

# Implement a Helpdesk Chatbot with Dialogflow & BigQuery ML

GSP431



Google Cloud Self-Paced Labs

# Overview

Wouldn't it be awesome to have an accurate estimate of how long it will take for tech support to resolve your issue? In this lab you will train a simple machine learning model for predicting helpdesk response time using BigQuery Machine Learning. You will then build a simple chatbot using Dialogflow, and learn how to integrate your trained BigQuery ML model with your helpdesk chatbot. The final solution will provide an estimate of response time to users at the moment a request is generated.

The exercises are ordered to reflect a common cloud developer experience:

1. Train a Model using BigQuery Machine Learning
2. Deploy a simple Dialogflow application
3. Use an inline code editor within Dialogflow for deploying a Node.js fulfillment script that integrates BigQuery
4. Test your chatbot

## What you'll learn

- How to train a machine learning model using BigQuery ML
- How to evaluate and improve a machine learning model using BigQuery ML
- How to import intents & entities into a Dialogflow agent
- How to implement custom Node.js fulfillment scripts
- How to integrate BigQuery with Dialogflow

## Prerequisites

- Basic concepts and constructs of Dialogflow. [Click here](#) for an introductory Dialogflow tutorial that covers basic conversational design and fulfillment using a webhook.
- Basic SQL and Node.js (or any coding language) knowledge.

## Solution Feedback/Lab Help

- For more information on this solution or feedback on this lab, please reach out to [gcct-ce-amer-specialists-ml@google.com](mailto:gcct-ce-amer-specialists-ml@google.com).

## Setup Lab

### 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 Pixelbook, 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!](#)


- 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.


  
**Sign in**  
Use your Google Account


[Forgot email?](#)


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

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

  
**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.

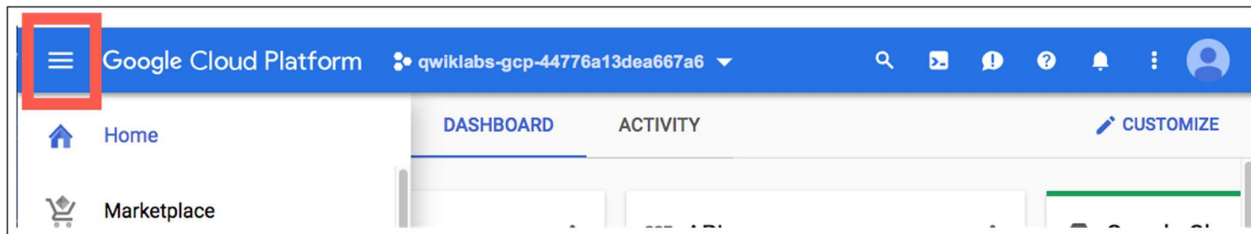
**Important:** You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

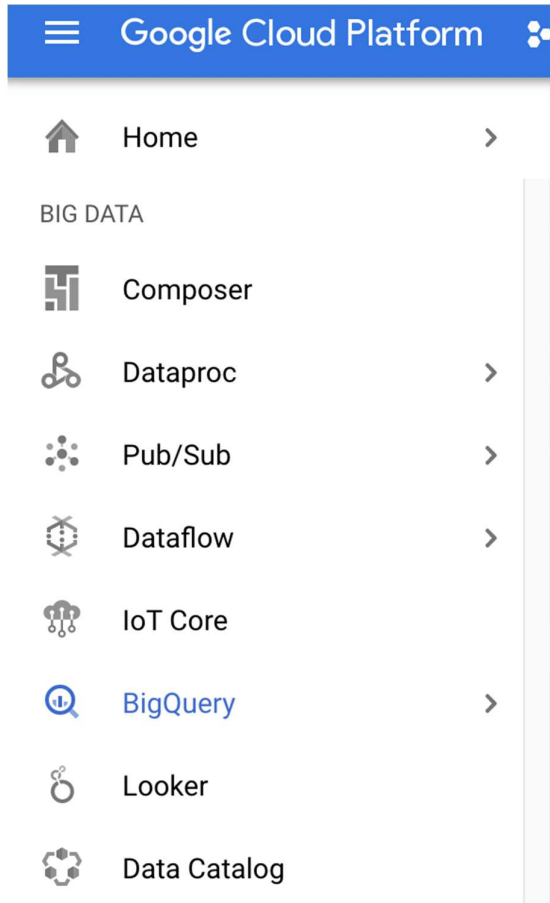
**Note:** You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



