Register and use ADK agents

Status: Approved

Date of last update: Jun 25, 2025

This page describes how to register and use agents developed using Google agent development kit (ADK) in Agentspace.

You can add generative agents to Agentspace using ADK and call them when needed. When agents are registered they can be selected using the Agentspace app and when you enter queries into your app, you can get answers from the Agents and ask follow-up questions.

Agent registration is available for Agentspace admins only, but after registration other users can also use the registered agents. (IAM permission for admins: `agents.manage`, which is part of the `Discovery Engine Admin` role).

Authorize your agents

This is an optional step that's needed if the agent wants to act on behalf of the end user, such as accessing BigQuery tables only the user has access to. In this case, the administrator can configure OAuth 2.0 authorizations for your agents.

Before you begin

1. Before you can begin the OAuth process, you must first register your application with your OAuth 2.0 provider. Google Cloud can provide OAuth 2.0 support for applications, for more details see Setting up OAuth 2.0. Make sure you always add `https://vertexaisearch.cloud.google.com/oauth-redirect` to the list of allowed redirect URIs in the OAuth application (both for Google and non-Google OAuth providers). When registering a new app, you will receive a Client ID and Client Secret which is needed for the registration process in Agentspace.
2. Scopes enable your application to only request access to the resources that it needs while also enabling users to control the amount of access that they grant to your application. For more information see Identify access scopes. For Google APIs, see OAuth 2.0 Scopes for Google APIs. You must configure the least privilege needed for

your agent. For example, if you only want to read Google Drive files, don't assign `email.write` OAuth 2.0 scope.

3. You will need to provide an authorization URI for your agent. This depends on the OAuth 2.0 provider, for Google see Set authorization parameters . The authorization URI includes the requested scopes, and it can include more than one. Usually an authorization URI contains a redirect URI, but you need not specify the redirect_uri field in it as Agentspace adds `https://vertexaisearch.cloud.google.com/oauth-redirect` automatically (see point 1. above).

4. You will also need a URI which points to an API endpoint to request OAuth 2.0 access tokens. For Google see Exchange authorization code for refresh and access tokens, which identifies the URI as `https://oauth2.googleapis.com/token`

Add an authorization resource to Agentspace

Once you have all the information from above, you can register an authorization resource with Agentspace. To create an authorization resource execute:

Shell

```
curl -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \

"https://discoveryengine.googleapis.com/v1alpha/projects/PROJECT_ID/loca
tions/global/authorizations?authorizationId=AUTH_ID" \
      -d '{
  "name": "projects/PROJECT_ID/locations/global/authorizations/AUTH_ID",
  "serverSideOauth2": {
    "clientId": "OAUTH_CLIENT_ID",
    "clientSecret": "OAUTH_CLIENT_SECRET",
    "authorizationUri": "OAUTH_AUTH_URI",
    "tokenUri": "OAUTH_TOKEN_URI"
  }
}'
```

`Replace the` following:

- PROJECT_ID: the ID of your Google Cloud project.
- AUTH_ID: the ID of the authorization resource. This is an arbitrary ID containing alphanumeric text defined by the user; it needs to be referenced later at the time when the Agent is registered (an Agent which requires OAuth support).
- OAUTH_CLIENT_ID: OAuth 2.0 client identifier issued to the client (see Prerequisites)
- OAUTH_CLIENT_SECRET: OAuth 2.0 client secret (see Prerequisites)
- OAUTH_AUTH_URI: Specifies the endpoint for obtaining an authorization code from a third-party authorization service for OAuth 2.0 (see Prerequisites).
- OAUTH_TOKEN_URI: Endpoint URL where the application can exchange an OAuth 2.0 authorization code for an access token (see Prerequisites).

The `name` field of the authorization resource must be used to reference this authorization resource later, when registering the corresponding Agent.

To delete an existing authorization resource execute:

Shell

```shell
curl -X DELETE \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \

"https://discoveryengine.googleapis.com/v1alpha/projects/PROJECT_ID/loca
tions/global/authorizations/AUTH_ID"
```

Replace the following:

- PROJECT_ID: the ID of your Google Cloud project.
- AUTH_ID: the ID of the authorization

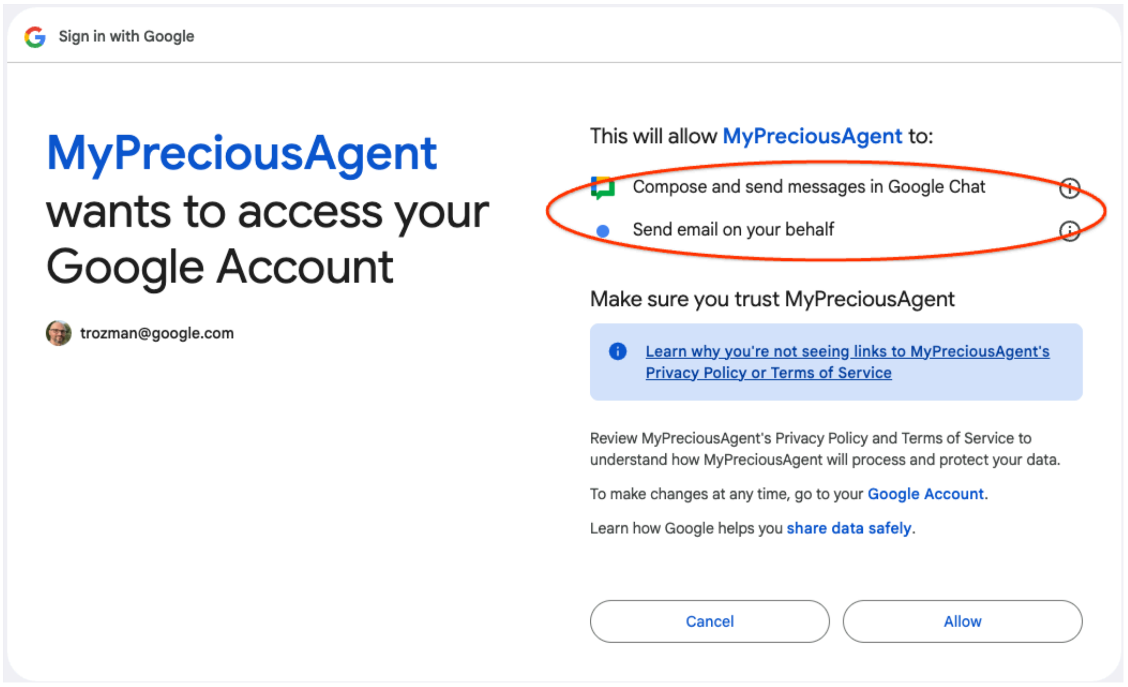
Example authorization URI with Google APIs

This section shows an example for an authorization URI and its corresponding consent screen that is displayed to the user during the authorization process.

