Implementing an AI Chatbot with Dialogflow

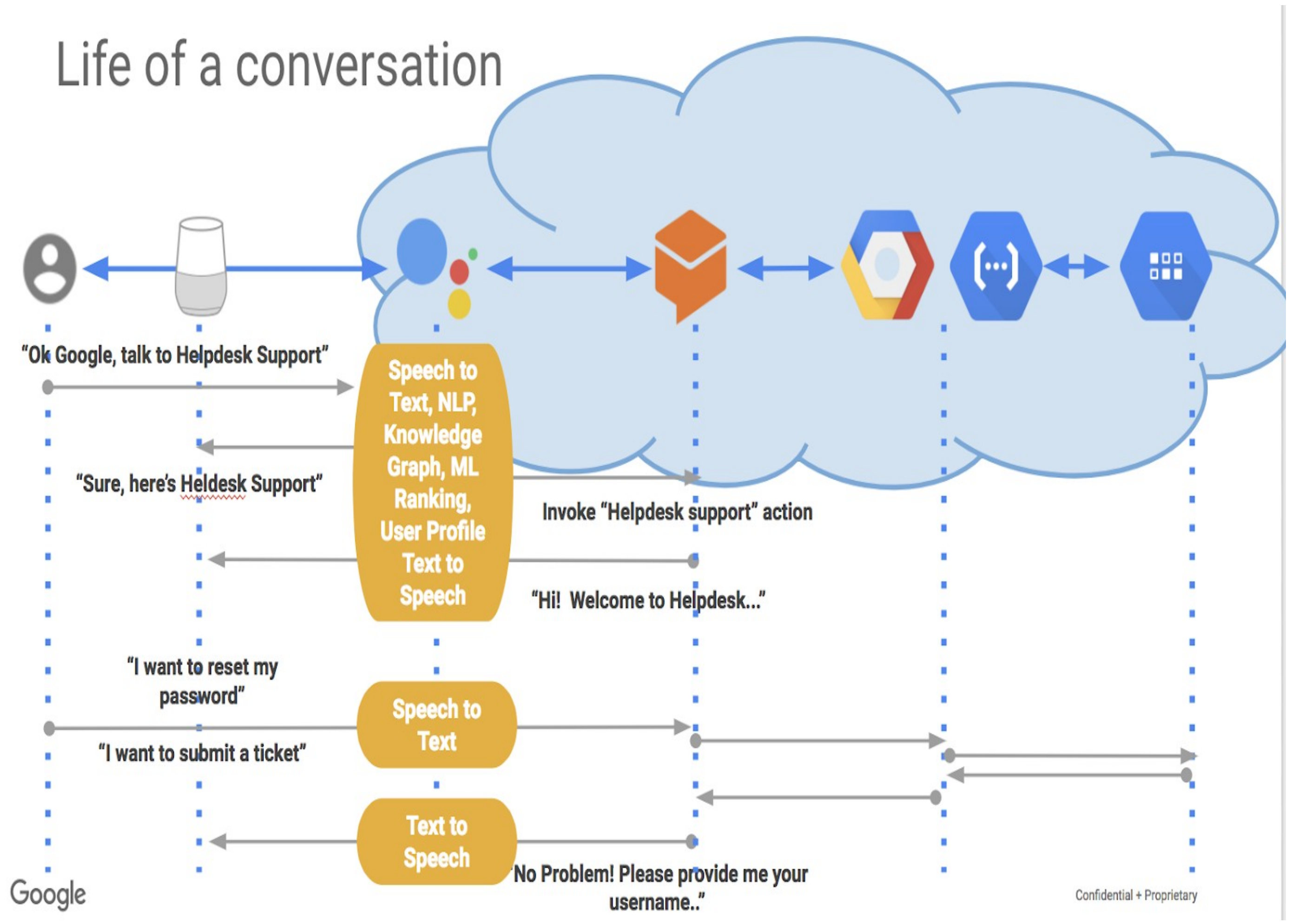
1 hourFree

GSP078

Overview

Dialogflow is a natural language understanding platform that makes it easy to design and integrate a conversational user interface into your mobile app, web application, device, bot, etc. Dialogflow can analyze multiple types of input from your customers, including text or audio inputs (like from a phone or voice recording). It can also respond to your customers either through text or with synthetic speech.

In this lab, you will build a Google Assistant chatbot that submits helpdesk tickets. The following is a diagram of the chatbot application on Google Cloud:


Life of a conversation
"Ok Google, talk to Helpdesk Support"
"Sure, here's Heldesk Support"
Speech to Text, NLP, Knowledge Graph, ML Ranking, User Profile Text to Speech
Invoke "Helpdesk support" action
"Hi! Welcome to Helpdesk..."
"I want to reset my password"
"I want to submit a ticket"
Speech to Text
Text to Speech
"No Problem! Please provide me your username.."
Google
Confidential + Proprietary

The exercises are ordered to reflect a common cloud developer process. You will:

- Set up your lab and learn how to work with Dialogflow and your Google Cloud environment.
- Deploy a simple Dialogflow application.
- Deploy a simple cloud function within Google Cloud to connect with Dialogflow.

- Test your chatbot.

What you'll learn

By the end of this lab, you will have an understanding of the following:

- Basics concepts and constructs of Dialogflow, including intent, entity and context.
- Chatbot workflow.
- Life of a conversation.

Prerequisites

This is an **fundamental level** lab. Before taking it, you should be comfortable with at least the basics of machine learning and natural language processing. Here are some Qwiklabs that can get you up to speed:

- Cloud Natural Language API: Qwik Start
- Google Cloud Speech API: Qwik Start
- Entity and Sentiment Analysis with the Natural Language API
- Classify Text into Categories with the Natural Language API

Once you are prepared, scroll down to dive into Dataflow.

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 Qwiklabs 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:

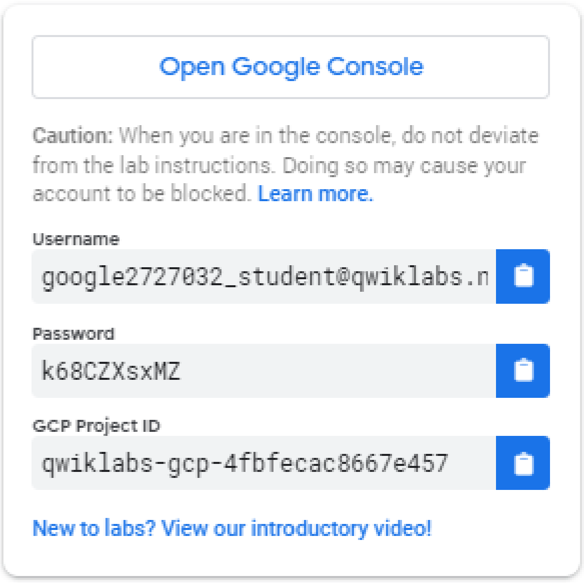
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

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.

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.


Open Google Console

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.

Username

google2727032_student@qwiklabs.n

Password

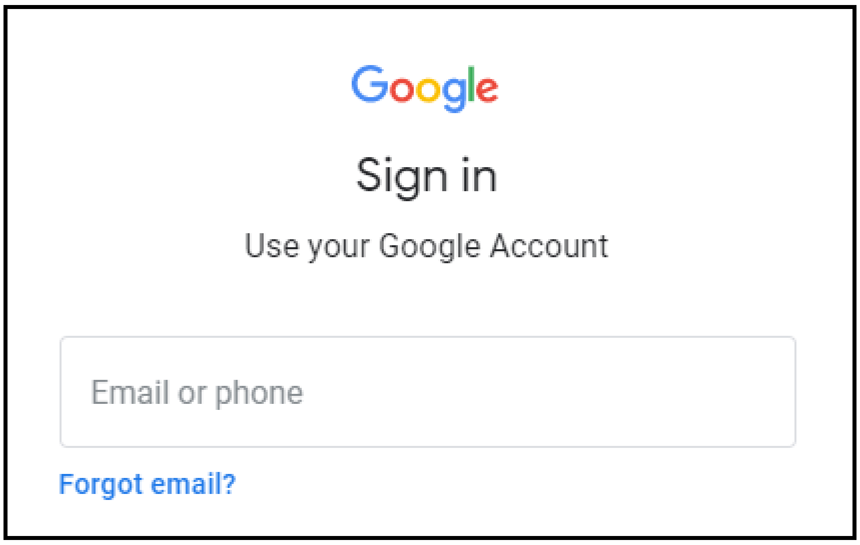
k68CZXsxMZ

GCP Project ID

qwiklabs-gcp-4fbfecac8667e457

New to labs? View our introductory video!

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.


Google

Sign in

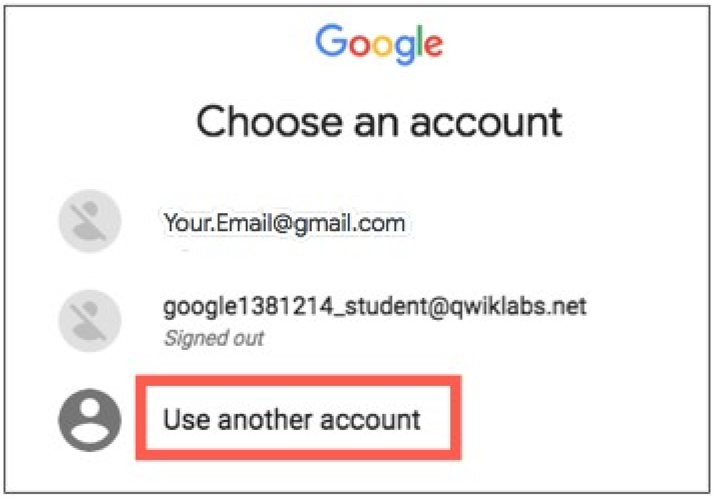
Use your Google Account

Email or phone

Forgot email?

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

Choose an account page, click **Use Another**


Google

Choose an account

Your.Email@gmail.com

google1381214_student@qwiklabs.net
Signed out

Use another account

Account.

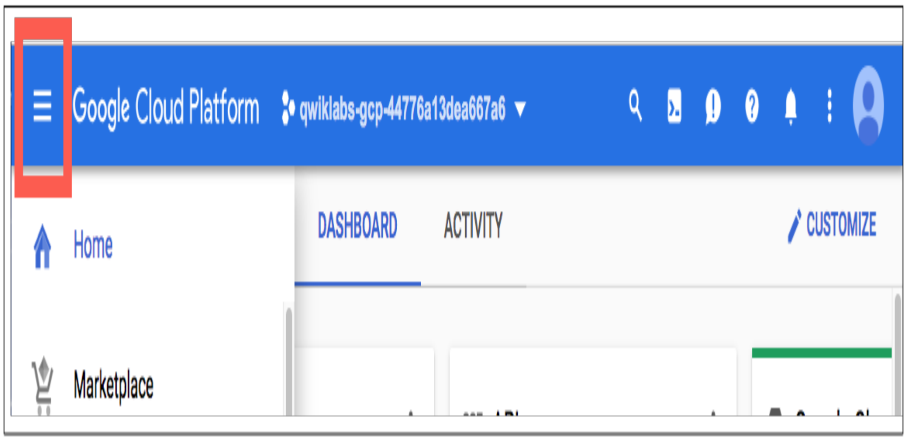
3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs 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. 

Dialogflow Concepts and Constructs

Dialogflow is a conversation building tool. It takes the human language and cleverly splits it into intents and arguments.

Agents are best described as NLU (Natural Language Understanding) modules. These can be included in your app, product, or service, and transforms natural user requests into actionable data. This transformation occurs when a user input matches one of the intents inside your agent.

Intents are the predefined or developer-defined components of agents that process a user's request. An intent represents a mapping between what a user says and what action should be taken by your software.

Intent interfaces have the following sections:

- User says
- Action
- Response
- Contexts

  Entities are powerful tools used for extracting parameter values from natural language inputs. Any important data you want to get from a user's request will have a corresponding entity.