# Train a Model Using BigQuery Machine Learning

## Open BigQuery Console

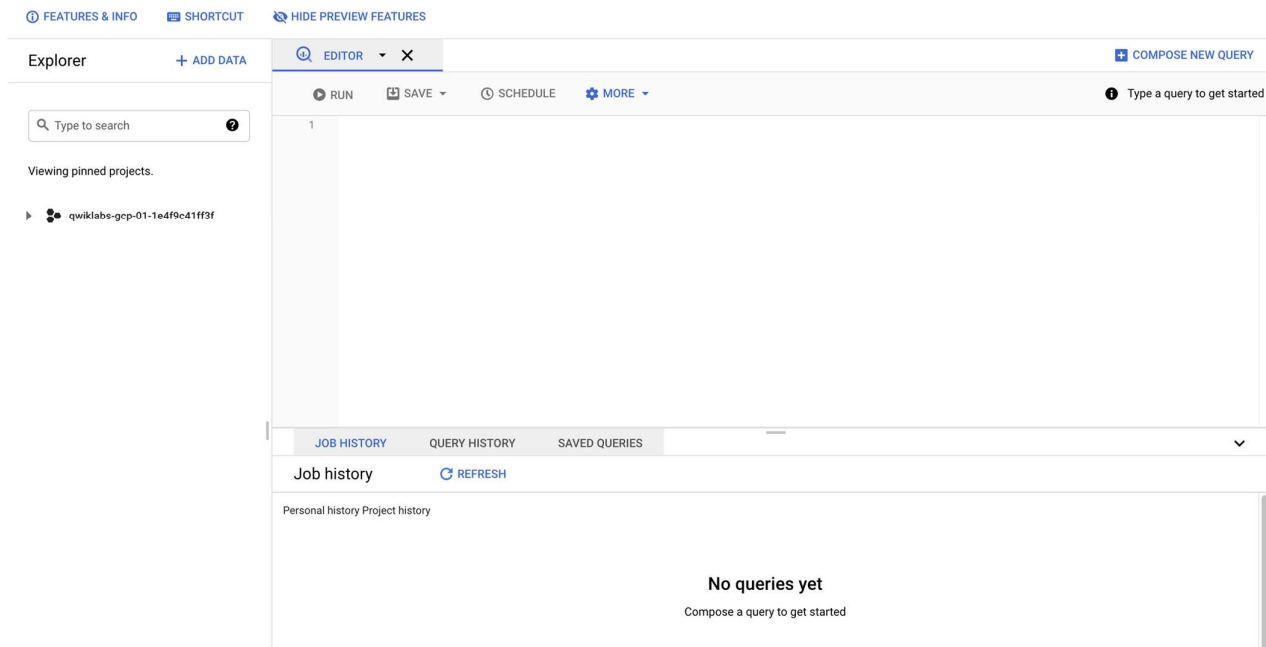
In the Google Cloud Console, select **Navigation menu** > **BigQuery**:



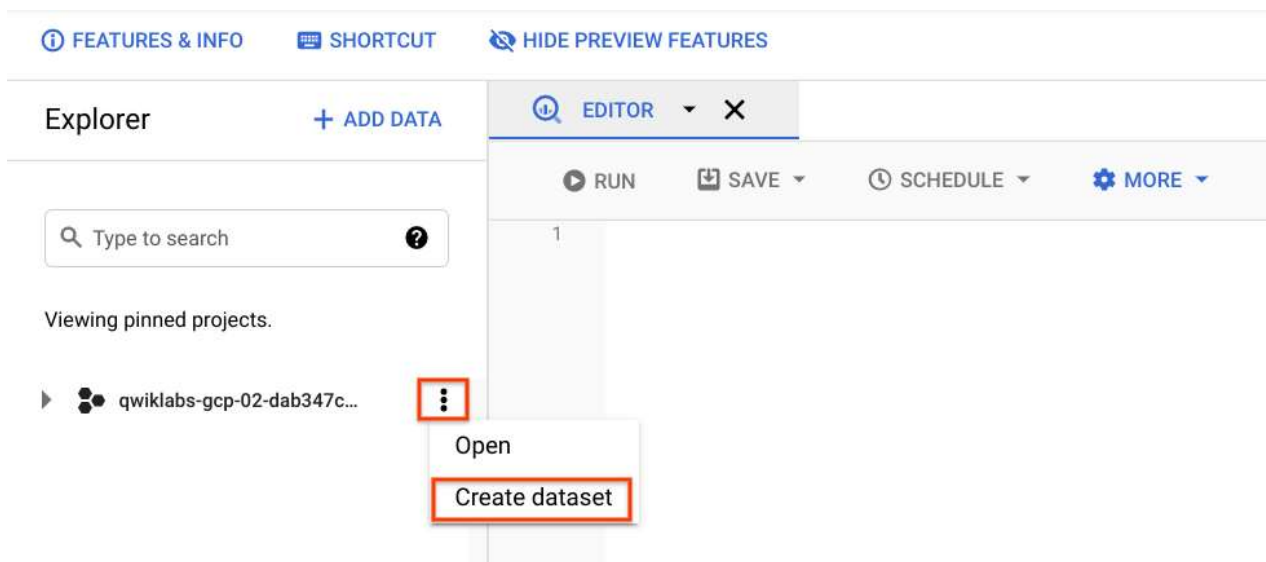
The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and the release notes.

Click **Done**.

The BigQuery console opens.



In the left pane, under **Explorer** section, click on the **View actions** icon next to your project ID, then click **Create dataset**.



For **Dataset ID**, type **helpdesk**. Leave **Default** selected as the location and click **Create dataset**.

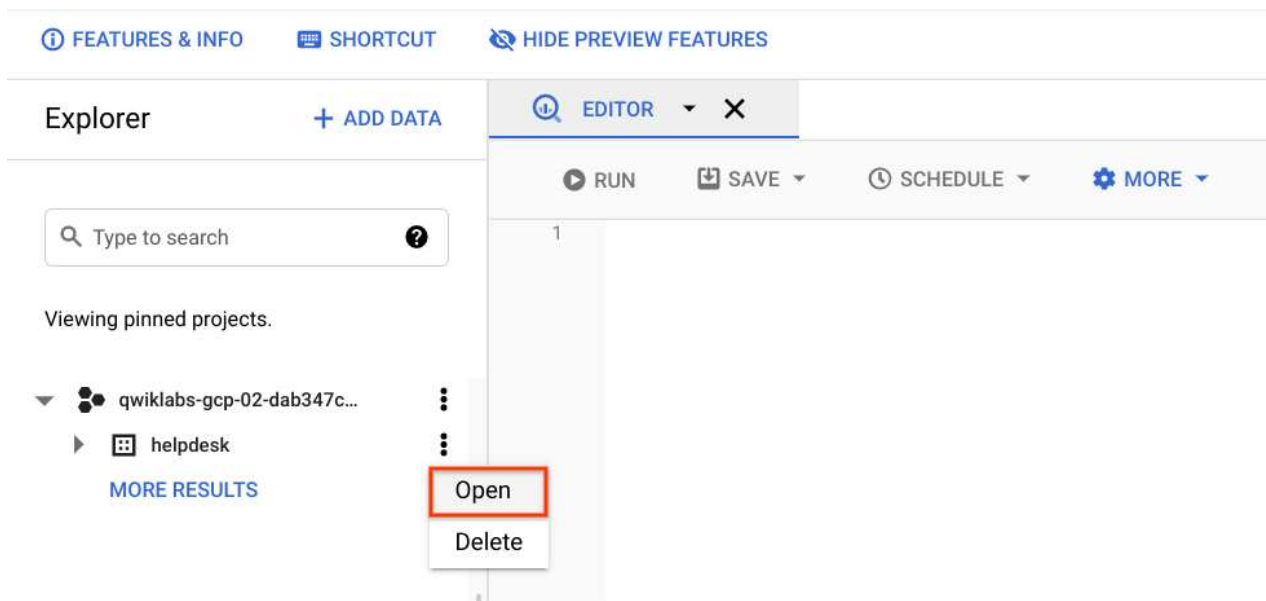
### Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

