Main menu

Build generative virtual agents with API integrations

Course · 1 hour 30 minutes Complete

Course overview

Challenge Lab

Use OpenAPI Tools with Conversational Agents

Build Generative Agents with API Integrations: Challenge Lab

Your Next Steps

Course Badge

Course Survey

Course > Build generative virtual agents with API integrations >

Quick tip: Review the prerequisites before you run the lab

Start Lab 01:30:00

Build Generative Agents with API Integrations: Challenge Lab

Lab 1 hour 30 minutes No cost Advanced

Rate Lab

This lab may incorporate AI tools to support your learning.

Lab instructions and tasks —/100

GENAI056

Challenge Lab Overview

Objective

Setup

Challenge Scenario

Your challenge

Task 1. Build a generative conversational agent using AI Applications

Task 2. Add a Tool to the bot using Cloud Run Functions

Task 3. Add extensions to the bot using Data Stores

Congratulations!

GENAI056

Previous Next

Challenge Lab Overview

This lab will challenge you to perform actions and automation across products. Instead of following step-by-step instructions, you are given a common business scenario and a set of tasks - you figure out how to complete them on your own! An automated scoring system (shown on this page) provides feedback on whether you have completed your tasks correctly.

When you take a Challenge Lab, you will not be taught Google Cloud concepts. You will need to use your skills to assess how to build the solution to the challenge presented. This lab is only recommended for students who have those skills. Are you up for the challenge?

Objective

The objective of this challenge lab is to demonstrate your proficiency in deploying conversational agents, creating a Cloud Run Function and providing it as an OpenAPI Tool to a conversational agent, and constructing a tailored Q&A system utilizing a data store. By completing this lab, you will demonstrate your ability to deploy advanced conversational solutions for customer opportunities.

Setup

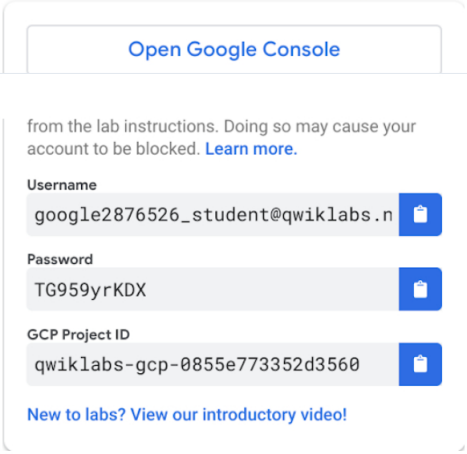
Qwiklabs setup

1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 02:00:00 and make sure you can finish in that time block.

There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click START LAB.

4. Note your lab credentials. You will use them to sign in to the Google Cloud Console.



5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

7. Accept the terms and skip the recovery resource page.

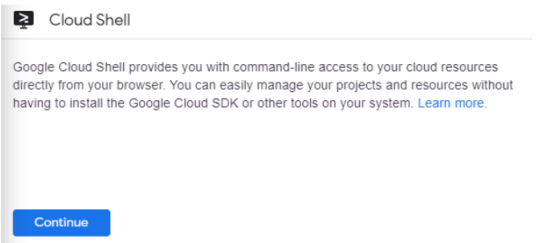
Do not click **End Lab** unless you are finished with the lab or want to restart it. This clears your work and removes the project.

Activate Cloud Shell

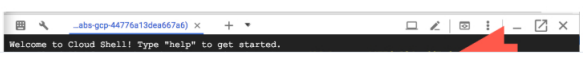
Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.

Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

(Example output)

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the gcloud command-line tool overview.

Challenge Scenario

Cymbal Shops is an American retail chain headquartered in Minneapolis that sells homewares, electronics, and clothing. Think of it as a digitally transforming retailer, inspired by clients like: Bed Bath & Beyond, Best Buy, Home Depot, and Nordstrom.

Your challenge

You work as an engineer at Cymbal Shops. Your project is to deploy a chatbot to help schedule tech support appointments for customers purchasing some of your more complicated products.

In this lab, you should:

1. Build and deploy a conversational agent using AI Applications.
2. Create extensions using Cloud Run Functions to enhance the functionality of the application.
3. Add generative capabilities using AI Applications and Gemini to provide dynamic responses and improve user interaction.