  The entities used in a particular agent will depend on the parameter values that are expected to be returned as a result of the agent functioning. In other words, a developer does not need to create entities for every possible concept mentioned in the agent – only for those needed for actionable data.

  There are 3 types of entities:

- **System**: defined by Dialogflow
- **Developer**: defined by a developer
- **User**: built for each individual end-user in every request

  It's important to distinguish between the three different types of system entities:

- **System mapping**: Has reference values
- **System enum**: Has no reference values
- **System composite**: Contains other entities with aliases and returns object type values

  Contexts represent the current context of a user's request. This is helpful for differentiating phrases which may be vague or have different meanings depending on the user's preferences, geographic location, the current page in an app, or the topic of conversation.

  For example, if a user is listening to music and finds a band that catches their interest, they might say something like: "I want to hear more of them". As a developer, you can include the name of the band in the context with the request, so that the agent can use it in other intents.

Fulfillment is a webhook that allows you to pass information from a matched intent into a web service and get a result from it.

All of this new information may be overwhelming, but not to panic — it should all come together once you start developing your Google Assistant chatbot in the following section.

Deploy a simple Dialogflow application to submit helpdesk tickets

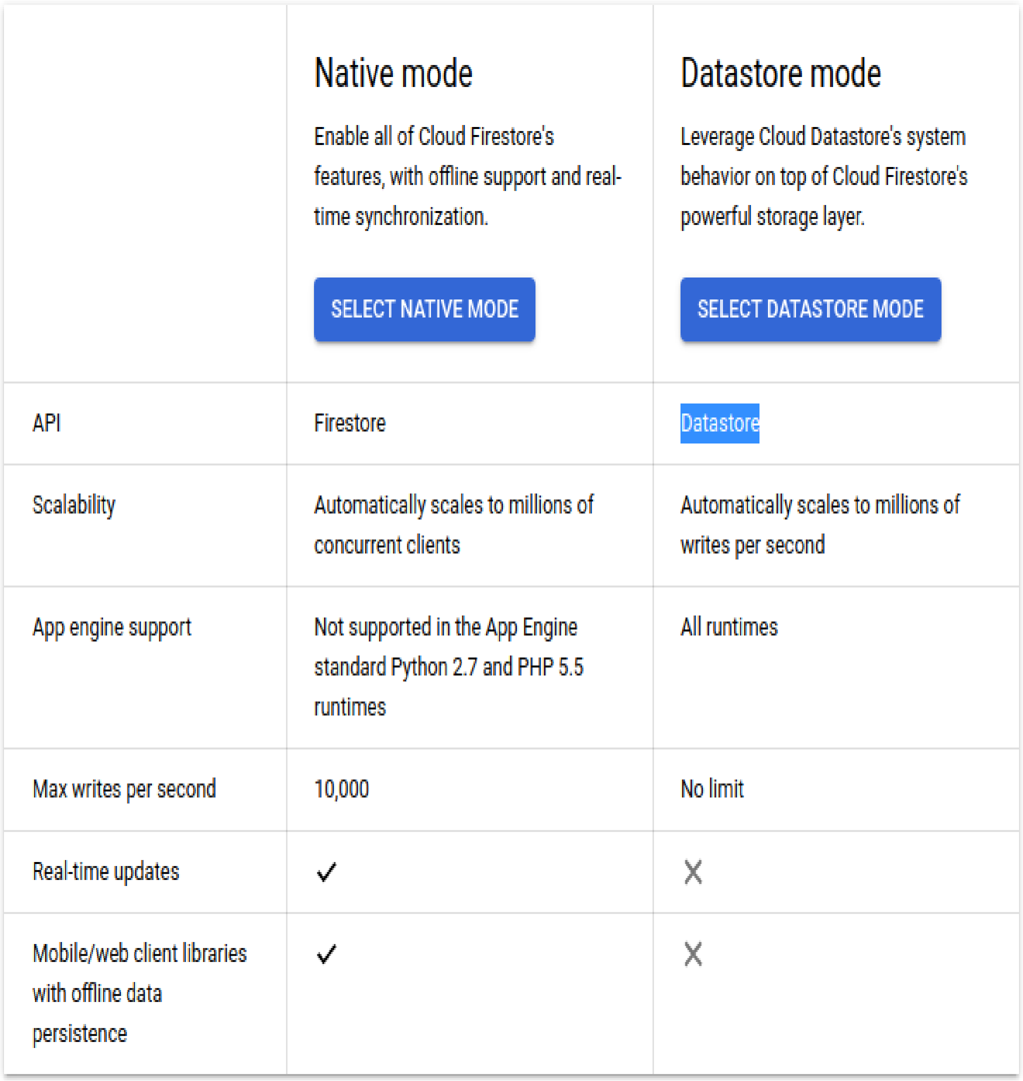
Enable Cloud Datastore

Cloud Datastore is a highly scalable NoSQL database for your applications. Cloud Datastore automatically handles sharding and replication, providing you with a highly available and durable database that scales automatically to handle your applications' load.

1. Create a new database instance, open the Datastore section in the Cloud Console.

2. In the Console, go to **Navigation menu** > **Datastore**

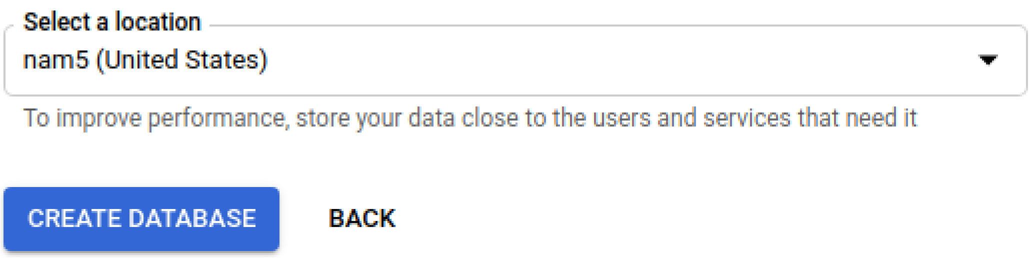
3. The screen should look like this:



	Native mode Enable all of Cloud Firestore's features, with offline support and real-time synchronization. SELECT NATIVE MODE	Datastore mode Leverage Cloud Datastore's system behavior on top of Cloud Firestore's powerful storage layer. SELECT DATASTORE MODE
API	Firestore	Datastore
Scalability	Automatically scales to millions of concurrent clients	Automatically scales to millions of writes per second
App engine support	Not supported in the App Engine standard Python 2.7 and PHP 5.5 runtimes	All runtimes
Max writes per second	10,000	No limit
Real-time updates	✓	✗
Mobile/web client libraries with offline data persistence	✓	✗

4. Click the **SELECT DATASTORE MODE**

5. From the dropdown **Select a location** choose **nam5(United State)**

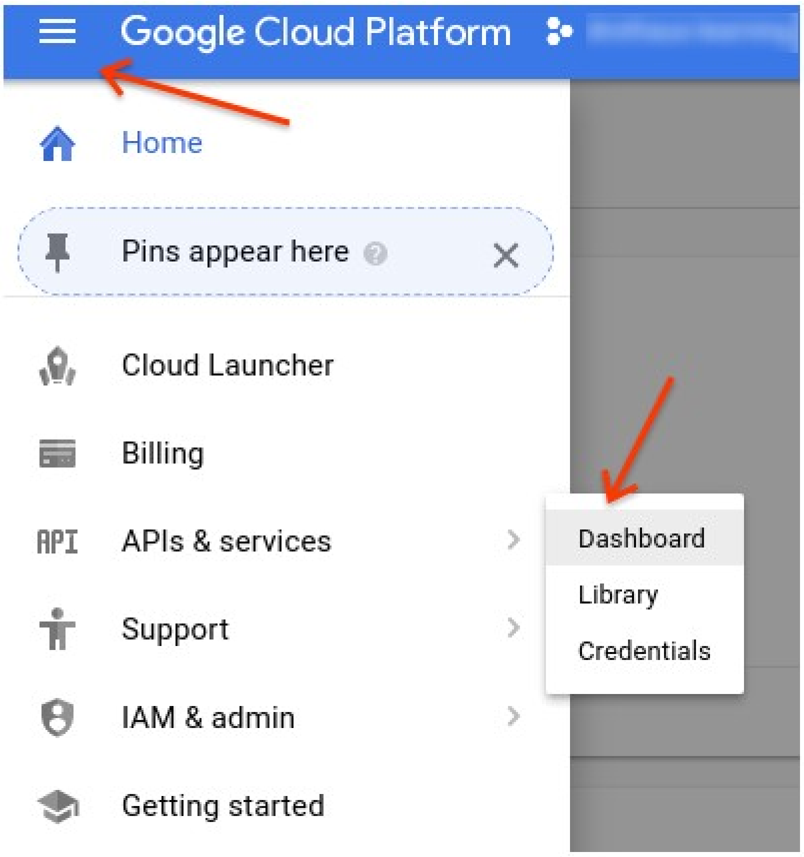


Select a location
nam5 (United States)
To improve performance, store your data close to the users and services that need it

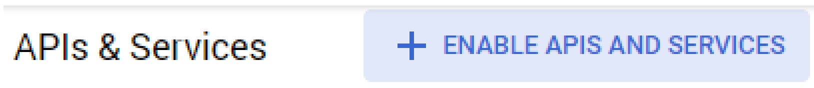
6. Click **CREATE DATABASE**

Create a Dialogflow agent

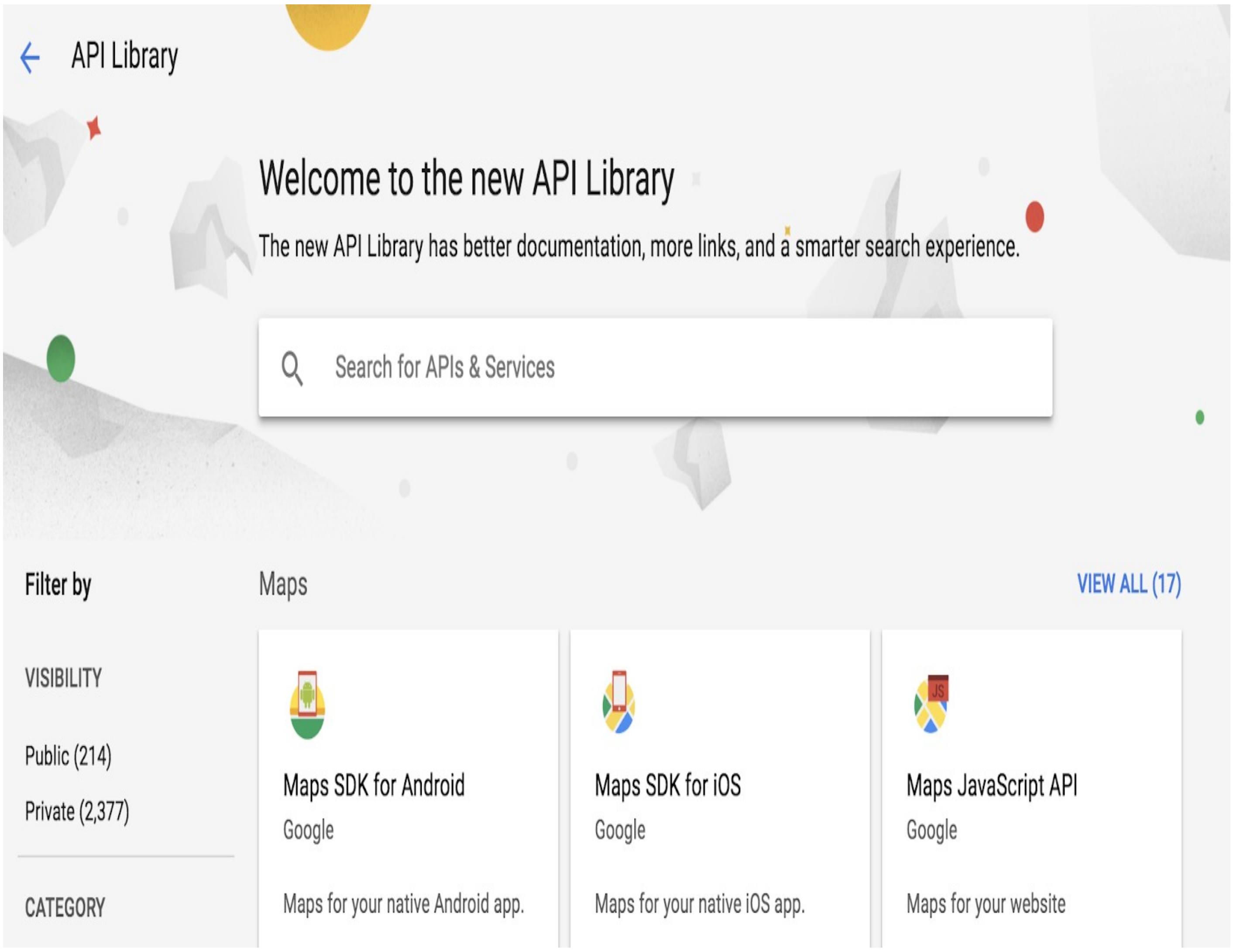
1. In the Console, go to **Navigation menu** > **APIs & Services** > **Dashboard**.