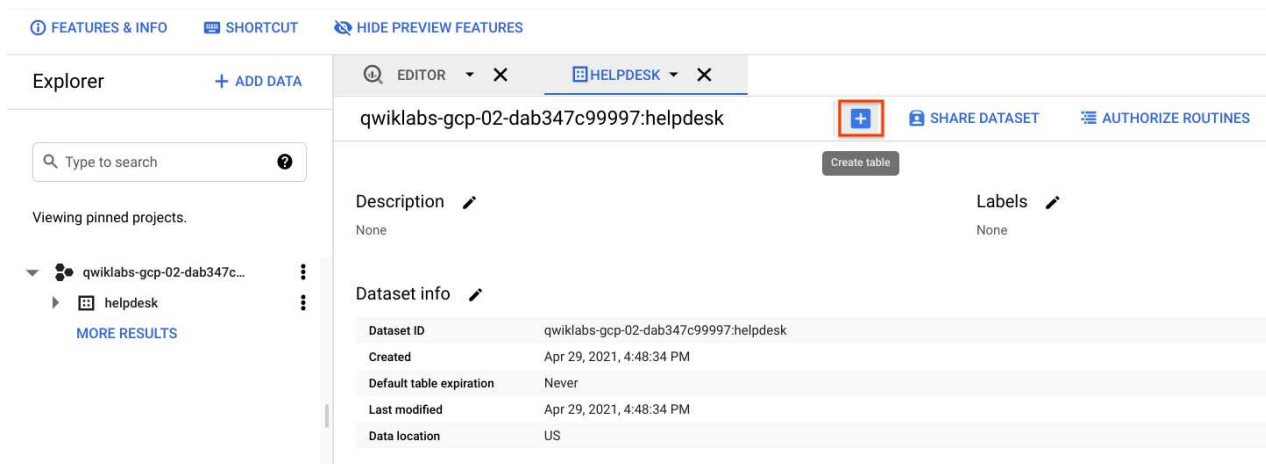
Create a BigQuery dataset

Check my progress

In the left menu, click the **View actions** icon next to **helpdesk** dataset you just created, then click **Open**.



Click **Create table**.



Use the following parameters to create a new table leaving the defaults for all other fields:

Properties	Values
Create table from	Google Cloud Storage
Select file from GCS bucket	gs://solutions-public-assets/smartenup-helpdesk/ml/issues.csv
File format	CSV
Table name	issues
Auto detect	Check the check-box for <b>Schema and input parameters</b>
Advanced options > Header rows to skip	1

Click **Create table**.



This will trigger a job that loads the source data into a new BigQuery table. It will take about 30 seconds for the job to complete, and you can view it by selecting **Job History** from the bottom.

### Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

Create a new table in BigQuery dataset

Check my progress

In BigQuery **EDITOR** add the following query:

```
SELECT * FROM `helpdesk.issues` LIMIT 1000
```

Click **Run** and examine the data.

In the next query, the data fields `category` and `resolutiontime` are used to build a machine learning model that predicts how long it will take to resolve an issue. The model type is a simple linear regression, and the trained model will be named **predict\_eta\_v0** in the helpdesk dataset.

Clear the previous query from the editor and run this query, it takes about **1** minute to complete:

```
CREATE OR REPLACE MODEL `helpdesk.predict_eta_v0`  
OPTIONS(model_type='linear reg') AS  
SELECT  
  category,  
  resolutiontime as label  
FROM  
  `helpdesk.issues`
```

### Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

Build an ML model to predicts time taken to resolve an issue

Check my progress

Clear the previous query from the editor and run the following query to evaluate the machine learning model you just created. The metrics generated by this query tell us how well the model is likely to perform.

```
WITH eval_table AS (  
  SELECT  
    category,  
    resolutiontime as label  
  FROM  
    helpdesk.issues  
)  
SELECT  
  *  
FROM
```

```
ML.EVALUATE (MODEL helpdesk.predict_eta_v0,  
TABLE eval_table)
```

Job information <a href="#">Results</a> JSON   Execution details						
Row	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
1	2.1690245063610343	8.939278218946114	0.20482886943747128	1.8381899074111185	0.16438582971020366	0.1643860312394838