An authorization URI is necessary to kick off the user authorization process. It contains the required scopes for the application / agent. Here's an example authorization URI that enables GMail email sending and Google chat message creation (scopes `https://www.googleapis.com/chat.messages.create` and `https://www.googleapis.com/gmail.send`):

```
None
https://accounts.google.com/o/oauth2/v2/auth?client_id=<client_id>&&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fchat.messages.create+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fgmail.send&include_granted_scopes=true&response_type=code&access_type=offline&prompt=consent
```

The corresponding consent screen for the above is similar to this:


Sign in with Google

MyPreciousAgent wants to access your Google Account

trozman@google.com

This will allow MyPreciousAgent to:

Compose and send messages in Google Chat

Send email on your behalf

Make sure you trust MyPreciousAgent

Learn why you're not seeing links to MyPreciousAgent's Privacy Policy or Terms of Service

Review MyPreciousAgent's Privacy Policy and Terms of Service to understand how MyPreciousAgent will process and protect your data.

To make changes at any time, go to your Google Account.

Learn how Google helps you share data safely.

Register an agent with Agentspace

To use a new agent with Agentspace, you must deploy it using the ADK. The deployment returns an agent engine resource name, which is necessary for the registration.

Before you begin

Before deploying the agent, do the following:

1. Enable the DiscoveryEngine API for the GCP project.
2. Enable the Vertex AI user and Vertex AI viewer role in your discoveryengine service account. This is required for Agentspace to call your ADK agent. Go to IAM in cloud console, search for discoveryengine and add permissions to that Service Account. To see the discoveryengine service account you need to check the "Include Google-provided role grants" on the IAM console screen.

To register a new agent with Agentspace, run the following curl command:

Shell
```shell
curl -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \
"https://discoveryengine.googleapis.com/v1alpha/projects/PROJECT_ID/loca
tions/global/collections/default_collection/engines/APP_ID/assistants/de
fault_assistant/agents" \
  -d '{
    "displayName": "DISPLAY_NAME",
    "description": "DESCRIPTION",
    "icon": {
       "uri": "ICON_URI"
     },
    "adk_agent_definition": {
      "tool_settings": {
        "tool_description": "TOOL_DESCRIPTION"
      },
      "provisioned_reasoning_engine": {
        "reasoning_engine":
"projects/PROJECT_ID/locations/REASONING_ENGINE_LOCATION/reasoningEngine
s/ADK_DEPLOYMENT_ID"
      },
      "authorizations": [
        "projects/PROJECT_ID/locations/global/authorizations/AUTH_ID"
        ]
      }
    }'
```

Please note that the "authorizations" tag is optional; it is only needed if the Agent needs to act on behalf of the users (when it needs OAuth 2.0 support, see Authorize your agents).

- `PROJECT_ID`: the ID of your Google Cloud project.
- `APP_ID`: the ID of the Agentspace app.
- `DISPLAY_NAME`: the display name of the agent.
- `DESCRIPTION`: the description of the agent, displayed on the frontend; it is only for the user's benefit.
  `ICON_URI`: The public URI of the icon to display near the name of the agent. Alternatively you can pass Base64-encoded image file contents, but in that case you have to use `icon.content` instead of `icon.uri.`
- `TOOL_DESCRIPTION`: the description / prompt of the agent used by the LLM to route requests to the agent. Must properly describe what the agent does. Never shown to the user.
- `ADK_DEPLOYMENT_ID`: the ID of the reasoning engine endpoint where the ADK agent is deployed.
- `REASONING_ENGINE_LOCATION`: The cloud location of the reasoning engine depending on which location you are creating an agent at. See Reasoning Engine Location
- `AUTH_ID`: the IDs of the authorization resources; can be omitted, can be one or can be more than one. See Authorize your agents on how to create such a resource.

The fields `DESCRIPTION` and `TOOL_DESCRIPTION` differ. For example, for an Invoice Scraper agent the description could be: *"Extract key information from uploaded invoices for business travel"*. The tool description prompt could be: *"You are an expert invoice data extractor for business travel expenses. Your task is to extract key information from user-uploaded invoice documents."*

The response of the above command returns all fields of the created Agent resource. The fields are the same as supplied by the command with the addition of the “name” field: this is the resource name of the newly created agent resource, it can be used to reference the agent later (e.g. when updating it). An example resource name is `"projects/PROJECT_ID/locations/global/collections/default_collection/engines/test-engine-1/assistants/default_assistant/agents/13570498627670476984”.`

Update the registration of an agent

All of the fields that were supplied during agent registration can be updated. The following fields are mandatory during update: `displayName, description, tool_settings, reasoning_engine`. Even if they are unchanged, you must provide these fields again in the update request.

To update the registration of an existing agent, run the following example curl command:

```
None
curl -X PATCH \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \
"https://discoveryengine.googleapis.com/v1alpha/AGENT_RESOURCE_NAME" \
  -d '{
    "displayName": "DISPLAY_NAME",
    "description": "DESCRIPTION",
    "adk_agent_definition": {
      "tool_settings": {
        "tool_description": "TOOL_DESCRIPTION"
      },
     "provisioned_reasoning_engine": {
        "reasoning_engine":
"projects/PROJECT_ID/locations/REASONING_ENGINE_LOCATION/reasoningEngines/ADK_DEPLOYMENT_ID"
      },
    }
  }'
```

Replace the following:

- `PROJECT_ID`: the ID of your Google Cloud project.
- `APP_ID`: the ID of the Agentspace app.
- `AGENT_RESOURCE_NAME`: the resource name of the agent registration to be updated.
- `DISPLAY_NAME`: the display name of the agent.
- `DESCRIPTION`: the description of the agent, displayed on the frontend; it is only for the user's benefit.
- `TOOL_DESCRIPTION`: the description / prompt of the agent used by the LLM to route requests to the agent. Must properly describe what the agent does. Never shown to the user.
- `REASONING_ENGINE_LOCATION`: The cloud location of the reasoning engine depending on which location you are creating an agent at. See Reasoning Engine Location
- `ADK_DEPLOYMENT_ID`: The id of the reasoning engine endpoint where the ADK agent is deployed.

