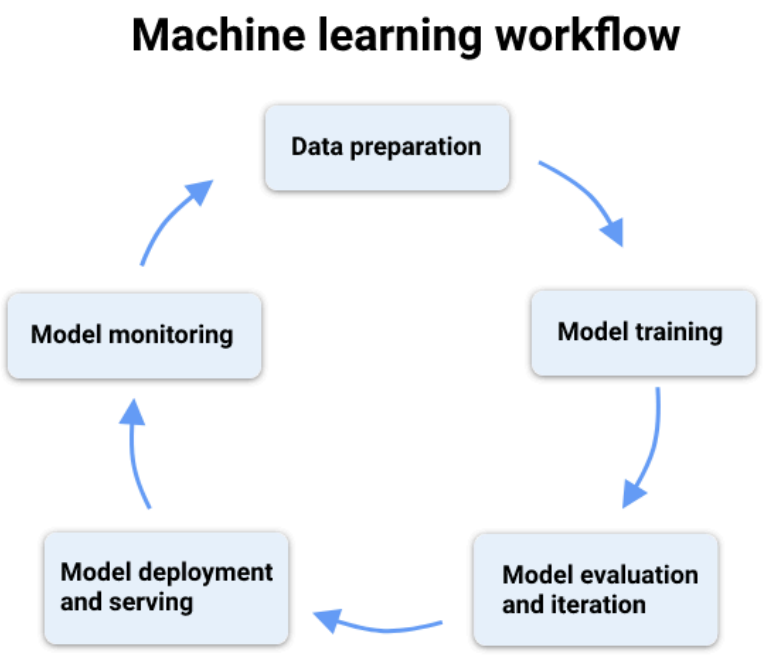
Agent Builder Advanced: Integrating External APIs

For section-specific guidelines and examples, see below

Vertex AI Overview

Vertex AI is a machine learning (ML) platform that lets you train and deploy ML models and AI applications. Vertex AI combines data engineering, data science, and ML engineering workflows, enabling your teams to collaborate using a common toolset. Agent Builder is a low-code, no-code platform that can help build things faster.



What are LLMs

Large language models (LLMs) are deep learning models trained on massive datasets of text. LLMs can translate language, summarize text, generate creative writing, generate code, power chatbots and virtual assistants, and complement search engines and recommendation systems. Creating an LLM requires massive amounts of data, significant compute resources, and specialized skills. Because LLMs require a big investment to create, they target broad rather than specific use cases. On Vertex AI, you can customize a foundation model for more specific tasks or knowledge domains by using prompt design and model tuning.

Build powerful AI agents, no code required. For complex goals, you can easily stitch together multiple agents, with one agent functioning as the main agent and others as subagents. Train with your data, automate tasks, and iterate with ease. Launch and analyze - all within a user-friendly platform.

Objectives:

- ✅ Introduce Agent Builder
- ✅ Cloud Functions for Integrating with external apis
- ✅ Build a use-case with a no-code/low-code platform

Task 0. 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 will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud 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.

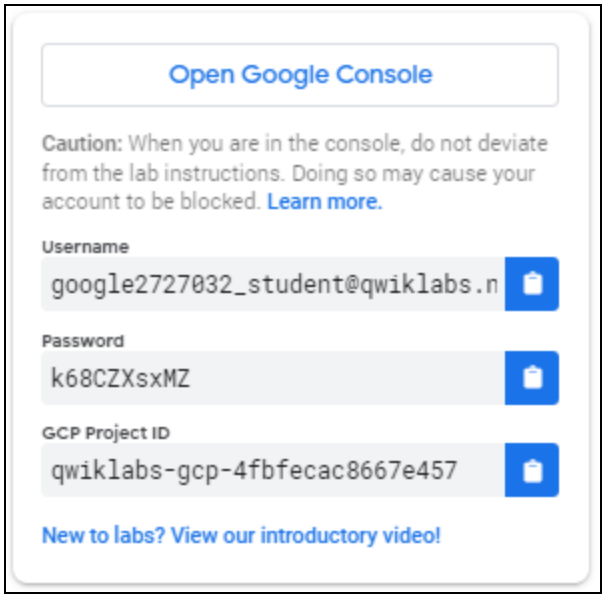
Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

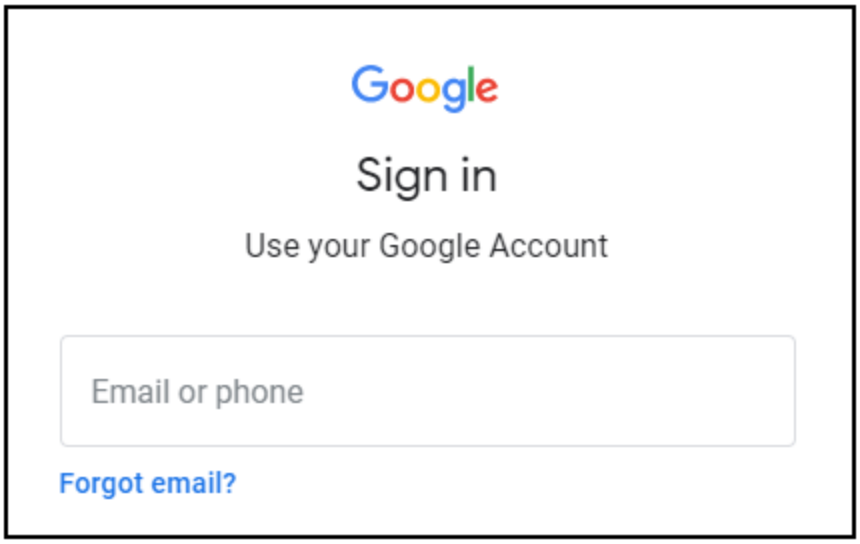
Note: If you are using a Chrome OS device, open an Incognito window to run this lab.

Alternatively, you can log out of all other Google / Gmail accounts before beginning the labs.

How to start your lab and sign in to the Google Cloud 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.

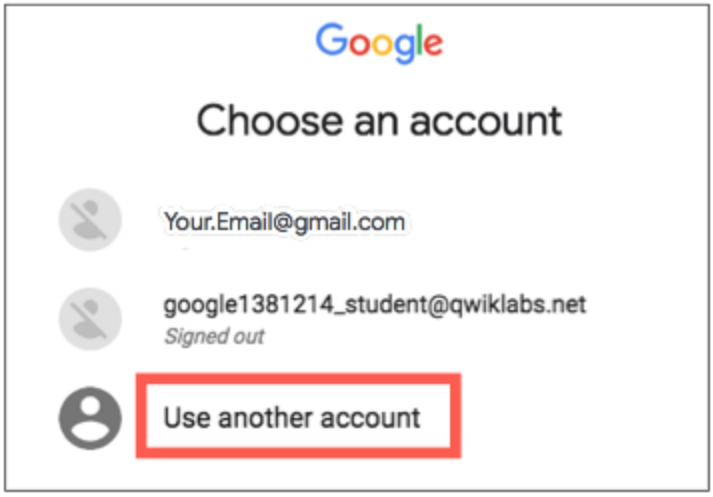




2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

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

   If you see the **Choose an account** page, click **Use Another Account**.



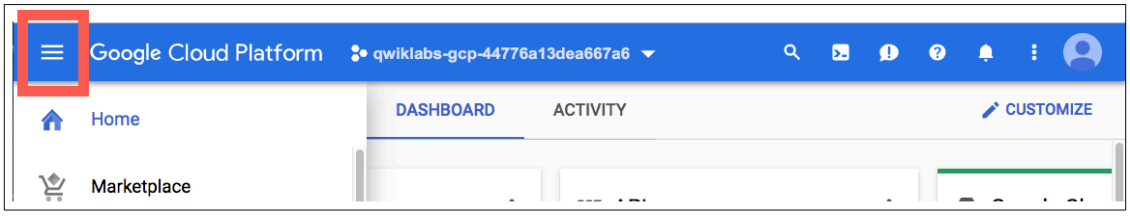
3. In the **Sign in** page, paste the username that you copied from the left panel. Then copy and paste the password.

Important: You must use the credentials from the left panel. Do not use your Google Cloud Training credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. 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.

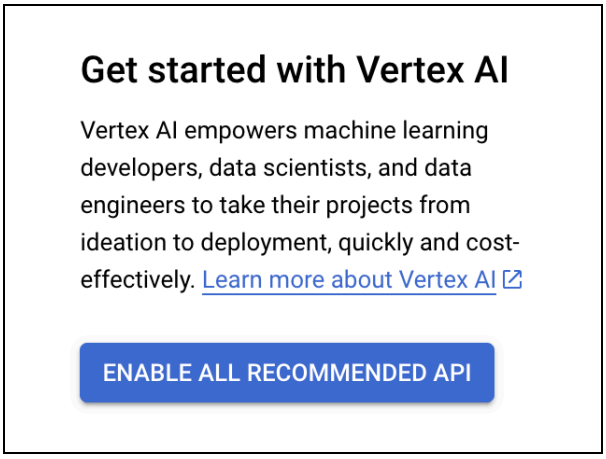
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Task 1. Set up your environment

Enable the Vertex AI API

1. In the Google Cloud Console, on the **Navigation menu**, click **Vertex AI** or Utilize the search bar to find **Vertex AI.**
2. Within the "Get Started with Vertex AI" section, locate and click on the option that says



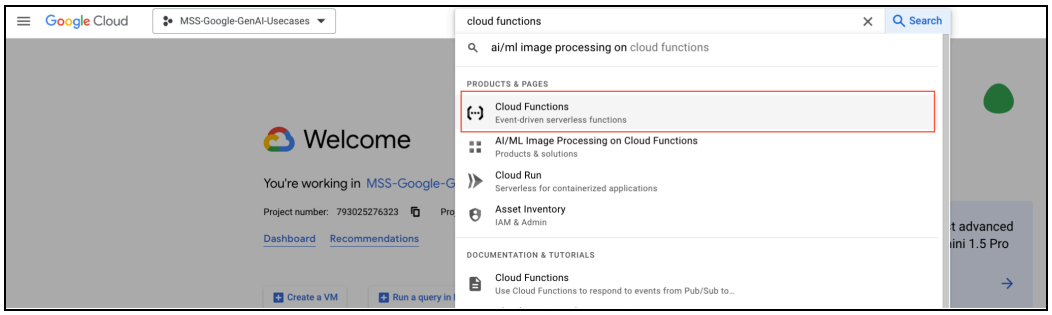
"ENABLE ALL RECOMMENDED API."