Task 1. Build a generative conversational agent using AI Applications

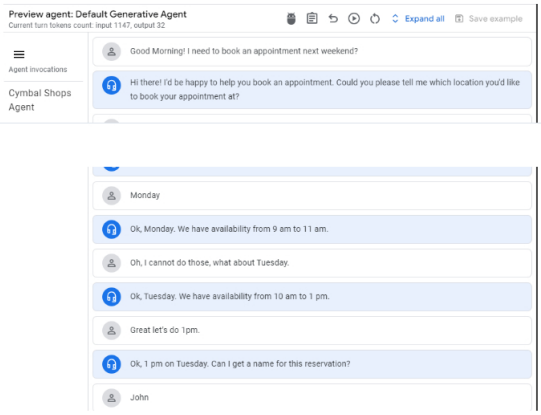
agent using AI Applications

In this section, you will use AI Applications to create a bot that can work through multiple steps to gather information from a user.

1. Enable the Dialogflow API and AI Applications API.
2. In AI Applications console create a new app of type **Conversational agent**.
3. Name your agent `Agent Name` and use the `global` region. For **conversation start** type, use **Playbook**.
4. In the agent-level **Settings** menu, enable the **Cloud Logging** and **conversation history** Logging options to generate logs for this agent.
5. Set the goal of the agent to `Help users book a tech support appointment.`

```
- Greet the user.
- Gather appointment details.
    - Ask for a location
    - Ask for a day of the week
- Coordinate a time based on available slots. Available
slots are:
    - Monday 9 am to 11 am
    - Tuesday 10 am to 1 pm
- Ask for the user's name for the appointment.
- Confirm the appointment.
```

The agent should generate dialogue similar to the provided sample transcript between a user and an agent.



Click **Check my progress** to verify the objective.

Build a conversational agent using AI Applications.

Check my progress

Task 2. Add a Tool to the bot using Cloud Run Functions

In this section, enable the agent to make an external API call to confirm up-to-date availability for appointments. Your task is to create a Cloud Run Function that will provide predefined availability dates. The agent must then communicate with this Cloud Run Function. To facilitate this communication, the agent application can connect to an external API using an OpenAPI tool by supplying the OpenAPI schema. By default, the agent application will make API calls on your behalf.

1. Grant the **Dialogflow Service Agent** the role it needs to invoke Cloud Run functions. You can see the Dialogflow Service Agent by enabling the "Include Google-provided role grants" in the **IAM** panel.
2. Create a Cloud Run function named `Cloud Function Name` in the `Lab GCP Region` region. The function should:
    - Use a **Runtime** in which you are comfortable writing a function. When you create the function, the default function that appears in the source code
    - You will write an HTTP function, with no additional Triggers required.

- The function should require authentication.
- For **Revision Scaling**, set the minimum and maximum number of instances to 0 and 1 respectively.

3. When you have created the function, update the source code so that it reads a value associated with the key `day` from the JSON input to the function. It should return JSON with the available appointment time ranges for that day. You can hard-code the available time slots to be only Monday 9 to 11 am and Tuesday 10 am to 1 pm.
4. Test the Cloud Run function to confirm it works as expected.
5. The agent now needs to be able to send requests to, and process responses from, the Cloud Run Function. In the Conversational Agents dashboard, create a Tool named `OpenAPI Tool Name` of type `OpenAPI`.
6. Provide the Tool an OpenAPI spec that indicates that the Tool should submit a successful response with a status of `200` so the agent knows how to interpret the response.
7. Update the agent's instructions to utilize the `OpenAPI Tool Name` tool instead of having available times hardcoded into its instructions.
8. Test the agent with the following sample dialog:

```
- I'd like to book an appointment
- New York
- Tomorrow, Tuesday
- 10 am
- Jeff
```

Click **Check my progress** to verify the objective.

Provide the agent a Tool that calls a Cloud Run Function.

Check my progress

Task 3. Add extensions to the bot using Data Stores

In this section, you will add a data store to the bot so that customers can ask questions. You will use a Cloud Storage location to create the data store. Follow the configurations below to deploy a chatbot built using AI Applications:

1. Add a data store Tool to the agent `Agent Name` so that users can ask questions about products from the Google Store. This Data store tool can be used by a generative agent for answers to end-user's questions from the data stores.
2. Create a data store named `Datastore Name` and configure it to point to unstructured data hosted in the following Cloud Storage bucket: `cloud-samples-data/dialogflow-cx/google-store`.
3. Test the agent by asking the following questions:

```
How long does the battery in the Pixel 7 Pro last?
Is the Pixel Watch water resistant?
Can I display my Google Photos on a Nest Hub.
```

Click **Check my progress** to verify the objective.

Provide the agent a Data Store.

Check my progress

Congratulations!

Congratulations!

You've successfully deployed a conversational agent and created an OpenAPI Tool that calls a Cloud Run Function. This accomplishment demonstrates your ability to deploy some critical components in advanced conversational solutions.

Manual Last Updated June 05, 2025

Lab Last Tested June 05, 2025