View your agent

To view your agent after you've registered it, run the following curl command:

```
None
curl -X GET \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \
"https://discoveryengine.googleapis.com/v1alpha/AGENT_RESOURCE_NAME"
```

Replace the following:

- PROJECT_ID: the ID of your Google Cloud project.
- APP_ID: the ID of the Agentspace app
- AGENT_RESOURCE_NAME: the resource name of the agent registration to be updated

List agents

To view your agents after you've created it, run the following curl command:

```
None
curl -X GET   -H "Authorization: Bearer $(gcloud auth
print-access-token)"
  -H "Content-Type: application/json"
  -H "X-Goog-User-Project: PROJECT_ID"
"https://discoveryengine.googleapis.com/v1alpha/projects/PROJECT_ID/loca
tions/global/collections/default_collection/engines/APP_ID/assistants/de
fault_assistant/agents"
```

Replace the following:

- PROJECT_ID: the ID of your Google Cloud project.
- APP_ID: the ID of the Agentspace app

Delete an agent

To delete a registration of an agent, run the following curl command:

```
None
curl -X DELETE \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \
"https://discoveryengine.googleapis.com/v1alpha/AGENT_RESOURCE_NAME"
```

Replace the following:

- `PROJECT_ID`: the ID of your Google Cloud project.
- `APP_ID`: the ID of the Agentspace app
- `AGENT_RESOURCE_NAME`: the resource name of the agent registration to be deleted

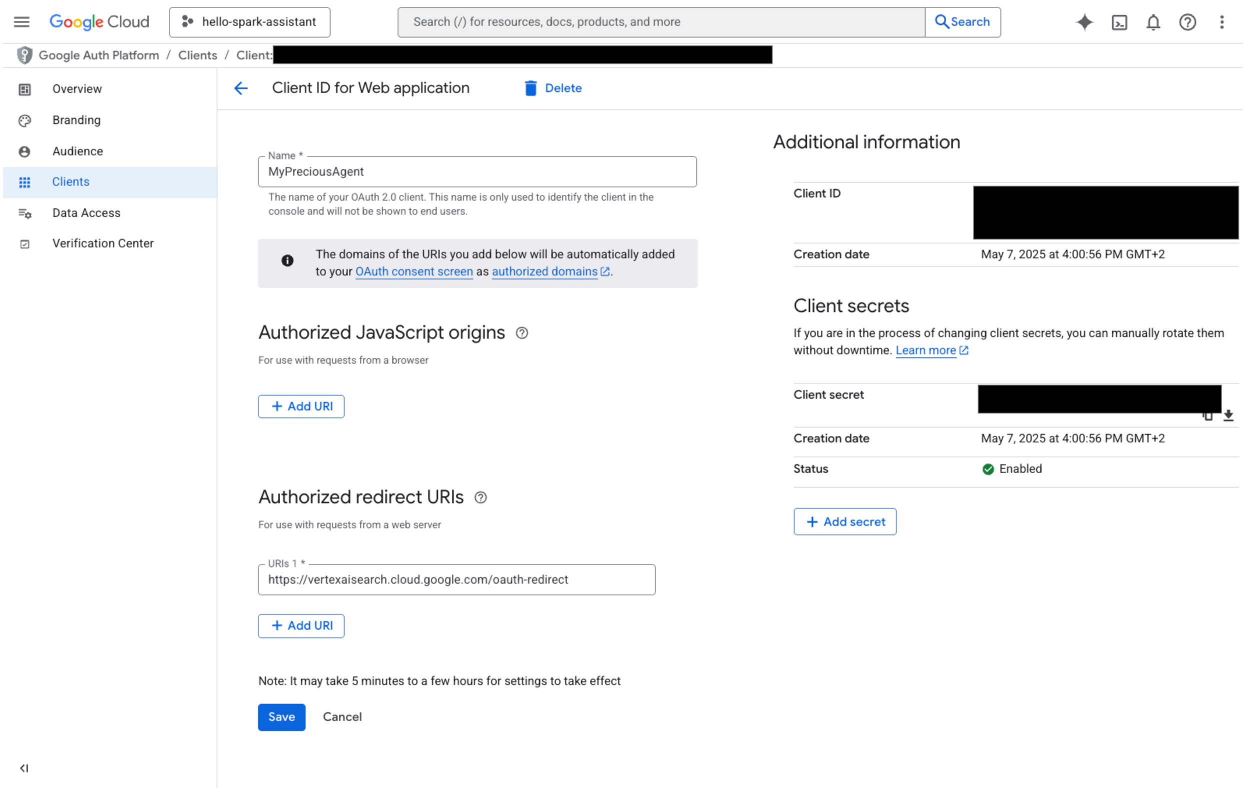
Reasoning Engine Location

Use following chart to find out what is the correct Reasoning Engine Location for your api call

Cloud location you are calling	Reasoning Engine Location
au	australia-southeast2
asia-northeast1	asia-northeast1
ca	northamerica-northeast2
in	asia-south2
de	europe-west3
us	us-central1
eu	europe-west1
europe-west2	europe-west2
others(including global)	us-central1

Get answers from an Agent using the Agentspace app

Let's consider the following OAuth credentials:



Client ID for Web application

Name *
MyPreciousAgent
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins
For use with requests from a browser

Authorized redirect URIs
For use with requests from a web server

URIs 1 *
https://vertexaisearch.cloud.google.com/oauth-redirect

Note: It may take 5 minutes to a few hours for settings to take effect

Additional information

Client ID	[redacted]
Creation date	May 7, 2025 at 4:00:56 PM GMT+2

Client secrets
If you are in the process of changing client secrets, you can manually rotate them without downtime. Learn more

Client secret	[redacted]
Creation date	May 7, 2025 at 4:00:56 PM GMT+2
Status	✓ Enabled

