



Agent Development Kit

Building AI Agents using FunctionTool, MCP, Agent Engine, and RAG

Arjun Prabhulal, Nov 2025



Agenda

1. Overview - Tools and Function Calling
2. Model Context Protocol (MCP)
 - Building Agents as MCP Clients
 - Building an MCP Server
3. Vertex AI Agent Engine
 - Overview of Agent Engine
 - Overview of Agent Starter Pack (ASP)
 - Deployment - Agent Engine ,GKE & Cloud Run
4. Building RAG agents with ADK , Agent Engine and ASP using
 - Vertex AI RAG Engine
 - Vertex AI Search



1. Overview - Tools and Function Calling

What are Tools and Function Calling ?



Tools are external functions, APIs that an LLM call to perform actions.

Tools = LLMs External Capabilities

Function Calling - Allows a LLM to connect with and **invoke tools**

Function Calling = How the LLM uses **tools**

Tools as Agents



Tools make agents actionable to interact with data, services, and APIs.

Types of Tools

- **Built-in** - Gemini tools - Web search , Code execution
- **Google Cloud tools** - BigQuery, Bigtable, Vertex AI search/RAG
- **Third-party tools** - GitHub, Notion, Hugging Face

ADK supports building own tools using

- **Function Calling** - Custom python,Java functions
- **MCP** (Model Context Protocol)
- OpenAPI-based tool (from specs)

Built-in Tools



Connecting ADK Agent using Build-in `google_search` tools

```
from google.adk.agents import Agent
from google.adk.tools import FunctionTool,google_search

root_agent = Agent(
    model="gemini-2.5-flash",
    name="support_agent",
    instruction="You are a helpful customer support assistant.",
    tools=[google_search]

)
```

A screenshot of a conversational AI application. At the top, there is a blue input bar with a white person icon on the right. A user message reads: "Hi . What is stock price of SP500 today ?". Below it, the AI's response is displayed in a dark gray box with rounded corners. The response text is: "On Monday, November 3, 2025, the S&P 500 rose to 6859 points, gaining 0.27% from the previous trading session. At the market open, the S&P 500 was at 6,882.32, up 0.62%. Earlier in the day, around 8:41 a.m. ET, the S&P 500 index was reported at 6,840.20, an increase of 0.26%. Later, by 11:54 AM ET, the S&P 500 had gained 10.83 points, or 0.16%. The index added 0.2% by the end of the trading day." At the bottom of the response box, there is a smaller white box containing a Google logo and the text "S&P 500 stock price today November 3 2025".



Third Party Tools

Connecting ADK Agent to an MCP Server - GitHub Example

MCPToolset() allows us to add custom tools via the MCP protocol.

```
import os
from google.adk.agents import Agent
from google.adk.tools.mcp_tool.mcp_session_manager import StreamableHTTPServerParams
from google.adk.tools.mcp_tool.mcp_toolset import MCPToolset

GITHUB_TOKEN = os.environ.get("GITHUB_TOKEN", "")

root_agent = Agent(
    model="gemini-2.5-pro",
    name="github_agent",
    instruction="Help users get information from GitHub",
    tools=[

        MCPToolset(
            connection_params=StreamableHTTPServerParams(
                url="https://api.githubcopilot.com/mcp/",
                headers={
                    "Authorization": f"Bearer {GITHUB_TOKEN}",
                    "X-MCP-Toolsets": "all",
                    "X-MCP-ReadOnly": "true"
                },
            ),
        ),
    ],
)
```

List 10 repos of GoogleCloudPlatform

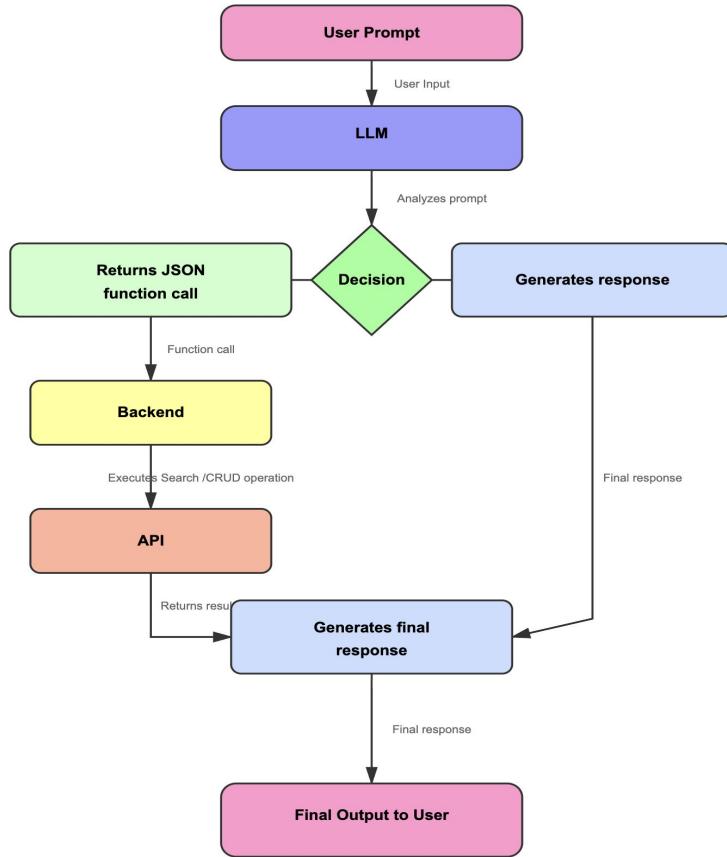
search_repositories

✓ search_repositories

Here are 10 repositories from GoogleCloudPlatform:

1. **microservices-demo**: Sample cloud-first application with 10 microservices showcasing Kubernetes, Istio, and gRPC. :stargazers: 19273, Language: Go, Forks: 9071 <https://github.com/GoogleCloudPlatform/microservices-demo>
2. **terraformer**: CLI tool to generate terraform files from existing infrastructure (reverse Terraform). Infrastructure to Code :stargazers: 14204, Language: Go, Forks: 1809 <https://github.com/GoogleCloudPlatform/terraformer>
3. **generative-ai**: Sample code and notebooks for Generative AI on Google Cloud, with Gemini on Vertex AI :stargazers: 11949, Language: Jupyter Notebook, Forks: 3481 <https://github.com/GoogleCloudPlatform/generative-ai>
4. **training-data-analyst**: Labs and demos for courses for GPU Training (<http://cloud.google.com/training>). :stargazers: 8340, Language: Jupyter Notebook, Forks: 6017 <https://github.com/GoogleCloudPlatform/training-data-analyst>
5. **python-docs-samples**: Code samples used on cloud.google.com :stargazers: 7875, Language: Jupyter Notebook, Forks: 6631 <https://github.com/GoogleCloudPlatform/python-docs-samples>
6. **kubecth-ai**: AI powered Kubernetes Assistant :stargazers: 6966, Language: Go, Forks: 629 <https://github.com/GoogleCloudPlatform/kubecth-ai>
7. **golang-samples**: Sample apps and code written for Google Cloud in the Go programming language. :stargazers: 4529, Language: Go, Forks: 1827 <https://github.com/GoogleCloudPlatform/golang-samples>
8. **professional-services**: Common solutions and tools developed by Google Cloud's Professional Services team. This repository and its contents are not an officially supported Google product. :stargazers: 2957, Language: Python, Forks: 1398 <https://github.com/GoogleCloudPlatform/professional-services>
9. **nodejs-docs-samples**: Node.js samples for Google Cloud Platform products. :stargazers: 2927, Language: JavaScript, Forks: 2009 <https://github.com/GoogleCloudPlatform/nodejs-docs-samples>
10. **tensorflow-without-a-phd**: A crash course in six episodes for software developers who want to become machine learning practitioners. :stargazers: 2829, Language: Jupyter Notebook, Forks: 914 <https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd>

Function Calling - Decision Tree



FunctionTool wraps a Python function as a callable tool, enabling the LLM agent to invoke code or APIs through natural-language instructions.

```
from google.adk.agents import Agent
from google.adk.tools import FunctionTool

def get_order_status(order_id: str):
    """
    Retrieves the status of an order by order ID.

    Args:
        order_id (str): The unique order identifier.
    """

    return {"order_id": order_id, "status": "Shipped", "delivery_date": "Nov 4, 2025"}
```

This code snippet shows a Python function `get_order_status` wrapped as a `FunctionTool`. The function retrieves the status of an order by its ID and returns a JSON response. It is used within an `Agent` configuration to provide customer support assistance.