2. Click on **Enable APIs and Services**:



3. Search for **Dialogflow**:


API Library

Welcome to the new API Library

The new API Library has better documentation, more links, and a smarter search experience.

Search for APIs & Services

Filter by

VISIBILITY

Public (214)

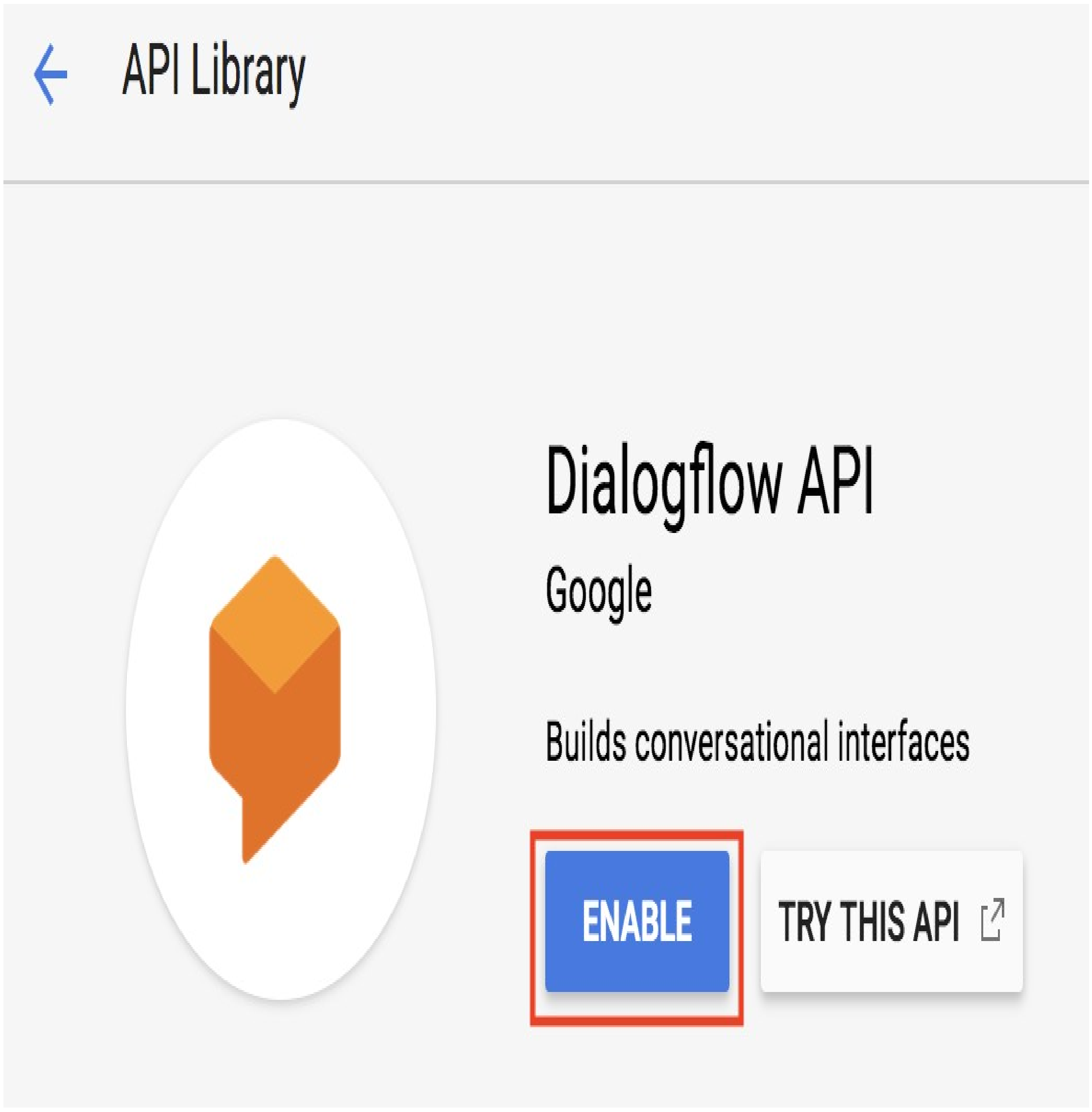
Private (2,377)

CATEGORY

Maps — VIEW ALL (17)

Maps SDK for Android	Maps SDK for iOS	Maps JavaScript API
Google	Google	Google
Maps for your native Android app.	Maps for your native iOS app.	Maps for your website

4. Click on the **Dialogflow API** and if the API is not Enabled, click **Enable**:



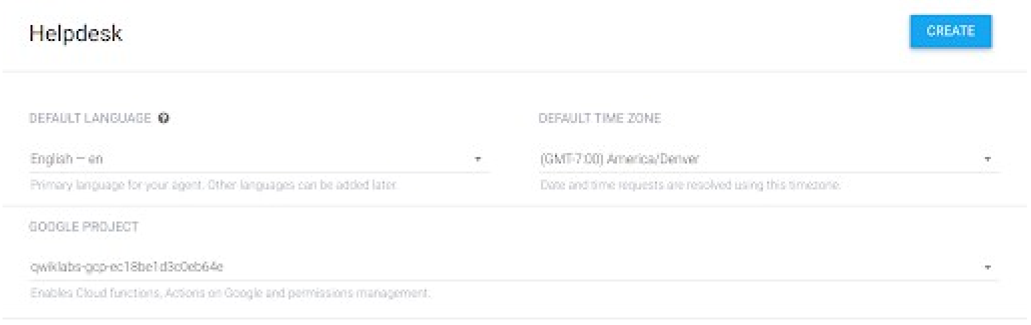
5. In a new tab, go to dialogflow.cloud.google.com. Click the Sign in with **Google** button, and select the credentials you logged into this lab.

6. Then **check the Terms of Service**. Click on **Accept**.

7. Click **Create Agent**.

8. Now add the agent information as you see in the screenshot below:

- **Agent name:** Helpdesk
- **Default Time zone:** America/Denver
- **Google Project:** Your Project ID


Helpdesk

DEFAULT LANGUAGE: English — en
Primary language for your agent. Other languages can be added later.

DEFAULT TIME ZONE: (GMT-7:00) America/Denver
Date and time requests are resolved using this timezone.

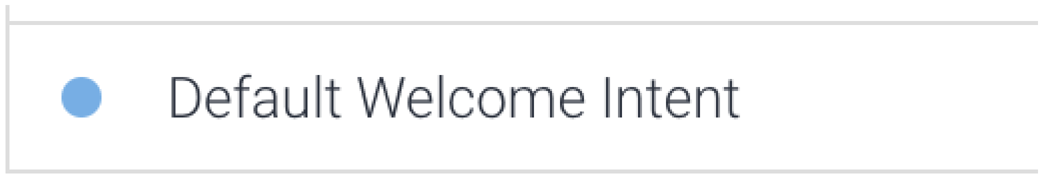
GOOGLE PROJECT: qwiklabs-gcp-ec18be1d3c0eb64e
Enables Cloud functions, Actions on Google and permissions management.

When you're ready, click **Create**.

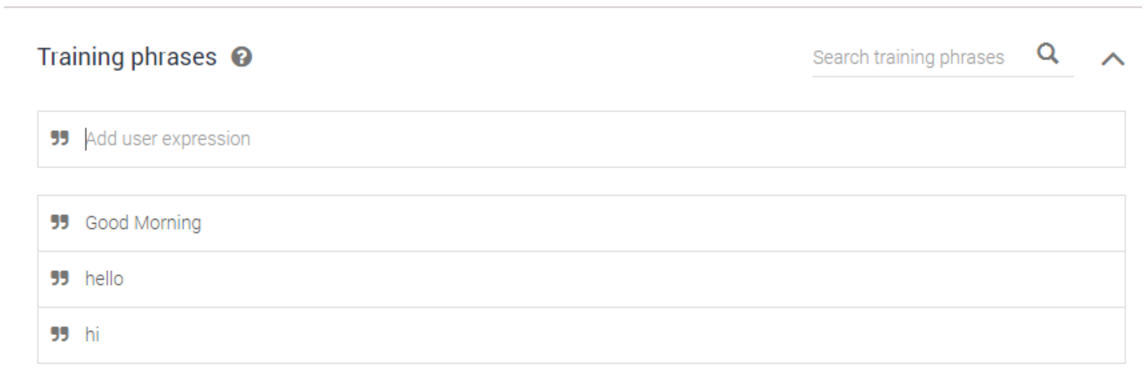
Test Completed Task
Click **Check my progress** to verify your performed task. If you have successfully created a Dialogflow agent, you will see an assessment score.

Now you're ready to get started!

The **Default Welcome Intent** is automatically created. Click on it to open it:


Default Welcome Intent

In the **Training phrases** section, add some expressions that a user could potentially say. The most common would be "Hi", "Hello", "Good morning". Add these now. Your screen should look something like this:


Training phrases

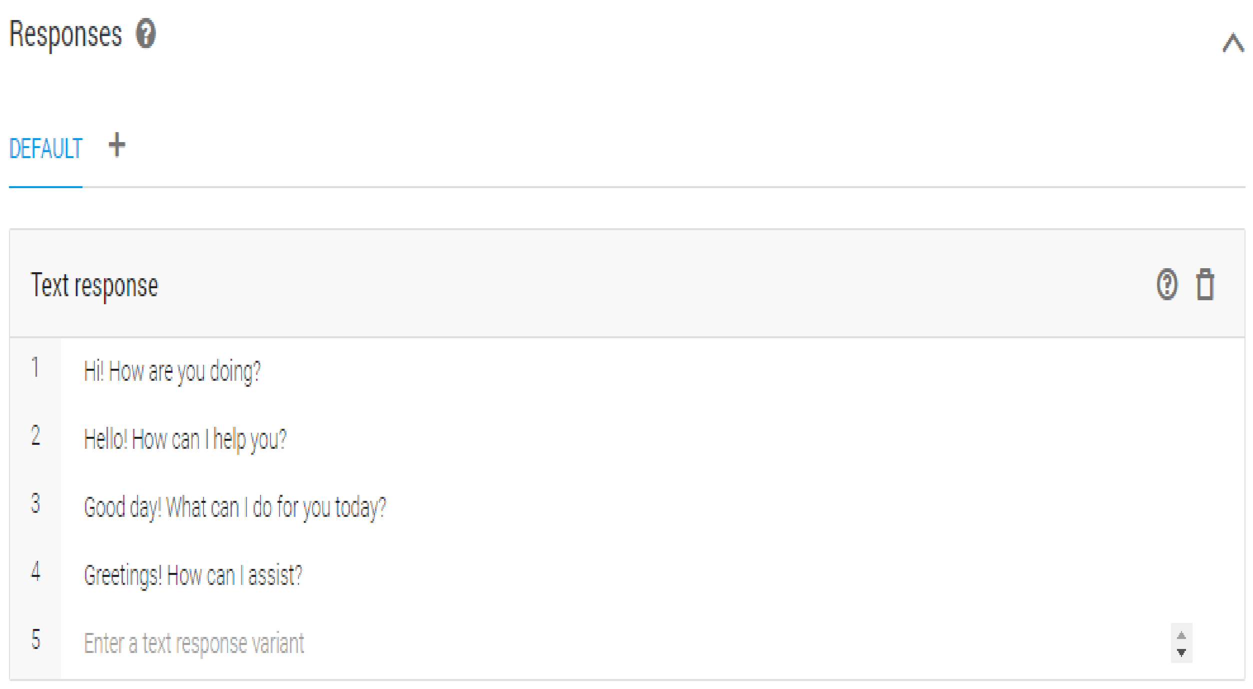
Add user expression

Good Morning

hello

hi

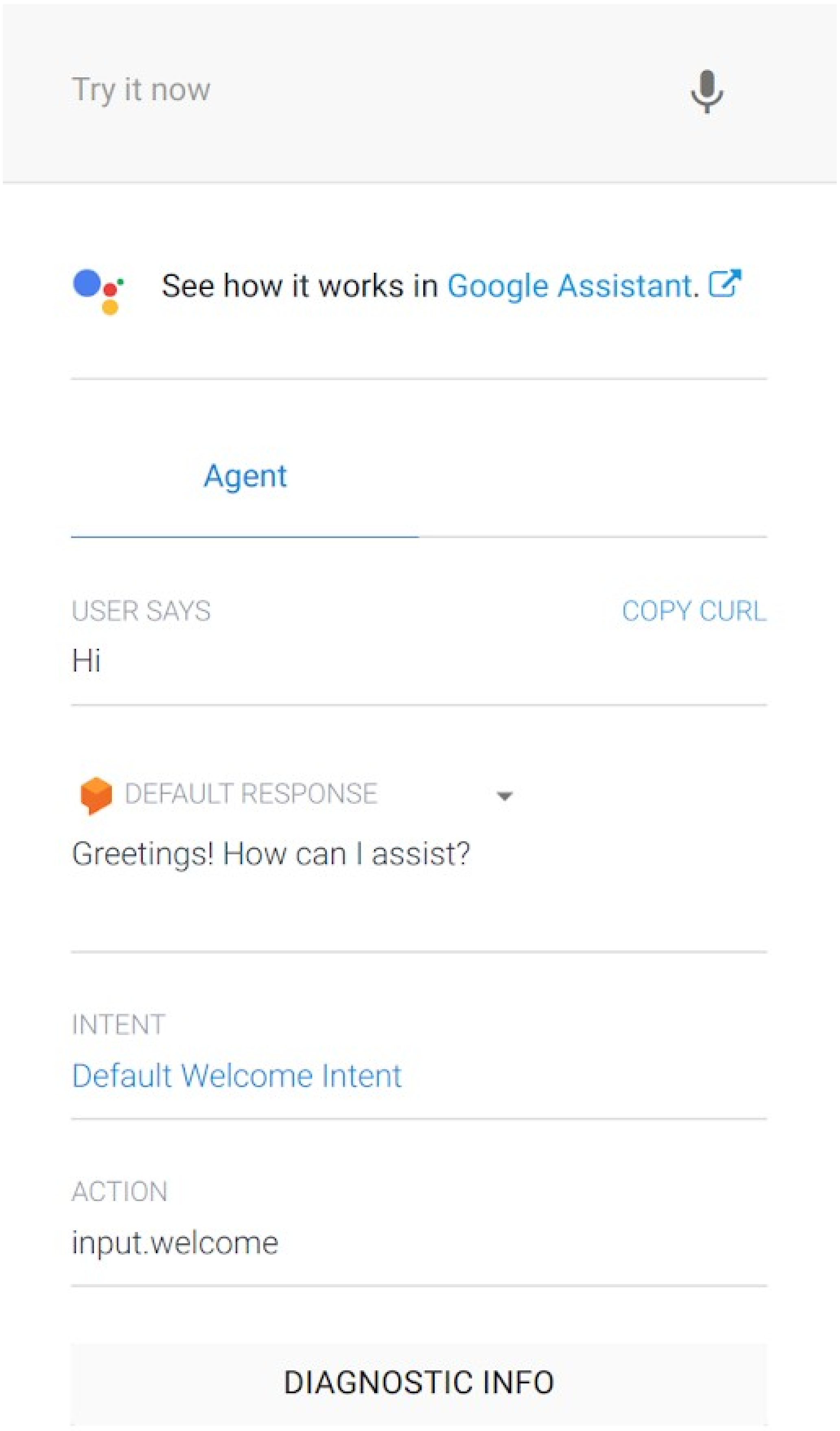
Scroll down to the **Responses** section. Here is where you'll set up the "Text response" that will automatically respond to users. These are all ready to go. If you want, add some more, like, "Hi! How can I help?"



1 Hi! How are you doing?
2 Hello! How can I help you?
3 Good day! What can I do for you today?
4 Greetings! How can I assist?
5 Enter a text response variant

Click **Save** in the top right corner.

Time to test what you built so far. On the right in the **Try it now** section, type "Hi" and press enter. You should see a Default Response.



Try it now

See how it works in Google Assistant.

Agent

USER SAYS

Hi

COPY CURL

DEFAULT RESPONSE

Greetings! How can I assist?

INTENT

Default Welcome Intent

ACTION

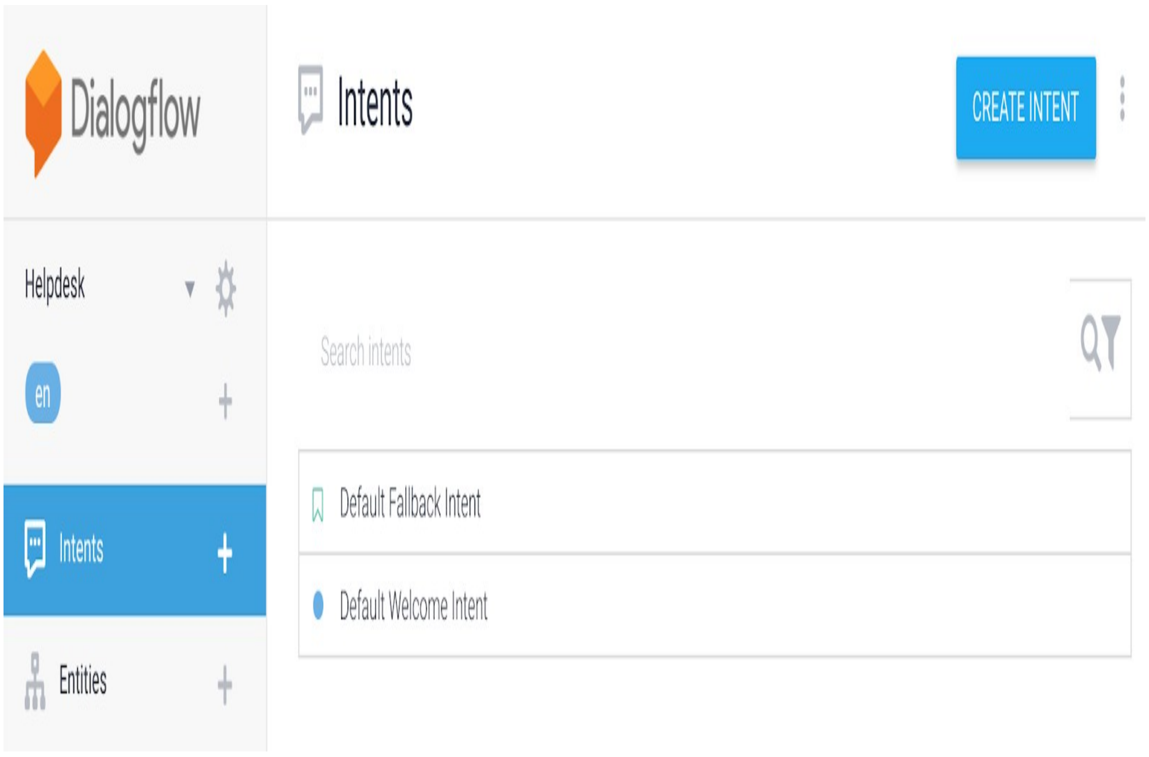
input.welcome

DIAGNOSTIC INFO

Pretty cool, right? Now things will get more interesting.

Create Intents

Click on **Intents** in the left pane, then click on **Create Intent**:



Name the intent "Submit Ticket".

Go to the "Training phrases" section and click on **Add Training Phrases**. Add the following:

- Ticket
- Submit ticket
- Problem
- Issue
- I want to submit a ticket
- I have a problem

Scroll down to the "Responses" section and click on **Add Response**. Enter the following:

- Sure! I can help you with that. Please provide your name for the ticket.

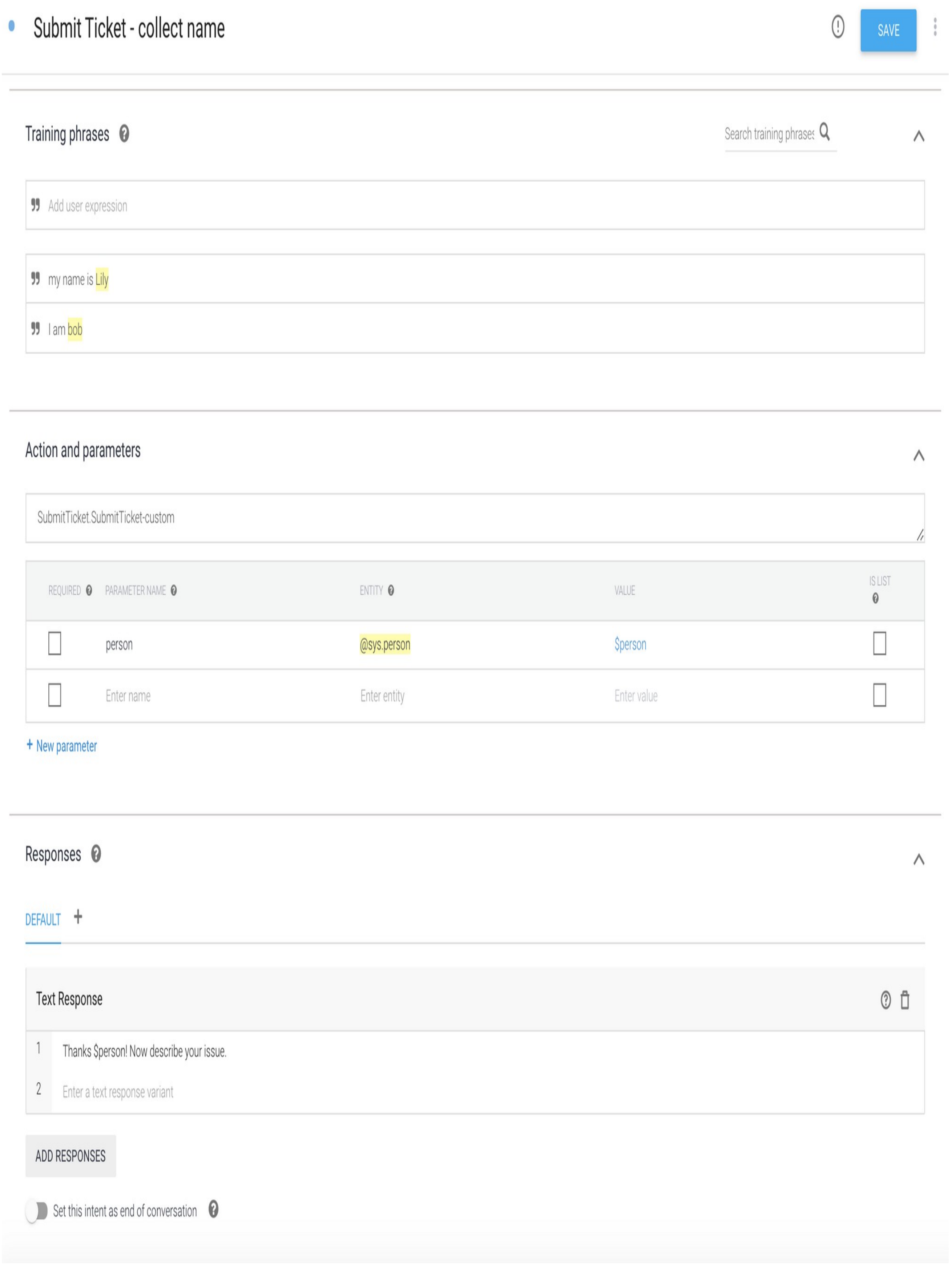
Click **Save** when you're done.

