Introduction to Agent Development Kit

A hands-on intro to Google's ADK

“Agents are the new apps.”

Dharmesh Shah
Co-founder, Hubspot

“There will be probably more AI agents than there are people in the world.”

Mark Zuckerberg
CEO, Meta

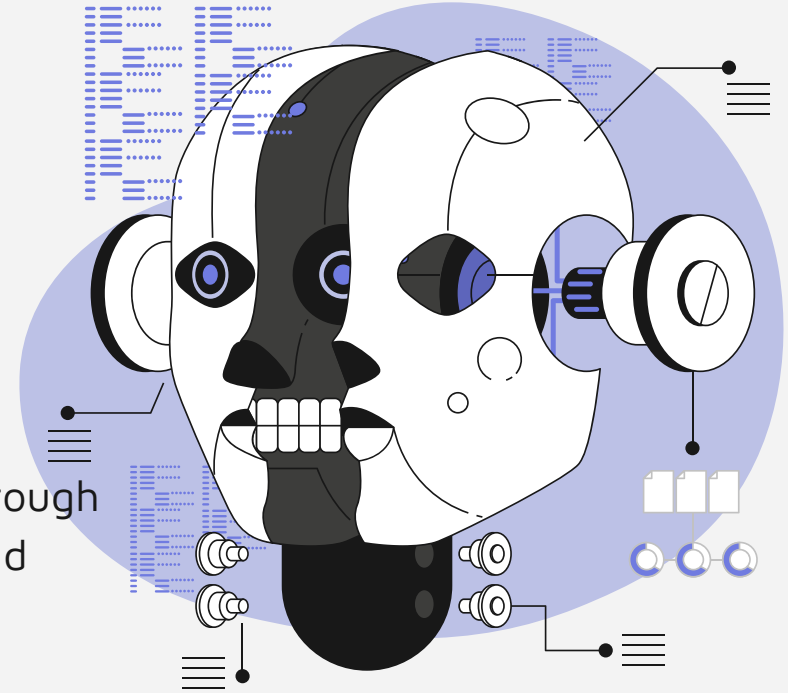
“There is going to be for the very first time agents sitting on top of tools.”

Jensen Huang
CEO, Nvidia

“AI agents will become the primary way we interact with computers in the future.”

Satya Nadella
CEO, Microsoft

AI Agents simplified



An AI agent is a system designed to reason through complex problems, create actionable plans, and execute these plans using a suite of tools.
It follows this continuous cycle:

- Think
- Plan
- Act
- Reflect

The Continuous Cycle of AI Agents

Think

The agent listen to what you're asking, then processes available data and context.

Plan

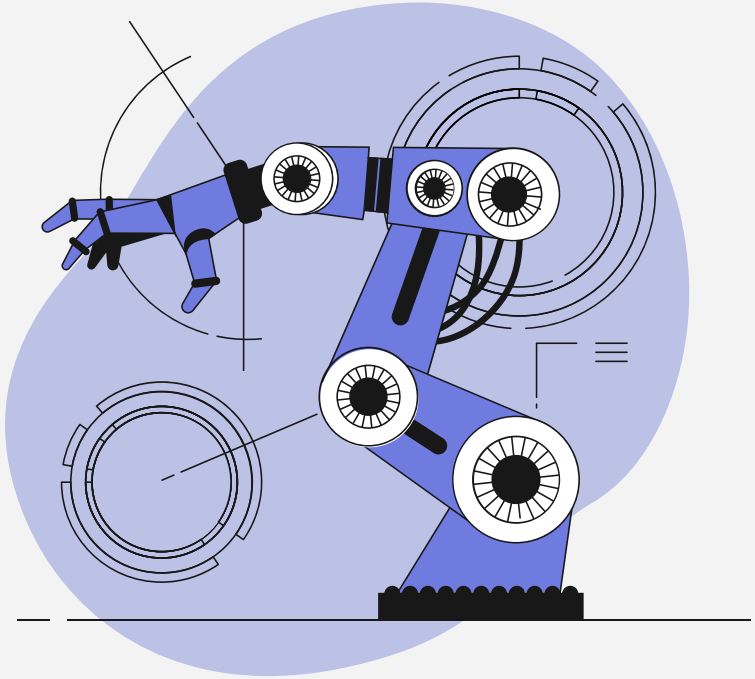
It decides on a strategy to achieve a goal or answer a question.

