Main menu

Integrate Generative AI Into Your Apps with Firebase Genkit

Course · 6 hours 20% complete

Course overview

Integrate Generative AI Into Your Apps with Firebase Genkit

- Getting Started with Firebase Genkit
- Function Calling with Firebase Genkit
- Deploy Firebase Genkit Apps to Google Cloud
- Build a RAG Solution with Firebase Genkit
- Build Generative AI Applications with Firebase Genkit: Challenge Lab

Your Next Steps

Course > Integrate Generative AI Into Your Apps with Firebase Genkit >

Quick tip: Review the prerequisites before you run the lab

End Lab 00:46:41

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Open Google Cloud Console

Username
student-02-dc294530688a

Password
ldTXC5d0fjY1

GCP Project ID
qwiklabs-gcp-00-5636ce5

Function Calling with Firebase Genkit

Lab 1 hour No cost Intermediate

★★★★½

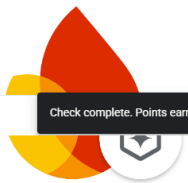
This lab may incorporate AI tools to support your learning.

Lab instructions and tas 100/100

Overview
Objective
Setup and requirements
Task 1. Set up your Genkit project
Task 2. Create a flow with a tool
Task 3. Explore and use the application
Congratulations!

Overview

Firebase Genkit is an open source framework that helps you build, deploy, and monitor production-ready AI-powered apps.



Check complete. Points earned: 40. Message: Assessment completed!

Firebase Genkit

Genkit is designed for app developers. It helps you to easily integrate powerful AI capabilities into your apps with familiar patterns and paradigms.

Use **Genkit** to create apps that generate custom content, use semantic search, handle unstructured inputs, answer questions with your business data, autonomously make decisions, orchestrate tool calls, and much more.

Previous Next

Objective

Check complete. Points earned: 40. Message: Assessment completed!

prompts, models, and tools to ask questions about a menu from a restaurant.

You learn how to:

- Create a new Firebase Genkit project.
- Create flows with Genkit and Gemini.
- Create prompts to provide structured input.
- Use Genkit's tools to leverage function calling in Gemini.
- Use the local Genkit developer UI.
- Explore how tool usage and responses are inspected in the Genkit development UI.

Setup and requirements

Before you click the Start Lab button

Note: Read these instructions.

Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

Check complete. Points earned: 40. Message: Assessment completed!

environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

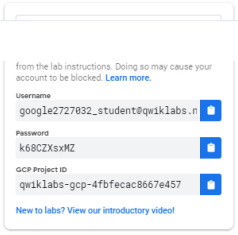
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Console

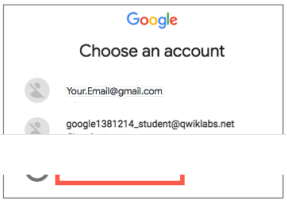
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Choose an account** page.

Note: Open the tabs in separate windows, side-by-side.

3. On the Choose an account page, click **Use Another Account**. The Sign in page opens.



4. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

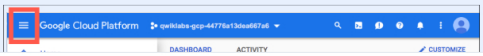
Note: You must use the credentials from the Connection Details panel. Do not use your Google Cloud Skills Boost credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

5. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Activate Google Cloud Shell

Google 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.

Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.

2. 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)
</mydomain></myaccount>
```

Example output:

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

- You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core]
project = <project_id>
</project_id>
```

Example output:

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

Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide .

Task 1. Set up your Genkit project

Initialize your Genkit environment

1. To initialize your environment for Google Cloud, run the following commands:

```
export GCLOUD_PROJECT=qwiklabs-gcp-00-5636ce5f9fd3
export GCLOUD_LOCATION=us-east1
```

2. To authenticate to Google Cloud and set up credentials for your project and user, run the following command:

```
gcloud auth application-default login
```

3. When prompted, type **Y**, and press **Enter**.
4. To launch the Google Cloud sign-in flow in a new tab, press Control (for Windows and Linux) or Command (for MacOS) and click the link in the terminal.
5. In the new tab, click the student email address.
6. When you're prompted to continue, click **Continue**.

click **Allow**.

Your verification code is displayed in the browser tab.

8. Click **Copy**.
9. Back in the terminal, where it says **Enter authorization code**, paste the code, and press **Enter**.

You are now authenticated to Google Cloud.

Set up your application

1. Create a directory for your project, and initialize a new **Node** project:

```
mkdir genkit-intro && cd genkit-intro
npm init -y
```

This command creates a `package.json` file in the `genkit-intro` directory.