Test Completed Task
Click **Check my progress** to verify your performed task. If you have successfully created a custom intent, you will see an assessment score.

Now click on **Intents** in the left-hand panel, and mouse over your newly created "Submit Ticket" intent. Click "Add follow up intent" and then select **custom**.

Click on the new "Submit Ticket - custom" intent to edit it. Fill in the details and make sure they resemble the screenshot below:

- Intent name: Submit Ticket - collect name
- Training phrases: I am bob
- Training phrases: My name is Lily
- Responses: Thanks $person! Now describe your issue.


Submit Ticket - collect name

Training phrases

Add user expression

my name is Lily

I am bob

Action and parameters

SubmitTicket.SubmitTicket-custom

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	person	@sys.person	$person	☐
☐	Enter name	Enter entity	Enter value	☐

+ New parameter

Responses

DEFAULT +

Text Response

1 Thanks $person! Now describe your issue.

2 Enter a text response variant

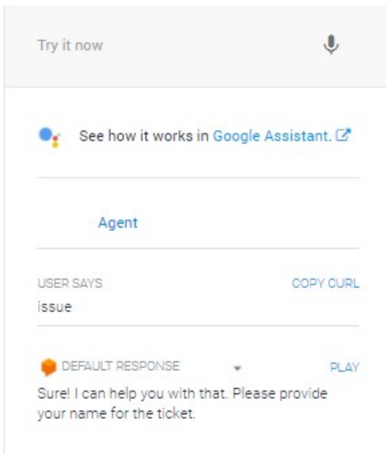
ADD RESPONSES

Set this intent as end of conversation

As you add your training phrases, notice that when you type "My name is Lily" a default entity is created called person (if it doesn't appear by default, please select @sys.person for Lily and bob manually). In the Text response section, a $ is added before person to represent a variable. This will let the chatbot echo the user's name back to them.

Click **Save** when you're done. Next you'll see how the variables are stored and recalled by Dialogflow.

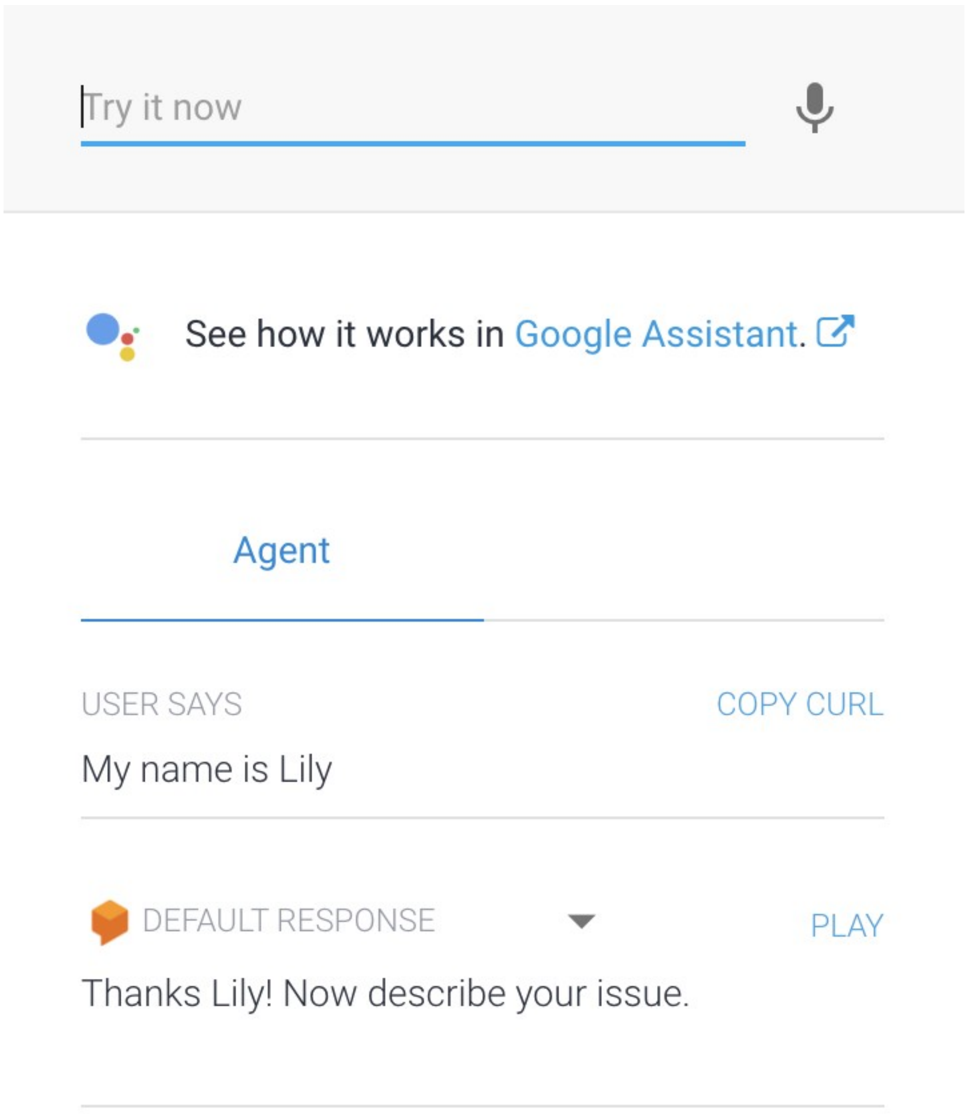
Type in "issue" in **Try it now** and check the response you get:



issue

Sure! I can help you with that. Please provide your name for the ticket.

Type in ‘My name is Lily' and check the response you get:


Try it now

See how it works in Google Assistant.

Agent

USER SAYS

COPY CURL

My name is Lily

DEFAULT RESPONSE

PLAY

Thanks Lily! Now describe your issue.

Notice under **Contexts** that Dialogflow automatically added a context of "SubmitTicket-followup".

Allow Fulfillment to Store Help Ticket Data

Now you'll enable Fulfillment so the help ticket information can be submitted to a database, Google Cloud Datastore in this case. It will take a few minutes for your resources to be available.

Click on **Fulfillment** in the left panel and switch the **Inline Editor** toggle to "Enabled".

You may get an error saying "Your Google Cloud resources are still being provisioned, please refresh the page and try again in a few minutes." If you do, just wait a short time and reload your page.

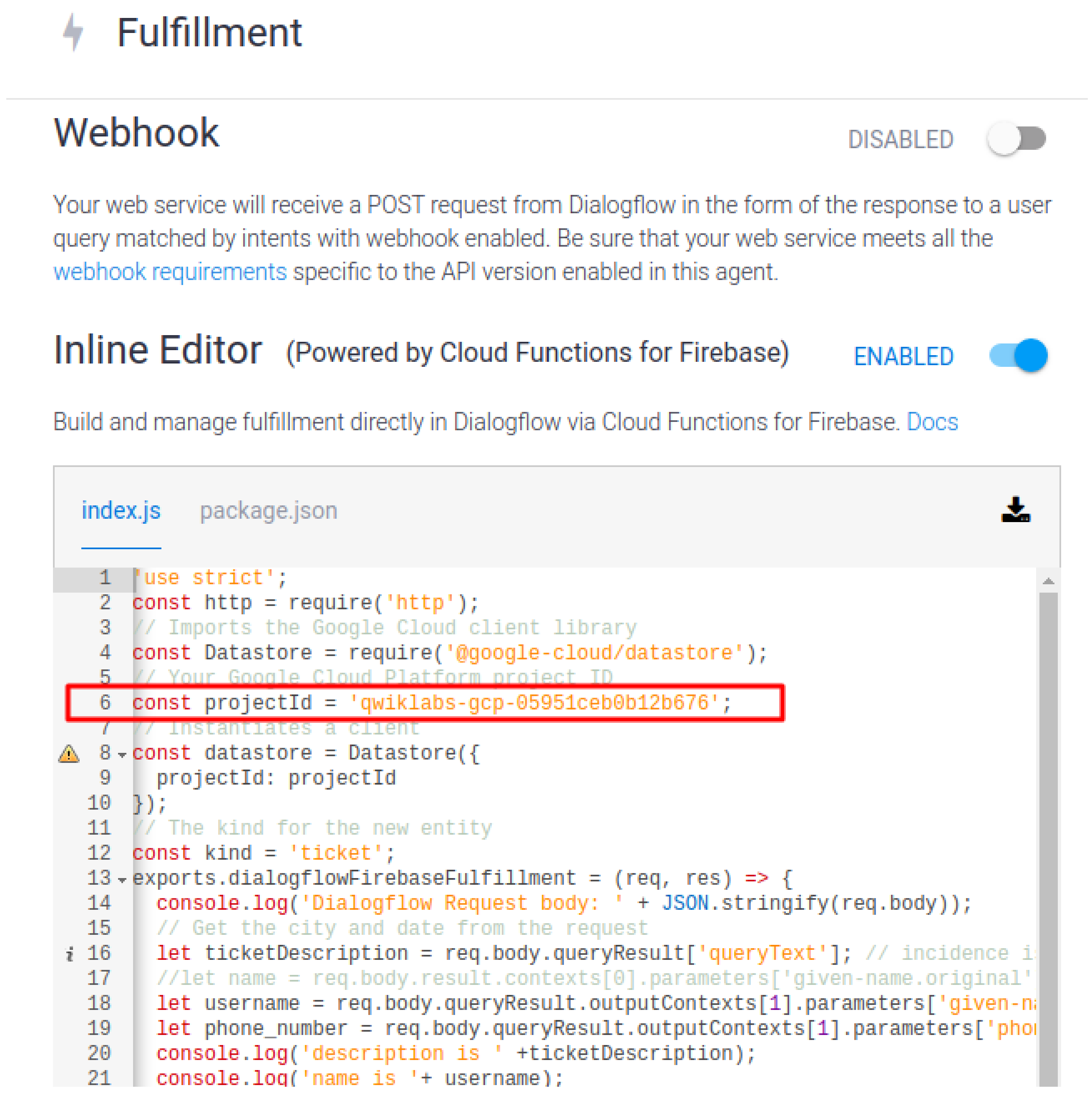
Copy the following code and paste it in the index.js tab, replacing the existing content:

```
'use strict';
const http = require('http');
// Imports the Google Cloud client library
const Datastore = require('@google-cloud/datastore');
// Your Google Cloud Platform project ID
const projectId = 'REPLACE_WITH_YOUR_PROJECT_ID';
// Instantiates a client
const datastore = Datastore({
   projectId: projectId
});
// The kind for the new entity
const kind = 'ticket';
exports.dialogflowFirebaseFulfillment = (req, res) => {
   console.log('Dialogflow Request body: ' + JSON.stringify(req.body));
   // Get the city and date from the request
   let ticketDescription = req.body.queryResult['queryText']; // incidence is a required param
   //let name = req.body.result.contexts[0].parameters['person.original'];
   let username = req.body.queryResult.outputContexts[1].parameters['person.original'];
```

```
    let phone_number = req.body.queryResult.outputContexts[1].parameters['phone-
number.original'];
    console.log('description is ' +ticketDescription);
    console.log('name is '+ username);
    console.log('phone number is '+ phone_number);
    function randomIntInc (low, high) {
        return Math.floor(Math.random() * (high - low + 1) + low);
    }
    let ticketnum = randomIntInc(11111,99999);
    // The Cloud Datastore key for the new entity
    const taskKey = datastore.key(kind);
    // Prepares the new entity
    const task = {
        key: taskKey,
        data: {
            description: ticketDescription,
            username: username,
            phoneNumber: phone_number,
            ticketNumber: ticketnum
        }
    };
    console.log("incidence is  " , task);
    // Saves the entity
    datastore.save(task)
    .then(() => {
        console.log(`Saved ${task.key}: ${task.data.description}`);
        res.setHeader('Content-Type', 'application/json');
        //Response to send to Dialogflow
        res.send(JSON.stringify({ 'fulfillmentText': "I have successfully logged your ticket, the ticket
number is " + ticketnum + ". Someone from the helpdesk will reach out to you within 24
hours."}));
        //res.send(JSON.stringify({ 'fulfillmentText': "I have successfully logged your ticket, the ticket
number is " + ticketnum + ". Someone from the helpdesk will reach out to you within 24 hours.",
'fulfillmentMessages': "I have successfully logged your ticket, the ticket number is " + ticketnum
+  ". Someone from the helpdesk will reach out to you within 24 hours."}));
    })
    .catch((err) => {
        console.error('ERROR:', err);
        res.setHeader('Content-Type', 'application/json');
        res.send(JSON.stringify({ 'speech': "Error occurred while saving, try again later",
'displayText': "Error occurred while saving, try again later" }));
    });
};
content_copy
```