Function Calling - Demo



```
INFO: Started server process [54098]
INFO: Waiting for application startup.
```

```
+-----+
| ADK Web Server started
|
| For local testing, access at http://127.0.0.1:8000.
+-----+
```

```
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

SESSION ID 69527cb9-9a5d-44ee-857b-2b992844fe93

Token Streaming + New Session ⌂

what is order status of XYZ123?

get_order_status

✓ get_order_status

Your order XYZ123 has been shipped and will be delivered on Nov 4, 2025.

Event 2 of 3 < > X

Event	Request	Response
		support_agent ← get_order_status

content:

- parts:
 - 0:
 - functionResponse:
 - id: "adk-b4372316-6adf-42bc-a31f-a4dd29e56110"
 - name: "get_order_status"
 - response:
 - order_id: "XYZ123"
 - status: "Shipped"
 - delivery_date: "Nov 4, 2025"
 - role: "user"
 - invocationId: "e-0a76a0f7-dd71-483a-afe1-bcee7f11a968"
 - author: "support_agent"
- actions:
 - stateDelta:
 - artifactDelta:
 - requestedAuthConfigs:
 - requestedToolConfirmations:
- id: "f0a46104-a9a0-4dcf-a7fb-682dbaebc301"
- timestamp: 1762194940.726673
- title: "functionResponse:get_order_status"

2025-11-03 13:35:39,026 - INFO - google_llm.py:133 - Sending out request, model: gemini-2.5-flash, backend: GoogleLLMVariant.VERTEX_AI, stream: False
2025-11-03 13:35:40,718 - INFO - _client.py:1740 - HTTP Request: POST https://us-central1-aiplatform.googleapis.com/v1beta1/projects/vertex-ai-experiment/locations/us-central1/publishers/google/models/gemini-2.5-flash:generateContent "HTTP/1.1 200 OK"
2025-11-03 13:35:40,721 - INFO - google_llm.py:186 - Response received from the model.
2025-11-03 13:35:40,722 - WARNING - types.py:6073 - Warning: there are non-text parts in the response: ['function_call', 'thought_signature'], returning concatenated text result from text parts. Check the full candidates.content.parts accessor to get the full model response.

Function Calling - Best Practices



- Use **Fewer Parameters** to reduce complexity.
- Prefer **Simple Data Types** over custom classes.
- Name **function and parameter** clearly to reflect purpose and to aid LLM interpretation.
- Design tools for parallel execution using **asynchronous handling**



2. Model Context Protocol (MCP)

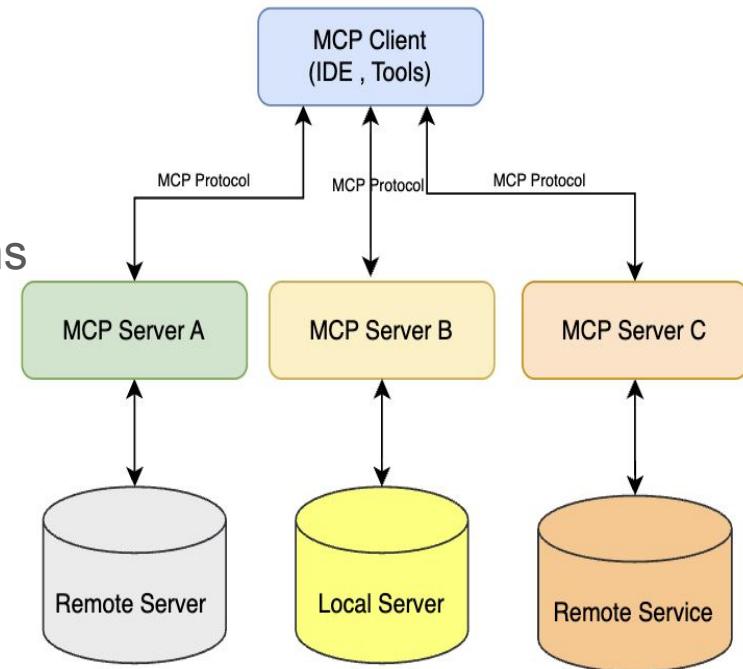
- Building Agents as MCP Clients
- Building MCP Server

What is (MCP) Model Context Protocol ?



MCP is an open-standard protocol that enables LLM interact with external data sources and tools ensuring interoperability, security, and consistency across platforms.

MCP follows a client-server architecture, defining how data (**resources**), interactive templates (**prompts**), and actionable functions (**tools**) are exposed by an MCP server and consumed by an MCP client



Function Calling vs MCP - When to Use Each



Feature	Function Calling	MCP
Scope	Local or simple APIs	External tools & services
Integration Type	Inline / direct	Client Server protocol
Use Case	Quick, lightweight tasks	Scalable, multi-tool orchestration
Communication	Single model context	Standardized over HTTP/SSE
Best For	One-off function calls	Production-grade agent ecosystems

ADK agent as MCP Client



An ADK agent acts as MCP client using MCP (local or remote) server

Setting up ADK Agent

```
# create a adk directory
mkdir adk-mcp-client-firecrawl
cd adk-mcp-client-firecrawl

# setup virtual env.
python3 -m venv .venv && source .venv/bin/activate

#install adk library
pip install google-adk

# Initialize agent
echo "from . import agent" > __init__.py
touch agent.py
```

```
from google.adk.agents.llm_agent import Agent
from google.adk.tools.mcp_tool.mcp_session_manager import StdioConnectionParams
from google.adk.tools.mcp_tool.mcp_toolset import MCPToolset
from mcp import StdioServerParameters
```

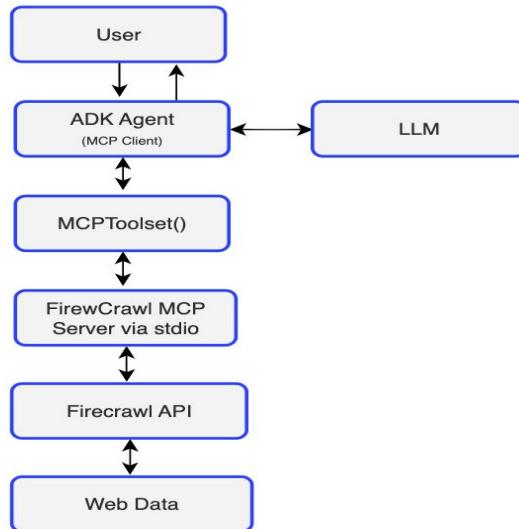
StdioConnectionParams is used to connect an ADK agent to a local MCP server by launching it as a subprocess through standard input/output (stdio).

ADK agent as MCP client - Local MCP Server



MCPToolset() automatically connects the agent to an MCP server, handles tool discovery, exposes tools to the agent, and proxies tool calls.

MCPToolset connects through **stdio** with the local MCP server as a local subprocess .



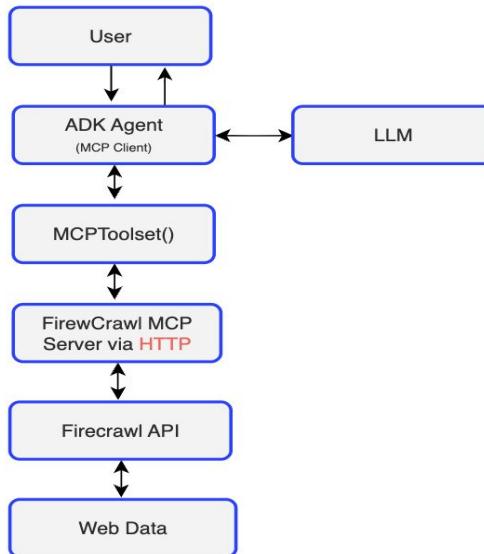
```
root_agent = Agent(
    model="gemini-2.5-pro",
    name="firecrawl_agent",
    description="A helpful assistant for scraping websites with Firecrawl",
    instruction="Help the user search for website content",
    tools=[MCPToolset(
        connection_params=StdioConnectionParams(
            server_params = StdioServerParameters(
                command="npx",
                args=["-y", "firecrawl-mcp"],
                env={
                    "FIRECRAWL_API_KEY": FIRECRAWL_API_KEY,
                }
            ),
            timeout=30,
        ),
        ],
    ],
)
```

ADK agent as MCP client - Remote MCP Server



MCPToolset connects MCP server via an HTTP streaming connection

StreamableHTTPServerParams - handles real-time communication with the remote MCP server via HTTP streaming (large responses or continuous scraping).



```
● ● ●  
from google.adk.agents.llm_agent import Agent  
from google.adk.tools.mcp_tool.mcp_session_manager import StreamableHTTPServerParams  
from google.adk.tools.mcp_tool.mcp_toolset import MCPToolset
```

```
● ● ●  
root_agent = Agent(  
    model="gemini-2.5-pro",  
    name="firecrawl_agent",  
    description="A helpful assistant for scraping websites with Firecrawl",  
    instruction="Help the user search for website content",  
    tools=[  
        MCPToolset(  
            connection_params=StreamableHTTPServerParams(  
                url=f"https://mcp.firecrawl.dev/{FIRECRAWL_API_KEY}/v2/mcp",  
            ),  
        ),  
    ],  
)
```

ADK agent as MCP client - Demo



hi what can you do ?

I can help you with web scraping, searching, and crawling tasks. What would you like to do?

scrape <https://google.github.io/adk-docs/>

firecrawl_scrape

✓ firecrawl_scrape

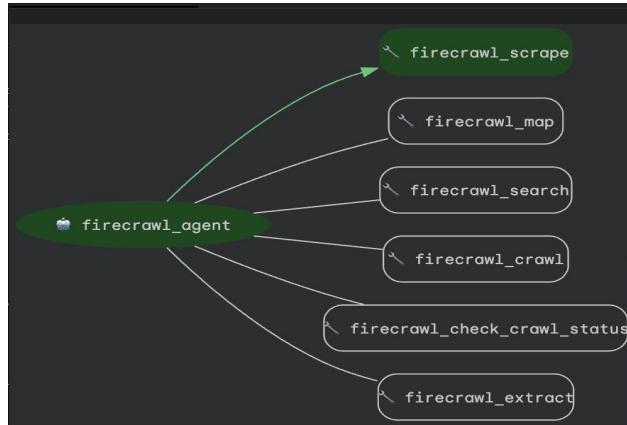
I have successfully scraped the content from the provided URL. Here is the summary of the content:

Agent Development Kit (ADK) is a flexible and modular framework for developing and deploying AI agents. It is model-agnostic, deployment-agnostic, and compatible with other frameworks. ADK is designed to make agent development feel more like software development, making it easier for developers to create, deploy, and orchestrate agentic architectures.

Key Features:

- Flexible Orchestration: Define workflows using workflow agents for predictable pipelines or leverage LLM-driven dynamic routing for adaptive behavior.
- Multi-Agent Architecture: Build modular and scalable applications by composing multiple specialized agents in a hierarchy.
- Robust Ecosystem: Equip agents with diverse capabilities using pre-built tools, custom functions, 3rd-party libraries, or even other agents as tools.
- Deployment Ready: Containerize and deploy your agents anywhere – locally, with Vertex AI Agent Engine, or in custom infrastructure using Cloud Run or Docker.
- Built-in Evaluation: Systematically assess agent performance by evaluating both the final response quality and the step-by-step execution trajectory.
- Safety and Security: Learn to build powerful and trustworthy agents by implementing security and safety patterns and best practices.

You can find more information at the source URL: <https://google.github.io/adk-docs/>



```
content:
  parts:
    0:
      functionResponse:
        id: "adk-170e2e12-4a5d-4476-b2f7-eda5e4ef00e8"
        name: "firecrawl_scrape"
      response:
        content:
          0:
            type: "text"
            text: "{ \"markdown\": \"[Skip to content](https://google.github.io/adk-docs/#learn-more)\\n\\n[Edit this page](https://github.com/google/adk-docs/edit/main/docs/index.md \\\"Edit this page\\\")\\n\\n[View source of this page](https://github.com/google/adk-docs/raw/main/docs/index.md \\\"View source of this page\\\")\\n\\n! [Agent Development Kit Logo](https://google.github.io/adk-docs/assets/agent-development-kit.png)\\n\\n# Agent Development Kit\\n\\nAgent Development Kit (ADK) is a flexible and modular framework for **developing** and **deploying** AI agents**. While optimized for Gemini
```

Building MCP Server



`adk_to_mcp_tool_type` - Imports a utility function that converts an ADK tool definition (`FunctionTool`) into an MCP-compatible schema (`mcp_types.Tool`).

`FunctionTool`, a wrapper class from the Agent Development Kit (ADK) that turns a normal Python function into an “ADK tool.”

<https://github.com/arjunprabhulal/mcp-youtube-search>