Task 2. Create a Cloud Function to Fetch Weather Data from Weatherbit.io

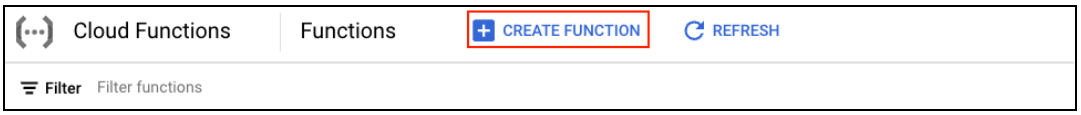
Open Google Cloud Console and create a new Cloud Function with name **get_current_weather**. Use Flask as the HTTP handler for the function's endpoint. Inside the function, make a call to the Weatherbit.io API to retrieve the current weather data.

Follow the steps below:

1. Search for the **Cloud Run Functions** in the GCP console, and select Cloud Functions from the search result



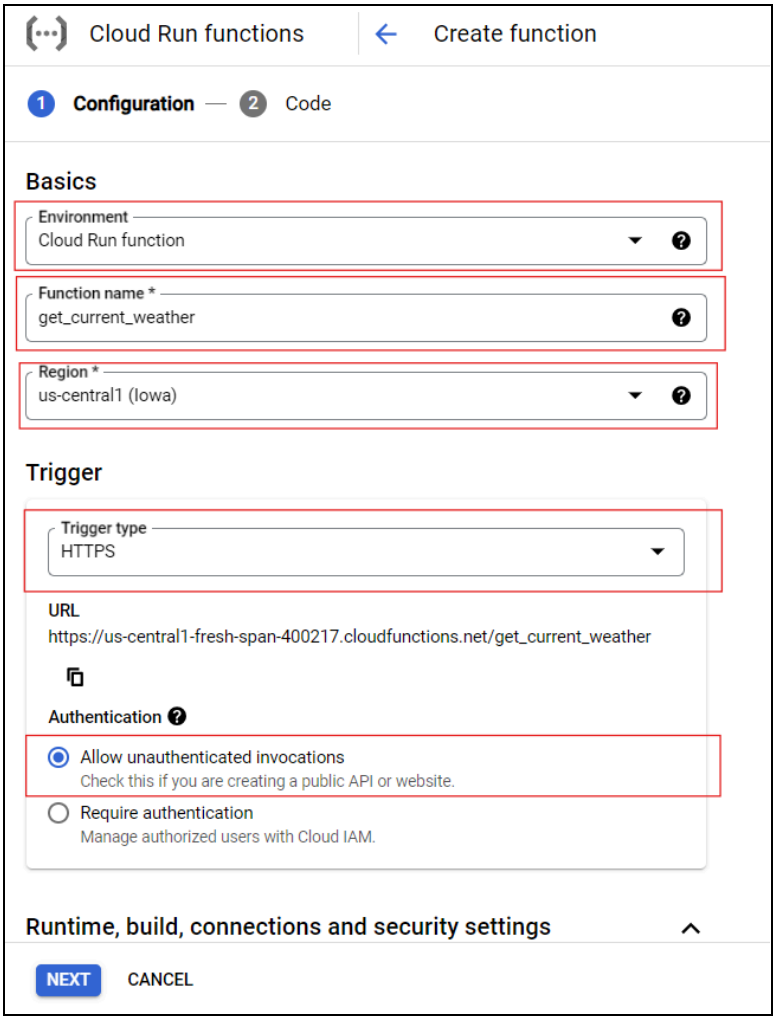
2. Create the Cloud Function
    a. Click on CREATE FUNCTION



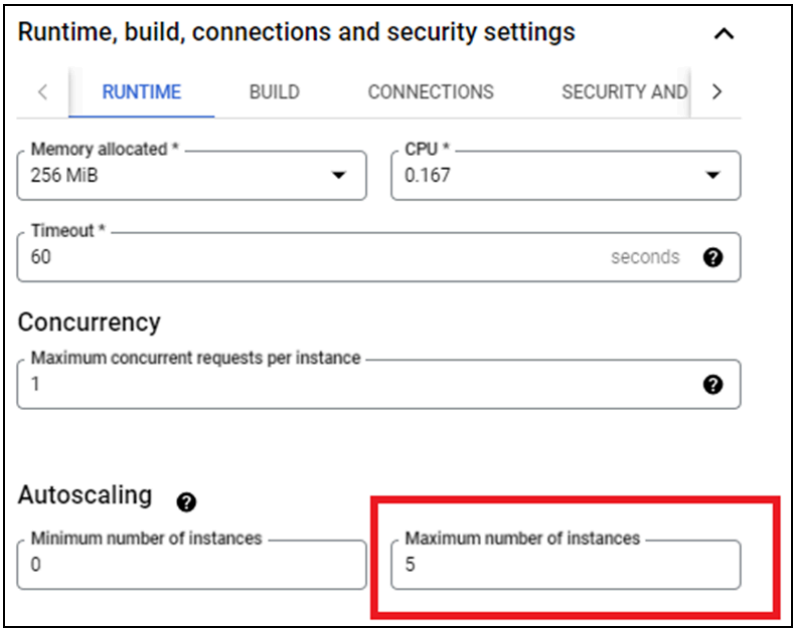
3. Update the parameters, and set the parameters as marked in **RED.**

Note: Function name is a name entered by the user

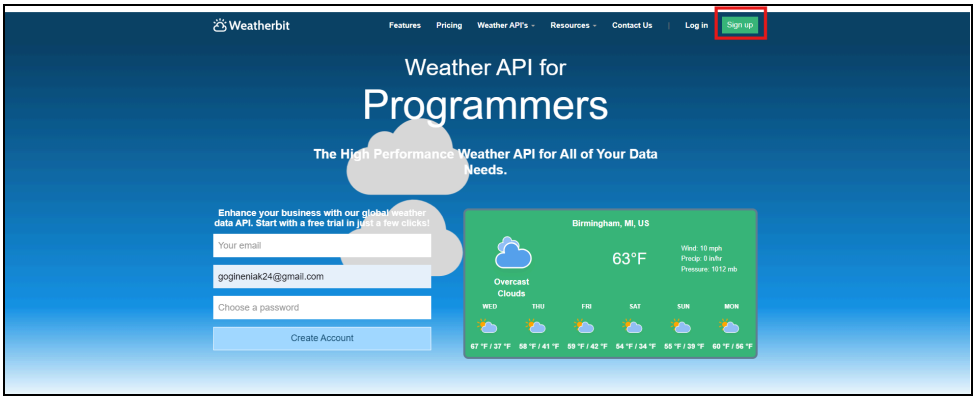
a. Environment : **Cloud Run Function**
b. Function name : **get_current_weather**
c. Region : **us-central1 (Iowa)**
d. Trigger type : **HTTPS**
e. Select "**Allow unauthenticated invocations**"


Cloud Run functions
Create function
1 Configuration — 2 Code
Basics
Environment
Cloud Run function
Function name *
get_current_weather
Region *
us-central1 (Iowa)
Trigger
Trigger type
HTTPS
URL
https://us-central1-fresh-span-400217.cloudfunctions.net/get_current_weather
Authentication
Allow unauthenticated invocations
Check this if you are creating a public API or website.
Require authentication
Manage authorized users with Cloud IAM.
Runtime, build, connections and security settings
NEXT
CANCEL

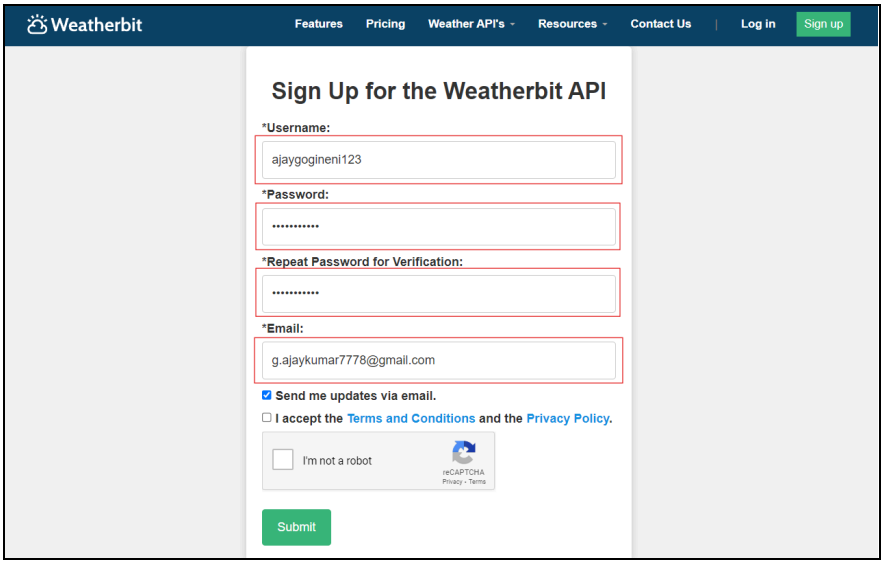
Note: Maximum numbers of instances should not exceed 5 (Refer below screenshot). This is the maximum number assigned for Qwiklabs.