Act

Executes the strategy (e.g., API calls, data retrieval, user interaction).

Reflect

Assesses actions, refines approaches, and improves performance.



AI isn’t just answering questions anymore — it’s booking your calendar, replying to emails, generating reports, and understanding workflows. We’ve officially entered the age of autonomous AI agents — software that doesn’t just respond, but reasons and executes.

Google’s answer to this shift is the Agent Development Kit (ADK) - a framework designed to let developers build intelligent agents using structured prompts, external tools, memory modules, and large language models like Gemini.

How to Build Agents with ADK

Get started:

Python Java

```
pip install google-adk
```

1. **Create & activate a virtual environment** (Recommended)

```
python -m venv .venv
```

```
# Mac / Linux
source .venv/bin/activate

# Windows CMD:
.venv\Scripts\activate.bat

# Windows PowerShell:
.venv\Scripts\Activate.ps1
```

```
pip install google-adk
```

2. Create Agent Project

```
parent_folder/
    multi_tool_agent/
        __init__.py
        agent.py
        .env
```

To get Create your project and get an API key:

https://console.cloud.google.com/

Incase you need more guide:
https://www.merge.dev/blog/gemini-api-key

https://console.cloud.google.com/

Google Cloud | Generative Language Client | Search (/) for resources, docs, products, and more | Search

Welcome

You're working in Generative Language Client

Project number:

Dashboard | Cloud Hub New

Create a VM | Run a query in BigQuery | Deploy an application

Create a storage bucket

Try Gemini 2.5 Flash

Try Gemini

Quick access

APIs & Services | IAM & Admin | Billing | Compute Engine

Select a project

New project

Search projects and folders

Recent Starred All

Name	Type
Generative Language Client	Project
search-api-test	Project
cheaper-shop	Project
folome	Project

Cancel

Google Cloud | Generative Language Client | Search (/) for resources, docs, products, and more | Search

API APIs & Services

- Enabled APIs & services
- Library
- Credentials
- OAuth consent screen
- Page usage agreements

Credentials

+ Create credentials | Delete | Restore deleted credentials

Create credentials to access your enabled APIs. Learn more

API Keys

Name	Creation date	Restrictions	Actions
Gemini API Practice	Jul 24, 2024	Generative L	

OAuth 2.0 Client IDs

Name	Creation date	Type	Client ID	Actions
ADK Voice Calendar Integration	Jul 8, 2025			

Service Accounts

Manage service accounts

Email	Name	Actions
No service accounts to display		

Google Cloud | Generative Language Client

Product details

Gemini API

Google

Build with latest models from Google Deepmind using the Gemini API for Developers

Manage API Enabled

Overview Pricing Documentation Related Products

Overview

The Gemini API allows developers to build generative AI applications using Gemini models. Gemini is our most capable model, built from the ground up to be multimodal. It can generalize and seamlessly