```
# MCP Server Imports
from mcp import types as mcp_types
from mcp.server.lowlevel import Server, NotificationOptions
from mcp.server.models import InitializationOptions
import mcp.server.stdio

# ADK Tool Imports
from google.adk.tools.function_tool import FunctionTool
# ADK <-> MCP Conversion Utility
from google.adk.tools.mcp_tool.conversion_utils import adk_to_mcp_tool_type
```



3. Vertex AI Agent Engine

[Agent Engine - Overview](#)

[Agent Starter Pack- Overview](#)

[Deploy - Agent Engine](#)

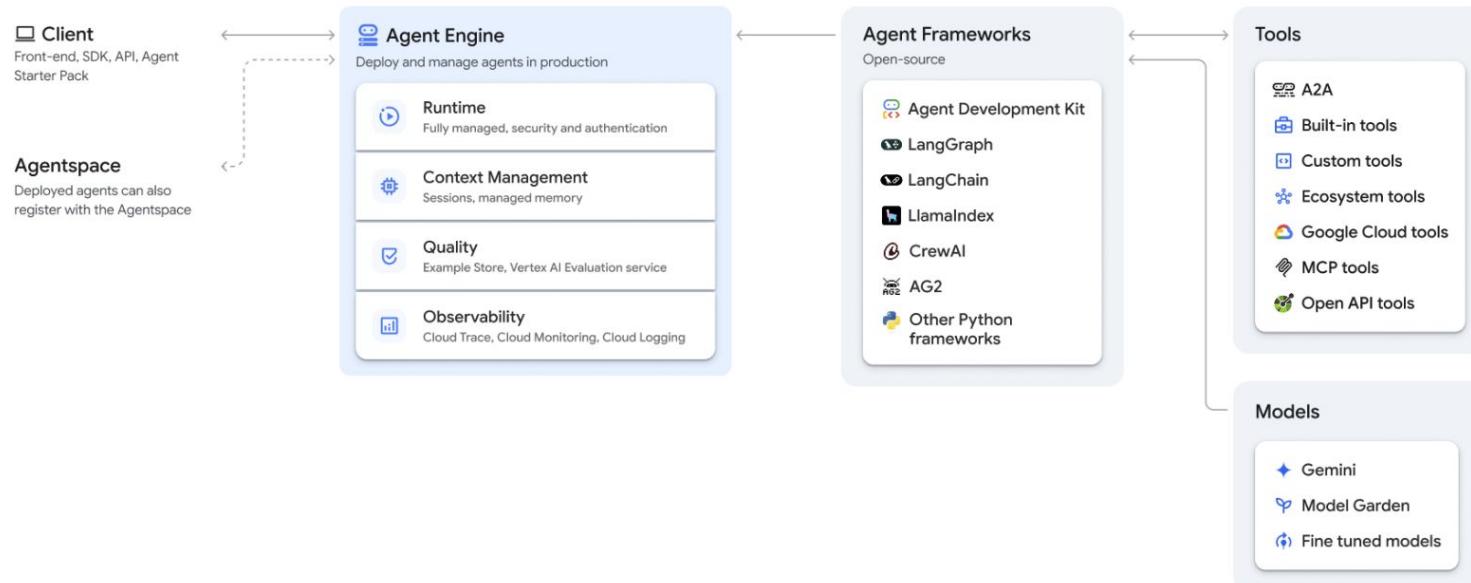
[Deploy - Cloud Run](#)

[Deploy - GKE](#)

Vertex AI Agent Engine - Overview



Agent Engine is a fully managed Google Cloud service enabling developers to deploy, manage, and scale AI agents in production.



Vertex AI Agent Engine - Features



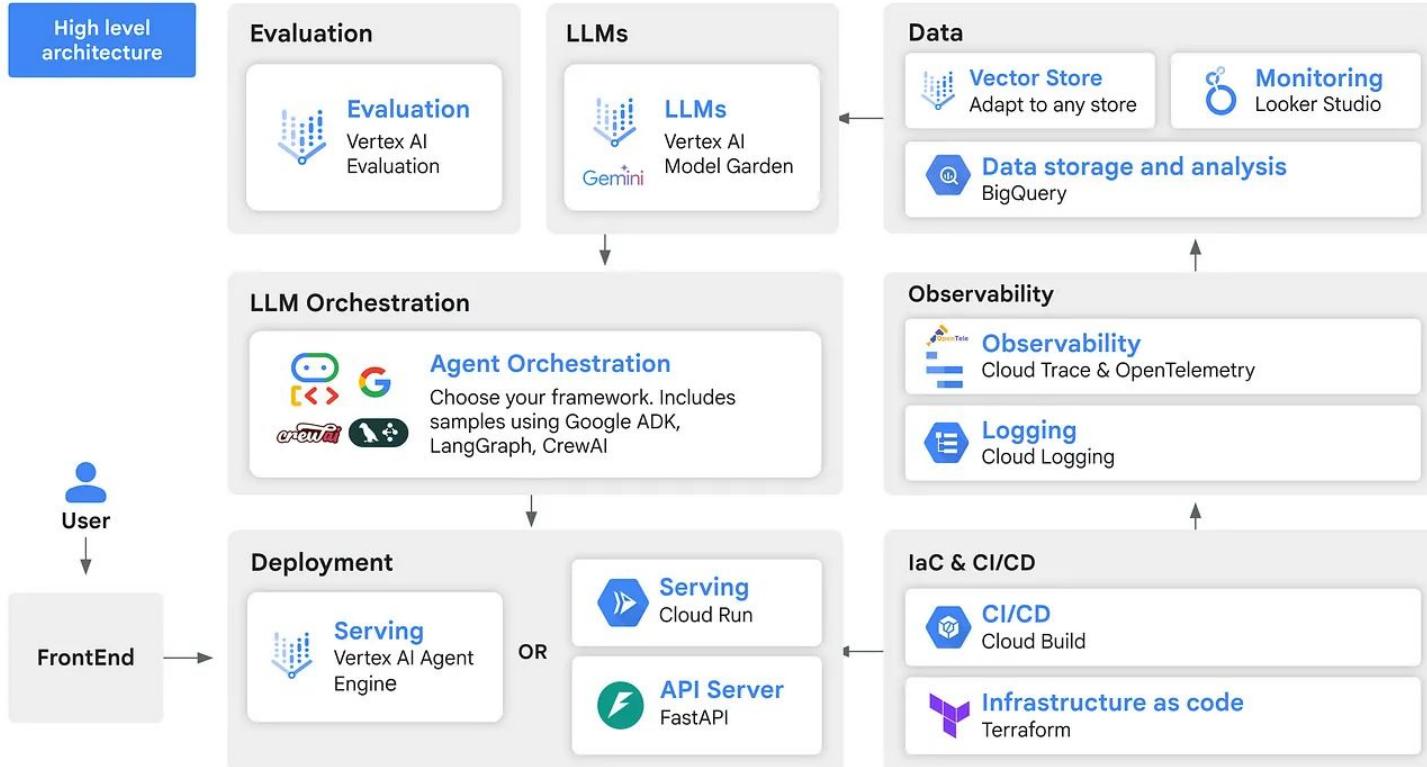
Managed runtime for agents: Agent Engine automatically handles deployment, scaling, containerization, and infrastructure security.

Sessions & memory bank: Built-in session management to retain context across interactions, and a persistent memory layer for long-term agent state.

Tool & model integration: Seamlessly connect LLMs with tools such as APIs or function calls, enabling end-to-end agent workflows deployable with minimal code

Security, compliance, observability: Enterprise-grade controls including VPC-SC, private networking, audit logging, and tracing dashboards for safe and transparent operations.

Agent Starter Pack(ASP)

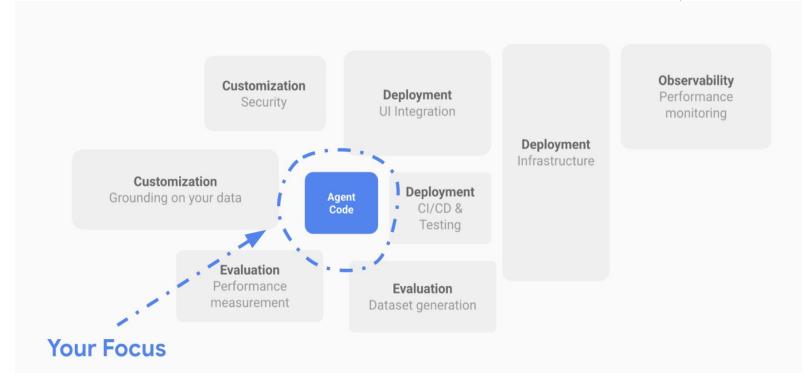


Agent Starter Pack(ASP) - Overview



Bridges the gap between experimentation and production

Collection of production-ready GenAI Agent Templates build for Google Cloud.



Provides **pre-built agent templates** (e.g., ReAct, RAG, multi-agent, multimodal/live API) designed for rapid generation of generative-AI agents on Google Cloud

Accelerates development by providing deployment infrastructure, observability/monitoring scaffolding, CI/CD pipelines, and best-practice project scaffolding

Agent Starter Pack(ASP) - Features



Production-ready foundation: ASP provides templates for deployment, CI/CD pipelines, infrastructure as code (**IaC**), API serving, and testing.

Observability & monitoring built in: Includes logging, tracing, dashboards,

Security & compliance scaffolding: GCP best practices for security controls

Data, UI & storage integrations: Supports connectivity for data sources, vector stores for RAG, UI playgrounds, and storage layers.

Focus on core logic: ASP allows us to focus on prompts, tools, orchestration, business logic.

Agent Starter Pack(ASP) - Usage



Prototype to Production: Accelerate deployment of ADK or LangChain agents using ready-made templates.

Enterprise Deployment: Deploy agents with built-in security, monitoring, scalability, and CI/CD.

Multi-Agent / RAG Workflows: Enable tools, retrieval, and orchestration with data-store & UI support.

Cross-Team Collaboration: Separate agent logic from infra so dev and ops teams work in parallel.

Agent Starter Pack - ADK_Base Project Creation



```
● ● ●  
# On macOS and Linux.  
curl -LsSf https://astral.sh/uv/install.sh | sh  
  
# On Windows.  
powershell -ExecutionPolicy Bypass -c "irm https://astral.sh/uv/install.ps1 | iex"  
  
#Agent Starter Pack Project creation  
uvx agent-starter-pack create adk-agent-demo
```

Installation ASP project creation

```
> uvx agent-starter-pack create adk-agent-demo  
Installed 60 packages in 229ms  
  
== Google Cloud Agent Starter Pack 🚀 ==  
Welcome to the Agent Starter Pack!  
This tool will help you create an end-to-end production-ready AI agent in Google Cloud!  
  
