Google Cloud tools - Agent Development Kit

Source URL: https://google.github.io/adk-docs/tools/google-cloud-tools/

Google Cloud Tools¶

Currently supported in Python

Google Cloud tools make it easier to connect your agents to Google Cloud's products and services. With just a few lines of code you can use these tools to connect your agents with:

- **Any custom APIs** that developers host in Apigee.
- **100s** of **prebuilt connectors** to enterprise systems such as Salesforce, Workday, and SAP.
- **Automation workflows** built using application integration.
- **Databases** such as Spanner, AlloyDB, Postgres and more using the MCP Toolbox for databases.

Google Cloud Tools

Apigee API Hub Tools¶

ApiHubToolset lets you turn any documented API from Apigee API hub into a tool with a few lines of code. This section shows you the step by step instructions including setting up authentication for a secure connection to your APIs.

Prerequisites

1. Install ADK
2. Install the Google Cloud CLI.
3. Apigee API hub instance with documented (i.e. OpenAPI spec) APIs
4. Set up your project structure and create required files

```
project_root_folder
 |
`-- my_agent
    |-- .env
    |-- __init__.py
    |-- agent.py
    `__ tool.py
```

Create an API Hub Toolset¶

Note: This tutorial includes an agent creation. If you already have an agent, you only need to follow a subset of these steps.

1. Get your access token, so that APIHubToolset can fetch spec from API Hub API. In your terminal run the following command

``` gcloud auth print-access-token # Prints your access token like 'ya29....'

`` 2. Ensure that the account used has the required permissions. You can use the pre-defined role` roles/apihub.viewer` or assign the following permissions:

1. **apihub.specs.get (required)**
2. apihub.apis.get (optional)
3. apihub.apis.list (optional)
4. apihub.versions.get (optional)
5. apihub.versions.list (optional)
6. apihub.specs.list (optional)
7. Create a tool with `APIHubToolset`. Add the below to `tools.py`

If your API requires authentication, you must configure authentication for the tool. The following code sample demonstrates how to configure an API key. ADK supports token based auth (API Key, Bearer token), service account, and OpenID Connect. We will soon add support for various OAuth2 flows.

``` from google.adk.tools.openapi_tool.auth.auth_helpers import token_to_scheme_credential from google.adk.tools.apihub_tool.apihub_toolset import APIHubToolset

# Provide authentication for your APIs. Not required if your APIs don't required authentication. auth_scheme, auth_credential =

token_to_scheme_credential( "apikey", "query", "apikey", apikey_credential_str )

sample_toolset_with_auth = APIHubToolset( name="apihub-sample-tool", description="Sample Tool", access_token="...", # Copy your access token generated in step 1 apihub_resource_name="...", # API Hub resource name auth_scheme=auth_scheme, auth_credential=auth_credential, )

```
```

For production deployment we recommend using a service account instead of an access token. In the code snippet above, use `service_account_json=service_account_cred_json_str` and provide your security account credentials instead of the token.

For apihub_resource_name, if you know the specific ID of the OpenAPI Spec being used for your API, use `projects/my-project-id/locations/us-west1/apis/my-api-id/versions/version-id/specs/spec-id`. If you would like the Toolset to automatically pull the first available spec from the API, use `projects/my-project-id/locations/us-west1/apis/my-api-id` 4. Create your agent file Agent.py and add the created tools to your agent definition:

``` from google.adk.agents.llm_agent import LlmAgent from .tools import sample_toolset

root_agent = LlmAgent( model='gemini-2.0-flash', name='enterprise_assistant', instruction='Help user, leverage the tools you have access to', tools=sample_toolset.get_tools(), )

`` 5. Configure your` **init**.py` to expose your agent

``` from . import agent

``` 6. Start the Google ADK Web UI and try your agent:

`` # make sure to run` adk web` from your project_root_folder adk web

```
```

Then go to http://localhost:8000 to try your agent from the Web UI.
```

Application Integration Tools¶

With **ApplicationIntegrationToolset** you can seamlessly give your agents a secure and governed to enterprise applications using Integration Connector’s 100+ pre-built connectors for systems like Salesforce, ServiceNow, JIRA, SAP, and more. Support for both on-prem and SaaS applications. In addition you can turn your existing Application Integration process automations into agentic workflows by providing application integration workflows as tools to your ADK agents.