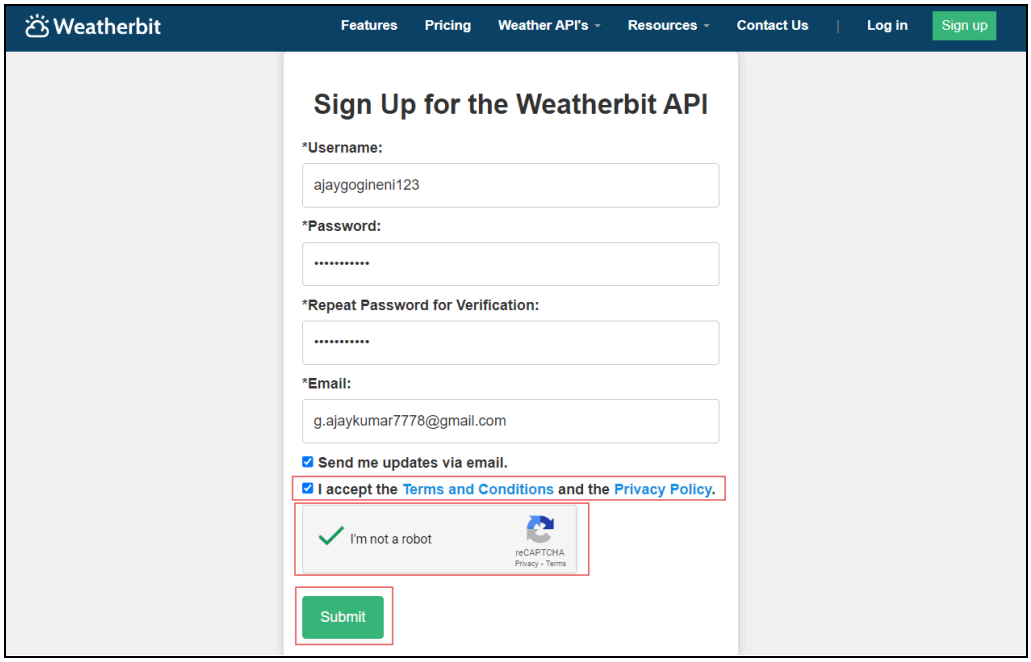
4. Get the Weather API Key from Weatherbit.io
   a. Go to https://www.weatherbit.io/
   b. Click on Sign up to create a new account



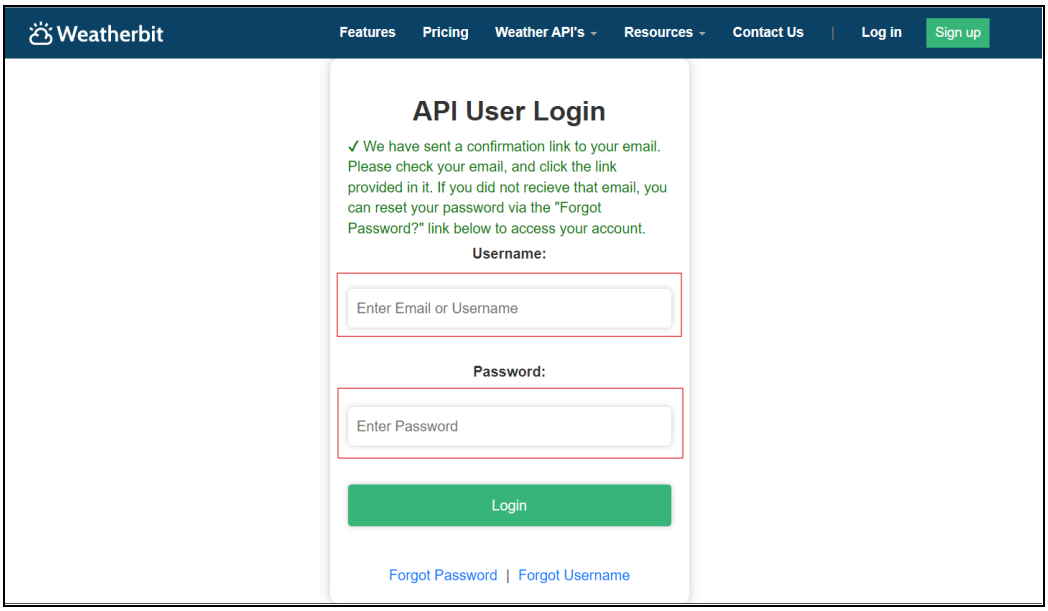
   c. Enter the email id and password



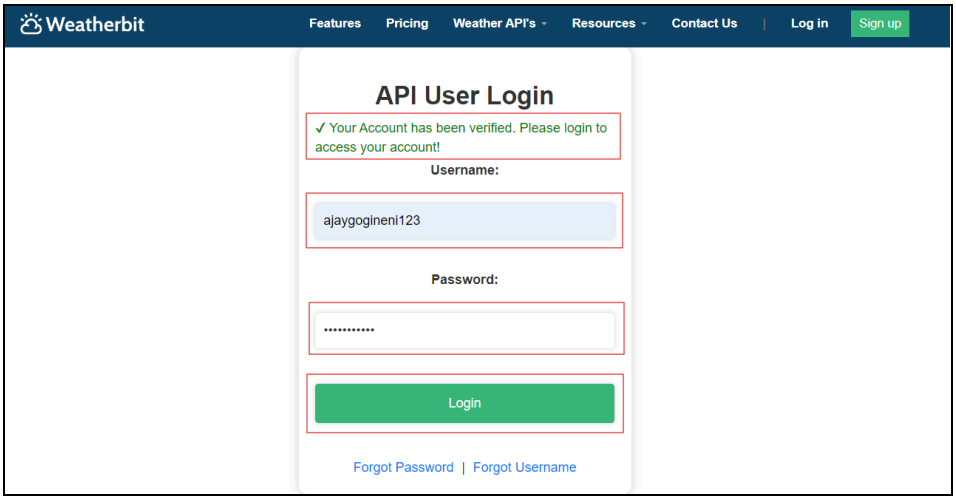
d. Accept the Terms and Conditions, Verify captcha and click on Submit



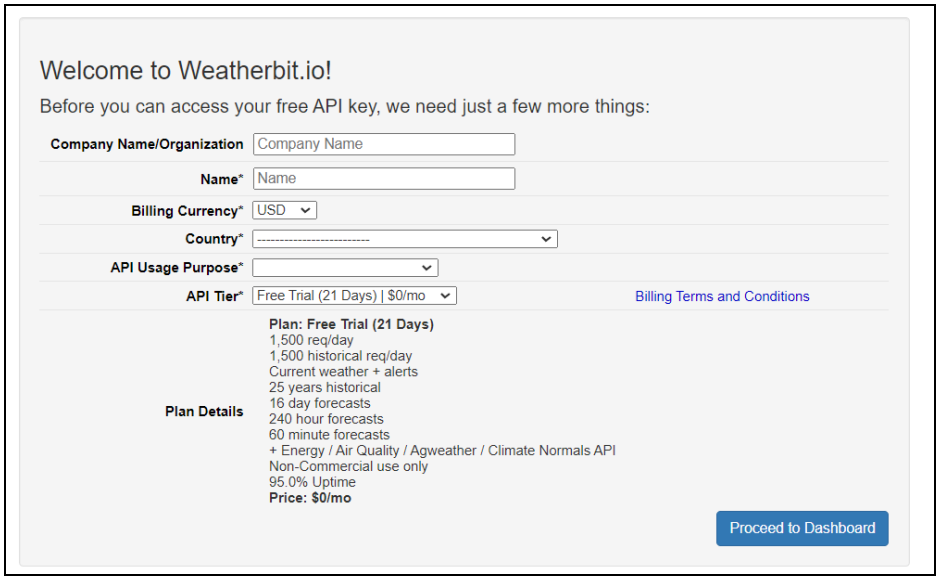
e. Verify the email id by clicking on the link sent over email.



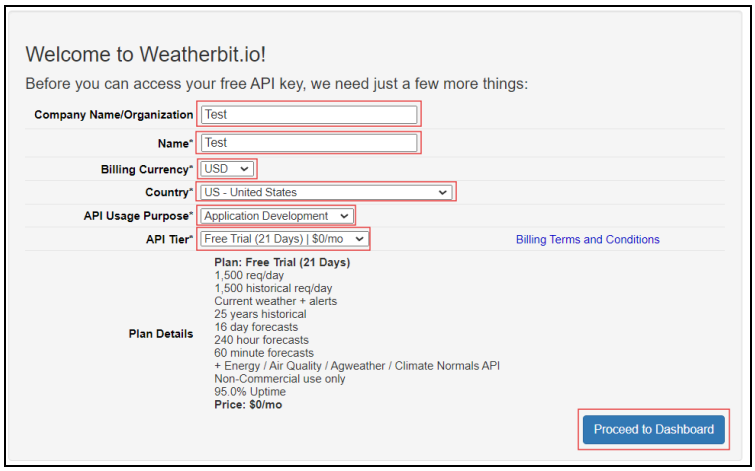
f. Login using the details after verifying the account



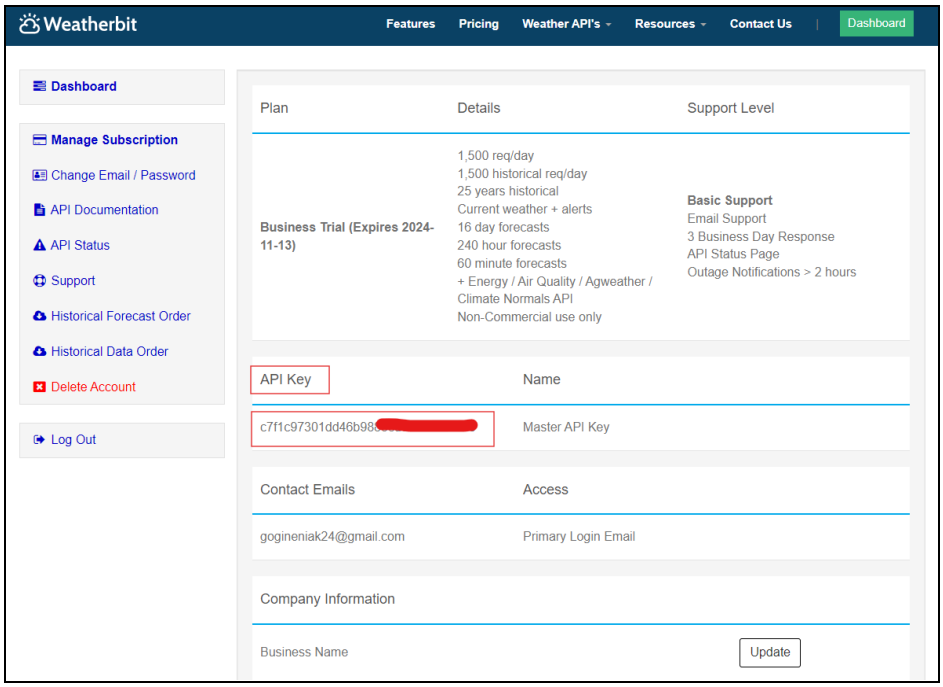
g. You will be redirected to a page where you can select your plan



h. Fill the details. Select Free Trial for API Tier. Click on Proceed to Dashboard