> Please select a agent to get started:  
1. adk_base - A base ReAct agent built with Google's Agent Development Kit (ADK)  
2. adk_a2a_base - A base ReAct agent built with Google's Agent Development Kit (ADK) and Agent2Agent (A2A) Protocol. #experimental  
3. adk_live - Real-time multimodal agent with ADK and Gemini Live API for low-latency voice and video interaction.  
4. agentic_rag - ADK RAG agent for document retrieval and Q&A. Includes a data pipeline for ingesting and indexing documents into Vertex AI Search or Vector Search.  
5. langgraph_base_react - An agent implementing a base ReAct agent using LangGraph  
6. crewai_coding_crew - A multi-agent system implemented with CrewAI created to support coding activities  
7. Browse agents from google/adk-samples - Discover additional samples  
  
Enter the number of your template choice (1): 1
```

Agent Starter Pack - ADK Base (ReAct Agent)



ReAct agent created using adk_base template and Vertex AI Agent Engine as deployment target

```
arjunprabhal@Arjuns-MacBook-Pro:~/workshop
./my-rag-agent (-zsh) 361
~/workshop (-zsh) 362

Welcome to the Agent Starter Pack!
This tool will help you create an end-to-end production-ready AI agent in Google Cloud!

> Please select a agent to get started:
1. adk_base - A base ReAct agent built with Google's Agent Development Kit (ADK)
2. adk_a2a_base - A base ReAct agent built with Google's Agent Development Kit (ADK) and Agent2Agent (A2A) Protocol. #experimental
3. adk_live - Real-time multimodal agent with ADK and Gemini Live API for low-latency voice and video interaction.
4. agentic_rag - ADK RAG agent for document retrieval and Q&A. Includes a data pipeline for ingesting and indexing documents into Vertex AI Search or Vector Search.
5. langgraph_base_react - An agent implementing a base ReAct agent using LangGraph
6. crewai_coding_crew - A multi-agent system implemented with CrewAI created to support coding activities
7. Browse agents from google/adk-samples - Discover additional samples

Enter the number of your template choice (1): 1

> Please select a deployment target:
1. Vertex AI Agent Engine - Vertex AI Managed platform for scalable agent deployments
2. Cloud Run - GCP Serverless container execution

Enter the number of your deployment target choice (1): 1

> Please select a CI/CD runner:
1. Google Cloud Build - Fully managed CI/CD, deeply integrated with GCP for fast, consistent builds and deployments.
2. GitHub Actions - GitHub Actions: CI/CD with secure workload identity federation directly in GitHub.

Enter the number of your CI/CD runner choice (1): 1

Enter desired GCP region (Gemini uses global endpoint by default) (us-central1):

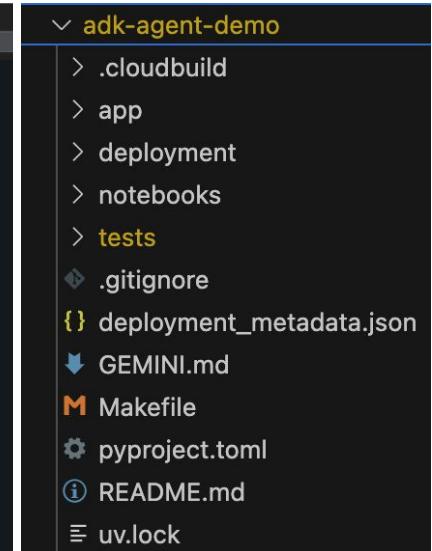
> You are logged in with account: 'arjun.prabhal@gmail.com'
> You are using project: 'vertex-ai-experiment'
> Do you want to continue? (The CLI will check if Vertex AI is enabled in this project) [Y/skip/edit] (Y): Y
> Successfully configured project: vertex-ai-experiment
> Testing GCP and Vertex AI Connection...
> ✓ Successfully verified connection to Vertex AI in project vertex-ai-experiment

> 🎉 Done. Execute the following command to get started:

> Success! Your agent project is ready.

Project README: cat adk-agent-demo/README.md
Online Development Guide: https://google/asp-dev

🚀 To get started, run the following command:
cd adk-agent-demo && make install && make playground
```



Agent Starter Pack - ADK Base Project Structure



Adk-agent-demo Project structure

```
adk-agent-demo/
└── app/                      # Core application code
    ├── agent.py                # Main agent logic
    ├── agent_engine_app.py     # Agent Engine application logic
    └── utils/                  # Utility functions and helpers
└── .cloudbuild/               # CI/CD pipeline configurations for Google Cloud Build
└── deployment/                # Infrastructure and deployment scripts
└── notebooks/                 # Jupyter notebooks for prototyping and evaluation
└── tests/                     # Unit, integration, and load tests
└── Makefile                   # Makefile for common commands
└── GEMINI.md                  # AI-assisted development guide
└── pyproject.toml             | # Project dependencies and configuration
```

Agent Engine - Local Development & Cloud Deploy



Install dependencies and launch local playground



```
# Install dependencies  
make install  
  
#and launch the local playground  
make playground
```

Vertex AI Agent Engine deployment



```
# Replace YOUR_DEV_PROJECT_ID with your actual Google Cloud  
Project ID  
gcloud config set project YOUR_DEV_PROJECT_ID  
  
# Provision GCP resources for dev. environment (optional)  
make setup-dev-env  
  
#Deploy Backend (Deploy to Agent Engine)  
make backend
```

Agent Engine - Demo



```
▶ make deploy
# Export dependencies to requirements file using uv export.
(uv export --no-hashes --no-header --no-dev --no-emit-project --no-annotate > .requirements.txt 2>/dev/null || \
    uv export --no-hashes --no-header --no-dev --no-emit-project > .requirements.txt) && \
    uv run -m app.agent_engine_app
INFO:root:Bucket vertex-ai-experiment-agent-engine already exists
INFO:root:Bucket vertex-ai-experiment-agent-engine already exists
```

DEPLOYING AGENT TO VERTEX AI AGENT ENGINE

Deployment successful

```
INFO:vertexai_genai.agentengines:Agent Engine created. To use it in another session:
INFO:vertexai_genai.agentengines:agent_engine=client.agent_engines.get('projects/913703049174/locations/us-central1/reasoningEngines/8742036632540020736')
INFO:root:Agent Engine ID written to deployment_metadata.json
```

Deployment successful! Test your agent: notebooks/adk_app_testing.ipynb
Service Account: service-913703049174@gcp-sa-aiplatform-re.iam.gserviceaccount.com

View in console: <https://console.cloud.google.com/vertex-ai/agents/locations/us-central1/agent-engines/8742036632540020736?project=vertex-ai-experiment>

Agent Engine - Validate Agent Engine



```
# "projects/PROJECT_ID/locations/us-central1/reasoningEngines/ENGINE_ID"
REASONING_ENGINE_ID = None

if REASONING_ENGINE_ID is None:
    try:
        with open("../deployment_metadata.json") as f:
            metadata = json.load(f)
            REASONING_ENGINE_ID = metadata.get("remote_agent_engine_id")
    except (FileNotFoundException, json.JSONDecodeError):
        pass

print(f"Using REASONING_ENGINE_ID: {REASONING_ENGINE_ID}")
# Get the existing agent engine
remote_agent_engine = client.agent_engines.get(name=REASONING_ENGINE_ID)

Using REASONING_ENGINE_ID: projects/913703049174/locations/us-central1/reasoningEngines/8742036632540020736
```

```
[8]: async for event in remote_agent_engine.async_stream_query(
    message="What is weather today in Atlanta ?", user_id="test"
):
    print(event)

{'content': {'parts': [{'thought_signature': 'CqYCAePx_17X880ofuRPpg1Un8FMZvCMou4D0UgDZWdYxc_V8s4cz60GX_6v_7qUrgDZ3CsCSid9leAqJUd1td5_jo2ZTxFrhrmHnIaE50AEzbh_bhGjctNhTgEW3ympaktLFK4n_FfeRujJA05EP2ZRDzFUsyUFkMvPirBCCRnVCDRKrIrk7e5uW8zewlTE_IMr65RjuixV7KxpzbhW0SUMLeaRvW-LDxq0rpDTGAdFt3qKhNv19ldy8Gyxu3z2Ip0Ij69T-zB_lG_DTX220-Ekxn9WafChx6sgCtzmdMy-boxTIKSh79FnIdauhWt74AaPwCbbK8LBV06XUQ1U6reyOev4rkPmLuzNZY4XbcdOp5Abn0VQCYG0zE', 'function_call': {'id': 'adk-90524973-34b7-4547-bfc9-2f6feab92d69', 'args': {'query': 'Atlanta'}, 'name': 'get_weather'}}, 'role': 'model', 'finish_reason': 'STOP', 'usage_metadata': {'candidates_token_count': 5, 'candidates_tokens_details': [{"modality": "TEXT", "token_count": 5}], 'prompt_tokens_details': [{"modality": "TEXT", "token_count": 152}], 'thoughts_token_count': 64, 'total_token_count': 221}, 'traffic_type': 'ON_DEMAND'}, 'avg_logprobs': -1.7934026718139648, 'invocation_id': 'e-65313a8b-8b1c-4dc3-b7af-1a64b9e7d797', 'author': 'root_agent', 'actions': {'state_delta': {}, 'artifact_delta': {}}, 'requested_auth_configs': {}, 'requested_tool_confirmations': {}, 'long_running_tool_ids': [], 'id': '2ae098df-f09a-4eb7-b509-90d9c37551ca', 'timestamp': 1762231191.768893}
{'content': {'parts': [{'function_response': {'id': 'adk-90524073-34b7-4547-bfc9-2f6feab92d60', 'name': 'get_weather', 'response': {'result': "It's 90 degrees and sunny."}}, 'role': 'user', 'invocation_id': 'e-65313a8b-8b1c-4dc3-b7af-1a64b9e7d797', 'author': 'root_agent', 'actions': {'state_delta': {}, 'artifact_delta': {}}, 'requested_auth_configs': {}, 'requested_tool_confirmations': {}, 'id': '43427014-c89f-4da4-a583-2d4a43dadf62', 'timestamp': 1762231192.689843}
{'content': {'parts': [{"text": "The weather in Atlanta is 90 degrees and sunny."}], 'role': 'model', 'finish_reason': 'STOP', 'usage_metadata': {'candidates_token_count': 12, 'candidates_tokens_details': [{"modality": "TEXT", "token_count": 12}], 'prompt_tokens_details': [{"modality": "TEXT", "token_count": 233}], 'total_token_count': 183, 'traffic_type': 'ON_DEMAND'}, 'avg_logprobs': -0.1963105003039042, 'invocation_id': 'e-65313a8b-8b1c-4dc3-b7af-1a64b9e7d797', 'author': 'root_agent', 'actions': {'state_delta': {}, 'artifact_delta': {}}, 'requested_auth_configs': {}, 'requested_tool_confirmations': {}, 'id': '5f36fad5-adeb-a489-50746798f216', 'timestamp': 1762231192.81134}
```

Cloud Run - ASP Project Creation



```
> uvx agent-starter-pack create adk-agent-cloudrun-demo

== Google Cloud Agent Starter Pack ==

Welcome to the Agent Starter Pack!
This tool will help you create an end-to-end production-ready AI agent in Google Cloud!