Then edit the index.js file:

- On line 6, replace REPLACE_WITH_YOUR_PROJECT_ID with your Project ID (in single quotes). Your project ID is on the Qwiklabs page with the credentials, where you started the lab.
  The result should look something like this (but with your own project name substituted):



Fulfillment

Webhook DISABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the webhook requirements specific to the API version enabled in this agent.

Inline Editor (Powered by Cloud Functions for Firebase) ENABLED

Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. Docs

index.js package.json

```
'use strict';
const http = require('http');
// Imports the Google Cloud client library
const Datastore = require('@google-cloud/datastore');
// Your Google Cloud Platform project ID
const projectId = 'qwiklabs-gcp-05951ceb0b12b676';
// Instantiates a client
const datastore = Datastore({
  projectId: projectId
});
// The kind for the new entity
const kind = 'ticket';
exports.dialogflowFirebaseFulfillment = (req, res) => {
  console.log('Dialogflow Request body: ' + JSON.stringify(req.body));
  // Get the city and date from the request
  let ticketDescription = req.body.queryResult['queryText']; // incidence i
  //let name = req.body.result.contexts[0].parameters['given-name.original'
  let username = req.body.queryResult.outputContexts[1].parameters['given-na
  let phone_number = req.body.queryResult.outputContexts[1].parameters['phon
  console.log('description is ' +ticketDescription);
  console.log('name is '+ username);
```

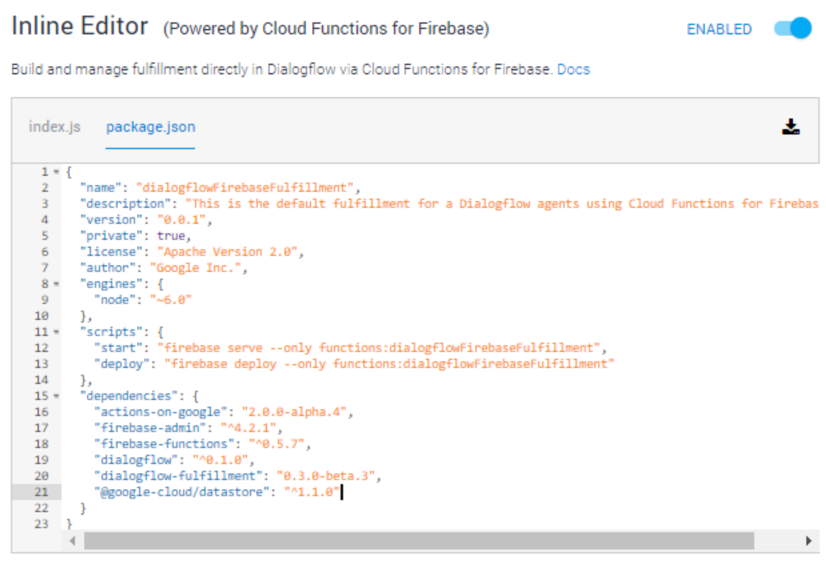
Then click on the **package.json** tab add this dependency:

```
"@google-cloud/datastore": "^1.1.0"
```

content_copy

Remember to add a comma after the last item in the dependency list.

The result should look something like this:


Inline Editor (Powered by Cloud Functions for Firebase)
ENABLED
Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. Docs
index.js
package.json

```
{
  "name": "dialogflowFirebaseFulfillment",
  "description": "This is the default fulfillment for a Dialogflow agents using Cloud Functions for Firebas
  "version": "0.0.1",
  "private": true,
  "license": "Apache Version 2.0",
  "author": "Google Inc.",
  "engines": {
    "node": "~6.0"
  },
  "scripts": {
    "start": "firebase serve --only functions:dialogflowFirebaseFulfillment",
    "deploy": "firebase deploy --only functions:dialogflowFirebaseFulfillment"
  },
  "dependencies": {
    "actions-on-google": "2.0.0-alpha.4",
    "firebase-admin": "^4.2.1",
    "firebase-functions": "^0.5.7",
    "dialogflow": "^0.1.0",
    "dialogflow-fulfillment": "0.3.0-beta.3",
    "@google-cloud/datastore": "^1.1.0"
  }
}
```

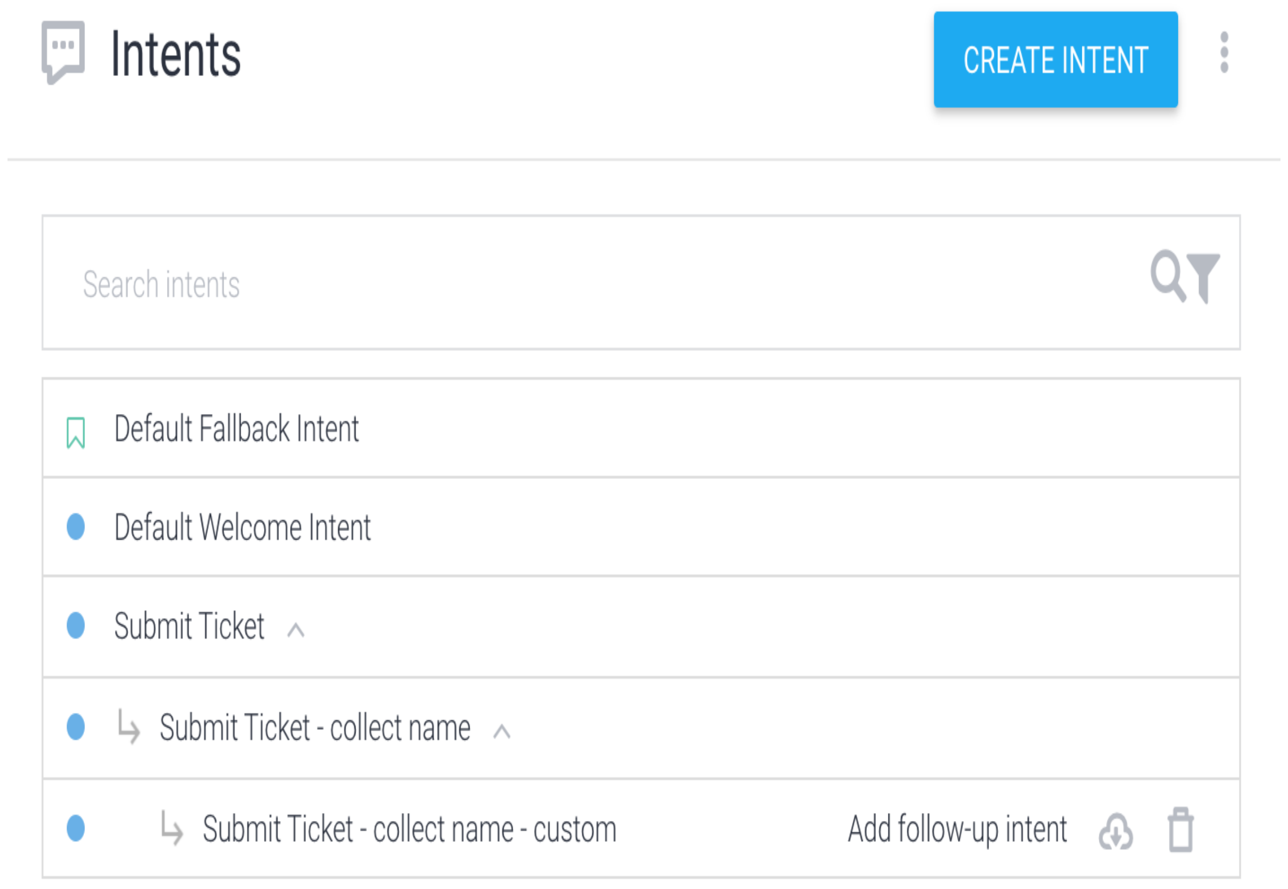
Then click the **Deploy** button. Wait until you see a message that the deployment was successful (this might take a little while).

Test Completed Task
Click **Check my progress** to verify your performed task. If you have successfully allowed Fulfillment to store Help Ticket Data, you will see an assessment score.

Next, go back to **Intents** in the left panel. Click the down arrow next to "Submit Ticket" to reveal its follow-up intents.

Mouse over "Submit Ticket - collect name" and click "Add follow up intent", then select **Custom**.

Click on the newly created intent "Submit Ticket - collect name - custom" to open it for editing.

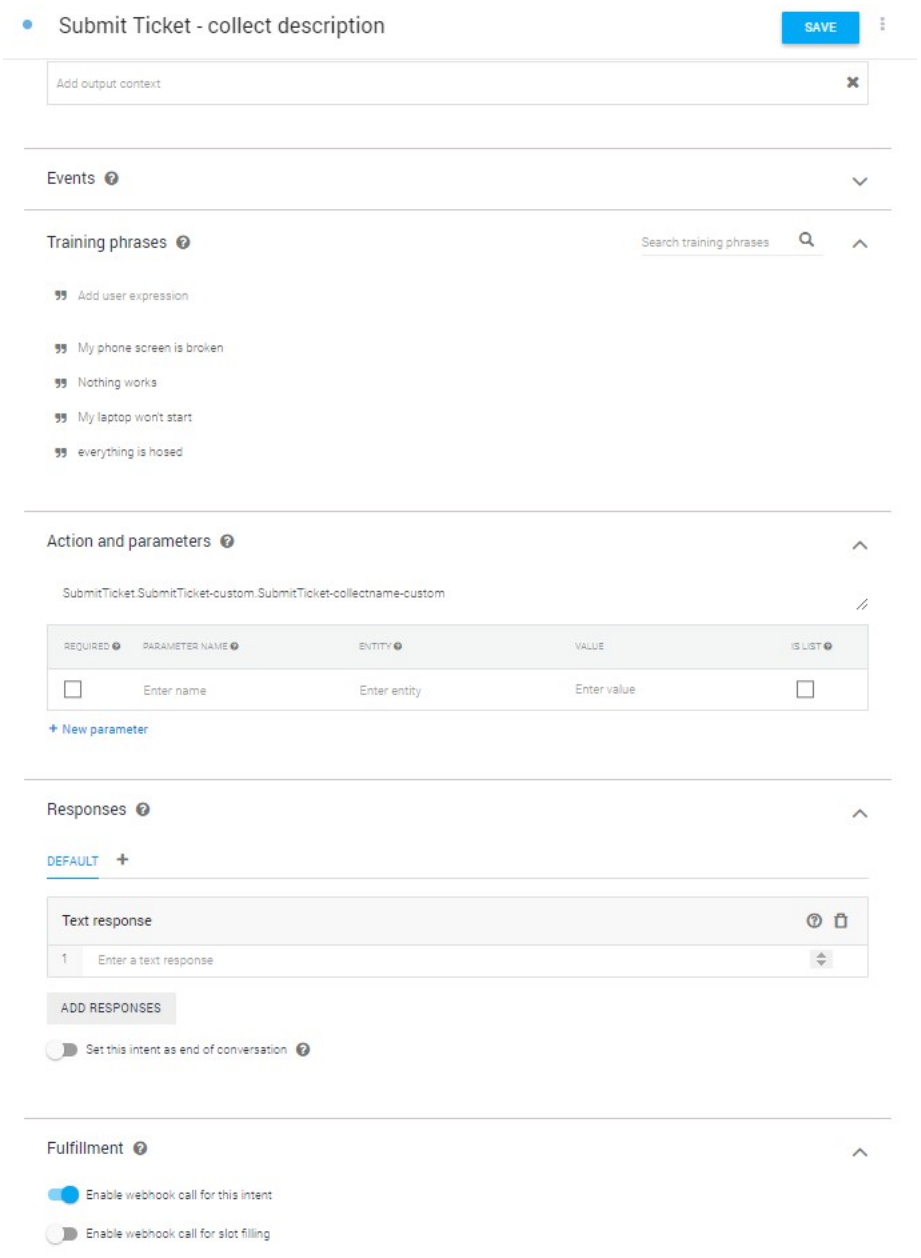


Name the intent "Submit Ticket - collect description".