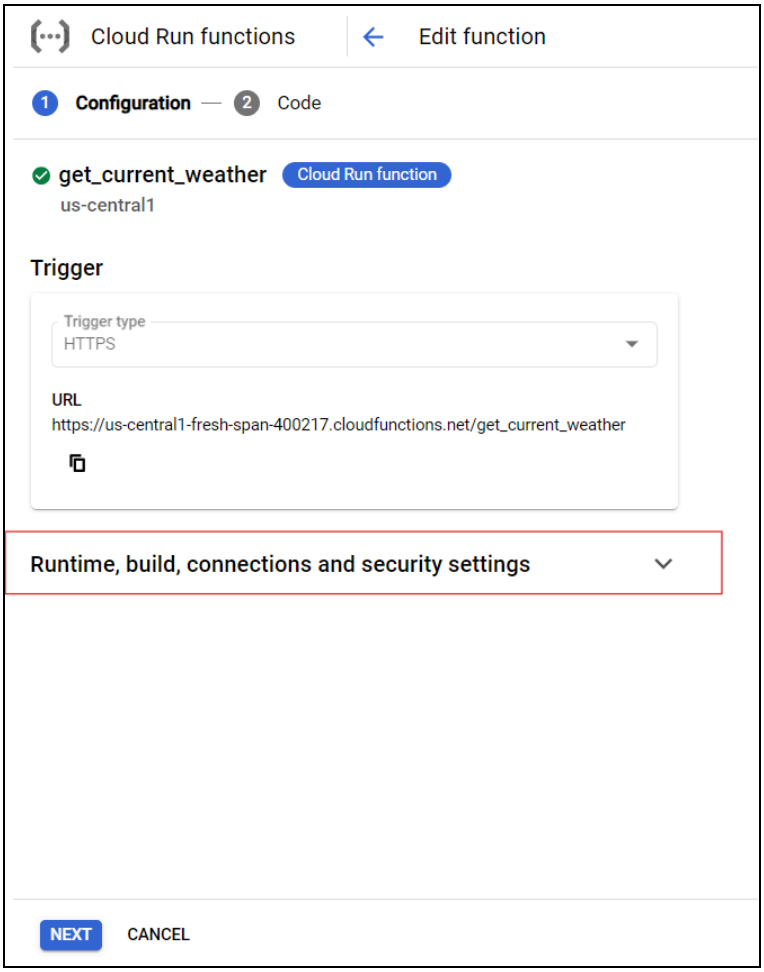


i. You will be redirected to the Dashboard where you can get the API key



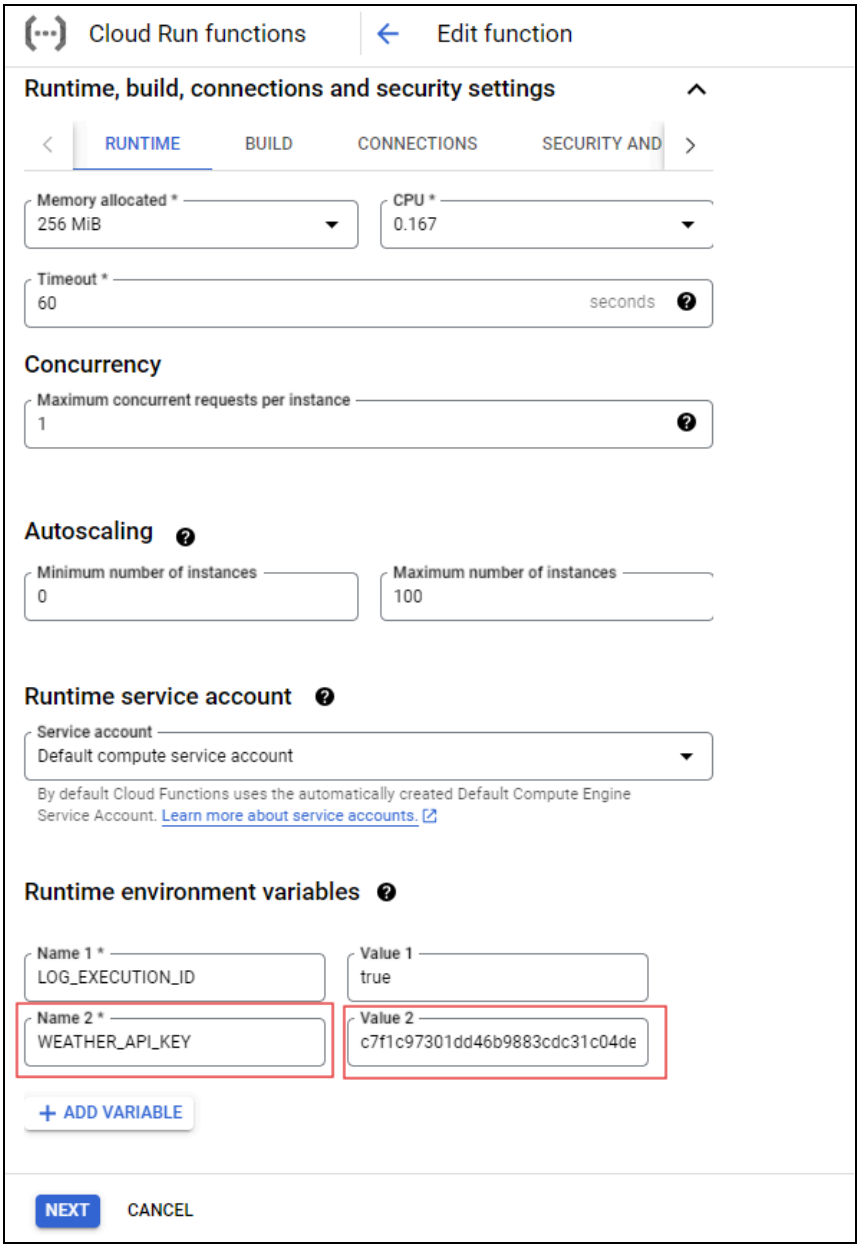
5. Add the API key as an environment variable in the Cloud Function

Expand Runtime, build, connections and security settings menu

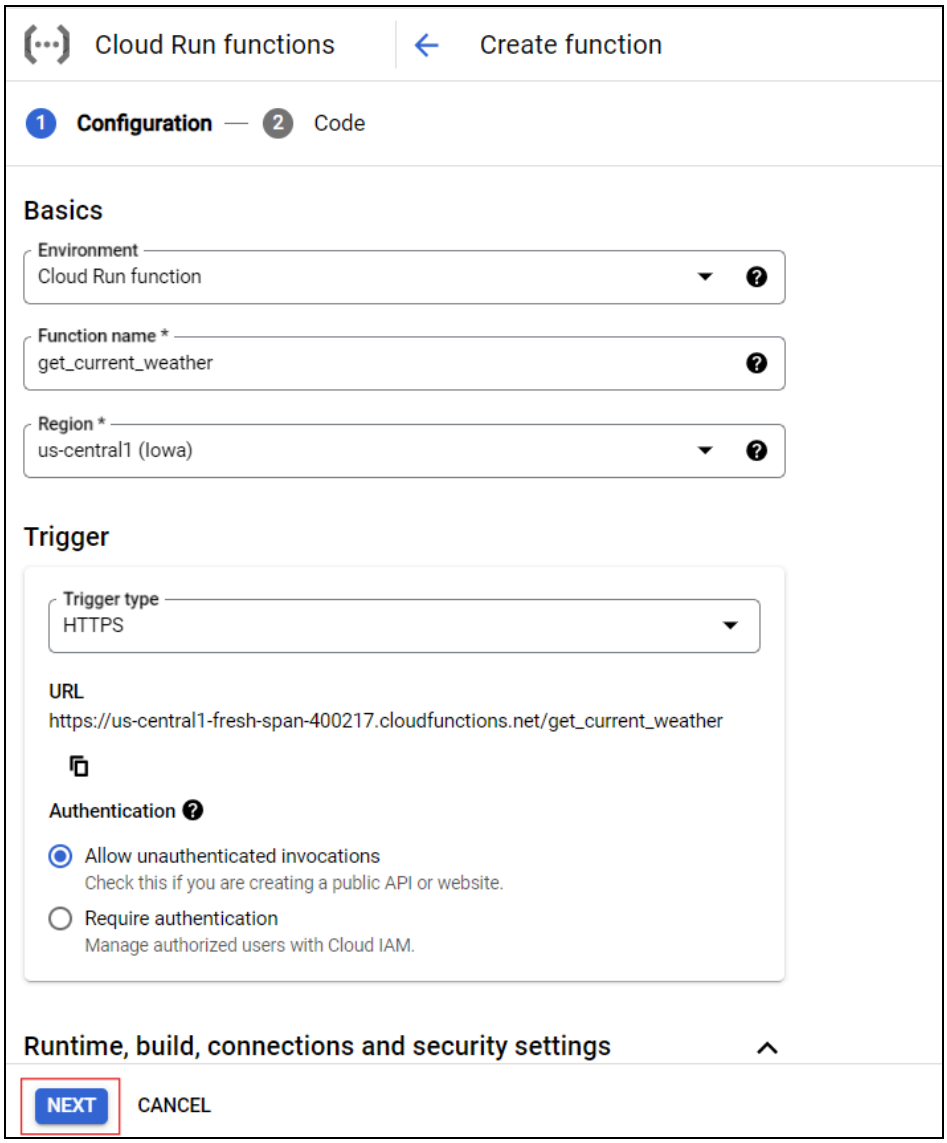


Go to Runtime environment variables

Add a variable with name WEATHER_API_KEY and value as the API KEY obtained from Weatherbit.io



6. Click on NEXT to open the Editor



6. Update the code to point to a Flask endpoint in the Editor.
(Note that the Function name from Step 1, Entry point and the function name in main.py should be same)

a. Select Python 3.12 in Runtime
b. Update Entry point to get_current_weather
c. Update the python code in main.py