> Please select a agent to get started:
1. adk_base - A base ReAct agent built with Google's Agent Development Kit (ADK)
2. adk_a2a_base - A base ReAct agent built with Google's Agent Development Kit (ADK) and Agent2Agent (A2A) Protocol. #experimental
3. adk_live - Real-time multimodal agent with ADK and Gemini Live API for low-latency voice and video interaction.
4. agentic_rag - ADK RAG agent for document retrieval and Q&A. Includes a data pipeline for ingesting and indexing documents into Vertex AI Search or Vector Search.
5. langgraph_base_react - An agent implementing a base ReAct agent using LangGraph
6. crewai_coding_crew - A multi-agent system implemented with CrewAI created to support coding activities
7. Browse agents from google/adk-samples - Discover additional samples

Enter the number of your template choice (1): 1

> Please select a deployment target:
1. Vertex AI Agent Engine - Vertex AI Managed platform for scalable agent deployments
2. Cloud Run - GCP Serverless container execution

Enter the number of your deployment target choice (1): 2

> Please select a session type:
1. In-memory session - Session data stored in memory - ideal for stateless applications
2. AlloyDB - Use AlloyDB for session management. Comes with terraform resources for deployment.
3. Vertex AI Agent Engine - Managed session service that automatically handles conversation history

Enter the number of your session type choice (1): 1

> Please select a CI/CD runner:
1. Google Cloud Build - Fully managed CI/CD, deeply integrated with GCP for fast, consistent builds and deployments.
2. GitHub Actions - GitHub Actions: CI/CD with secure workload identity federation directly in GitHub.

Enter the number of your CI/CD runner choice (1): 1
```

Cloud Run - ASP Project Structure



```
> uvx agent-starter-pack create adk-agent-cloudrun-demo
== Google Cloud Agent Starter Pack ==
Welcome to the Agent Starter Pack!
This tool will help you create an end-to-end production-ready AI agent in Google Cloud!

> Please select a agent to get started:
1. adk_base - A base ReAct agent built with Google's Agent Development Kit (ADK)
2. adk_a2a_base - A base ReAct agent built with Google's Agent Development Kit (ADK) and Agent2Agent (A2A) Protocol. #experimental
3. adk_live - Real-time multimodal agent with ADK and Gemini Live API for low-latency voice and video interaction.
4. agentic_rag - ADK RAG agent for document retrieval and Q&A. Includes a data pipeline for ingesting and indexing documents into Vertex AI Search or Vector Search.
5. langgraph_base_react - An agent implementing a base ReAct agent using LangGraph
6. crewai_coding_crew - A multi-agent system implemented with CrewAI created to support coding activities
7. Browse agents from google/adk-samples - Discover additional samples

Enter the number of your template choice (1): 1

> Please select a deployment target:
1. Vertex AI Agent Engine - Vertex AI Managed platform for scalable agent deployments
2. Cloud Run - GCP Serverless container execution

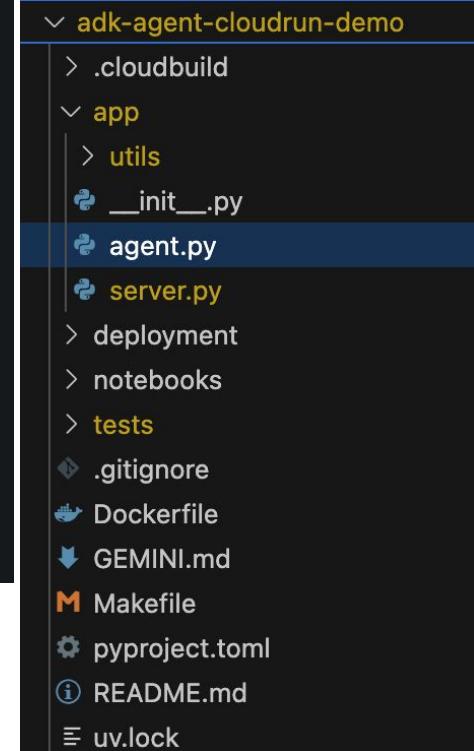
Enter the number of your deployment target choice (1): 2

> Please select a session type:
1. In-memory session - Session data stored in memory - ideal for stateless applications
2. AlloyDB - Use AlloyDB for session management. Comes with terraform resources for deployment.
3. Vertex AI Agent Engine - Managed session service that automatically handles conversation history

Enter the number of your session type choice (1): 1

> Please select a CI/CD runner:
1. Google Cloud Build - Fully managed CI/CD, deeply integrated with GCP for fast, consistent builds and deployments.
2. GitHub Actions - GitHub Actions: CI/CD with secure workload identity federation directly in GitHub.

Enter the number of your CI/CD runner choice (1): 1
```

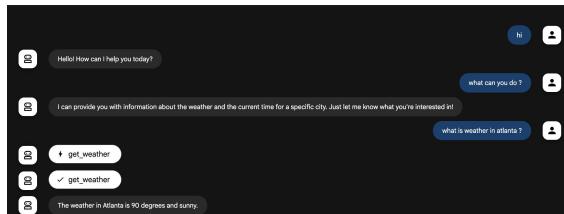


Cloud Run - Deploy and Validation



```
> make deploy
fatal: not a git repository (or any of the parent directories): .git
PROJECT_ID=$(gcloud config get-value project) && \
  gcloud beta run deploy adk-agent-cloudrun-demo \
    --source . \
    --memory "4Gi" \
    --project $PROJECT_ID \
    --region "us-central1" \
    --no-allow-unauthenticated \
    --no-cpu-throttling \
    --labels "created-by=adk" \
    --update-build-env-vars "AGENT_VERSION=0.1.0" \
    --set-env-vars \
    "COMMIT_SHA=" \
    \\\nBuilding using Dockerfile and deploying container to Cloud Run service [adk-agent-cloudrun-demo] in project [vertex-ai-experimental] region [us-central1]
✓ Building and deploying new service... Done.
✓ Validating Service...
✓ Uploading sources...
✓ Building Container... Logs are available at [https://console.cloud.google.com/cloud-build/builds;region=us-central1/dbfd46d5-2098-4b43-8018-73cc1c1dff6f?project=913703049174].
✓ Creating Revision...
✓ Routing traffic...
Done.
Service [adk-agent-cloudrun-demo] revision [adk-agent-cloudrun-demo-00001-z9n] has been deployed and is serving 100 percent of traffic.
Service URL: https://adk-agent-cloudrun-demo-913703049174.us-central1.run.app
```

```
> gcloud run services proxy adk-agent-cloudrun-demo \
  --region us-central1 \
  --project vertex-ai-experimental
Proxying to Cloud Run service [adk-agent-cloudrun-demo] in project [vertex-ai-experimental] region [us-central1]
http://127.0.0.1:8080 proxies to https://adk-agent-cloudrun-demo-rjx3zyr6ua-uc.a.run.app
```



GKE - ADK Automated Deployment



GKE is the Google Cloud managed Kubernetes service. It allows you to deploy and manage containerized applications using Kubernetes.

```
# setting up GCP env. variable
export GOOGLE_CLOUD_PROJECT="vertex-ai-experiment"
export GOOGLE_CLOUD_LOCATION=us-central1
export GOOGLE_GENAI_USE_VERTEXAI=True
export GOOGLE_CLOUD_PROJECT_NUMBER=$(gcloud projects describe --format json
$GOOGLE_CLOUD_PROJECT | jq -r ".projectNumber")

# Enable permissions
gcloud services enable container.googleapis.com \
    artifactregistry.googleapis.com \
    cloudbuild.googleapis.com \
    aiplatform.googleapis.com
```

GKE - Setting up Pre-requisites



Grant roles to compute service account and setting up GKE

```
# Grant Roles to Service Account
ROLES_TO_ASSIGN=(
    "roles/artifactregistry.writer"
    "roles/storage.objectViewer"
    "roles/logging.viewer"
    "roles/logging.logWriter"
)

# create GKE cluster
cloud container clusters create-auto adk-cluster \
--location=$GOOGLE_CLOUD_LOCATION \
--project=$GOOGLE_CLOUD_PROJECT

# Deploy Agents in automated way
adk deploy gke \
--project vertex-ai-experiment \
--cluster_name adk-cluster \
--region us-central1 \
--with_ui \
--log_level info \
~/workshop/adk-gke-demo/
```

ADK - Deploy to GKE



Run ADK deploy gke with GCP project and cluster and ADK repo

```
▶ adk deploy gke \
--project vertex-ai-experiment \
--cluster_name adk-cluster \
--region us-central1 \
--with_ui \
--log_level info \
~/workshop/adk-gke-demo/
/Users/arjunprabhulal/workshop/adk-gke-demo/.venv/lib/python3.14/site-packages/google/cloud/aiplatform/models.py:52:
  f google-cloud-aiplatform. Please upgrade to google-cloud-storage >= 3.0.0.
    from google.cloud.aiplatform.utils import gcs_utils

❶ Starting ADK Agent Deployment to GKE...
-----
Project:      vertex-ai-experiment
Region:       us-central1
Cluster:      adk-cluster
-----

STEP 1: Preparing build environment...
- Using temporary directory: /var/folders/8q/3xqw2gjj54vbtxmvlhz753wr0000gn/T/gke_deploy_src/20251104_071736
- Copying agent source code...
✓ Environment prepared.

STEP 2: Generating deployment files...
- Creating Dockerfile...
✓ Dockerfile generated: /var/folders/8q/3xqw2gjj54vbtxmvlhz753wr0000gn/T/gke_deploy_src/20251104_071736/Dockerfile

STEP 3: Building container image with Cloud Build...
(This may take a few minutes. Raw logs from gcloud will be shown below.)
INFO: Not using ignore file.
Creating temporary archive of 15777 file(s) totalling 498.3 MiB before compression.
```

GKE - ADK Deployment



Under the hood - ADK deploy gke does

- Containerization
- Build Docker image and pushing to gcr container registry
- Generate kubernetes deployment and service yaml manifest
- Applies manifes in GKE cluster

```
FETCHSOURCE
Fetching storage object: gs://vertex-ai-experiment_cloudbuild/source/1762258660.583335-1f5c4b4fc2b74fe4b246d1e6c71b3352.tgz#1762258692236953
Copying gs://vertex-ai-experiment_cloudbuild/source/1762258660.583335-1f5c4b4fc2b74fe4b246d1e6c71b3352.tgz#1762258692236953...
\ [1 files][103.5 MiB/103.5 MiB]
Operation completed over 1 objects/103.5 MiB.