Additional details

Type: SaaS & APIs

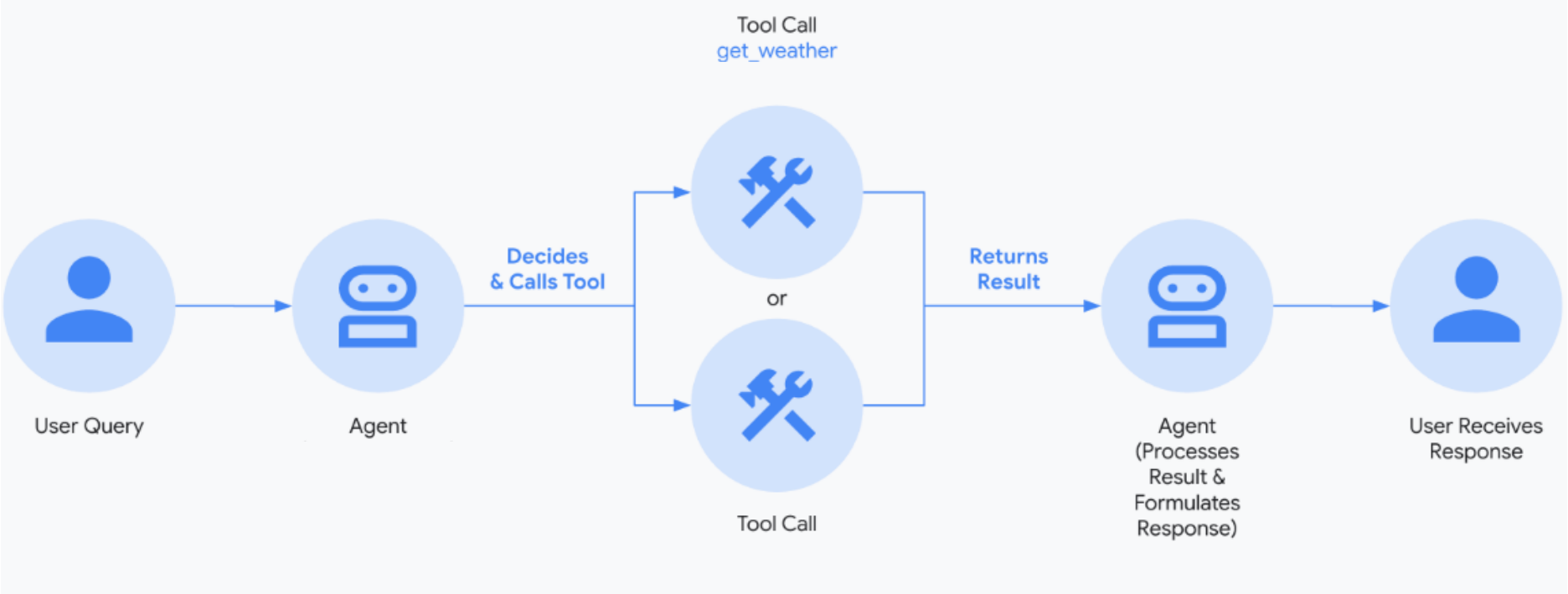
Last product update: 3/17/25

Model variants

The Gemini API offers different models that are optimized for specific use cases. Here's a brief overview of Gemini variants that are available:

Model variant	Input(s)	Output	Optimized for
Gemini 2.5 Pro `gemini-2.5-pro`	Audio, images, videos, text, and PDF	Text	Enhanced thinking and reasoning, multimodal understanding, advanced coding, and more
Gemini 2.5 Flash `gemini-2.5-flash`	Audio, images, videos, and text	Text	Adaptive thinking, cost efficiency
Gemini 2.5 Flash-Lite Preview `gemini-2.5-flash-lite-preview-06-17`	Text, image, video, audio	Text	Most cost-efficient model supporting high throughput
Gemini 2.5 Flash Native Audio `gemini-2.5-flash-preview-native-audio-dialog` &	Audio, videos, and text	Text and audio, interleaved	High quality, natural conversational audio outputs, with or without thinking

How our agent will function



Run your agent

Using the terminal, navigate to the parent directory of your agent project (e.g. using `cd ..`):

```
parent_folder/      <-- navigate to this directory
    multi_tool_agent/
        __init__.py
        agent.py
        .env
```

There are multiple ways to interact with your agent:

Then run

adk web

Or you can run directly in the terminal

adk run greeting_agent

Tools

Types of tools

How to add tools

Tool calling in action

What is a Tool?

In the context of ADK, a Tool represents a specific capability provided to an AI agent, enabling it to perform actions and interact with the world beyond its core text generation and reasoning abilities. What distinguishes capable agents from basic language models is often their effective use of tools.

Technically, a tool is typically a modular code component—**like a Python/ Java function**, a class method, or even another specialized agent—designed to execute a distinct, predefined task. These tasks often involve interacting with external systems or data.

Key Characteristics

Action-Oriented: Tools perform specific actions, such as:

- Querying databases
- Making API requests (e.g., fetching weather data, booking systems)
- Searching the web
- Executing code snippets
- Retrieving information from documents (RAG)
- Interacting with other software or services

Extends Agent capabilities: They empower agents to access real-time information, affect external systems, and overcome the knowledge limitations inherent in their training data.

Execute predefined logic: Crucially, tools execute specific, developer-defined logic. They do not possess their own independent reasoning capabilities like the agent's core Large Language Model (LLM). The LLM reasons about which tool to use, when, and with what inputs, but the tool itself just executes its designated function.