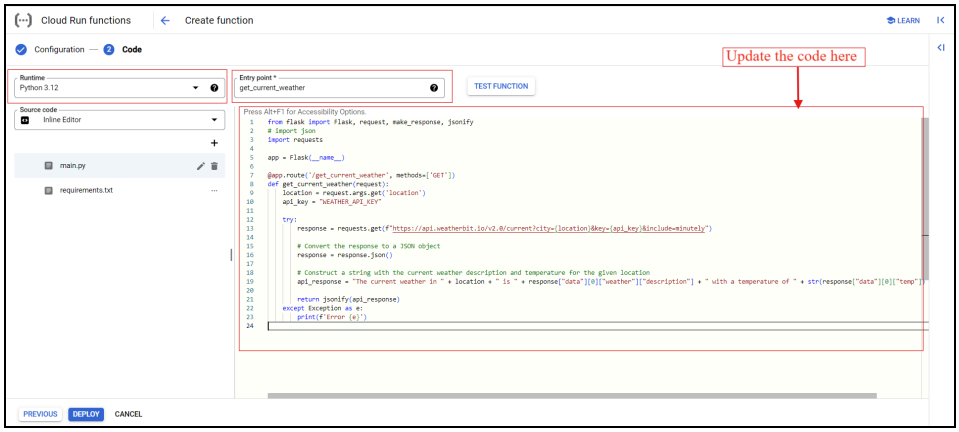
This code gets weather through an API code

Code for main.py:

Note: "Please verify code indentation after copying." You can copy the code in a text format from the github location: https://github.com/themlguy-tf/generative-ai/tree/main/agent-builder

---

```
from flask import Flask, request, make_response, jsonify
import requests
import os
app = Flask(__name__)

@app.route('/get_current_weather', methods=['GET'])
def get_current_weather(request):
    location = request.args.get('location')
    api_key = os.getenv('WEATHER_API_KEY')

    try:
        response =
requests.get(f"https://api.weatherbit.io/v2.0/current?city={location}&key={api_key}&include
=minutely")

        # Convert the response to a JSON object
        response = response.json()

        # Construct a string with the current weather description and temperature for the given
location
        api_response = "The current weather in " + location + " is " +
response["data"][0]["weather"]["description"] + " with a temperature of " +
str(response["data"][0]["temp"]) + "°C."

        return jsonify(api_response)
    except Exception as e:
        print(f'Error {e}')
```

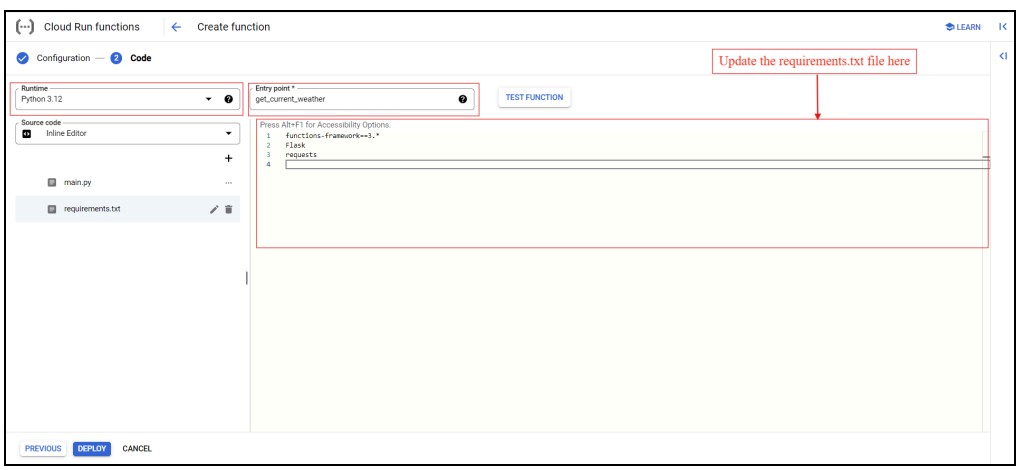


d. Update the requirements.txt file

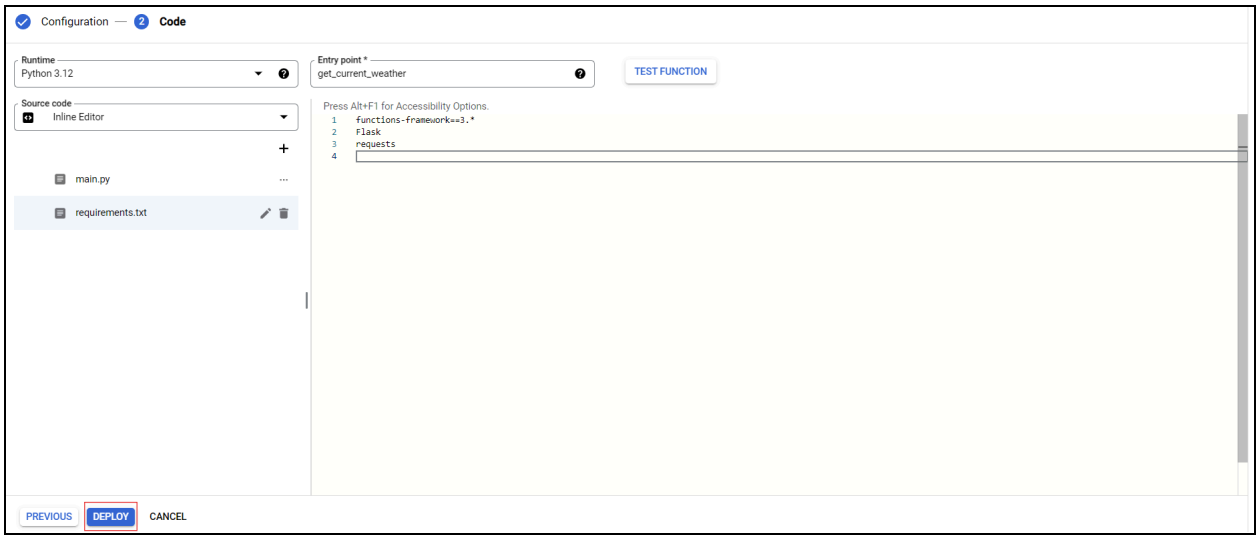
e. **requirements.txt**

---

functions-framework==3.*
Flask
requests

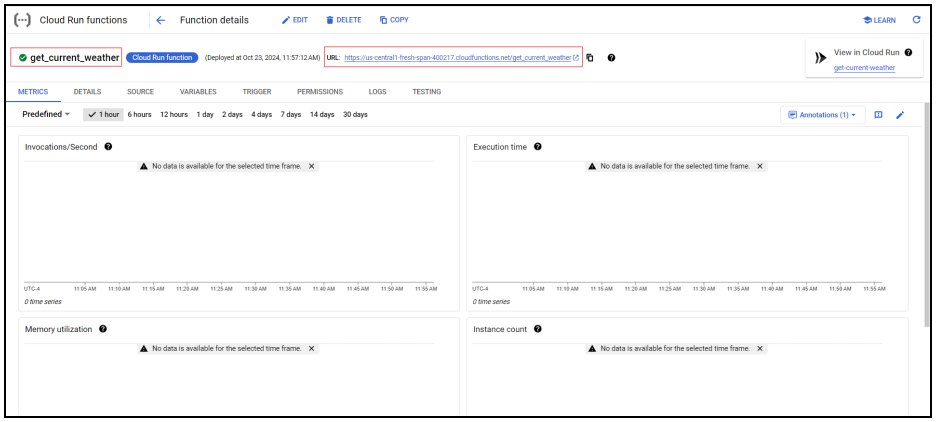


6. Click on Deploy to deploy the cloud function



This will deploy the cloud function that can get weather from Weatherbit.io. The cloud function is deployed here:
https://us-central1-fresh-span-400217.cloudfunctions.net/get_current_weather
* Note: This url will be different for your application

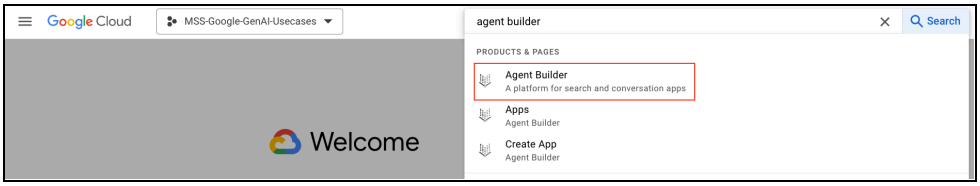
Debug Steps: Make sure the Entry point and function name are same



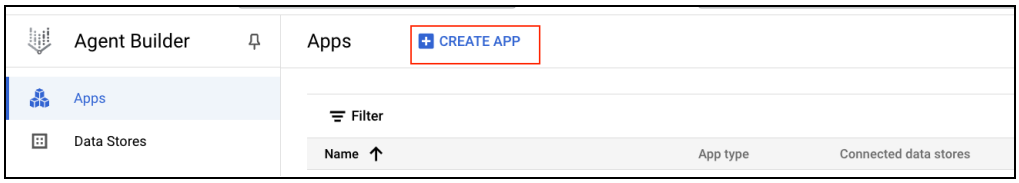
Task 3. Navigate to the Agent Builder service and set up an OpenAPI tool for the get_current_weather cloud function.

This tool will be used as an input to the agent. A description of the tool and the corresponding schema should be provided here. The output format of get_current_weather cloud function and the input of this tool should be the same. If there is a mismatch, there will be errors.