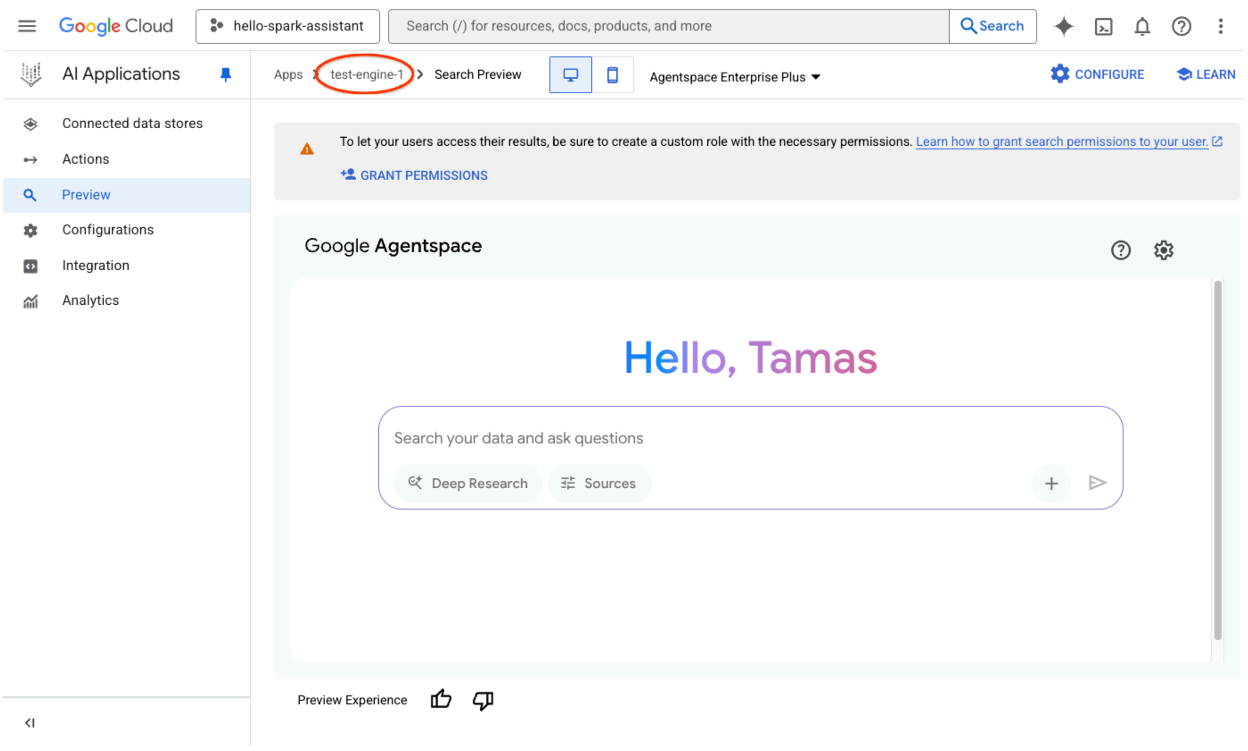
In this example, the OAuth credentials have the name `MyPreciousAgent`. Suppose that the OAuth config (see [Authorization]) of this Agent is set up in Agentspace with `MyPreciousAgent` credentials and with the following authorization URL:

```
https://accounts.google.com/o/oauth2/v2/auth?client_id=
OAUTH_CLIENT_ID
&&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fgmail.send&incl
ude_granted_scopes=true&response_type=code&access_type=offline&pr
ompt=consent
```

In this authorization URL, the authorization requires scope `https://www.googleapis.com/gmail.send`. This means that Agentspace asks you for consent to send GMail emails (and nothing else!) on the user's behalf for `MyPreciousAgent`. Note that the redirect URI is missing from the authorization URL, that's because Agentspace automatically adds `https://vertexaisearch.cloud.google.com/oauth-redirect` as per [Authorization].

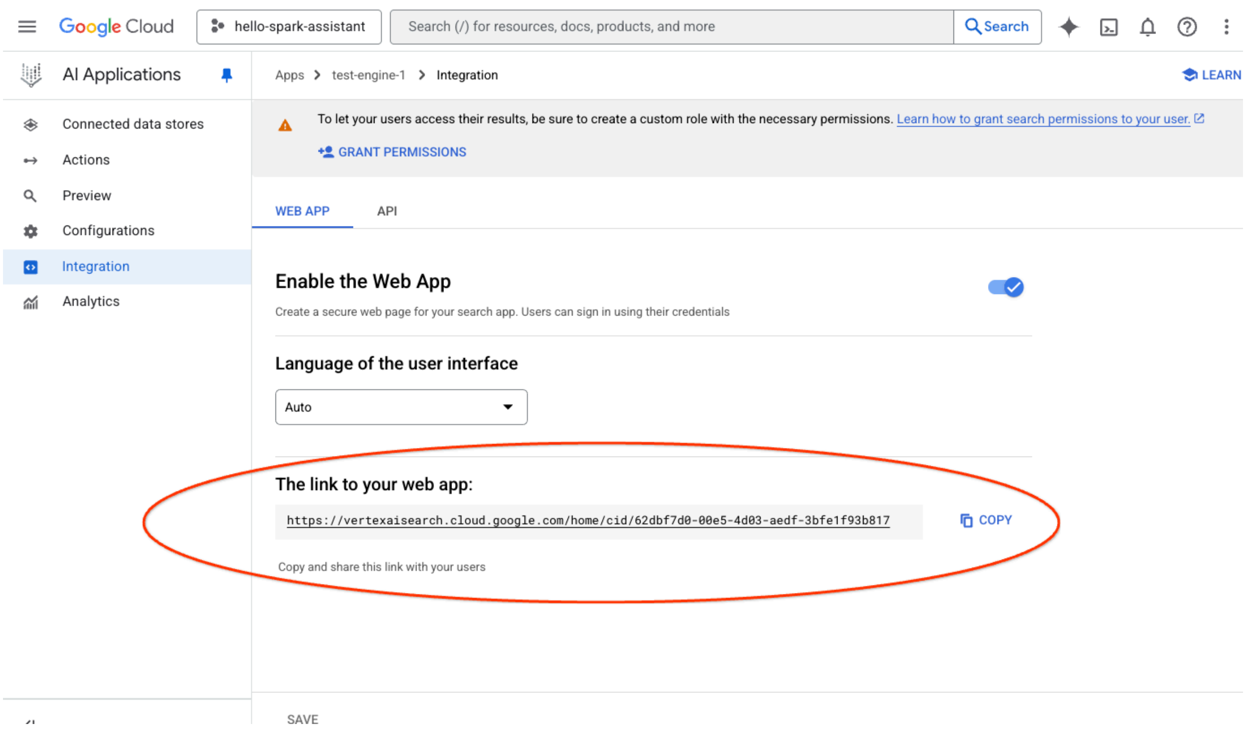
To start a conversation with an agent, do the following:

1. In the Google Cloud console, go to the **Agentspace** page.
2. On the Apps page, select the Agentspace app to which you added the agents.