2. To install the **Genkit CLI**, run the following command:

3. To install the core Genkit packages and dependencies for your app, run the following command:

```
npm install genkit@1.0.4 --save
npm install @genkit-ai/vertexai@1.0.4 @genkit-ai/google-cloud@1.0.4 @genkit-ai/express@1.0.4 --save
```

Create the Genkit app code

1. Create the application source folder and main file:

```
mkdir src && touch src/index.ts
```

2. From the **Cloud Shell** menu, click on **Open Editor**.

3. Open the `genkit-intro/package.json` file, and review all the dependencies that were added.

   The dependency list should look similar to this:

```
    "@genkit-ai/express": "^1.0.4",
    "@genkit-ai/google-cloud": "^1.0.4",
    "@genkit-ai/vertexai": "^1.0.4",
    "genkit": "^1.0.4"
},
```

4. Go to the `src` folder, and open the `src/index.ts` file. Add the following import library references to the index.ts file:

```
import { z, genkit } from 'genkit';
import { vertexAI } from '@genkit-ai/vertexai';
import { gemini20Flash001 } from '@genkit-ai/vertexai';
import { logger } from 'genkit/logging';
import { enableGoogleCloudTelemetry } from '@genkit-ai/google-cloud';
import { startFlowServer } from '@genkit-ai/express';
```

5. After the import statements, add code to initialize and configure the genkit instance with the required plugin:

```
const ai = genkit({
    plugins: [
        vertexAI({ location: 'us-east1'}),
    ]
```

```
logger.setLogLevel('debug');
enableGoogleCloudTelemetry();
```

The Vertex AI plugin provides access to the Gemini models. The initialization code also sets the log level to debug for troubleshooting, and enables tracing and metrics for monitoring.

6. After changes to the `src/index.ts` file are saved, copy the `package.json` and `index.ts` files to a Cloud Storage Bucket.

   Run these commands in your Cloud Shell terminal:

```
gcloud storage cp ~/genkit-intro/package.json gs://qwiklabs-gcp-00-5636ce5f9fd3
gcloud storage cp ~/genkit-intro/src/index.ts gs://qwiklabs-gcp-00-5636ce5f9fd3
```

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

Setup your Genkit Project

Check my progress

Assessment completed!

Task 2. Create a flow with a tool

In this task, you use **tools**. They provide an abstraction layer that uses function calling

from Gemini to determine if an external service needs to be called, extract the parameters, identify and execute a function, and pass the results back to the LLM.

You implement this functionality to retrieve menu items from a file that contains data for a restaurant's daily menu. The file is stored locally, but it could also be hosted behind an API, enabling your flow to communicate with third-party services.

Create the today's menu data file

1. In Cloud Shell, create a `data` directory:

```
mkdir ~/genkit-intro/data
```

2. Create the file with today's menu items:

```
cat <<EOF > ~/genkit-intro/data/menu.json
[
    {
        "title": "Mozzarella Sticks",
        "price": 8,
        "description": "Crispy fried mozzarella sticks
served with marinara sauce."
    },
    {
        "title": "Chicken Wings",
        "price": 10,
        "description": "Crispy fried chicken wings tossed
in your choice of sauce."
    },
    {
    "title": "Nachos",
        "price": 12,
        "description": "Crispy tortilla chips topped with
melted cheese, chili, sour cream, and salsa."
    },
    {
        "title": "Onion Rings",
        "price": 7,
        "description": "Crispy fried onion rings served
with ranch dressing."
    },
    {
        "title": "French Fries",
```

```
    },
    {
        "title": "Mashed Potatoes",
        "price": 6,
        "description": "Creamy mashed potatoes."
    },
    {
        "title": "Coleslaw",
        "price": 4,
        "description": "Homemade coleslaw."
    },
    {
        "title": "Classic Cheeseburger",
        "price": 12,
        "description": "A juicy beef patty topped with
melted American cheese, lettuce, tomato, and onion on a
toasted bun."
    },
    {
        "title": "Bacon Cheeseburger",
        "price": 14,
        "description": "A classic cheeseburger with the
addition of crispy bacon."
    },
    {
        "title": "Mushroom Swiss Burger",
        "price": 15,
        "description": "A beef patty topped with sautéed
mushrooms, melted Swiss cheese, and a creamy horseradish
```

```
    {
        "title": "Chicken Sandwich",
        "price": 13,
        "description": "A crispy chicken breast on a
toasted bun with lettuce, tomato, and your choice of
sauce."
    },
    {
        "title": "Pulled Pork Sandwich",
        "price": 14,
        "description": "Slow-cooked pulled pork on a
toasted bun with coleslaw and barbecue sauce."
    },
    {
        "title": "Reuben Sandwich",
        "price": 15,
        "description": "Thinly sliced corned beef, Swiss
cheese, sauerkraut, and Thousand Island dressing on rye
bread."
    },
    {
        "title": "House Salad",
        "price": 8,
        "description": "Mixed greens with your choice of
dressing."
    },
    {
        "title": "Caesar Salad",
        "price": 9,
```

```
    },
    {
        "title": "Greek Salad",
        "price": 10,
        "description": "Mixed greens with feta cheese,
olives, tomatoes, cucumbers, and red onions."
    },
    {
        "title": "Chocolate Lava Cake",
        "price": 8,
        "description": "A warm, gooey chocolate cake with a
molten chocolate center."
    },
    {
        "title": "Apple Pie",
        "price": 7,
        "description": "A classic apple pie with a flaky
crust and warm apple filling."
    },
    {
        "title": "Cheesecake",
        "price": 8,
        "description": "A creamy cheesecake with a graham
cracker crust."
    }
]
EOF
```

```
cat ~/genkit-intro/data/menu.json
```