1. Go to Agent Builder on Google cloud console

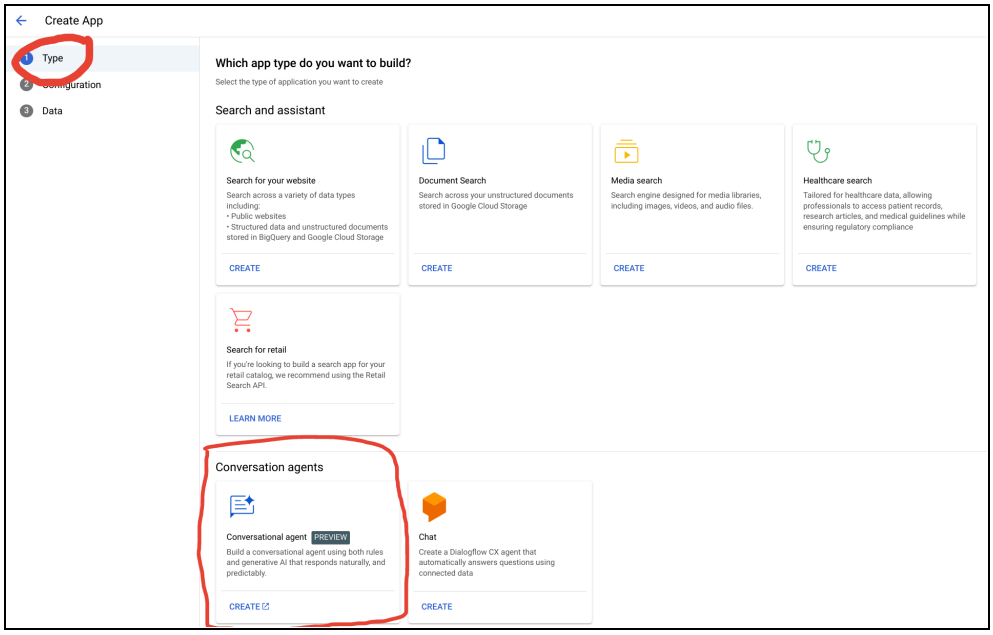


2. Click on Create App to start building a new Agent builder

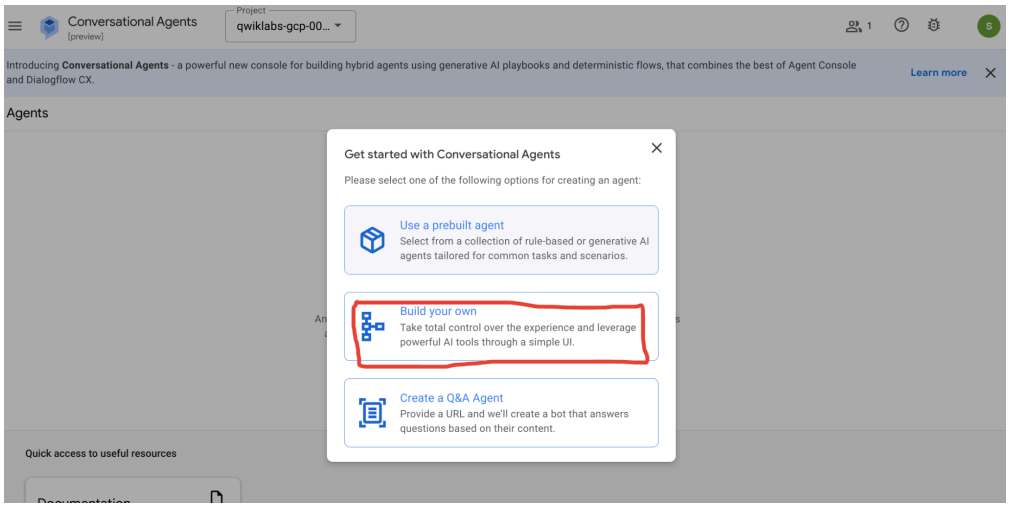


3. Select Apps > Type > Agent

In Create App, select the Conversation agent

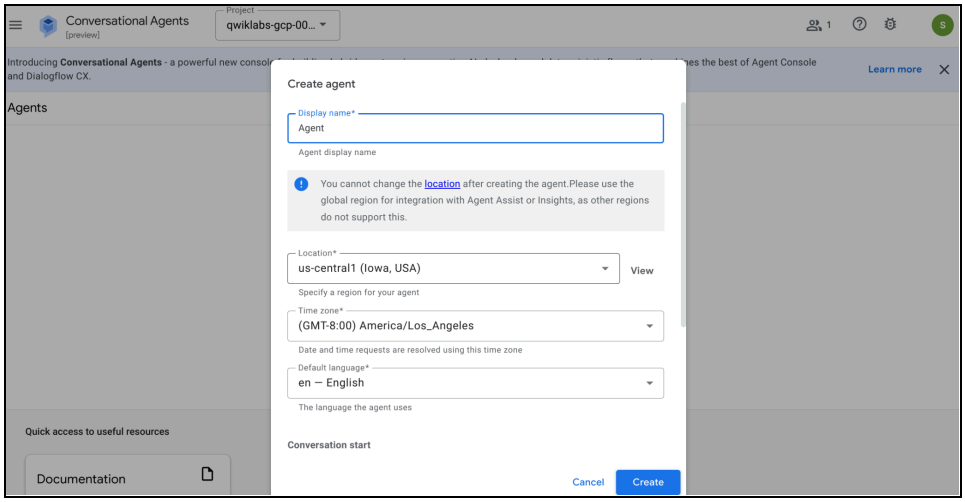


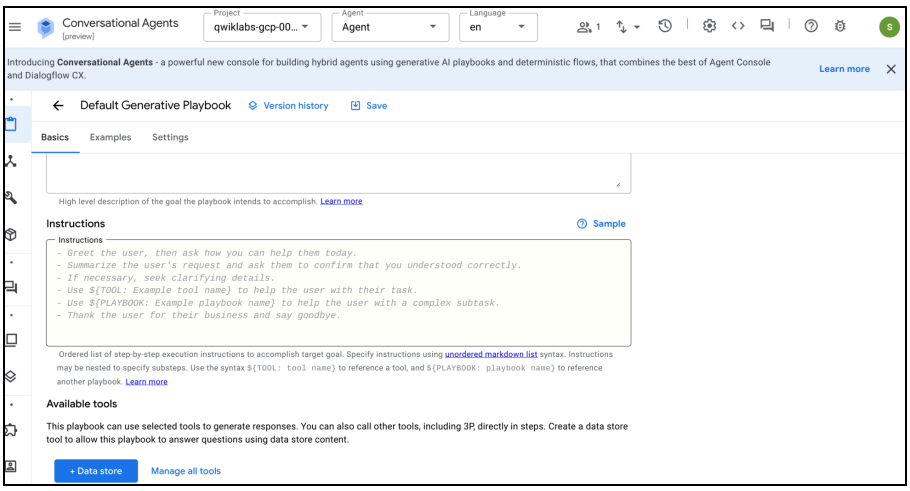
Select “Build your own”



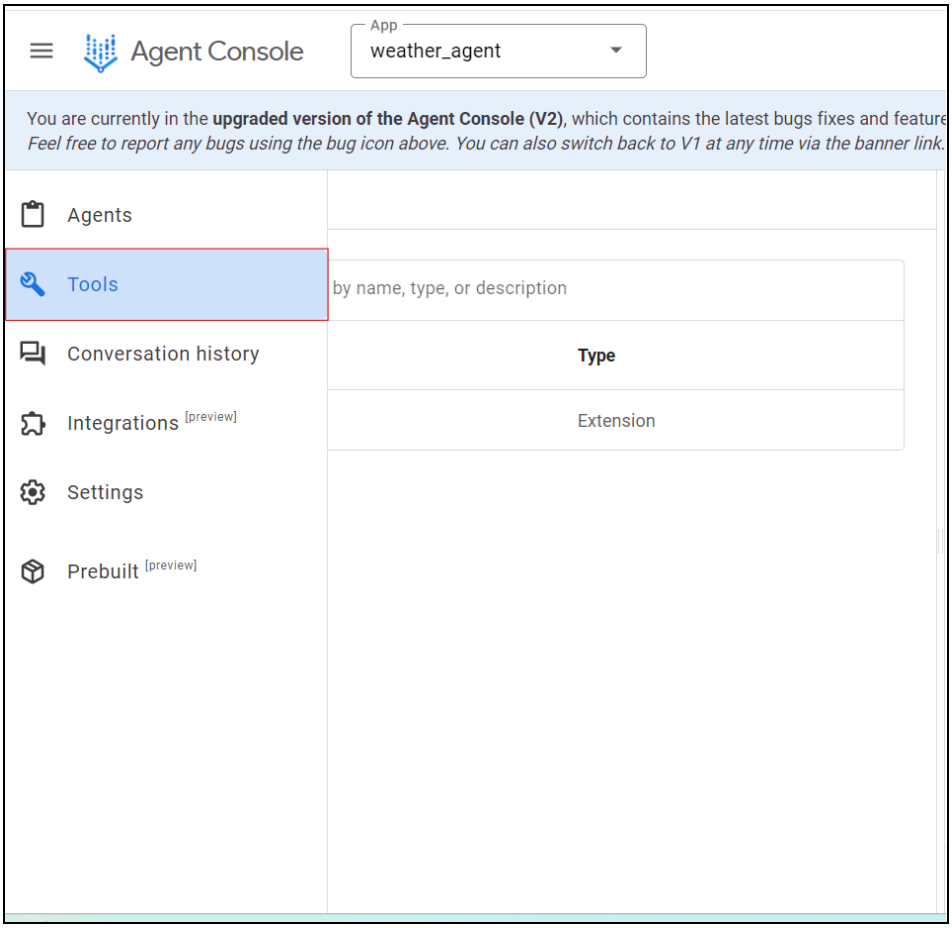
Click “Create”. Please note that it is a Playbook - Generative AI agent.