BUILD
Already have image (with digest): gcr.io/cloud-builders/gcb-internal
Step 1/13 : FROM python:3.11-slim
3.11-slim: Pulling from library/python
d7ecded7702a: Pulling fs layer
1ee9c106547f: Pulling fs layer
f002d17b63fe: Pulling fs layer
65868bb001a40: Pulling fs layer
65868bb001a40: Verifying Checksum
65868bb001a40: Download complete
1ee9c106547f: Verifying Checksum
1ee9c106547f: Download complete
f002d17b63fe: Verifying Checksum
f002d17b63fe: Download complete
d7ecded7702a: Verifying Checksum
d7ecded7702a: Download complete
d7ecded7702a: Pull complete
1ee9c106547f: Pull complete
f002d17b63fe: Pull complete
65868bb001a40: Pull complete
Digest: sha256:d4eee051786fd345977c9352635478919b5e595da3bf5a25e9662e5dadcc77157
Status: Downloaded newer image for python:3.11-slim
--> 870925f75741
Step 2/13 : WORKDIR /app
--> Running in fb594f6d7637
Removing intermediate container fb594f6d7637
--> 43542dfcd70
Step 3/13 : RUN adduser --disabled-password --gecos "" myuser
--> Running in 115641fcacae
usermod: no changes
Removing intermediate container 115641fcacae
--> ffac7ab8ef59
Step 4/13 : USER myuser
--> Running in e858fb8ed966
Removing intermediate container e858fb8ed966
--> 06305244709
Step 5/13 : ENV PATH="/home/myuser/.local/bin:$PATH"
--> Running in f776f504e100
Removing intermediate container f776f504e100
--> 4d2b83fb0387
```

GKE - ADK Deployment Complete



Deployment and Service yaml manifest are applied during deployment process

```
latest: digest: sha256:72bc4f77db6497b33a3aaea4831edc9c22d2837dd9266aa046128fa0068c66d5 size: 2206
DONE

INFO: Display format:
  table(
    id,
    createTime.date("%Y-%m-%dT%H:%M:%S%Oz", undefined='-'),
    duration(start=startTime,end=finishTime,precision=0,calendar=false,undefined='-').slice(2:).join(''):label=DURATION,
    build_source(undefined=''):label=SOURCE,
    build_images(undefined=''):label=IMAGES,
    status
  )
ID           CREATE_TIME        DURATION SOURCE                                IMAGES
316a64f8-f57d-4112-89c7-0129d8bc5a0 2025-11-04T12:23:12+00:00 2M6S      gs://vertex-ai-experimentent_cloudbuild/source/1762258963.54857-b15a1273fff4436892a34a338b69b4ee.tgz  gcr.io/vertex-ai-experimentent/adk-de
fault-service-name (+1 more) SUCCESS
✓ Container image built and pushed successfully.
- Creating Kubernetes deployment.yaml...
✓ Kubernetes deployment manifest generated: /var/folders/8q/3xqw2gjj54vbttxmv1hz753wr0000gn/T/gke_deploy_src/20251104_072239/deployment.yaml

STEP 4: Applying deployment to GKE cluster...
- Getting cluster credentials...
Fetching cluster endpoint and auth data.
Kubeconfig entry generated for adk-cluster.
- Applying Kubernetes manifest...
- The following resources were applied to the cluster:
- deployment.apps/adk-default-service-name created
- service/adk-default-service-name created

STEP 5: Cleaning up...
- Removing temporary directory: /var/folders/8q/3xqw2gjj54vbttxmv1hz753wr0000gn/T/gke_deploy_src/20251104_072239

🎉 Deployment to GKE finished successfully!
```

```
▶ k get pod
NAME                               READY   STATUS    RESTARTS   AGE
adk-default-service-name-7697989d4d-chnz2  1/1     Running   0          29m

▶ k get svc
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
adk-default-service-name  LoadBalancer  34.118.235.100  34.31.172.147  80:31037/TCP  136m
kubernetes        ClusterIP  34.118.224.1    <none>        443/TCP    8h
```

GKE - ADK Demo



Not Secure 34.31.172.147/dev-ui/?app=adk-gke-demo&session=e47aa715-9039-48c4-8b93-...

Agent Development Kit

SESSION e47aa715-9039-48c4-8b93-85a12a34677e

adk-gke-demo

Trace Events State Artifacts Sessions

I can retrieve the capital city of a given country.

what is capital of France ?

⚡ get_capital_city

✓ get_capital_city

Paris

What can you do ?

what is capital of France ?

Choosing the Right Deployment for AI Agents



Vertex AI Agent Engine - For Managed Agent Runtimes

- Fully managed runtime for AI agents — handles context, memory, and scaling.
- Focus on agent logic, not infrastructure.
- Built-in memory, evaluation, and Example Store integrations.
- Secure sandbox for code execution via Code Execution Tool.

Cloud Run - For Serverless, Stateless Agents

- Ideal for stateless, event-driven agents or APIs.
- Auto-scales to zero for cost-efficient, bursty workloads.
- Simple, fully managed serverless platform with minimal ops overhead.

GKE (Google Kubernetes Engine) - For Complex or Stateful Agents

- Best for custom infra / networking or stateful workloads.
- Full control over Kubernetes clusters, storage, and deployments.
- Fits large teams with mature DevOps and CI/CD pipelines.
- Supports hybrid microservice + agent architectures.



4. RAG agent with ADK and ASP

Vertex AI Rag Engine - Deploy to Agent Engine

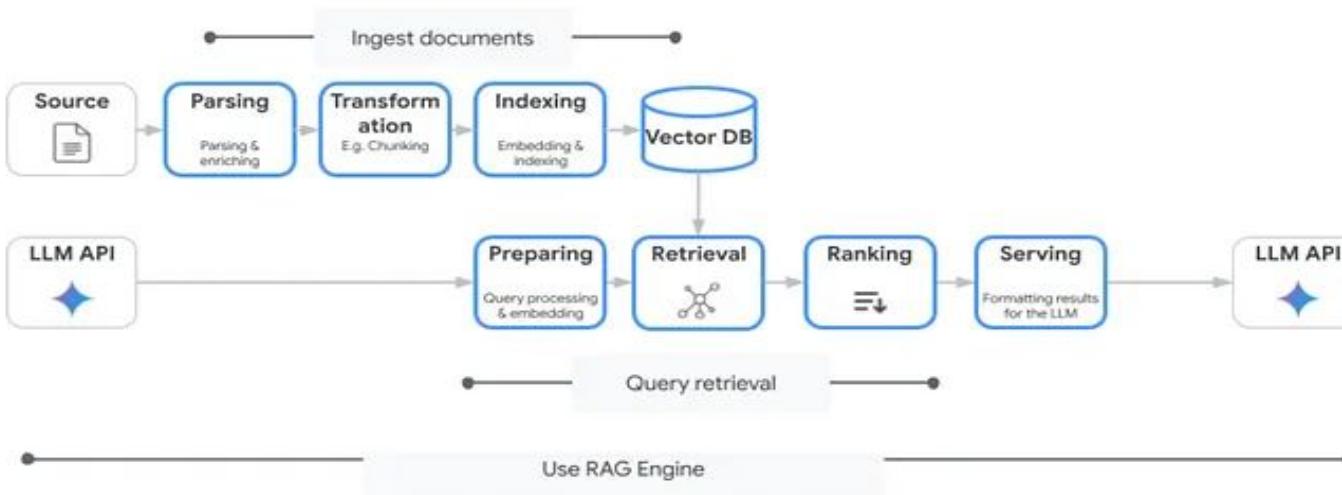
Vertex AI Search - Deploy to CloudRun

Vertex AI Rag Engine



Vertex AI RAG Engine is part of Google Cloud Vertex AI, built to enable Retrieval-Augmented Generation (RAG) workflows.

RAG lets combine LLM intelligence with **real-time data retrieval**, so responses are accurate, grounded, and enterprise-ready.



Vertex AI Rag Engine - Features



Managed Vector Store : Built-in storage for document embeddings.

Seamless Gemini Integration : Works natively with Gemini models.

Real-Time Retrieval : Fetches context instantly during LLM generation.

Enterprise-Grade Reliability : Powered by Google Cloud Spanner.

Tiers:

Basic: 100 processing units

Scaled: 1000 processing units for production workloads

Vertex AI Rag Engine - RAG Project creation



Install ASP for Agentic RAG using RAG Engine

```
#Authenticate GCP login  
gcloud auth login  
  
#Set default GCP Project ID  
gcloud config set project YOUR_PROJECT_ID  
  
#Agent Starter Pack using rag Template identifier  
uvx agent-starter-pack create rag-agent-engine-demo -a adk@rag
```

“-a” flag specifies the Template identifier to use.

Vertex AI Rag Engine - RAG Project Structure



Project creation using ADK RAG template

```
> uvx agent-starter-pack create adk-rag-agent-engine-demo -a adk@rag
== Welcome to google/adk-samples! ✨ ==
Powered by Google Cloud - Agent Starter Pack

This tool will help you create an end-to-end production-ready AI agent in Google Cloud!

> Fetching template: python/agents/rag
🔒 Remote template requires agent-starter-pack version 0.19.1
👉 Switching to version 0.19.1...
Installed 60 packages in 165ms
✅ Using version-locked template

> Please select a deployment target:
1. Vertex AI Agent Engine - Vertex AI Managed platform for scalable agent deployments
2. Cloud Run - GCP Serverless container execution

Enter the number of your deployment target choice (1): 1

> Please select a CI/CD runner:
1. Google Cloud Build - Fully managed CI/CD, deeply integrated with GCP for fast, consistent builds and deployments.
2. GitHub Actions - GitHub Actions: CI/CD with secure workload identity federation directly in GitHub.

Enter the number of your CI/CD runner choice (1): 1

Enter desired GCP region (Gemini uses global endpoint by default) (us-central1):
Gemini uses global endpoint by default

> You are logged in with account: 'arjun.prabhulal@gmail.com'
> You are using project: 'vertex-ai-experiment'
> Do you want to continue? (The CLI will check if Vertex AI is enabled in this project) [Y/skip/edit] (Y): Y
> Successfully configured project: vertex-ai-experiment
> Testing GCP and Vertex AI Connection...
> ✓ Successfully verified connection to Vertex AI in project vertex-ai-experiment

> 🤗 Done. Execute the following command to get started:

> Success! Your agent project is ready.

