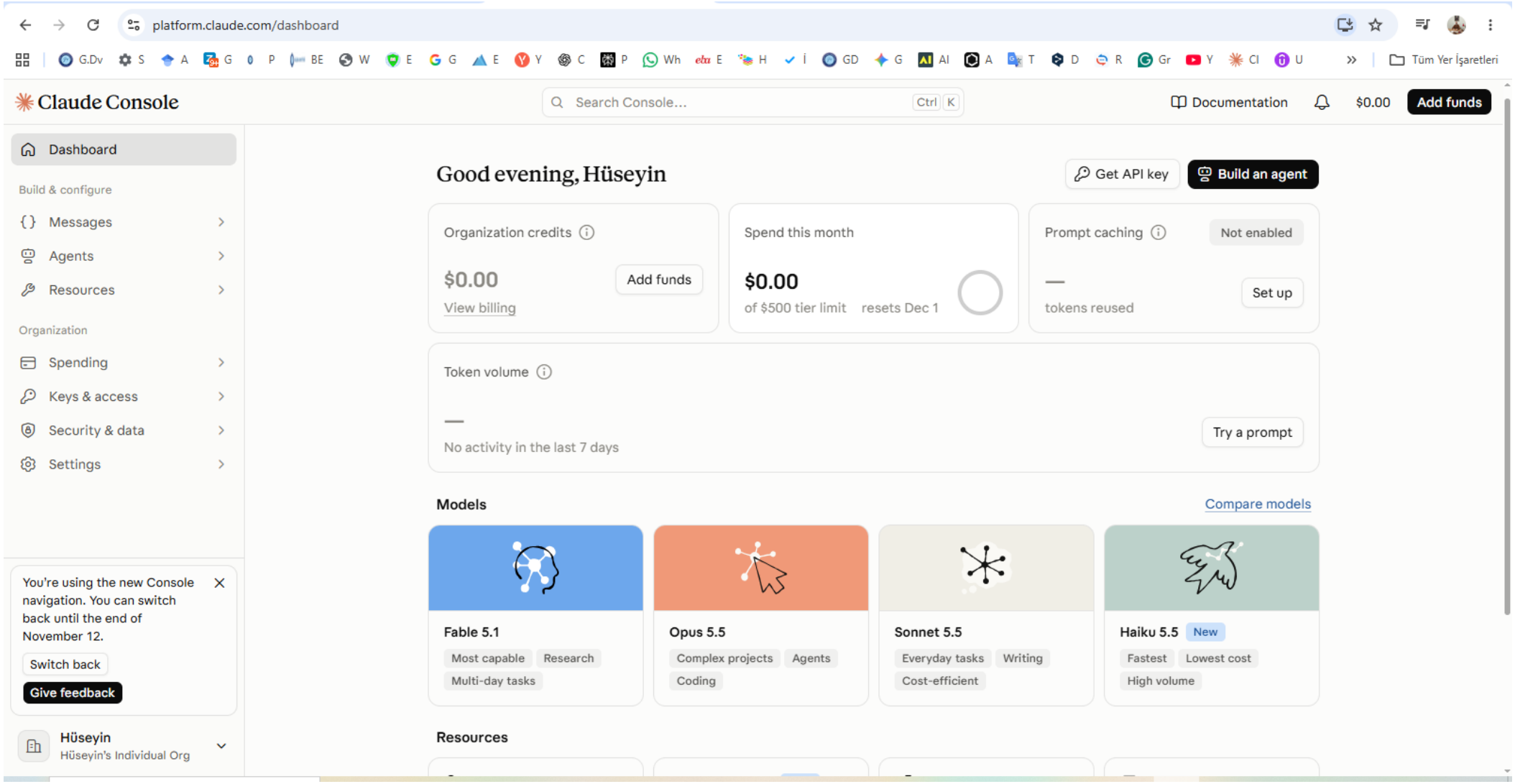
1.Agent SDK ile bir proje oluşturun.
2.İçinde hatalı kod bulunan bir dosya oluşturun.
3.Hataları otomatik olarak bulup düzelten bir aracı çalıştırın.

Yapay Zeka ve Uygulamaları - Copyright © 2026

Önkoşullar

- **Node.js 18.14+** veya **Python 3.10+**
- Bir **Anthropic hesabı** . Eğer hesabınız yoksa, buradan kaydolun .