How Agents Use Tools

Agents leverage tools dynamically through mechanisms often involving function calling. The process generally follows these steps:

1. **Reasoning:** The agent's LLM analyzes its system instruction, conversation history, and user request.
2. **Selection:** Based on the analysis, the LLM decides on which tool, if any, to execute, based on the tools available to the agent and the docstrings that describes each tool.
3. **Invocation:** The LLM generates the required arguments (inputs) for the selected tool and triggers its execution.
4. **Observation:** The agent receives the output (result) returned by the tool.
5. **Finalization:** The agent incorporates the tool's output into its ongoing reasoning process to formulate the next response, decide the subsequent step, or determine if the goal has been achieved.

Think of the tools as a specialized toolkit that the agent's intelligent core (the LLM) can access and utilize as needed to accomplish complex tasks.

Tool Types in ADK ¶

ADK offers flexibility by supporting several types of tools:

1. **Function Tools:** Tools created by you, tailored to your specific application's needs.
    - **Functions/Methods:** Define standard synchronous functions or methods in your code (e.g., Python def).
    - **Agents-as-Tools:** Use another, potentially specialized, agent as a tool for a parent agent.
    - **Long Running Function Tools:** Support for tools that perform asynchronous operations or take significant time to complete.
2. **Built-in Tools:** Ready-to-use tools provided by the framework for common tasks. Examples: Google Search, Code Execution, Retrieval-Augmented Generation (RAG).
3. **Third-Party Tools:** Integrate tools seamlessly from popular external libraries. Examples: LangChain Tools, CrewAI Tools.

Extra:

- **Sessions and memory**: make agents remember things between different conversations.
- **Persistent storage**: make agent save data (sessions & memory); database functionality (InMemorySessionService & DatabaseSessionService).
- **Multi agent**: multiple agents working together.
- **Workflows for agents**: when agents work in a specific order:
    - Sequential agent
    - Parallel agent
    - Loop agent

Some adk samples:

https://github.com/benjaminogbonna/adk-samples

.github	Updated CODEOWNERS for image-scoring (google#226)	last month
java	fix: update tools.yaml for Forecasting agent (google#228)	3 weeks ago
python	chore: update auto-insurance-agent to latest ADK version (g...	last week
.gitignore	Added agent data files and .whl files to .gitignore (google#2...	2 weeks ago
CONTRIBUTING.md	initial commit with readmes	3 months ago
LICENSE	initial commit with readmes	3 months ago
README.md	Adding a new sample imagen_scoring (google#163)	last month

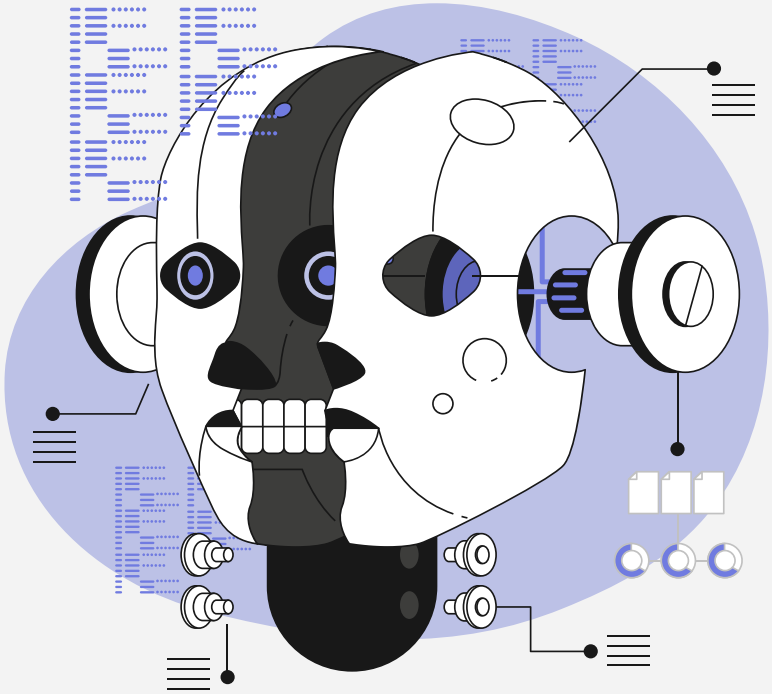
README License

Agent Development Kit (ADK) Samples

License Apache 2.0

Thanks!

Questions?



@ Pie & AI Abuja