📘 Project README: cat adk-rag-agent-engine-demo/README.md
Online Development Guide: https://goole/asp-dev

📝 To get started, run the following command:
cd adk-rag-agent-engine-demo && make install && make playground
```

The screenshot shows a file explorer interface with the following directory structure:

- adk-rag-agent-engine-demo (highlighted)
- .cloudbuild
- deployment
- eval
- notebooks
- rag
- tests
- .env.example
- .gitignore
- deployment_metadata.json
- GEMINI.md
- Makefile
- pyproject.toml
- RAG_architecture.png
- RAG_workflow.png
- README.md
- starter_pack_README.md
- uv.lock

Vertex AI Rag Engine - Corpus Creation



RAG Engine

Create corpus

Configure RAG Engine

Vertex AI RAG Engine is a data framework for developing context-augmented large language model (LLM) applications. Context augmentation occurs when you apply an LLM to your data. This implements retrieval-augmented generation (RAG).

[Learn more ↗](#)

Region

us-central1 (Iowa)



Name	Status	Description	Vector database
rag-corpus	Ready	rag-corpus containing LLM PDF	Ragmanaged Vector Store
rag-agent-corpus	Ready	LLM PDFs	Ragmanaged Vector Store

Vertex AI Rag Engine - Prerequisites



Grant Agent Engine service account permission to query the RAG corpus

```
## Setting Up Permissions

### Step 1: Create Custom IAM Role

gcloud iam roles create ragCorpusQueryRole \
--project=your-project-id \
--title="RAG Corpus Query Role" \
--permissions="aiplatform.ragCorpora.query"

### Step 2: Grant Role to Service Account

PROJECT_NUMBER=$(gcloud projects describe your-project-id --format="value(projectNumber)")

gcloud projects add-iam-policy-binding your-project-id \
--member="serviceAccount:@gcp-sa-aiplatform-re.iam.gserviceaccount.com">service-${PROJECT_NUMBER}@gcp-sa-aiplatform-re.iam.gserviceaccount.com" \
--role="projects/your-project-id/roles/ragCorpusQueryRole"
```

Vertex AI Rag Engine - Deploy to Agent Engine



make deploy - Deploys RAG agent to Agent Engine

```
# make deploy
# Export dependencies to requirements file using uv export.
(uv export --no-hashes --no-header --no-dev --no-emit-project --no-annotate > .requirements.txt 2>/dev/null || \
uv export --no-hashes --no-header --no-dev --no-emit-project > .requirements.txt) && \
uv run -m rag.agent_engine_app
INFO:root:Bucket vertex-ai-experiment-agent-engine already exists
INFO:root:Bucket vertex-ai-experiment-agent-engine already exists

[ INFO ] DEPLOYING AGENT TO VERTEX AI AGENT ENGINE [ INFO ]
```

WARNING:google_genai._api_client:Both GOOGLE_API_KEY and GEMINI_API_KEY are set. Using GOOGLE_API_KEY.
INFO:google_genai._api_client:The user provided project/location will take precedence over the Vertex AI API key from the environment variable.
INFO:adk_rag_agent: config: {agent: <_main_.AgentEngineApp object at 0x118bddd90>, 'config': AgentEngineConfig(
description='Answer questions related to Vertex AI documentation using Vertex AI RAG Engine',
display_name='adk-rag-agent-engine-demo',
env_vars={
 'NUM_WORKERS': '1'
},
extra_packages=[
 './rag',
],
gcs_dir_name='adk-rag-agent-engine-demo',
labels={},
requirements=[
 'absolutify==0.3.1',
 'aioboto3==1.11.1',
 'aiohttp==3.13.0',
 'aionetworking==1.4.0',
 'aiosignal==0.21.0',
 'aiosqlite==0.21.0',
 ... 164 more items ...>
,
staging_bucket='gs://vertex-ai-experiment-agent-engine'
)
INFO:httplibx:HTTP Request: GET https://us-central1-aiplatform.googleapis.com/v1beta1/projects/vertex-ai-experiment/locations/us-central1/reasoningEngines "HTTP/1.1 200 OK"
INFO:root:
[INFO] Creating new agent: adk-rag-agent-engine-demo

```
INFO:vertexai_genai.agentengines:Using bucket vertex-ai-experiment-agent-engine
INFO:vertexai_genai.agentengines:Wrote to gs://vertex-ai-experiment-agent-engine/adk-rag-agent-engine-demo/agent_engine.pkl
INFO:vertexai_genai.agentengines:Writing to gs://vertex-ai-experiment-agent-engine/adk-rag-agent-engine-demo/requirements.txt
INFO:vertexai_genai.agentengines:Creating in-memory tarfile of extra_packages
INFO:vertexai_genai.agentengines:Writing to gs://vertex-ai-experiment-agent-engine/adk-rag-agent-engine-demo/dependencies.tar.gz
INFO:httplibx:HTTP Request: POST https://us-central1-aiplatform.googleapis.com/v1beta1/projects/vertex-ai-experiment/locations/us-central1/reasoningEngines "HTTP/1.1 200 OK"
INFO:vertexai_genai.agentengines:View progress and logs at https://console.cloud.google.com/logs/query?project=vertex-ai-experiment
```

```
INFO:httplibx:HTTP Request: GET https://us-central1-aiplatform.googleapis.com/v1beta1/projects/913703049174/locations/us-central1/reasoningEngines/6538931989825716224/operations/7371216610779987968 "HTTP/1.1 200 OK"
INFO:vertexai_genai.agentengines:Agent Engine created. To use it in another session:
INFO:vertexai_genai.agentengines:agent_engine=<client.agent_engines.get('projects/913703049174/locations/us-central1/reasoningEngines/6538931989825716224')>
INFO:root:Agent Engine ID written to deployment_metadata.json

[ INFO ] Deployment successful! Test your agent: notebooks/adk_app_testing.ipynb [ INFO ]
Service Account: service-913703049174@gcp-sa-aiplatform-re.iam.gserviceaccount.com [ INFO ]

[ INFO ] View in console: https://console.cloud.google.com/vertex-ai/agents/locations/us-central1/agent-engines/6538931989825716224?project=vertex-ai-experiment
```

Vertex AI Rag Engine - Demo



Agent Engine

💡 A more intuitive and powerful Agent Engine UI is coming. We're launching a major UI update in November to help you better manage your agents.

View and manage agent instances in this Cloud project.

[Develop agent](#) [Deployment guide](#)

Region —
us-central1 (Iowa)

Filter Enter property name or value

Name	Resource name	Created	Framework	Description
adk-rag-agent-engine-demo	.../6538931989825716224	Nov 4, 2025, 1:01:53 AM	google-adk	Answer questions related to Vertex AI documentation using Vertex AI RAG Engine

Detailed Values:

Event #1

Response: A Mixture of Experts (MoE) is an architecture that combines multiple specialized sub-models, known as "experts," to enhance overall performance, particularly for complex tasks. It differs from simple ensemble learning by using a "gating network" to intelligently route different parts of the input to the most relevant experts, rather than just aggregating all expert predictions.
Confidence Scores: 0.9869, 0.9172, 0.8471, 0.6053, 0.6359, 0.9863, 0.9752, 0.6736, 0.6176, 0.9595, 0.6093, 0.9935, 0.6526, 0.9635, 0.6905, 0.9821, 0.9898
Avg Confidence: 0.8286
Tokens: 494/263/820
Log Probabilities: -0.164920
Finish Reason: STOP

FINAL RESPONSE

Answer to 'What is Mixture of Experts (MoE)?'

A Mixture of Experts (MoE) is an architecture that combines multiple specialized sub-models, known as "experts," to enhance overall performance, particularly for complex tasks. It differs from simple ensemble learning by using a "gating network" to intelligently route different parts of the input to the most relevant experts, rather than just aggregating all expert predictions.

The main components of an MoE include:

- * **Experts**: These are individual sub-models, typically transformer-based architectures in the context of large language models, designed to handle specific subsets of input data or particular tasks.
- * **Gating Network (Router)**: This crucial component learns to direct the input to the appropriate expert(s). It analyzes the input and generates a probability distribution indicating each expert's contribution to the final prediction. The gating network is usually a neural network.
- * **Combination Mechanism**: This component combines the outputs of the experts, weighted by the probabilities from the gating network, to produce the final prediction, often through a weighted average.

This approach allows experts to specialize, improving performance and potentially reducing computational cost through "sparse activation," where only a subset of experts is activated for any given input.

Citations:

- 1) foundational-large-language-models-text-generation.pdf

✓ All displays executed successfully!

Vertex AI Search



Vertex AI Search: Google-grade Search and Answers + Gemini powering your website, app or your enterprise

Specialized solutions for verticals	Retail Product search and recommendations optimized for revenue	Media Media search and recommendations optimized f. watchtime	Healthcare Patient EHR search optimized for quality and compliance	Website Site search and recommendations optimized for CTR	Intranet Internal content search and assistance built for privacy & productivity	Custom Build Agents, Gen AI Experiences that use search in Agent Builder
OOB solution for Searching your data and the web	Vertex AI Search & Recommendations					
Foundational Building blocks and Platform for DIY implementations	Out-of-the-box Performance Set up in minutes Advanced crawling, parsing, doc understanding Search pages / pdfs / images / videos Leverages Web signals and Knowledge Graph Semantic Search + Hybrid search Spelling correction and autocomplete Multimodal Embeddings Reverse image search	Tuning & Customization Events-based reranking Events-based autocomplete Synonyms & Redirects Relevance filtering Extract length customization Boost/Bury/Filters/Facets Search adaptors Bring your own embeddings	Generative AI Grounded Answers and Generation Blending multiple sources in generated answer Sources citations and corroborations Conversational AI AI planning, multi source and multi step search Advanced self revision Customizable AI Natural Language to Filters			
Connectors to enterprise sources and databases	Document AI Documents understanding, processing, extracting	RAG APIs Embeddings, Ranking, Check grounding	Vector Search Google Scale and performance Vector Search for your data	Grounded Gemini Generative AI grounded on Google Search and your data		

Your Data Sources +
User data Activity



Vertex AI Search - Overview



Vertex AI Search is Google Cloud enterprise-grade retrieval and search service that lets us build AI-powered search and retrieval systems across private data such as documents, websites, databases, or cloud storage without managing any search infrastructure.

Vertex AI Search part of the Vertex AI Agent Platform, and often used inside RAG (Retrieval-Augmented Generation) pipelines or chatbots to fetch relevant information before LLM reasoning.

Vertex AI Search - Features



Feature	Description
Unified Search API	Query across structured & unstructured data (PDFs, docs, websites, BigQuery, etc.) via a single API.
Vector & keyword retrieval	Combines semantic vector search (embedding-based) with keyword relevance for best results.
Automatic indexing	Handles ingestion, chunking, and embedding of your content into a data store , no manual pipeline needed.
LLM Integration	Connects directly to Vertex AI Generative models or Agent Engine for context-aware responses.
Access Control	Integrates with IAM, Cloud KMS, and ACLs for secure search over private enterprise data.
Multimodal & multi-source	Supports documents, web data, APIs, and databases within one index.
Low-latency retrieval	Fully managed service optimized for millisecond-scale search.

Vertex AI Search - Agentic RAG ASP Template



```
> uvx agent-starter-pack create adk-vertexsearch-cloudrun

==== Google Cloud Agent Starter Pack =====
Welcome to the Agent Starter Pack!
This tool will help you create an end-to-end production-ready AI agent in Google Cloud!