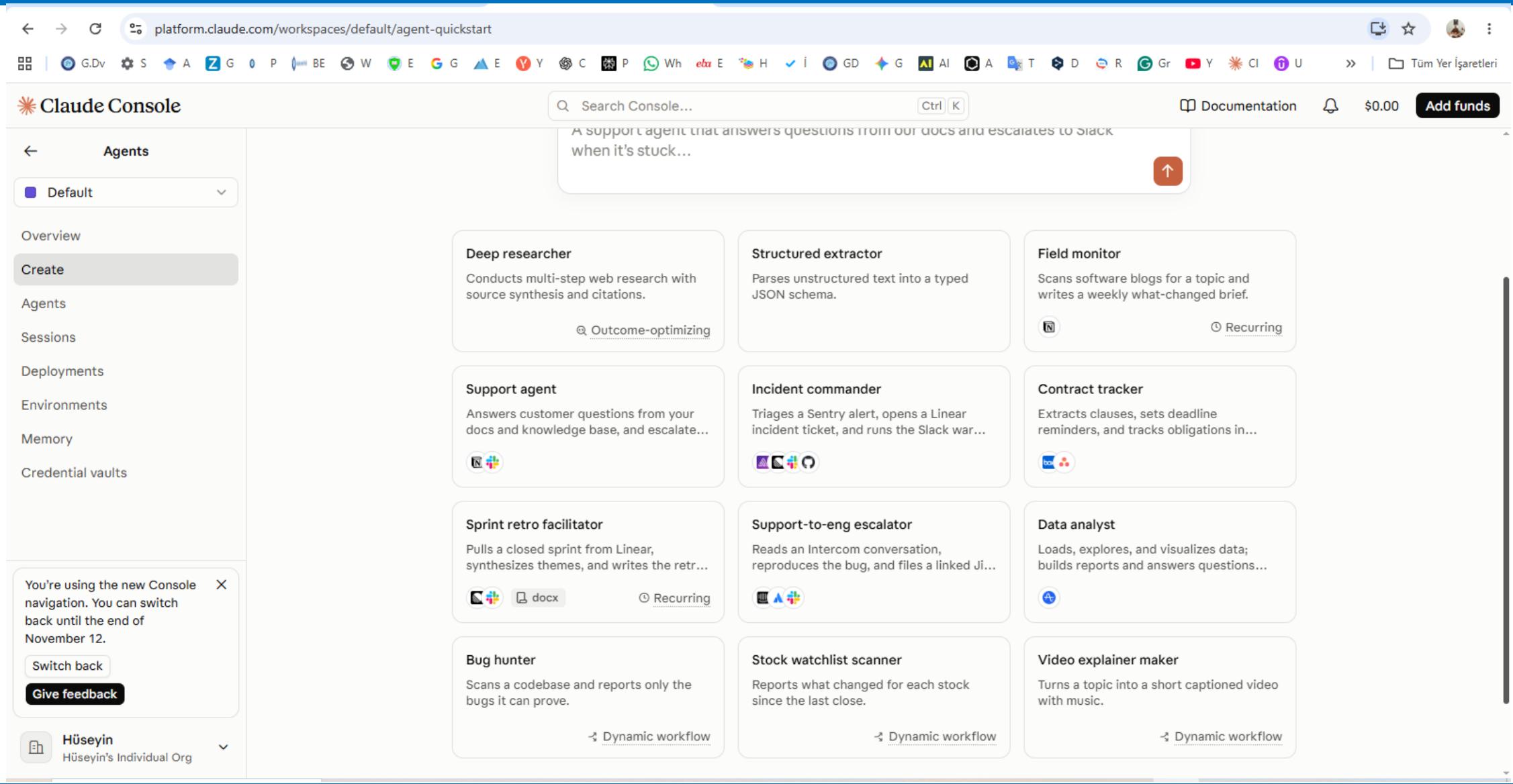
https://platform.claude.com/

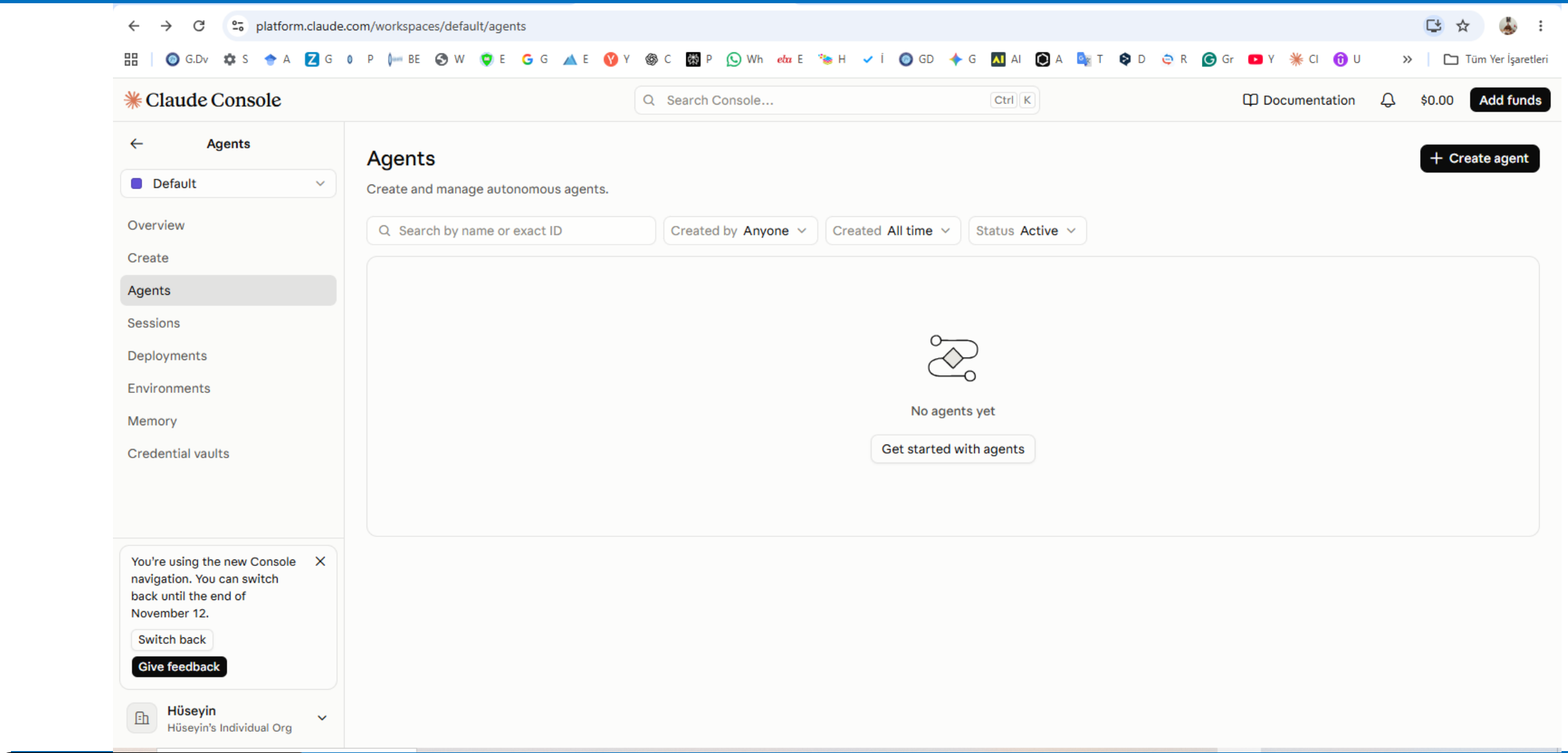


Yapay Zeka ve Uygulamaları - Copyright © 2026






platform.claude.com/workspaces/default/agent-quickstart
Claude Console
Search Console...
Ctrl K
Documentation
$0.00
Add funds
Agents
Default
Overview
Create
Agents
Sessions
Deployments
Environments
Memory
Credential vaults
when it's stuck...
Deep researcher
Conducts multi-step web research with source synthesis and citations.
Outcome-optimizing
Structured extractor
Parses unstructured text into a typed JSON schema.
Field monitor
Scans software blogs for a topic and writes a weekly what-changed brief.
Recurring
Support agent
Answers customer questions from your docs and knowledge base, and escalate...
Incident commander
Triages a Sentry alert, opens a Linear incident ticket, and runs the Slack war...
Contract tracker
Extracts clauses, sets deadline reminders, and tracks obligations in...
Sprint retro facilitator
Pulls a closed sprint from Linear, synthesizes themes, and writes the retr...
docx
Recurring
Support-to-eng escalator
Reads an Intercom conversation, reproduces the bug, and files a linked Ji...
Data analyst
Loads, explores, and visualizes data; builds reports and answers questions...
Bug hunter
Scans a codebase and reports only the bugs it can prove.
Dynamic workflow
Stock watchlist scanner
Reports what changed for each stock since the last close.
Dynamic workflow
Video explainer maker
Turns a topic into a short captioned video with music.
Dynamic workflow
You're using the new Console navigation. You can switch back until the end of November 12.
Switch back
Give feedback
Hüseyin
Hüseyin's Individual Org



Yapay Zeka ve Uygulamaları - Copyright © 2026

Kurmak

1. Bir proje klasörü oluşturun.

Bu hızlı başlangıç kılavuzu için yeni bir dizin oluşturun:

mkdir my-agent cd my-agent

Kendi projeleriniz için SDK'yı herhangi bir klasörden çalıştırabilirsiniz; varsayılan olarak o dizindeki ve alt dizinlerindeki dosyalara erişimi olacaktır.

2 (yeni proje)

SDK'yı yükleyin Python (pip)

Sanal bir ortam oluşturun ve etkinleştirin, ardından paketi yükleyin.