Enter some user expressions into **Training phrases** as shown below. Here is what we selected for **Training phrases**:

- Everything is hosed
- My laptop won't start
- Nothing works
- My phone screen is broken

Use the following screenshot as a reference:


Submit Ticket - collect description

Add output context

Events

Training phrases

Add user expression

My phone screen is broken

Nothing works

My laptop won't start

everything is hosed

Action and parameters

SubmitTicket.SubmitTicket-custom.SubmitTicket-collectname-custom

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	Enter name	Enter entity	Enter value	☐

Responses

Text response

1 Enter a text response

Set this intent as end of conversation

Fulfillment

Enable webhook call for this intent

Enable webhook call for slot filling

Scroll to the bottom of the screen click on the **Fulfillment** arrow to toggle the section. Click on **Enable webhook call for this intent**.

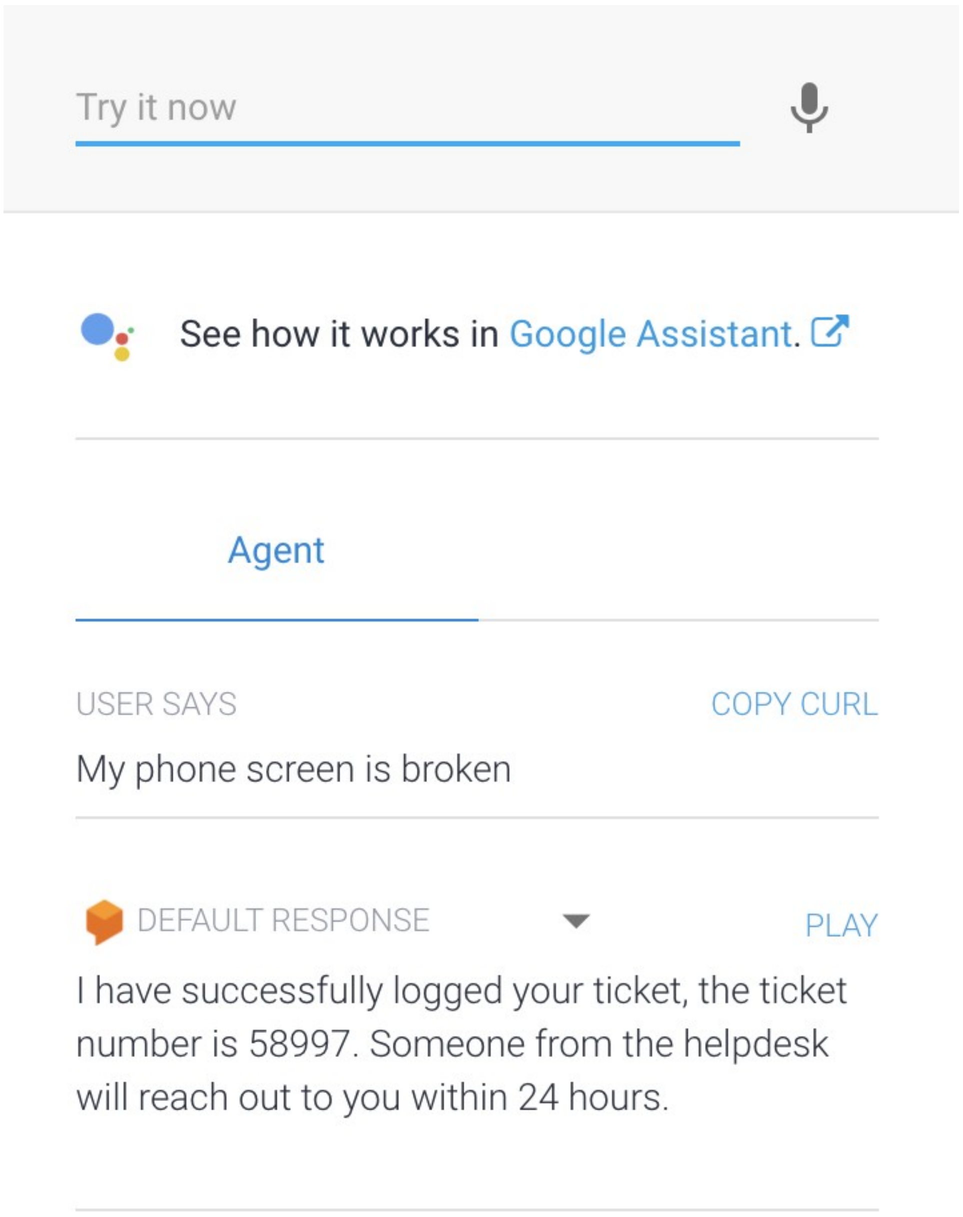
When you're finished, click **Save**.

At this point, the Dialogflow should be set up. Test it in the "Try it now" panel by entering the following conversation:

1. Hi
2. I would like to submit a ticket
3. My name is John

4. My phone screen is broken

   You should see a default response that resembles the following:



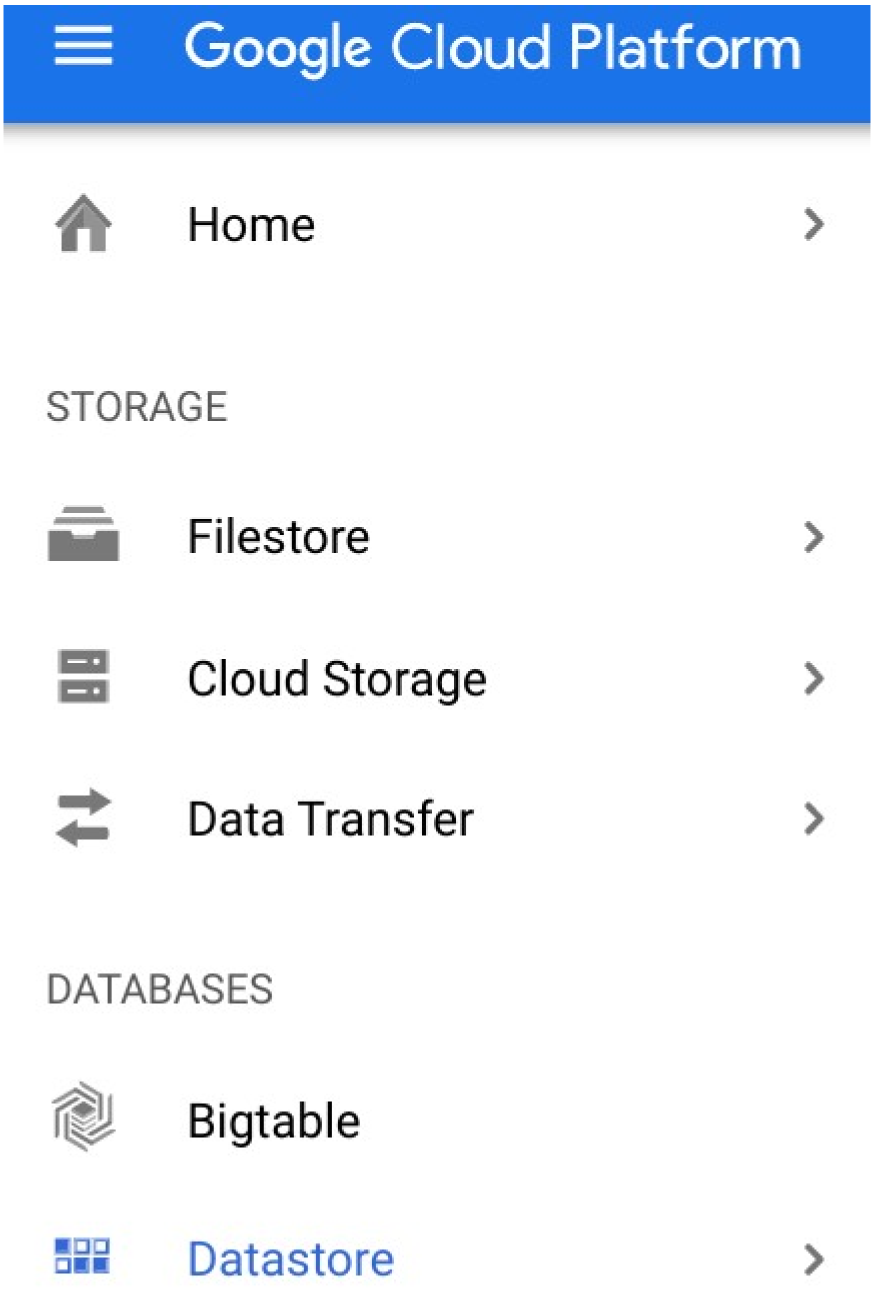
My phone screen is broken

I have successfully logged your ticket, the ticket number is 58997. Someone from the helpdesk will reach out to you within 24 hours.

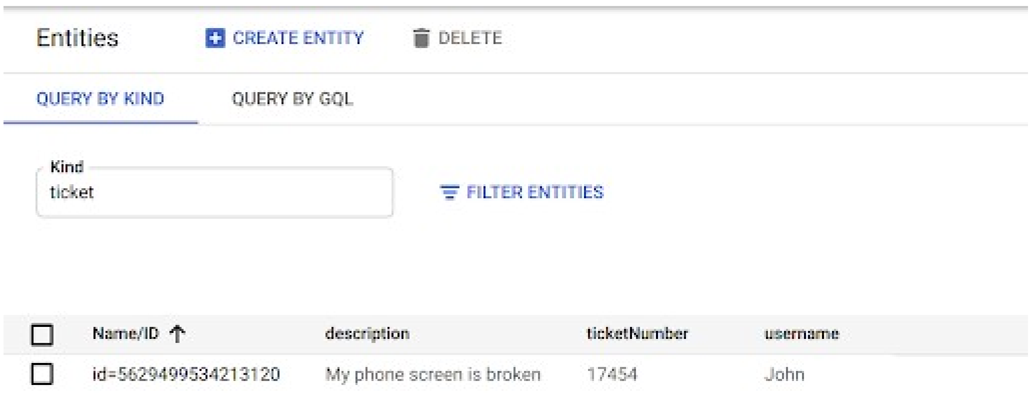
Verify that Tickets are Logged in Datastore

Now verify that the support ticket is getting logged in Datastore.

From the Cloud Console Navigation menu, go to **Datastore**.



You should see the entry as shown below.