> Please select a agent to get started:
1. adk_base - A base ReAct agent built with Google's Agent Development Kit (ADK)
2. adk_a2a_base - A base ReAct agent built with Google's Agent Development Kit (ADK) and Agent2Agent (A2A) Protocol. #experimental
3. adk_live - Real-time multimodal agent with ADK and Gemini Live API for low-latency voice and video interaction.
4. agentic_rag - ADK RAG agent for document retrieval and Q&A. Includes a data pipeline for ingesting and indexing documents into Vertex AI Search or Vector Search.
5. langgraph_base_react - An agent implementing a base ReAct agent using LangGraph
6. crewai_coding_crew - A multi-agent system implemented with CrewAI created to support coding activities
7. Browse agents from google/adk-samples - Discover additional samples

Enter the number of your template choice (1): 4

> This agent includes a data ingestion pipeline.
> Please select a datastore type for your data:
1. Vertex AI Search - Managed, serverless document store that enables Google-quality search and RAG for generative AI.
2. Vertex AI Vector Search - Scalable vector search engine for building search, recommendation systems, and generative AI applications. Based on ScaNN algorithm.

Enter the number of your choice [1/2] (1): 1

> Please select a deployment target:
1. Vertex AI Agent Engine - Vertex AI Managed platform for scalable agent deployments
2. Cloud Run - GCP Serverless container execution

Enter the number of your deployment target choice (1): 2

> Please select a session type:
1. In-memory session - Session data stored in memory - ideal for stateless applications
2. AlloyDB - Use AlloyDB for session management. Comes with terraform resources for deployment.
3. Vertex AI Agent Engine - Managed session service that automatically handles conversation history

Enter the number of your session type choice (1): 1

> Please select a CI/CD runner:
1. Google Cloud Build - Fully managed CI/CD, deeply integrated with GCP for fast, consistent builds and deployments.
2. GitHub Actions - GitHub Actions: CI/CD with secure workload identity federation directly in GitHub.

Enter the number of your CI/CD runner choice (1): 1

Enter desired GCP region (Gemini uses global endpoint by default) (us-central1): 
```

Vertex AI Search - Agentic RAG ASP Template



```
Enter desired GCP region (Gemini uses global endpoint by default) (us-central1):
```

```
> You are logged in with account: 'arjun.prabhulal@gmail.com'  
> You are using project: 'vertex-ai-experiment'  
> Do you want to continue? (The CLI will check if Vertex AI is enabled in this project) [Y/skip/edit] (Y): Y  
> Successfully configured project: vertex-ai-experiment  
> Testing GCP and Vertex AI Connection...  
> ✓ Successfully verified connection to Vertex AI in project vertex-ai-experiment
```

===== DATA INGESTION SETUP =====

This agent uses a datastore for grounded responses.

The agent will work without data, but for optimal results:

1. Set up dev environment:

```
  export PROJECT_ID=vertex-ai-experiment && cd adk-vertexsearch-cloudrun && make setup-dev-env
```

See deployment/README.md for more info

2. Run the data ingestion pipeline:

```
  export PROJECT_ID=vertex-ai-experiment && cd adk-vertexsearch-cloudrun && make data-ingestion
```

See data_ingestion/README.md for more info

```
>  Done. Execute the following command to get started:
```

```
> Success! Your agent project is ready.
```

```
💡 Project README: cat adk-vertexsearch-cloudrun/README.md  
Online Development Guide: https://goo.gle/asp-dev
```

```
🚀 To get started, run the following command:
```

```
cd adk-vertexsearch-cloudrun && make install && make playground
```

Vertex AI Search - DataStore Setup



Data stores

+ Create data store

Feedback on AI Applications

Learn

Filter Enter property name or value					?	☰
Name	Connected apps	Created ↓	ID	Location		
rag-datastore	N/A	Nov 1, 2025	rag-datastore	global	⋮	

Data stores > rag-datastore > Data

Feedback on AI Applications

Learn

[rag-datastore](#) [Edit](#)

Data store ID	rag-datastore
Type	Unstructured data
Serving state	Enabled Edit
Region	global
Language	N/A Edit
Connected apps	N/A
Datastore size	6.81 MiB ?
Number of documents	1
Last document import	Nov 1, 2025, 8:01:55 PM View details
Exclude from generative AI features	False Edit

Documents Events Activity Processing Config [Preview](#)

[+ Import data](#) [Purge Data](#)

ID	URI	Index Status	Actions
9b8dd75fb63dda9cf7ea6fcf2cc7f51b	gs://agents-gemini-embedding/Foundational-Large-Language-Models-Text-Generation.pdf ?	✓ Indexed: Nov 1, 2025, 8:13:16 PM, America/New_York	

Vertex AI Search - Cloud Run Deploy



`make deploy` - Install adk application to cloud Run using in memory (stateless)

```
> make deploy
fatal: not a git repository (or any of the parent directories): .git
PROJECT_ID=$(gcloud config get-value project) && \
  gcloud beta run deploy adk-vertexsearch-cloudrun \
    --source . \
    --memory "4Gi" \
    --project $PROJECT_ID \
    --region "us-central1" \
    --no-allow-unauthenticated \
    --no-cpu-throttling \
    --labels "created-by=adk" \
    --update-build-env-vars "AGENT_VERSION=0.1.0" \
    --set-env-vars \
    "COMMIT_SHA=,DATA_STORE_ID=rag-datastore,DATA_STORE_REGION=global" \
    \"

Building using Dockerfile and deploying container to Cloud Run service [adk-vertexsearch-cloudrun] in project [vertex-ai-experimental] region [us-central1]
✓ Building and deploying new service... Done.
  ✓ Validating Service...
  ✓ Uploading sources...
  ✓ Building Container... Logs are available at [https://console.cloud.google.com/cloud-build/builds;region=us-central1/36ba0d50-83d1-4f77-9bd2-0ac99b916d98?project=913703049174].
  ✓ Creating Revision...
  ✓ Routing traffic...
Done.
Service [adk-vertexsearch-cloudrun] revision [adk-vertexsearch-cloudrun-00001-n97] has been deployed and is serving 100 percent of traffic.
Service URL: https://adk-vertexsearch-cloudrun-913703049174.us-central1.run.app
```

Use CloudRun proxy for IAM authenticated or use make IAP=True

```
> gcloud run services proxy adk-vertexsearch-cloudrun \
  --region us-central1 \
  --project vertex-ai-experimental
Proxying to Cloud Run service [adk-vertexsearch-cloudrun] in project [vertex-ai-experimental] region [us-central1]
http://127.0.0.1:8080 proxies to https://adk-vertexsearch-cloudrun-rjx3zyr6ua-uc.a.run.app
```

Vertex AI Search - Demo



Hi there! How can I help you today?

hi

I can help you with a variety of tasks, including:

- **Answering your questions:** I can provide information on a wide range of topics.
- **Finding information:** I can use tools to search for information and provide you with relevant details.
- **Generating text:** I can help you write different kinds of creative content.
- **Running code snippets:** I can execute Python code to assist with calculations or data processing.

Essentially, I'm here to provide information and help you with your queries! What would you like to do?

what can you do ?

What is Mixture of Experts ?

A Mixture of Experts (MoE) is an architecture that combines multiple specialized sub-models, known as "experts," to enhance overall performance, particularly for complex tasks. It functions as a form of ensemble learning but differs in its approach: instead of simply averaging the predictions of all experts, it intelligently routes different parts of the input to specific experts. This specialization allows each expert to become proficient in a particular sub-domain or aspect of the data.

Here's a breakdown of the main components of an MoE:

- **Experts:** These are individual sub-models designed to handle specific subsets of input data or particular tasks. In the context of large language models, these are typically transformer-based architectures.
- **Gating Network (Router):** This crucial component learns to route the input to the appropriate expert or experts. It takes the input and generates a probability distribution over the experts, determining each expert's contribution to the final prediction. The gating network is usually also a neural network.
- **Combination Mechanism:** This mechanism combines the outputs from the experts, weighting them by the probabilities provided by the gating network, to produce the final prediction. A common method for this is a weighted average.

The process involves both the experts and the gating network receiving the input. Each expert processes the input and generates its output, while the gating network analyzes the input to produce a probability distribution over the experts, indicating their contribution. These probabilities then weight the experts' outputs, and the weighted combination forms the final prediction. This architecture allows different experts to specialize in handling specific data types or sub-tasks, improving performance and potentially reducing computational costs through "sparse activation," where only a subset of experts is activated for any given input.

Vertex AI Search vs RAG Engine



Vertex AI Search = Google Grade , Enterprise managed semantic search Q & A

Vertex AI RAG Engine = Customizable Vector DB, developer-driven RAG infrastructure

Features	Vertex AI Search	Vertex AI RAG Engine
Purpose	Pre-built enterprise semantic search solution	Developer-centric framework for custom RAG pipelines
Use Case	Search & Q&A over documents	Real-time retrieval-augmented generation with Gemini models
Data Storage	Managed Search Datastore (auto-indexed, text-optimized)	Managed Vector Store (embeddings stored in Spanner DB)
Integration Level	No-code / low-code via Vertex Console & API	Deep LLM integration via ADK & function calling
Customization	Limited schema & ranking controls	Full control over retrieval, chunking, ranking, and grounding
Model Coupling	Works with Gemini for retrieval & Q&A	Built for Gemini RAG — injects context into model prompts
Best For	Fast setup of enterprise search and knowledge assistants	Developers building custom LLM apps with dynamic retrieval



Q & A

Feedback