Windows'ta:

py -m venv .venv
.venv\Scripts\Activate.ps1
pip install claude-agent-sdk

py -m venv .venv .venv\Scripts\Activate.ps1 pip install claude-agent-sdk

Eğer PowerShell Activate.ps1yürütme ilkesi hatasıyla bloke olursa, *Set-ExecutionPolicy -Scope Process RemoteSigned* önce şunu çalıştırın.

Yapay Zeka ve Uygulamaları - Copyright © 2026

3. API anahtarınızı ayarlayın.

Claude Konsolundan bir API anahtarı alın , ardından bu anahtarı ajanınızı çalıştıracağınız kabukta bir ortam değişkeni olarak ayarlayın:

macOS / Linux

Windows (PowerShell)

```
export ANTHROPIC_API_KEY=your-api-key
```

SDK, anahtarı aracınızı çalıştıran işlemin ortamından okur; dosyaları otomatik olarak yüklemez .env. Anahtarı bir dosyada saklıyorsanız , SDK'yı çağırmadan önce .env, örneğin paket aracılığıyla, kendiniz yükleyin .dotenv

SDK ayrıca üçüncü taraf API sağlayıcıları aracılığıyla kimlik doğrulamayı da destekler:

Amazon Bedrock : CLAUDE_CODE_USE_BEDROCK=1Ortam değişkenini ayarlayın ve AWS kimlik bilgilerini yapılandırın.

AWS üzerinde Claude Platformu : Ayarları yapın CLAUDE_CODE_USE_ANTHROPIC_AWS=1ve ANTHROPIC_AWS_WORKSPACE_IDardından AWS kimlik bilgilerini yapılandırın.

Google Cloud'un Aracı Platformu : CLAUDE_CODE_USE_VERTEX=1Ortam değişkenini ayarlayın ve Google Cloud kimlik bilgilerini yapılandırın.

Microsoft Foundry : CLAUDE_CODE_USE_FOUNDRY=1Ortam değişkenini ayarlayın ve Azure kimlik bilgilerini yapılandırın.

Hatalı bir dosya oluşturun

Bu hızlı başlangıç kılavuzu, kodda hataları bulup düzeltebilen bir aracı oluşturmayı adım adım anlatmaktadır. Öncelikle, aracının düzeltmesi için kasıtlı olarak bazı hatalar içeren bir dosyaya ihtiyacınız var. Bu dosyayı **utils.py** dizinde oluşturun **my-agent** ve aşağıdaki kodu yapıştırın:

```
def calculate_average(numbers):
    total = 0
    for num in numbers:
        total += num
    return total / len(numbers)
def get_user_name(user):
    return user["name"].upper()
```

Bu kodda iki hata var:

calculate_average([]) sıfıra bölme ile ilgili çökmeler

get_user_name(None)TypeError hatasıyla çöküyor

Hataları bulan ve düzelten bir ajan oluşturun.

agent.py Python SDK kullanıyorsanız veya **agent.ts** TypeScript kullanıyorsanız oluşturun. **agent.mts** Mevcut projeniz CommonJS kullanıyorsa bunun yerine şunu kullanın:

```
import asyncio
from claude_agent_sdk import query, ClaudeAgentOptions, AssistantMessage, ResultMessage
async def main():
    # Agentic loop: streams messages as Claude works
    async for message in query(
        prompt="Review utils.py for bugs that would cause crashes. Fix any issues you find.",
        options=ClaudeAgentOptions(
            allowed_tools=["Read", "Edit", "Glob"],  # Auto-approve these tools
            permission_mode="acceptEdits",  # Auto-approve file edits
        ),
    ):
        # Print human-readable output
        if isinstance(message, AssistantMessage):
            for block in message.content:
                if hasattr(block, "text"):
                    print(block.text)  # Claude's reasoning
                elif hasattr(block, "name"):
                    print(f"Tool: {block.name}")  # Tool being called
        elif isinstance(message, ResultMessage):
            print(f"Done: {message.subtype}")  # Final result
asyncio.run(main())
```

Temsilcinizi çalıştırın

Ajanınız hazır. Aşağıdaki komutla çalıştırın:

Python (pip)

Sanal ortamınız hala etkin durumdayken:

python agent.py

İşlem sırasında, ajan gerekçelerini ve çağırdığı her aracı yazdırır ve sonunda **.** ile bitirir **Done: success.** Çalıştırdıktan sonra kontrol edin **utils.py** Boş listeleri ve boş kullanıcıları ele alan savunma kodunu göreceksiniz. Ajanınız otomatik olarak şunları yapar:

utils.pyKodu anlamak için okuyun.

Mantığı analiz ettim ve çökmeye neden olabilecek uç durumları belirledim.

Hata ayıklama işlemlerini doğru şekilde eklemek için dosya düzenlendi.

Agent SDK'yı farklı kılan da bu: Claude, sizden araçları uygulamanızı istemek yerine, araçları doğrudan çalıştırıyor.