

Third party tools - Agent Development Kit

Source URL: <https://google.github.io/adk-docs/tools/third-party-tools/>

Third Party Tools

Currently supported in **Python**

ADK is designed to be **highly extensible, allowing you to seamlessly integrate tools from other AI Agent frameworks** like CrewAI and LangChain. This interoperability is crucial because it allows for faster development time and allows you to reuse existing tools.

1. Using LangChain Tools

ADK provides the `LangchainTool` wrapper to integrate tools from the LangChain ecosystem into your agents.

Example: Web Search using LangChain's Tavily tool

[Tavily](#) provides a search API that returns answers derived from real-time search results, intended for use by applications like AI agents.

1. Follow [ADK installation and setup](#) guide.
2. **Install Dependencies:** Ensure you have the necessary LangChain packages installed. For example, to use the Tavily search tool, install its specific dependencies:

```
''' pip install langchain_community tavily-python
```

```
''' 3. Obtain a Tavily API KEY and export it as an environment variable.
```

```
''' export TAVILY_API_KEY=
```

```
` ` 4. **Import:** Import the LangchainTool wrapper from ADK  
and the specific LangChain tool you wish to use  
(e.g, TavilySearchResults).
```

```

''' from google.adk.tools.langchain_tool import LangchainTool from
langchain_community.tools import TavilySearchResults

`` 5. **Instantiate & Wrap:** Create an instance of your
LangChain tool and pass it to the `LangchainTool` constructor.

''' # Instantiate the LangChain tool tavily_tool_instance =
TavilySearchResults( max_results=5, search_depth="advanced",
include_answer=True, include_raw_content=True, include_images=True, )

# Wrap it with LangchainTool for ADK adk_tavily_tool =
LangchainTool(tool=tavily_tool_instance)

`` 6. **Add to Agent:** Include the
wrapped `LangchainTool` instance in your agent's `tools` list during
definition.

''' from google.adk import Agent

# Define the ADK agent, including the wrapped tool my_agent =
Agent( name="langchain_tool_agent", model="gemini-2.0-flash",
description="Agent to answer questions using TavilySearch.", instruction="I can
answer your questions by searching the internet. Just ask me anything!",
tools=[adk_tavily_tool] # Add the wrapped tool here )

'''

```

Full Example: Tavily Search

Here's the full code combining the steps above to create and run an agent using the LangChain Tavily search tool.

```

import os

from google.adk import Agent, Runner
from google.adk.sessions import InMemorySessionService
from google.adk.tools.langchain_tool import LangchainTool
from google.genai import types
from langchain_community.tools import TavilySearchResults

# Ensure TAVILY_API_KEY is set in your environment

```

```

if not os.getenv("TAVILY_API_KEY"):
    print("Warning: TAVILY_API_KEY environment variable not set.")

APP_NAME = "news_app"
USER_ID = "1234"
SESSION_ID = "session1234"

# Instantiate LangChain tool
tavily_search = TavilySearchResults(
    max_results=5,
    search_depth="advanced",
    include_answer=True,
    include_raw_content=True,
    include_images=True,
)

# Wrap with LangchainTool
adk_tavily_tool = LangchainTool(tool=tavily_search)

# Define Agent with the wrapped tool
my_agent = Agent(
    name="langchain_tool_agent",
    model="gemini-2.0-flash",
    description="Agent to answer questions using TavilySearch.",
    instruction="I can answer your questions by searching the internet",
    tools=[adk_tavily_tool] # Add the wrapped tool here
)

session_service = InMemorySessionService()
session = session_service.create_session(app_name=APP_NAME, user_id=USER_ID)
runner = Runner(agent=my_agent, app_name=APP_NAME, session_service=session)

# Agent Interaction
def call_agent(query):
    content = types.Content(role='user', parts=[types.Part(text=query)])
    events = runner.run(user_id=USER_ID, session_id=SESSION_ID, new_message=content)

```

```

    for event in events:
        if event.is_final_response():
            final_response = event.content.parts[0].text
            print("Agent Response: ", final_response)

call_agent("stock price of GOOG")

```

2. Using CrewAI tools

ADK provides the `CrewaiTool` wrapper to integrate tools from the CrewAI library.