Define the flow and related objects

1. Use the Cloud Shell editor to open the `genkit-intro/src/index.ts` file, and append the code to define these input and output objects:

    - **MenuItemSchema:** Describes a menu item, and contains 3 fields: title, description, and price. This matches the data in the `menu.json` file that we added above.
    - **MenuItem:** A type for the MenuItemSchema definition.
    - **MenuQuestionInputSchema:** A string that contains the input string from the user.
    - **AnswerOutSchema:** A string that contains the response from the LLM that is sent back to the user.

```
export const MenuItemSchema = z.object({
    description: z.string()
        .describe('Details, including ingredients and
preparation'),
    price: z.number()
        .describe('Price, in dollars'),
});

export type MenuItem = z.infer<typeof MenuItemSchema>;

// Input schema for a question about the menu
export const MenuQuestionInputSchema = z.object({
    question: z.string(),
});

// Output schema containing an answer to a question
export const AnswerOutputSchema = z.object({
    answer: z.string(),
});
```

2. Next, define the menu data and the menu tool to access the data. The menu data is loaded from local storage, but it could also be an API call to a third party service.

```
const menuData: Array<MenuItem> =
require('../data/menu.json')
```

```
(
    name: 'todaysMenu',
    description: "Use this tool to retrieve all the items
on today's menu",
    inputSchema: z.object({}),
    outputSchema: z.object({
        menuData: z.array(MenuItemSchema)
            .describe('A list of all the items on the
menu'),
    }),
},
async () => Promise.resolve({ menuData: menuData })
);
```

The LLM will use the description of the tool and schema to determine whether the tool will be helpful for a given prompt.

3. Define the prompt that uses the tool, and specify the prompt text.

```
export const dataMenuPrompt = ai.definePrompt(
{
    name: 'dataMenu',
    model: gemini20Flash001,
    input: { schema: MenuQuestionInputSchema },
    output: { format: 'text' },
    tools: [menuTool]
```

```
You are acting as a helpful AI assistant named Walt that
can answer
questions about the food available on the menu at Walt's
Burgers.

Answer this customer's question, in a concise and helpful
manner,
as long as it is about food on the menu or something
harmless like sports.
Use the tools available to answer food and menu questions.
DO NOT INVENT ITEMS NOT ON THE MENU.

Question:
{{question}} ?
`
);
```

The prompt includes the model, input schema, output format, and an array of tools, as well as the prompt that will be sent to the model.

4. Define the flow that prompts the model using the `dataMenuPrompt`.

```
export const menuQuestionFlow = ai.defineFlow(
{
    name: 'menuQuestion',
```

```
},
async (input) => {
    const response = await dataMenuPrompt({
        question: input.question,
    });
    return { answer: response.text };
}
);
```

Note that the flow does not specify the use of the tool. The LLM will note the tools available to it and use the tool if it would be helpful.

5. Add the following line in the index.ts file, which starts the flow server and exposes your flows as HTTP endpoints:

```
startFlowServer({
    flows: [menuQuestionFlow],
    port: 8080,
    cors: {
        origin: '*',
    },
});
```

6. After changes to the `src/index.ts` file are saved, copy the `menu.json` and

Run these commands in your Cloud Shell terminal:

```
gcloud storage cp -r ~/genkit-intro/data/menu.json
gs://qwiklabs-gcp-00-5636ce5f9fd3
gcloud storage cp -r ~/genkit-intro/src/index.ts
gs://qwiklabs-gcp-00-5636ce5f9fd3
```

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

Create a flow with a tool to consult the menu

Check my progress

Assessment completed!