Update the Display name of the agent to **Weather Agent** and click on Create

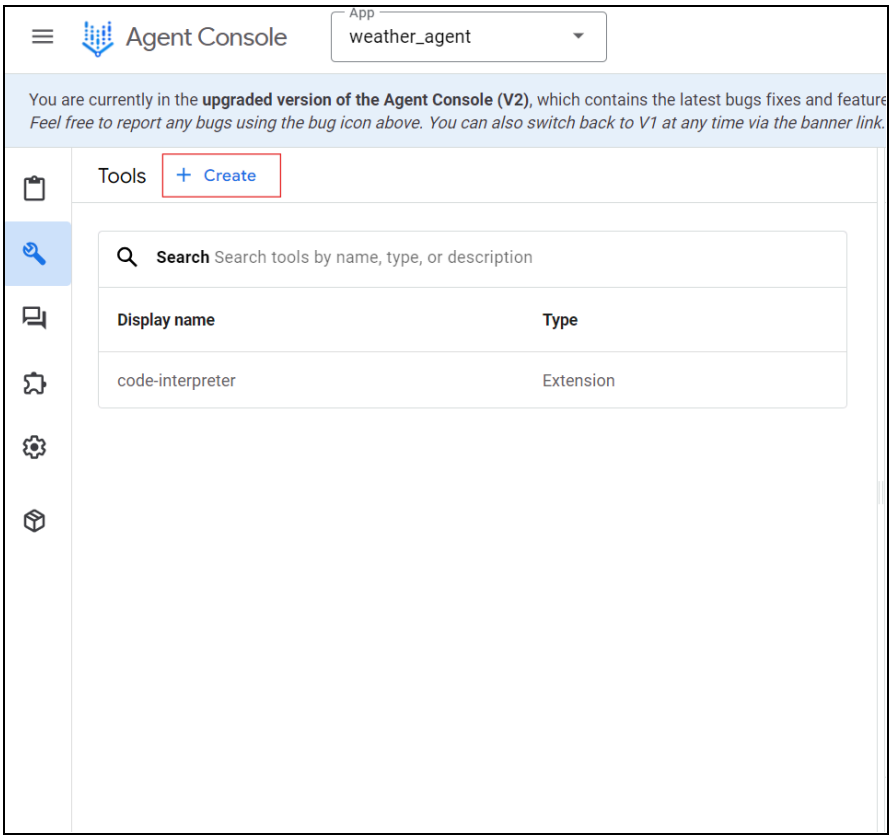




4. Select Tools from the left panel



5. Click on Create



6. Update the following parameters:
    a. Tool name : get_current_weather
    b. Type : OpenAPI
    c. Update the Description to:
    Use this tool to get the real time weather at a particular location. Pass the location as an input parameter to get the weather at that location.
    d. Select YAML under schema
    e. Update the schema to:
    * Note: Use the url corresponding to get_current_weather cloud function created previously in the schema. Set the paths to get_current_weather.

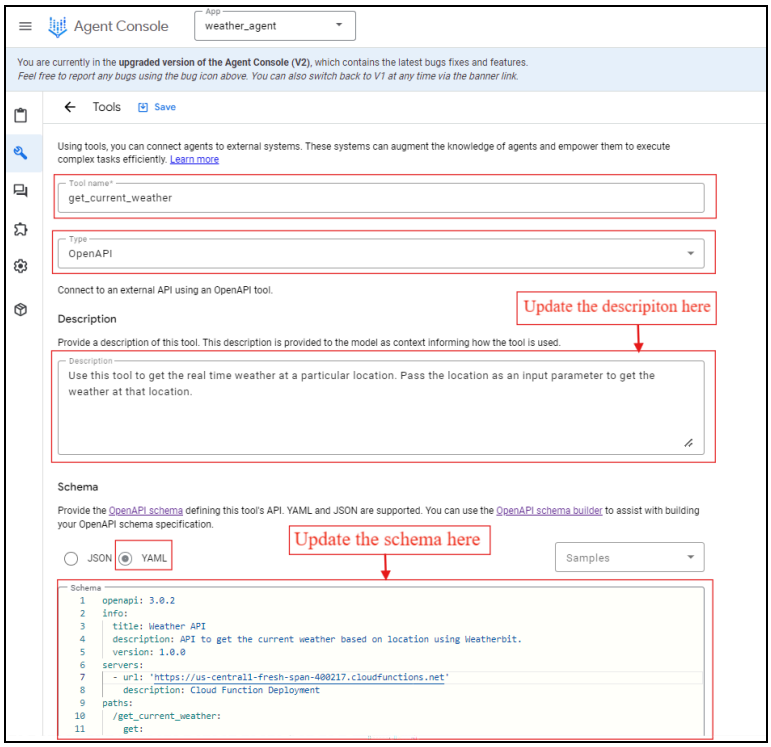
Note: "Please verify code indentation after copying." You can copy the code in a text format from the github location: https://github.com/themlguy-tf/generative-ai/tree/main/agent-builder

---

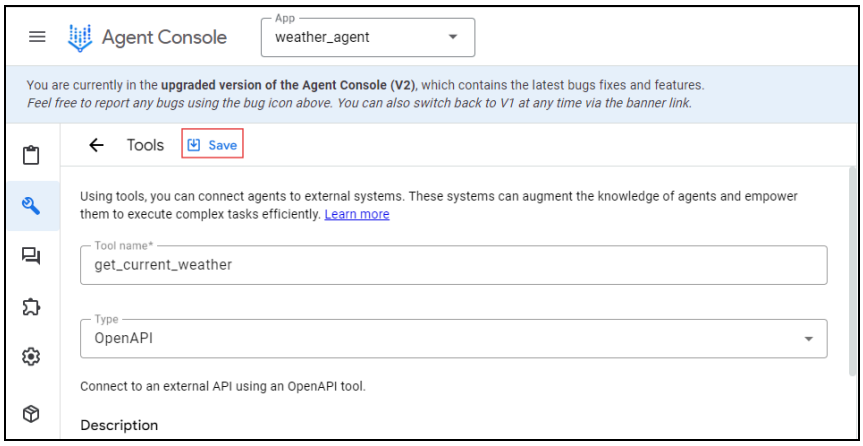
```
openapi: 3.0.2
info:
  title: Weather API
```

```yaml
  description: API to get the current weather based on location using Weatherbit.
  version: 1.0.0
servers:
  - url: 'https://us-central1-genaiinprogress.cloudfunctions.net'
    description: Cloud Function Deployment
paths:
  /get_current_weather:
    get:
      summary: Get Current Weather
      description: Retrieve the current weather for a specified location.
      parameters:
        - in: query
          name: location
          schema:
            type: string
          required: true
          description: The name of the city to get the weather for.
      responses:
        '200':
          description: Successful response with weather information
          content:
            application/json:
              schema:
                type: object
                properties:
                  weather:
                    type: string
                    example: >-
                      The current weather in London is Clear sky with a
                      temperature of 13.3°C.
        '400':
          description: Bad Request (missing or invalid location parameter)
        '500':
          description: 'Internal Server Error (e.g., API issues or connectivity failure)'
components:
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      in: header
```

```
    name: WEATHER_API_KEY
security:
  - ApiKeyAuth: []
```

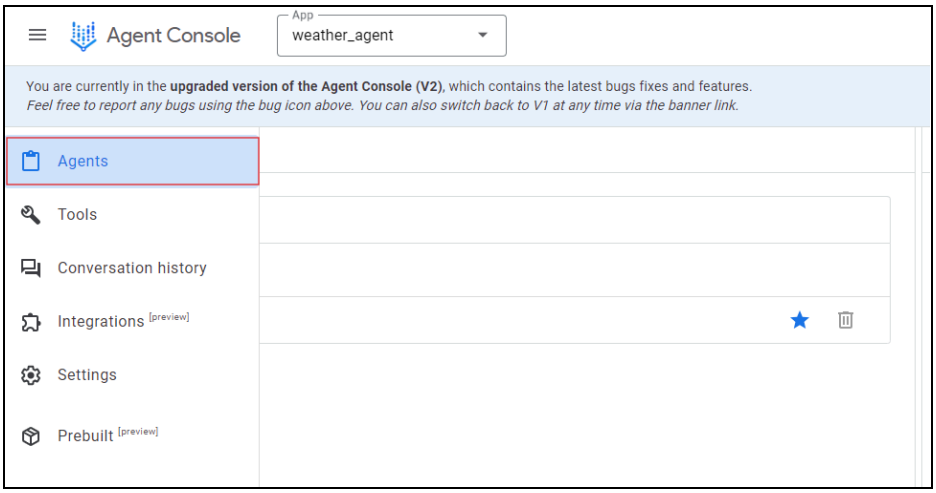


7. Save the Tool

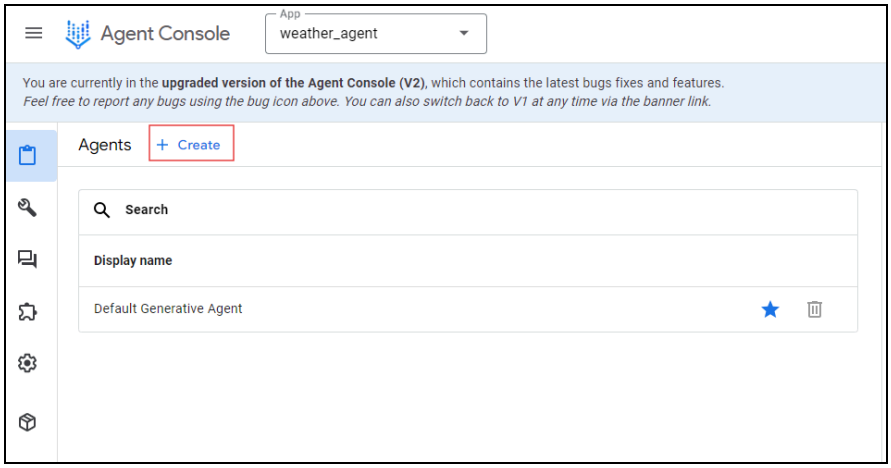


Step 3: Create an Agent using the tool created in Step 2 as one of the inputs

1. Click on “Playbook” in the left panel



2. Click on Create to create a new Agent



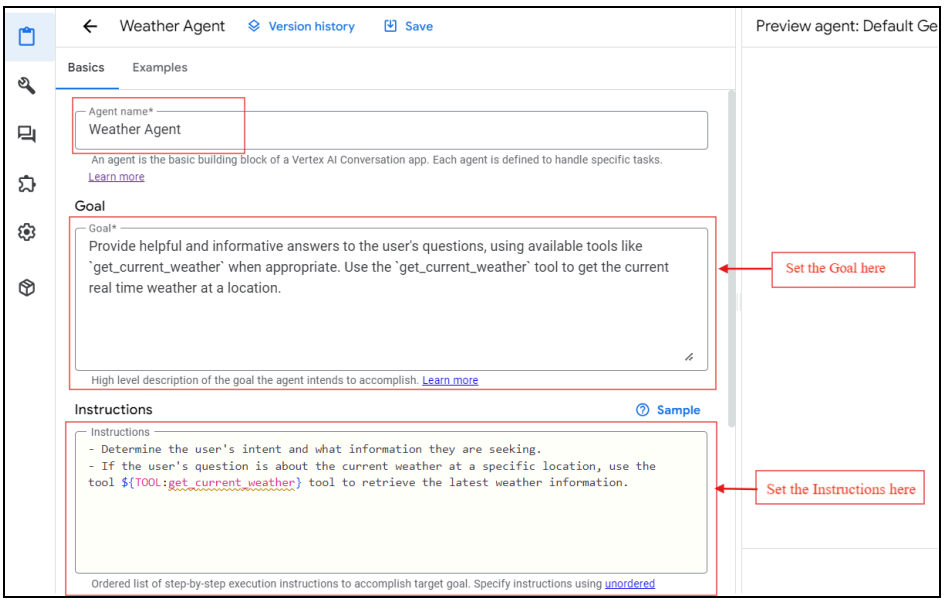
3. Update the following fields:

   a. Agent name : Weather Agent *Note: Change the name of the agent as per requirement
   b. Set the **Goal** of the Agent to :