Example: Web Search using CrewAI's Serper API

[Serper API](#) provides access to Google Search results programmatically. It allows applications, like AI agents, to perform real-time Google searches (including news, images, etc.) and get structured data back without needing to scrape web pages directly.

1. Follow [ADK installation and setup](#) guide.
2. **Install Dependencies:** Install the necessary CrewAI tools package. For example, to use the SerperDevTool:

```

` pip install crewai-tools

```

```

` 3. Obtain a Serper API KEY and export it as an environment variable.

```

```

` export SERPER_API_KEY=

```

```

` 4. **Import:** Import CrewaiTool from ADK and the desired CrewAI tool (e.g, SerperDevTool).

```

```

` from google.adk.tools.crewai_tool import CrewaiTool from crewai_tools
import SerperDevTool

```

```

` 5. **Instantiate & Wrap:** Create an instance of the CrewAI tool. Pass it to the CrewaiTool constructor. Crucially, you

```

must provide a name and description to the ADK wrapper, as these are used by ADK's underlying model to understand when to use the tool.

```
''' # Instantiate the CrewAI tool serper_tool_instance =
SerperDevTool( n_results=10, save_file=False, search_type="news", )

# Wrap it with CrewaiTool for ADK, providing name and description
adk_serper_tool = CrewaiTool( name="InternetNewsSearch",
description="Searches the internet specifically for recent news articles using
Serper.", tool=serper_tool_instance )

`` 6. **Add to Agent:** Include the
wrapped CrewaiTool instance in your agent's tools` list.

''' from google.adk import Agent

# Define the ADK agent my_agent = Agent( name="crewai_search_agent",
model="gemini-2.0-flash", description="Agent to find recent news using the
Serper search tool.", instruction="I can find the latest news for you. What topic
are you interested in?", tools=[adk_serper_tool] # Add the wrapped tool here )

'''
```

Full Example: Serper API

Here's the full code combining the steps above to create and run an agent using the CrewAI Serper API search tool.

```
import os

from google.adk import Agent, Runner
from google.adk.sessions import InMemorySessionService
from google.adk.tools.crewai_tool import CrewaiTool
from google.genai import types
from crewai_tools import SerperDevTool

# Constants
APP_NAME = "news_app"
USER_ID = "user1234"
SESSION_ID = "1234"
```

```

# Ensure SERPER_API_KEY is set in your environment
if not os.getenv("SERPER_API_KEY"):
    print("Warning: SERPER_API_KEY environment variable not set.")

serper_tool_instance = SerperDevTool(
    n_results=10,
    save_file=False,
    search_type="news",
)

adk_serper_tool = CrewaiTool(
    name="InternetNewsSearch",
    description="Searches the internet specifically for recent news and articles",
    tool=serper_tool_instance
)

serper_agent = Agent(
    name="basic_search_agent",
    model="gemini-2.0-flash",
    description="Agent to answer questions using Google Search.",
    instruction="I can answer your questions by searching the internet",
    # Add the Serper tool
    tools=[adk_serper_tool]
)

# Session and Runner
session_service = InMemorySessionService()
session = session_service.create_session(app_name=APP_NAME, user_id=USER_ID)
runner = Runner(agent=serper_agent, app_name=APP_NAME, session_service=session_service)

# Agent Interaction
def call_agent(query):
    content = types.Content(role='user', parts=[types.Part(text=query)])
    events = runner.run(user_id=USER_ID, session_id=SESSION_ID, new_message=content)

    for event in events:

```

```
if event.is_final_response():  
    final_response = event.content.parts[0].text  
    print("Agent Response: ", final_response)  
  
call_agent("what's the latest news on AI Agents?")
```