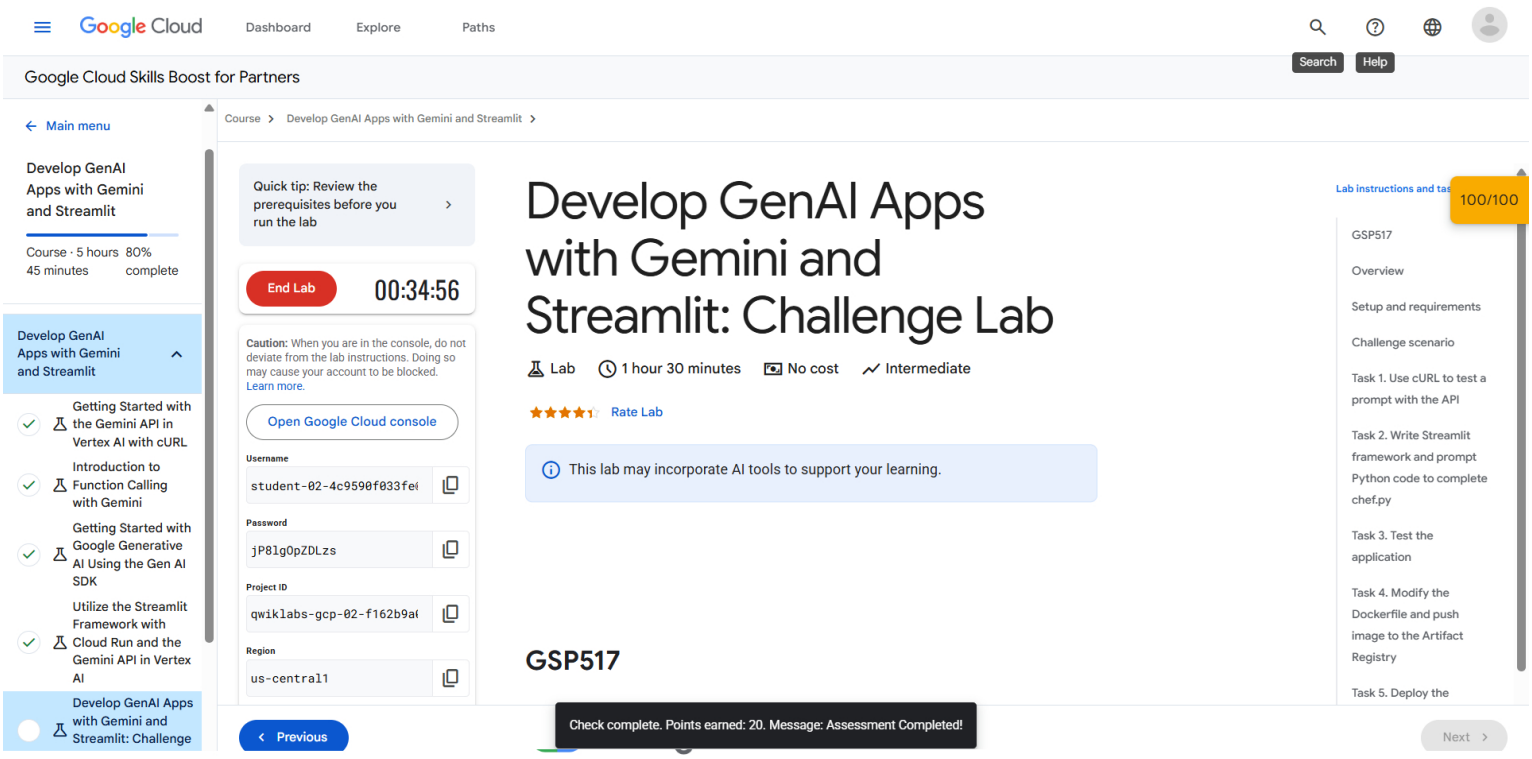


Overview

In a challenge lab you're given a scenario and a set of tasks. Instead of following step-by-step instructions, you will use the skills learned from the labs in the course to figure out how to complete the tasks on your own! An automated scoring system (shown on this page) will provide feedback on whether you have completed your tasks correctly.

When you take a challenge lab, you will not be taught new Google Cloud concepts. You are expected to extend your learned skills, like changing default values and reading and researching error messages to fix your own mistakes.

To score 100% you must successfully complete all tasks within the time period!

This lab is recommended for students who enrolled in the Develop GenAI Apps with

Check complete. Points earned: 20. Message: Assessment Completed!

Setup and requirements

Before you click the Start Lab button

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 are made available to you.

This hands-on lab lets you do the lab activities in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

Check complete. Points earned: 20. Message: Assessment Completed!

Note: Use an Incognito (recommended) or private browser window to run this lab. This prevents conflicts between your personal account and the student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab—remember, once you start, you cannot pause a lab.