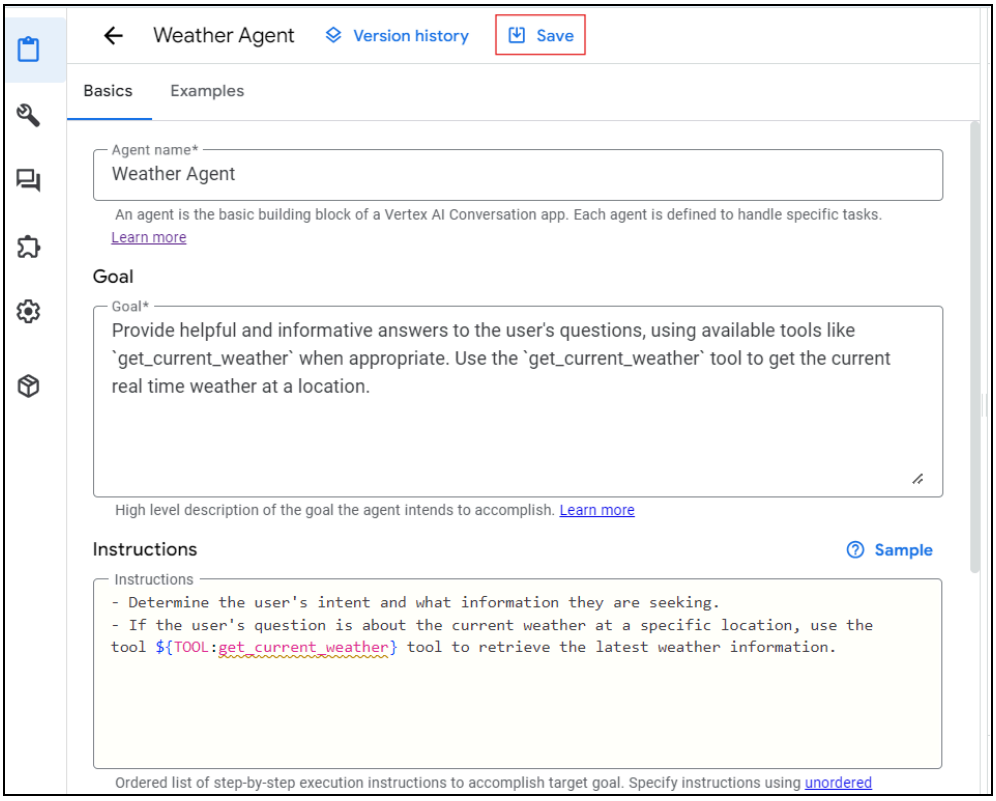
Provide helpful and informative answers to the user's questions, using available tools like `get_current_weather` when appropriate. Use the `get_current_weather` tool to get the current real time weather at a location.

c. Specify the **instructions** to be followed by the Agent. To reference a tool, use the syntax **${TOOL: tool name}**:

- Determine the user's intent and what information they are seeking.
- If the user's question is about the current weather at a specific location, use the ${TOOL:get_current_weather} tool to retrieve the latest weather information.



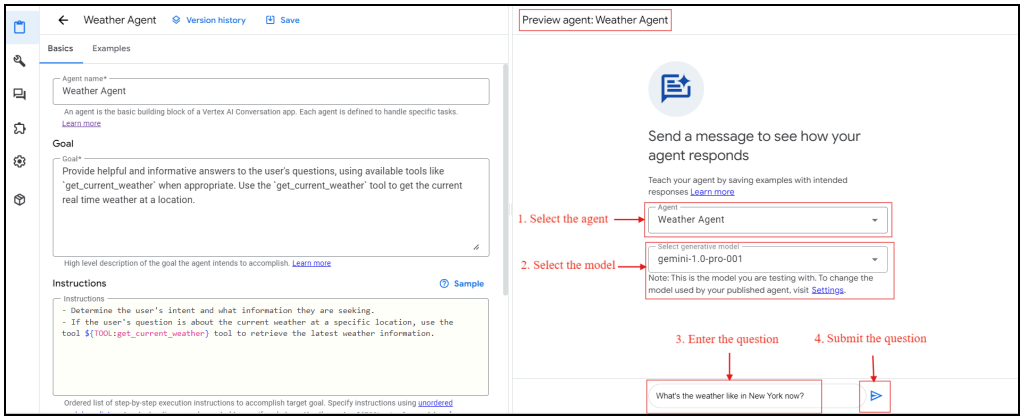
4. Save the Agent



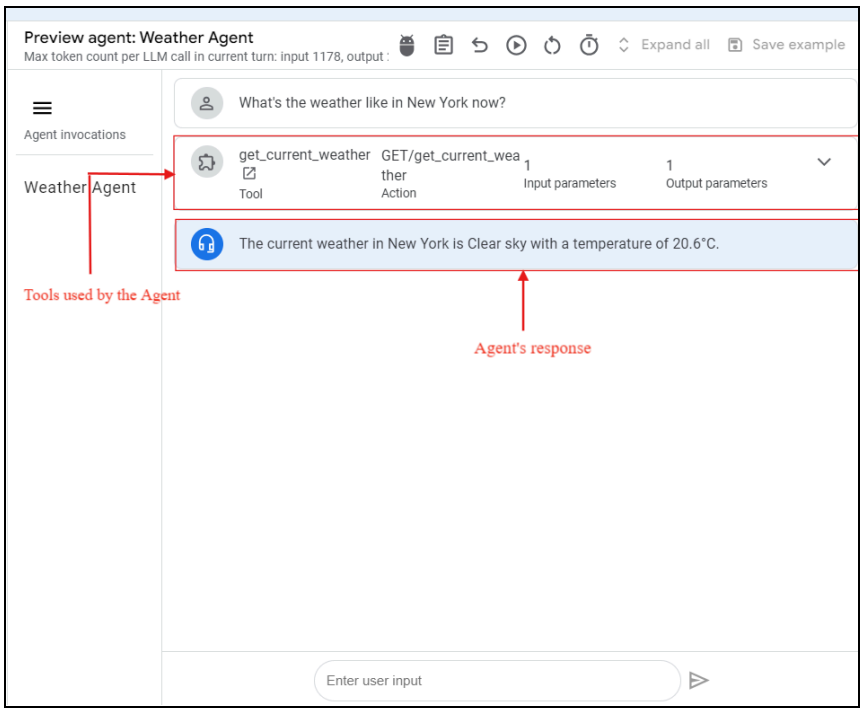
Step 4: Testing the Agent created

Follow these steps to test the Agent Builder:
Use the Preview agent: section to ask questions

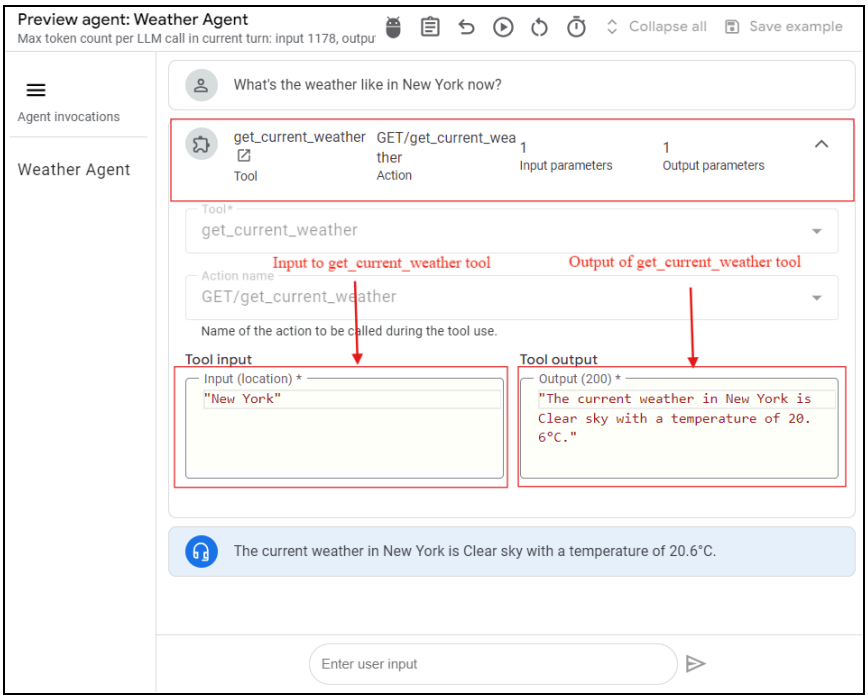
1. Go to Conversation History from the tool bar on the left side
2. Select the agent created, Weather Agent. *Note: Select the correct agent
3. Select the GenAI model, **gemini-1.0-pro-001 Or the latest version of Gemini model**
4. Enter the question in the text box. Example question: 'What's the weather like in New York now?'
5. Click the **submit** button or press Enter



Get a list of tools used by the agent to come up with the final answer



Expand the tool to view its inputs and outputs in detail.



Congratulations!

In this lab, you have successfully learnt about how to build an agent using the Agent Builder using external APIs

Lab Author: Naresh Jasotani(nareshjasotani@google.com)


You're Awesome!


Kudos