Prerequisites

1. Install ADK
2. An existing Application Integration workflow or Integrations Connector connection you want to use with your agent
3. To use tool with default credentials: have Google Cloud CLI installed. See installation guide.

Run:

```
 gcloud config set project <project-id>
gcloud auth application-default login
gcloud auth application-default set-quota-project <project-id>
```

1. Set up your project structure and create required files

`` `project_root_folder |-- .env` -- my_agent |-- **init**.py |-- agent.py `__ tools.py

```

When running the agent, make sure to run adk web in project_root_folder

## Use Integration Connectors¶

Connect your agent to enterprise applications using Integration Connectors.
```

Prerequisites

1. To use a connector from Integration Connectors, you need to provision Application Integration in the same region as your connection by clicking on "QUICK SETUP" button.

Google Cloud Tools

1. Go to Connection Tool template from the template library and click on "USE TEMPLATE" button.

Google Cloud Tools 2. Fill the Integration Name as **ExecuteConnection** (It is mandatory to use this integration name only) and select the region same as the connection region. Click on "CREATE". 3. Publish the integration by using the "PUBLISH" button on the Application Integration Editor.

Google Cloud Tools

Steps:

1. Create a tool with `ApplicationIntegrationToolset` within your `tools.py` file

``` from google.adk.tools.application_integration_tool.application_integration_toolset import ApplicationIntegrationToolset

connector_tool = ApplicationIntegrationToolset( project="test-project", # TODO: replace with GCP project of the connection location="us-central1", #TODO: replace with location of the connection connection="test-connection", #TODO: replace with connection name entity_operations={"Entity_One": ["LIST","CREATE"], "Entity_Two": []},#empty list for actions means all operations on the entity are supported. actions=["action1"], #TODO: replace with actions service_account_credentials='{...}', # optional. Stringified json for service account key tool_name_prefix="tool_prefix2", tool_instructions="..." )

```

Note:

- You can provide service account to be used instead of using default credentials by generating Service Account Key and providing right

  Application Integration and Integration Connector IAM roles to the service account.
- To find the list of supported entities and actions for a connection, use the connectors apis: listActions or listEntityTypes

`ApplicationIntegrationToolset` now also supports providing auth_scheme and auth_credential for dynamic OAuth2 authentication for Integration Connectors. To use it, create a tool similar to this within your `tools.py` file:

``` from google.adk.tools.application_integration_tool.application_integration_toolset import ApplicationIntegrationToolset from google.adk.tools.openapi_tool.auth.auth_helpers import dict_to_auth_scheme from google.adk.auth import AuthCredential from google.adk.auth import AuthCredentialTypes from google.adk.auth import OAuth2Auth

oauth2_data_google_cloud = { "type": "oauth2", "flows": { "authorizationCode": { "authorizationUrl": "https://accounts.google.com/o/oauth2/auth", "tokenUrl": "https://oauth2.googleapis.com/token", "scopes": { "https://www.googleapis.com/auth/cloud-platform": ( "View and manage your data across Google Cloud Platform" " services" ), "https://www.googleapis.com/auth/calendar.readonly": "View your calendars" }, } }, }

oauth_scheme = dict_to_auth_scheme(oauth2_data_google_cloud)

auth_credential = AuthCredential( auth_type=AuthCredentialTypes.OAUTH2, oauth2=OAuth2Auth( client_id="...", #TODO: replace with client_id client_secret="...", #TODO: replace with client_secret ), )

connector_tool = ApplicationIntegrationToolset( project="test-project", # TODO: replace with GCP project of the connection location="us-central1", #TODO: replace with location of the connection connection="test-connection", #TODO: replace with connection name entity_operations={"Entity_One": ["LIST","CREATE"], "Entity_Two": []},#empty list for actions means all operations on the entity are supported. actions=["GET_calendars/%7BcalendarId%7D/events"], #TODO: replace with actions. this one is for list events service_account_credentials='{...}', # optional. Stringified json for service account key tool_name_prefix="tool_prefix2", tool_instructions="...", auth_scheme=oauth_scheme, auth_credential=auth_credential )
```