Note: Use only the student account for this lab. If you use a different Google Cloud

account, you may incur charges to that account.

Challenge scenario

You onboarded at Cymbal Health just a few months ago. Cymbal Health is an

and coverage under one health plan to make it easier for patients to receive high quality care at an affordable cost.

As a value added service, Cymbal Health is interested in improving customer Healthy Living and Wellness, with tips, and advice within apps. One particular area they want to focus on is improving customer nutrition.

By harnessing the power of the Gemini, a multimodal model for generating text, audio, images, and video, Cymbal Health can build applications that generate meal recommendations for its customers.

As an example, your team has been working to create a AI-Based Chef app, that

Your job is to build, test and deploy a Proof Of Concept (POC) for this Chef app built on the Gemini model, Streamlit framework, and Cloud Run. As part of this POC, they have a list of tasks they would like to see you do in an allotted period of time in a sandbox environment.

Your challenge

Your tasks include the following:

- Use cURL to test a prompt with the API
- Write Streamlit framework and prompt Python code to complete `chef.py`
- Test the application
- Modify the Dockerfile and push the Docker image to the Artifact Registry
- Deploy the application to Cloud Run and test

a prompt with the API

Before you can begin to create the Chef app in Vertex AI, you should test connectivity with the Gemini API.

1. In the Google Cloud console, on the **Navigation menu** (≡), click **Vertex AI > Workbench**.

2. Find the `generative-ai-jupyterlab` instance and click on the **Open JupyterLab** button.

The JupyterLab interface for your Workbench instance opens in a new browser tab.

3. From the left hand menu, modify `prompt.ipynb` to include your project_ID and region within cell 3. You can get these in the left panel of the lab instructions.

4. From the left hand menu, modify `prompt.ipynb` to use the following prompt with cURL within cell 5, by replacing the existing prompt.


```
to include recipes that use ingredients associated with a
peanuts food allergy. I have ahi tuna, fresh ginger, and
edamame in my kitchen and other ingredients. The customer
wine preference is red. Please provide some for meal
recommendations. For each recommendation include
preparation instructions, time to prepare and the recipe
title at the beginning of the response. Then include the
wine paring for each recommendation. At the end of the
```

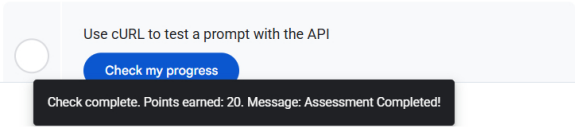
```
recommendation provide the calories associated with the
meal and the nutritional facts.
```

5. Run all cells and observe the results.

6. Save `prompt.ipynb`.

Once you are satisfied with the results, verify the objective.

To verify the objective, click **Check my progress**.



Task 2. Write Streamlit framework and prompt Python code to complete chef.py

For this task you will clone a GitHub repo, and download the `chef.py` file. Then you will add Streamlit framework code in the `chef.py` file for the wine preference, to complete the user interface for the application. You will also include a custom Gemini prompt (similar to the one in task 1), but this one includes variables.

1. Using Cloud Shell clone the repo below from the default directory.

```
git clone
https://github.com/GoogleCloudPlatform/generative-ai.git
```

Check complete. Points earned: 20. Message: Assessment Completed!

```
cd generative-ai/gemini/sample-apps/gemini-streamlit-
cloudrun
```

3. Specify the dependencies in the requirements.txt file:

```
google-cloud-logging
```

> **Important:** All work in this challenge lab should be done within this directory. If you do not download the `chef.py` file here and make changes to it here, it will not be able to access the Streamlit framework. You will also not be able to test it in Cloud Shell (Task 3), or build the docker container (Task 4), and deploy then test it Cloud Run (Task 5).

4. Download the `chef.py` file using the following command.

```
gsutil cp gs://qwiklabs-gcp-02-f162b9a0b607-gemini/chef.py
```

5. Open the chef.py file in the Cloud Shell Editor and review the code.

> **Note:** The chef.py file already includes the Streamlit framework user interface code for the cuisine, dietary_preference, allergy, ingredient_1, ingredient_2, and ingredient_3 variables. Review this interface code before completing the next step.

6. For Project ID, use `qwiklabs-gcp-02-f162b9a0b607`, and for Location, use `us-central1`.

7. Add Streamlit framework radio button option for the wine variable. Include options for Red, White and None.

Click here for hint!