Entities
CREATE ENTITY
DELETE
QUERY BY KIND
QUERY BY GQL
Kind: ticket
FILTER ENTITIES

Name/ID	description	ticketNumber	username
id=5629499534213120	My phone screen is broken	17454	John

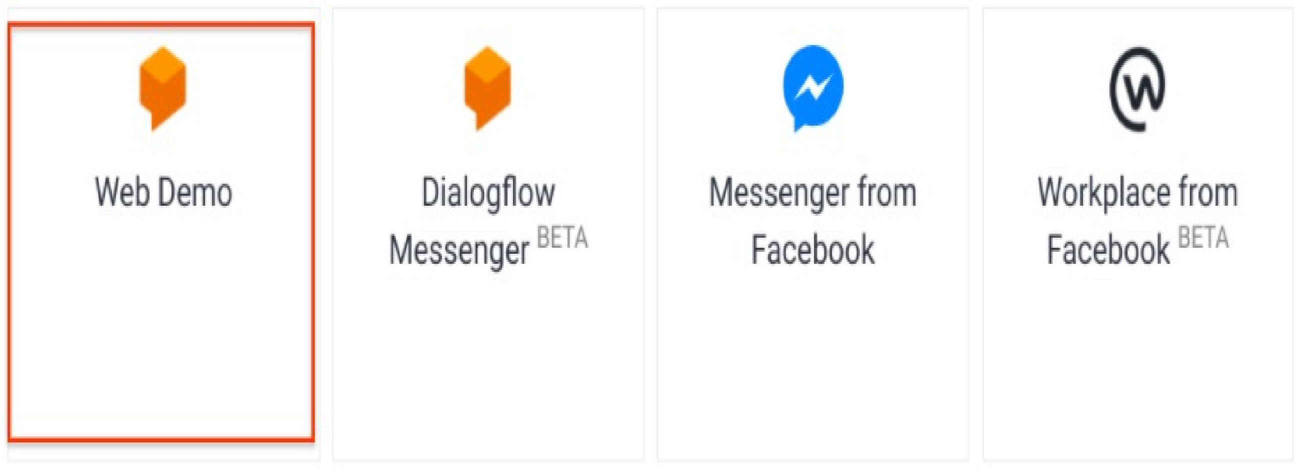
This verifies that your chatbot is working and logging tickets, awesome!

Test Completed Task
Click **Check my progress** to verify your performed task. If tickets get successfully logged in Datastore, you will see an assessment score.

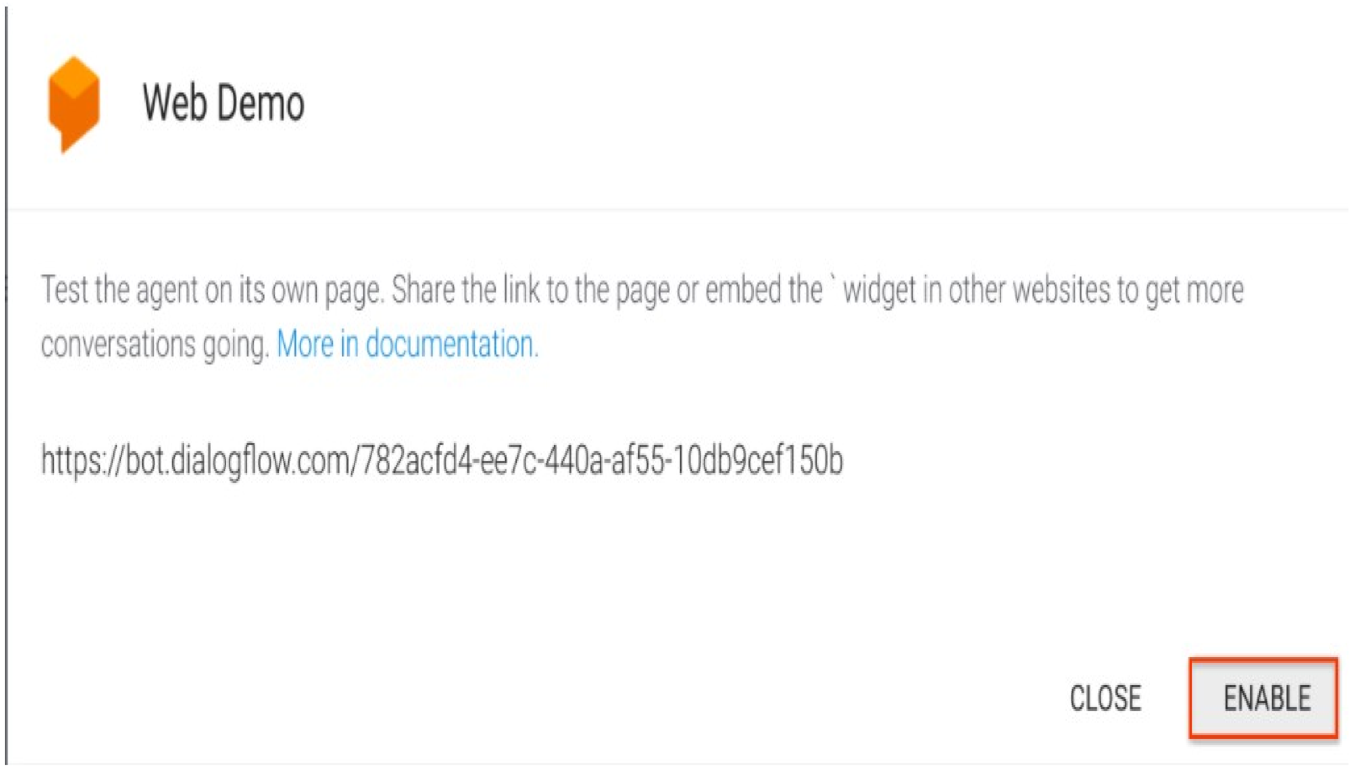
Testing your Chatbot

Dialogflow provides many types of integrations for your chatbot. Now take a look at a sample web user interface for the chatbot.

Click on **Integrations** in the Dialogflow left panel and then click on the **Web Demo** integration.



Click **Enable**.



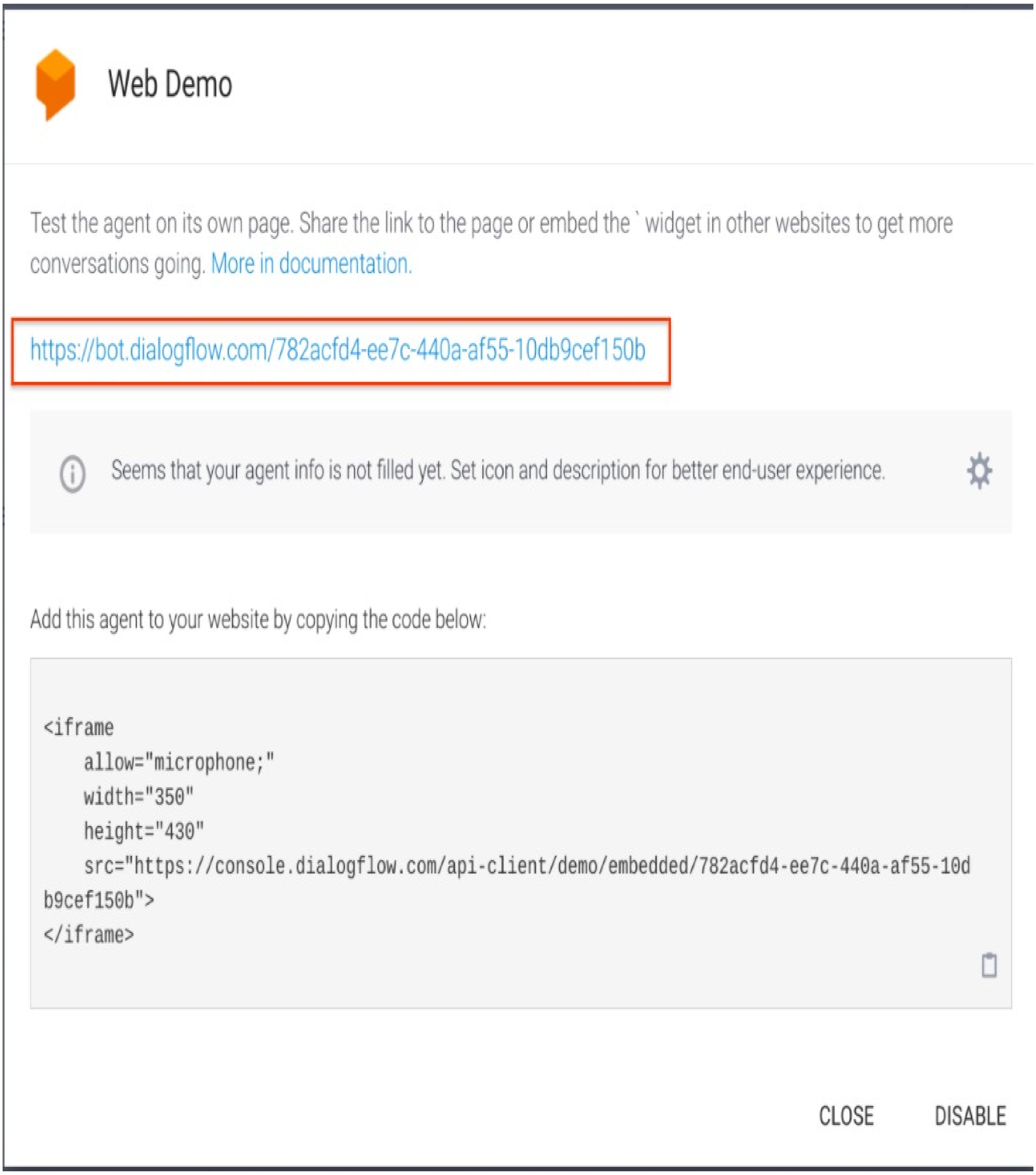
Web Demo

Test the agent on its own page. Share the link to the page or embed the ` widget in other websites to get more conversations going. More in documentation.

https://bot.dialogflow.com/782acfd4-ee7c-440a-af55-10db9cef150b

Now, click on the URL link to launch Web

Demo:


Web Demo

Test the agent on its own page. Share the link to the page or embed the ` widget in other websites to get more conversations going. More in documentation.

https://bot.dialogflow.com/782acfd4-ee7c-440a-af55-10db9cef150b

Seems that your agent info is not filled yet. Set icon and description for better end-user experience.

Add this agent to your website by copying the code below:

```
<iframe
    allow="microphone;"
    width="350"
    height="430"
    src="https://console.dialogflow.com/api-client/demo/embedded/782acfd4-ee7c-440a-af55-10d
b9cef150b">
</iframe>
```

Start using the chat interface by typing in the Ask something... section! If you are using a Chrome browser, if you click the microphone icon and you can speak your questions to the chatbot. Start chatting with the chatbot using the following conversation:

- Type "Hi" and hit Enter. The chatbot should respond as before.
- Then enter/say "Submit ticket"
- Provide the name "My name is Lily"
- Provide the ticket details of "My phone screen is broken"
  You should receive a dialog output that says a ticket has been submitted. You can also check in datastore to see if the ticket has been logged. The web demo is in it's early stages, so if it doesn't work as expected try to run the commands again or refresh the page.

Test your Understanding

Below are a multiple choice questions to reinforce your understanding of this lab's concepts. Answer them to the best of your abilities.

Training phrases
Response
Action and parameters
Intent name

<ql-true-false-probe stem="Contexts give you more control over intent matching and let you manage state". answer="true">

Congratulations!

You're now a chatbot developer!

Finish your quest
This self-paced lab is part of the Qwiklabs Quest Machine Learning APIs. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

Take your next lab
Try out another lab on Machine Learning APIs, like AI Platform: Qwik Start or Entity and Sentiment Analysis with the Natural Language API.

Next steps
- Sign up for the full Coursera Course on Machine Learning with TensorFlow on Google Cloud Platform

Google Cloud Training & 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 August 20, 2021

Lab Last Tested August 20, 2021