Task 3. Explore and use the application

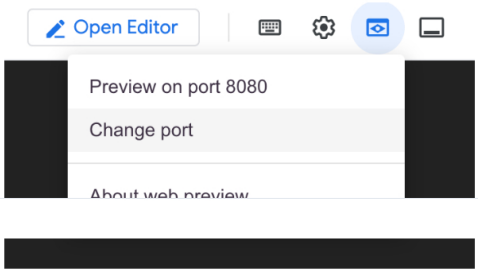
1. In the Cloud Shell terminal, start the **Genkit Developer UI** by running the following command:

```
cd ~/genkit-intro
npx genkit start -- npx tsx src/index.ts | tee -a
output.txt
```

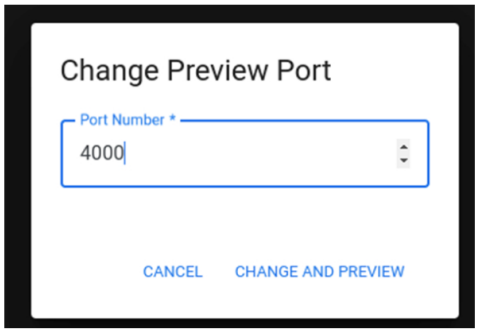
2. When prompted, press Enter to continue.

   Wait for the command to return the **Genkit Developer UI** URL in the output before continuing to the next step.

3. In the **Web Preview** menu, click **Change port**.

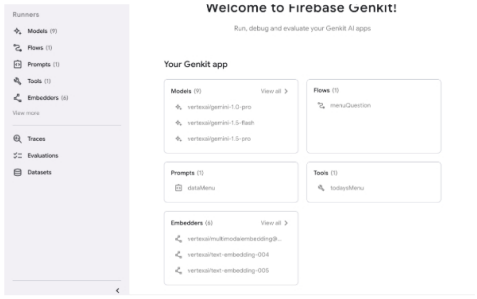


4. For the **Port Number**, type **4000**, and click **Change and Preview**.



   This opens the Genkit developer UI in a separate tab in your browser.

5. Explore the **Genkit development UI** from your browser.



   Verify that the left panel contains flows, prompts, models, and tools. Upon opening each section, confirm the presence of elements defined in your code.

Test the flow

1. In the Genkit developer UI, navigate to the **Flows** section. Click **menuQuestion**, and type the following question to provide the input (JSON):

```
    "question": "What is on the menu for today?"
}
```

2. Click **Run**.

   You should get a response that may look similar to this:

   Output:

```
{
  "answer": "We have a great menu today! We have appetizers like
mozzarella sticks, chicken wings, and nachos. For entrees, we have
cheeseburgers, chicken sandwiches, pulled pork sandwiches, and
more. We also have salads and desserts.  What can I get for you?
\n"
}
```

   If you get an error, re-run the flow.

View trace information

1. At the bottom, click **View traces**.

Observe the timeline of each step on the left, and the corresponding metadata associated with the request on the right.

2. To see how the model identified the function to be called, click **vertexai/gemini-2.0-flash-001**.

3. To view the input and output to and from the tool, click **todaysMenu**.

4. To view the formatted response that Gemini provided for the menu input, click **vertexai/gemini-2.0-flash-001**.

Try different prompts

1. Navigate back to **Runners > Flows**, and click the **menuQuestion** flow.

```
{
    "question": "Are there any chicken options in today's
menu?"
}
```

```
{
    "question": "What burgers do you have on today's menu?"
}
```

2. To test the tool, in the left panel, navigate to **Tools**, select the **todaysMenu** tool, and click **Run**.

   Observe the tool function's response as it would appear when called directly.

3. Navigate to **Prompts**, select the **dataMenu** prompt, and provide the prompt:

```
{
    "question": "What is on the menu for today?"
}
```

4. Click **Run**.

   - **User:** the prompt from your code with the question you asked inserted in the prompt.
   - **Model:** a response to the user's questions, augmented by the data provided by the tool. In this example, the model gives a short version of the menu the way a waiter would do so.

Save your work

1. To exit the Genkit UI, close your browser tab, and type **CTRL+C** in your Cloud Shell terminal to stop the application.

2. Copy the `output.txt` file to a Cloud Storage bucket:

```
gcloud storage cp ~/genkit-intro/output.txt gs://qwiklabs-
gcp-00-5636ce5f9fd3
```

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

Explore and use the application

Check my progress

Assessment completed!

Congratulations!

You have successfully created a Genkit project and leveraged function calling from Gemini. You created a prompt, added a tool to interact with the model, and encapsulated the process in a flow. You tested the application using the Genkit developer UI.

Manual Last Updated May 15, 2025

Lab Last Tested May 15, 2025

Lab Last Tested May 15, 2023

Course Badge