To let your users access their results, be sure to create a custom role with the necessary permissions. Learn how to grant search permissions to your user.

3. In the main menu, select **Integration**.



Google Cloud | hello-spark-assistant | Search (/) for resources, docs, products, and more | Search

AI Applications

Apps > test-engine-1 > Integration

LEARN

- Connected data stores
- Actions
- Preview
- Configurations
- Integration
- Analytics

To let your users access their results, be sure to create a custom role with the necessary permissions. Learn how to grant search permissions to your user.

GRANT PERMISSIONS

WEB APP | API

Enable the Web App [✓]

Create a secure web page for your search app. Users can sign in using their credentials

Language of the user interface

Auto

The link to your web app:

https://vertexaisearch.cloud.google.com/home/cid/62dbf7d0-00e5-4d03-aedf-3bfe1f93b817

COPY

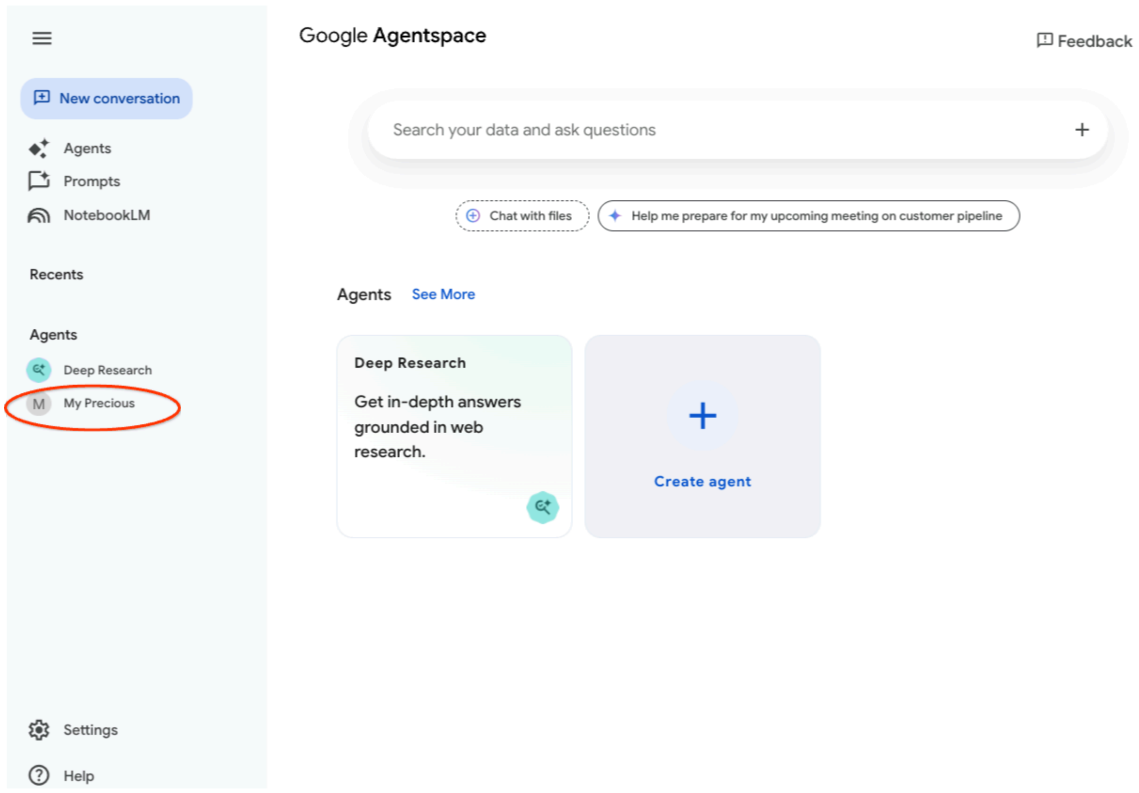
Copy and share this link with your users

SAVE

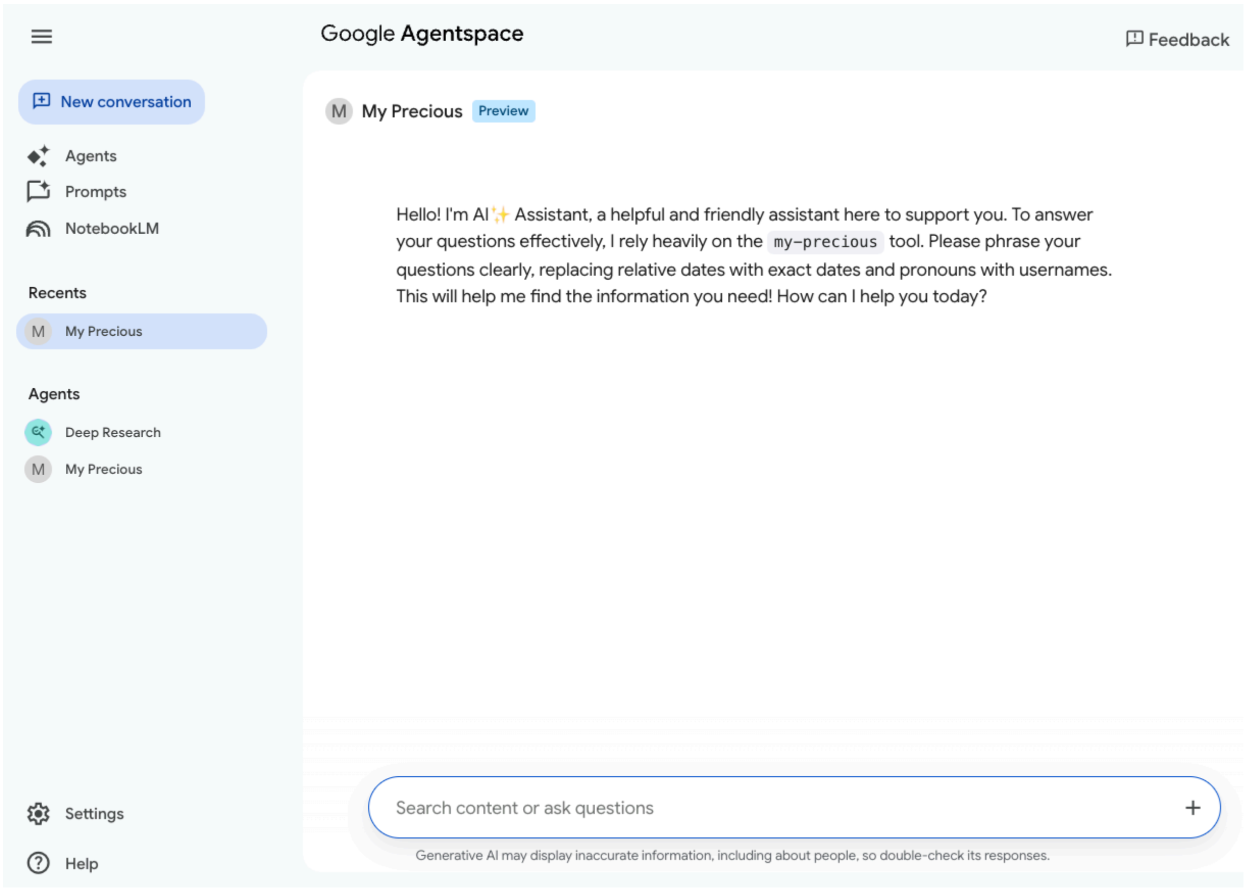
4. Make sure that **Enable the Web App** is enabled.
5. In **The link to your web app**, click **Copy** and navigate to this link in your browser.