8. Save the `chef.py` file.

9. Add the new Gemini prompt below in Python code.

```
\n
However, don't include recipes that use ingredients with
the customer's {allergy} allergy. \n
I have {ingredient_1}, \n
{ingredient_2}, \n
```

```
{ingredient_2}, \n
and {ingredient_3} \n
in my kitchen and other ingredients. \n
The customer's wine preference is {wine} \n
Please provide some for meal recommendations.
For each recommendation include preparation instructions,
time to prepare
and the recipe title at the beginning of the response.
Then include the wine paring for each recommendation.
At the end of the recommendation provide the calories
associated with the meal
and the nutritional facts.
"""
```

10. Save the `chef.py` file.

Once you are satisfied with the Gemini prompt code you added in `chef.py`, upload the file to **`qwiklabs-gcp-02-f162b9a0b607`-generative-ai** bucket by running below

```
gcloud storage cp chef.py gs://qwiklabs-gcp-02-f162b9a0b607-generative-ai/
```

To verify the objective, click **Check my progress**.

Write prompt and Streamlit framework Python code to complete chef.py

Check my progress

Note: Make sure to run above command after making any changes in the `chef.py` file. So that the updated `chef.py` file is present in the bucket.

For this task you will use the terminal in Cloud Shell to run and test your application.

Make sure your are still in this path, `generative-ai/gemini/sample-apps/gemini-streamlit-cloudrun`.

1. Setup the python virtual environment and install the dependencies.
2. Set environment variables for PROJECT (as your Project ID) and REGION (as the region you are using in the lab environment).
3. Run the `chef.py` application and test it.

Click here for hint!

Once you tested the application in Cloud Shell and confirmed it is performing as designed, without errors, verify the objective.

To verify the objective, click **Check my progress**.

Test the application

Task 4. Modify the Dockerfile and push image to the Artifact Registry

In this task you modify the sample `Dockerfile` to use your `chef.py` file and push the Docker image to the Artifact Registry.

Important: Before completing the steps in this task we recommend you set environment variables for PROJECT (as your Project ID) and REGION (as the region you are using in the lab environment) as you did in a previous task.

1. Use the Cloud Shell editor to modify the Dockerfile to use `chef.py`, then save the file.

Variable	Value
AR_REPO	chef-repo
SERVICE_NAME	chef-streamlit-app

Note: We recommend you combine this command and the following two commands as a single command, as the process to create the Artifact Registry and submit the build to Cloud Build, takes approximately 8 minutes.

3. Create the Artifact Registry repository with the `gcloud artifacts repositories create` command and the following parameters.

Parameter	Value
repo name	$AR_REPO
location	$REGION
repository format	Docker

4. Submit the build with the `gcloud builds submit` command and the following parameters.

Parameter	Value
tag	"$REGION-docker.pkg.dev/$PROJECT/$AR_REPO/$SERVICE_NAME"

Click here for hint!

Click here for hint!

5. Wait for the command to complete.

Once the command is complete, verify the objective.

To verify the objective, click **Check my progress**.

Modify the Dockerfile and push the Docker image to the Artifact Registry

Check my progress

Task 5. Deploy the application to Cloud Run and test

In this task you deploy the application (as a Docker Artifact) to Cloud Run and then test the application as running from the Cloud Run service endpoint.

1. In Cloud Shell deploy the application (as a Docker Artifact), using `gcloud run deploy` command and the following parameter values:

Parameter	Value
port	8080
image	"$REGION-docker.pkg.dev/$PROJECT/$AR_REPO/$SERVICE_NAME"
flag	--allow-unauthenticated
platform	managed
project	PROJECT
set-env-vars	PROJECT=$PROJECT,REGION=$REGION

Click here for hint!

Note: You may see a prompt asking "Do you want enable these APIs to continue (this will take a few minutes)?" If you do, select Y for yes.

The deployment will take a few minutes to complete and you will be provided a URL to the Cloud Run service. You can visit that in the browser to view the Cloud Run application that you just deployed.

2. Test the application with the link provided.

Once you successfully tested the application running on Cloud Run, verify the objective.

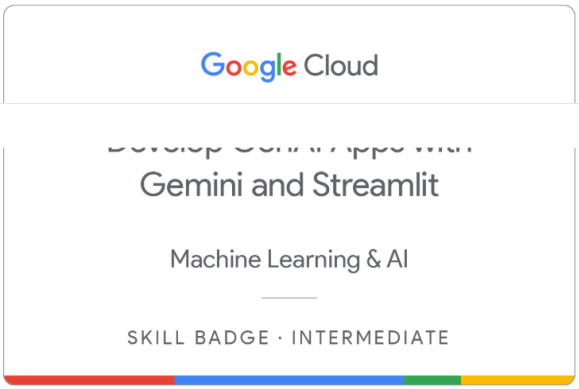
To verify the objective, click **Check my progress**.

Check my progress

Assessment Completed!

Congratulations!

By completing this challenge lab, you verified your skills with Gen AI application development with Gemini and how you can apply these to AI based chef application.



Next steps / Learn more

- Check out the Generative AI on Vertex AI documentation.
- Learn more about Generative AI on the Google Cloud Tech YouTube channel.
- Google Cloud Generative AI official repo
- Example Gemini notebooks

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated May 28, 2025

Lab Last Tested May 28, 2025

Lab