`` 2. Add the tool to your agent. Update your agent.py` file

``` from google.adk.agents.llm_agent import LlmAgent from .tools import connector_tool

root_agent = LlmAgent( model='gemini-2.0-flash', name='connector_agent', instruction="Help user, leverage the tools you have access to", tools=[connector_tool], )

`` 3. Configure your **init**.py` to expose your agent

``` from . import agent

``` 4. Start the Google ADK Web UI and try your agent.

`` # make sure to run adk web` from your project_root_folder adk web

```

Then go to http://localhost:8000, and choose my_agent agent (same as the agent folder name)

Use App Integration Workflows¶

Use existing Application Integration workflow as a tool for your agent or create a new one.

Steps:

1. Create a tool with `ApplicationIntegrationToolset` within your `tools.py` file

``` integration_tool = ApplicationIntegrationToolset( project="test-project", # TODO: replace with GCP project of the connection location="us-central1", #TODO: replace with location of the connection integration="test-integration", #TODO: replace with integration name triggers=["api_trigger/test_trigger"],#TODO: replace with trigger id(s). Empty list would mean all api triggers in the integration to be considered. service_account_credentials='{...}', #optional. Stringified json for service account key tool_name_prefix="tool_prefix1", tool_instructions="..." )

```

Note: You can provide service account to be used instead of using default credentials by generating Service Account Key and providing right Application Integration and Integration Connector IAM roles to the service account. 2. Add the tool to your agent. Update your `agent.py` file

``` from google.adk.agents.llm_agent import LlmAgent from .tools import integration_tool, connector_tool

root_agent = LlmAgent( model='gemini-2.0-flash', name='integration_agent', instruction="Help user, leverage the tools you have access to", tools=[integration_tool], )

`` 3. Configure your`__init__.py` to expose your agent

``` from . import agent

``` 4. Start the Google ADK Web UI and try your agent.

`` # make sure to run`adk web` from your project_root_folder adk web

```

Then go to http://localhost:8000, and choose my_agent agent (same as the agent folder name)

Toolbox Tools for Databases¶

MCP Toolbox for Databases is an open source MCP server for databases. It was designed with enterprise-grade and production-quality in mind. It enables you to develop tools easier, faster, and more securely by handling the complexities such as connection pooling, authentication, and more.

Google's Agent Development Kit (ADK) has built in support for Toolbox. For more information on getting started or configuring Toolbox, see the documentation.

GenAI Toolbox

Configure and deploy¶

Toolbox is an open source server that you deploy and manage yourself. For more instructions on deploying and configuring, see the official Toolbox documentation:

- Installing the Server
- Configuring Toolbox

Install client SDK¶

ADK relies on the `toolbox-core` python package to use Toolbox. Install the package before getting started:

```
pip install toolbox-core
```

Loading Toolbox Tools¶

Once you’re Toolbox server is configured and up and running, you can load tools from your server using ADK:

```
from google.adk.agents import Agent
from toolbox_core import ToolboxSyncClient

toolbox = ToolboxSyncClient("https://127.0.0.1:5000")

# Load a specific set of tools
tools = toolbox.load_toolset('my-toolset-name'),
# Load single tool
tools = toolbox.load_tool('my-tool-name'),

root_agent = Agent(
    ...,
    tools=tools # Provide the list of tools to the Agent
)
```

Advanced Toolbox Features¶

Toolbox has a variety of features to make developing Gen AI tools for databases. For more information, read more about the following features:

- Authenticated Parameters: bind tool inputs to values from OIDC tokens automatically, making it easy to run sensitive queries without potentially leaking data
- Authorized Invocations: restrict access to use a tool based on the users Auth token
- OpenTelemetry: get metrics and tracing from Toolbox with OpenTelemetry