To start a conversation with an agent, do the following:

1. In the app navigation menu, in the **Agents** section, click the agent that you created.



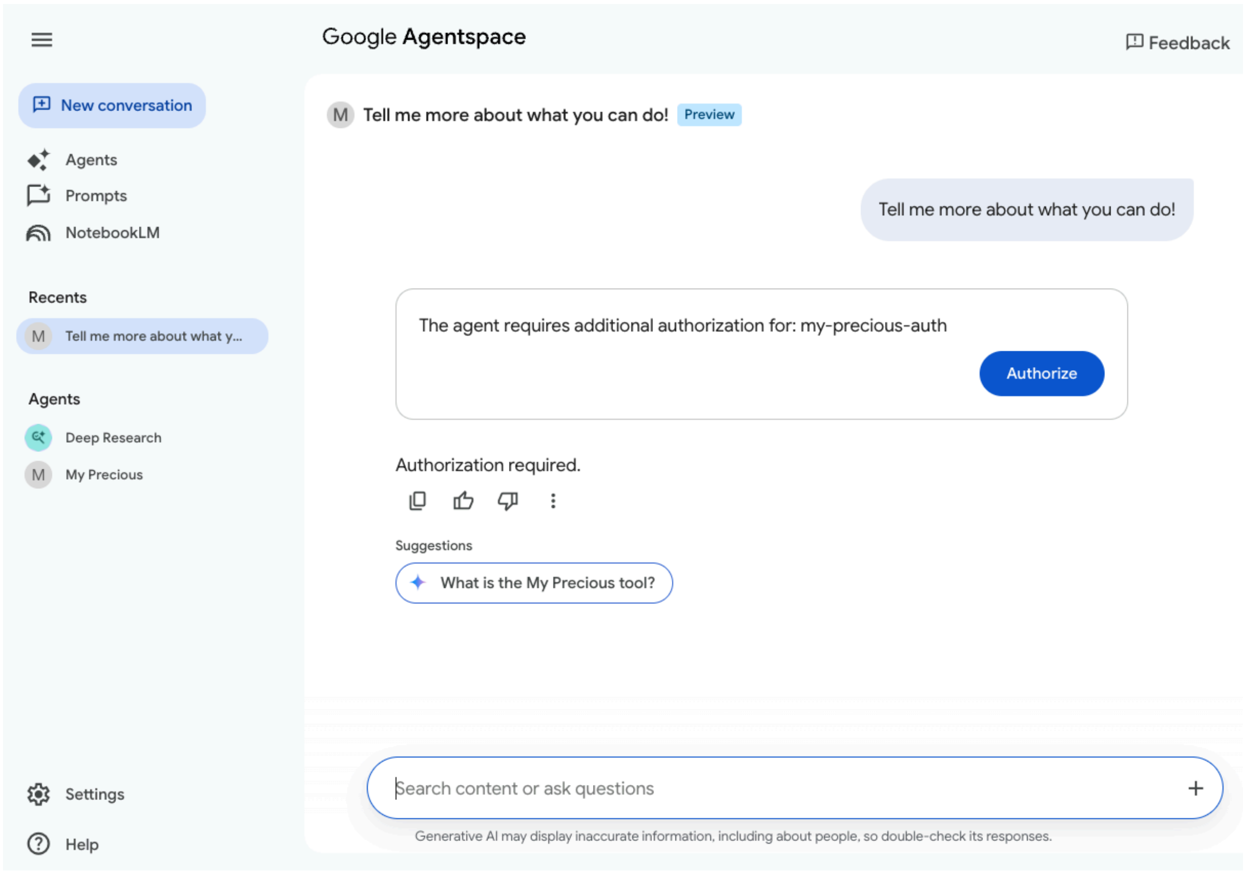
2. In the selected agent, search for the required content or ask questions:



Hello! I'm AI ✨ Assistant, a helpful and friendly assistant here to support you. To answer your questions effectively, I rely heavily on the `my-precious` tool. Please phrase your questions clearly, replacing relative dates with exact dates and pronouns with usernames. This will help me find the information you need! How can I help you today?

Search content or ask questions
Generative AI may display inaccurate information, including about people, so double-check its responses.

3. If this is the first time you're trying to use the agent, Agentspace requires authorization. Click **Authorize** to start the OAuth authorization:



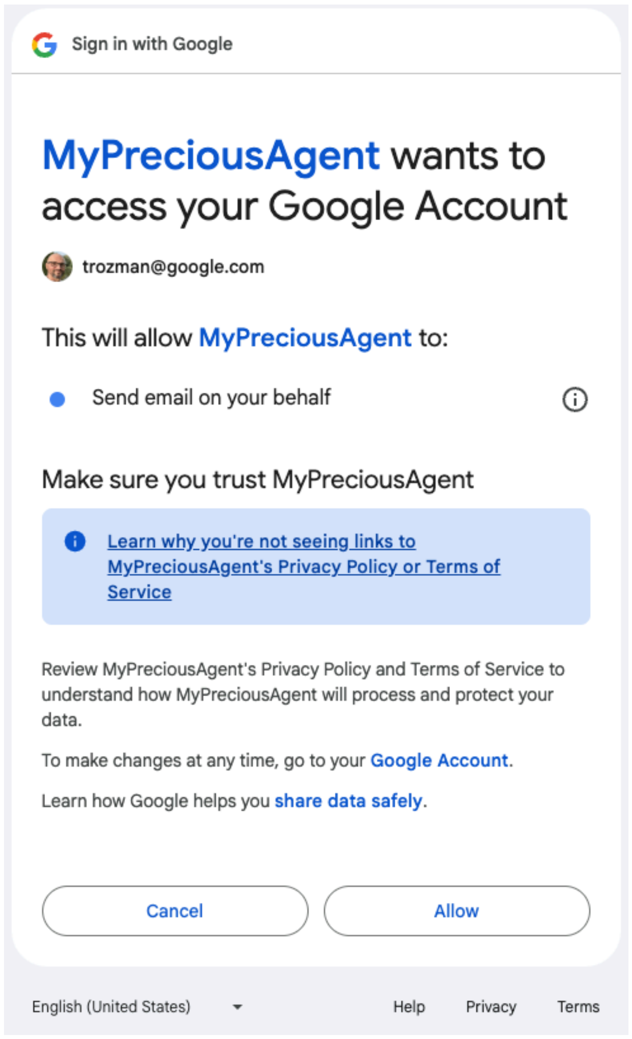
Tell me more about what you can do!

The agent requires additional authorization for: my-precious-auth

Authorization required.

Generative AI may display inaccurate information, including about people, so double-check its responses.

4. In the OAuth dialog, click **Allow** to allow the selected agent to send emails on your behalf:



Sign in with Google

MyPreciousAgent wants to access your Google Account

trozman@google.com

This will allow MyPreciousAgent to:

- Send email on your behalf

Make sure you trust MyPreciousAgent

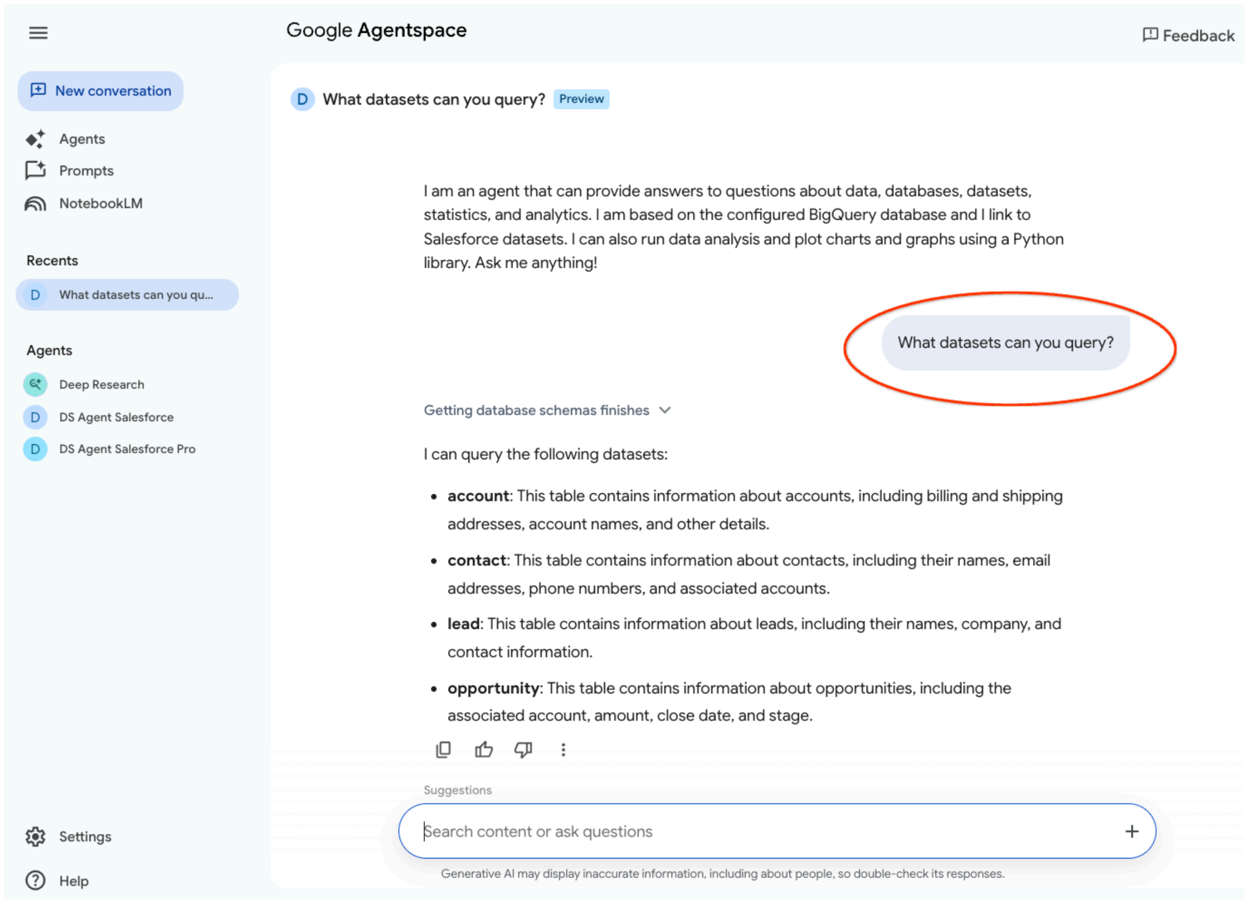
Learn why you're not seeing links to MyPreciousAgent's Privacy Policy or Terms of Service

Review MyPreciousAgent's Privacy Policy and Terms of Service to understand how MyPreciousAgent will process and protect your data.

To make changes at any time, go to your Google Account.

Learn how Google helps you share data safely.

After the consent is granted Agentspace invokes the agent :



I am an agent that can provide answers to questions about data, databases, datasets, statistics, and analytics. I am based on the configured BigQuery database and I link to Salesforce datasets. I can also run data analysis and plot charts and graphs using a Python library. Ask me anything!

What datasets can you query?

Getting database schemas finishes

I can query the following datasets:

- **account**: This table contains information about accounts, including billing and shipping addresses, account names, and other details.
- **contact**: This table contains information about contacts, including their names, email addresses, phone numbers, and associated accounts.
- **lead**: This table contains information about leads, including their names, company, and contact information.
- **opportunity**: This table contains information about opportunities, including the associated account, amount, close date, and stage.

Generative AI may display inaccurate information, including about people, so double-check its responses.

Get answers using the API

To answers from your agent using the assistant API, run the following curl command:

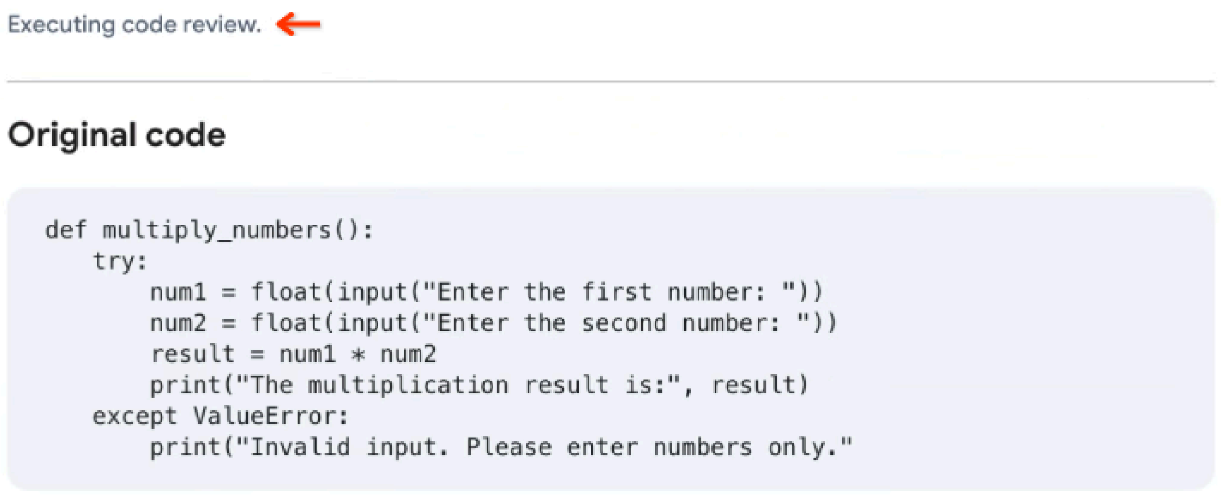
```
None
curl -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -H "X-Goog-User-Project: PROJECT_ID" \
"https://discoveryengine.googleapis.com/v1alpha/projects/PROJECT_ID/locations/global/collections/default_collection/engines/APP_ID/assistants/default_assistant:streamAssist" \
-d '{
  "name":
"projects/PROJECT_ID/locations/global/collections/default_collection/engines/APP_ID/assistants/default_assistant",
  "query": {
    "text": "QUERY"
  },
  "session":
"projects/PROJECT_ID/locations/global/collections/default_collection/eng
ines/APP_ID/sessions/-",
  "assistSkippingMode": "REQUEST_ASSIST",
  "answerGenerationMode": "AGENT",
  "agentsConfig": {
    "agent": "AGENT_RESOURCE_NAME"
  }
}'
```

Replace the following:

- PROJECT_ID: the ID of your Google Cloud project.
- APP_ID: the ID of the app.
- QUERY: the query.
- AGENT_RESOURCE_NAME: the resource name of the registered agent you want to chat with.

Status Updates from your agent

ADK agents can generate status update messages during query execution, which are displayed on the Agentspace UI.



Executing code review.

Original code

```
def multiply_numbers():
    try:
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))
        result = num1 * num2
        print("The multiplication result is:", result)
    except ValueError:
        print("Invalid input. Please enter numbers only."
```

Application status updates can be shown on the Agentspace UI by configuring the `ui:status_update` value in the state dictionary of the ADK session context.

The following is a brief example illustrating how an agent can emit these status updates:

```Python
from typing import Optional
from functools import partial
from google.adk.agents import LlmAgent


def update_ui_status(
    callback_context: CallbackContext, status_msg: str
) -> Optional[types.Content]:
  """Updates the status of the execution on the Agentspace UI.

  Args:
    callback_context: The callback context.
    status_msg: The status message to display on the UI.
  """
  callback_context.state["ui:status_update"] = status_msg


# Code Writer Agent
# Takes the initial specification (from user query) and writes code.
code_writer_agent = LlmAgent(
    name="CodeWriterAgent",
    model="gemini-2.0-flash-001",
    instruction="""You are a Python Code Generator.
Based *only* on the user's request, write Python code that fulfills the
requirement.
Output *only* the complete Python code block, enclosed in triple backticks
(```python ... ```).
Do not add any other text before or after the code block.
""",
    description="Writes initial Python code based on a specification.",
    output_key="generated_code",  # Stores output in state['generated_code']
    before_agent_callback=partial(
        update_ui_status, status_msg="Starting code generation."
    ),
    after_agent_callback=partial(
        update_ui_status, status_msg="Code generation complete."
    ),
)
```

Using access tokens in your ADK Agent

If you configured an Authorization, Agentspace will prompt the user to authorize and send OAuth access tokens to your ADK agent. Since Vertex AI Search calls only access documents that are accessible by the caller, we can use the OAuth token to forward the user identity to the search call and get access e.g. to the user's files on Google Drive:

```python
from google.adk.agents import LlmAgent
from google.adk.tools.tool_context import ToolContext
import requests

BASE_URL="https://discoveryengine.googleapis.com"
ENGINE_NAME=f"projects/{PROJECT_ID}/locations/global/collections/default_collec
tion/engines/{ENGINE_ID}"

def search(query: str, tool_context: ToolContext) -> str:
  # Access token is stored in the temp namespace to avoid saving to session
storage.
  access_token = tool_context.state[f"temp:{AUTH_ID}"]
  engine_url = f"{BASE_URL}/v1alpha/{ENGINE_NAME}"
  url = f"{engine_url}/servingConfigs/default_search:search"
  headers = {"Authorization": f"Bearer {access_token}"}
  r = requests.post(url, data={'query': query}, headers=headers).json()
```

```
  return
r['results'][0]['document']['derivedStructData']['snippets'][0]['snippet']

search_agent = LlmAgent(
    name="SearchAgent",
    model="gemini-2.0-flash-001",
    instruction="Use the search tool to access all sorts of information",
    description="Searches",
    tools=[search]
)
```