Using the evaluation metrics, we can see that the model doesn't perform very well. When the `r2\_score` and `explained\_variance` metrics are close to 0, our algorithm is having difficulty distinguishing the signal in our data from the noise. We are going to use a few more fields during training and see if there is an improvement: **seniority, experience & type**. The final trained model will be named **predict\_eta**. Clear the previous query from the editor and run the query below:

```
CREATE OR REPLACE MODEL `helpdesk.predict_eta`  
OPTIONS(model_type='linear_reg') AS  
SELECT  
  seniority,  
  experience,  
  category,  
  type,  
  resolutiontime as label  
FROM  
  `helpdesk.issues`
```

Now, run the following query to evaluate the updated machine learning model you just created.

```
WITH eval_table AS (  
  SELECT  
    seniority,  
    experience,  
    category,  
    type,  
    resolutiontime as label  
  FROM  
    helpdesk.issues  
)  
SELECT  
  *  
FROM  
  ML.EVALUATE (MODEL helpdesk.predict_eta,  
    TABLE eval_table)
```

## Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

Run the query to evaluate the ML model

Check my progress

After adding the additional fields during training, we can see that our model has improved. When the metrics `r2\_score` and `explained\_variance` are close to 1, there is evidence to suggest that our model is capturing a strong linear relationship. We can also see that our error metrics are lower than before, which means our model will likely perform better.

Job information <u>Results</u> JSON   Execution details						
Row	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
1	1.2304085222106764	3.158537555822176	0.1156468563003275	0.9131213415977868	0.7047503529486896	0.7047507586451964

Now we can execute a query to get a prediction of resolution time for a given scenario:

```
WITH pred_table AS (
SELECT
  5 as seniority,
  '3-Advanced' as experience,
  'Billing' as category,
  'Request' as type
)
SELECT
  *
FROM
  ML.PREDICT(MODEL `helpdesk.predict_eta`,
    TABLE pred_table)
```

Job information <u>Results</u> JSON   Execution details						
Row	predicted_label	seniority	experience	category	type	
1	3.661129542367746	5	3-Advanced	Billing	Request	

When seniority is **5**, experience is **3-Advanced**, category is **Billing**, and type is **Request**, our model is saying that the average response time is **3.74 days**.

# Create a Dialogflow Agent

In a new browser tab, create an Agent at Dialogflow.

1. Click this [Dialogflow.com](https://dialogflow.com) link.
2. If required, click **Sign-in with Google**, then use the credentials you logged into this lab with.
3. Check **Yes, I have read and accept the agreement**, and then **Accept** the agreement.
4. Click **Create Agent**.
5. Name your agent and select other properties such as **language** and **time zone**.
6. Set **Google Project** to your Qwiklabs ID.

☰	DFflow-agent	CREATE
<hr/>		
DEFAULT LANGUAGE ⓘ		DEFAULT TIME ZONE
English — en ▼		(GMT-5:00) America/New_York ▼
Primary language for your agent. Other languages can be added later.		Date and time requests are resolved using this timezone.
<hr/>		
GOOGLE PROJECT		
qwiklabs-gcp-62e4adc78c393ff9 ▼		
Enables Cloud functions, Actions on Google and permissions management.		
<hr/>		

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

## Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

Create a Dialogflow Agent

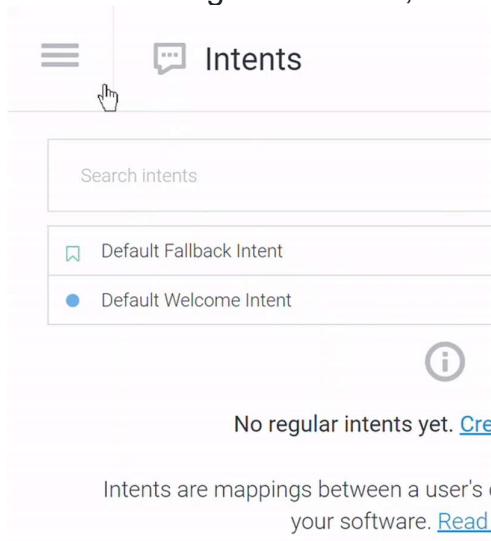
Check my progress

# Import Intents & Entities for a Simple Helpdesk Agent

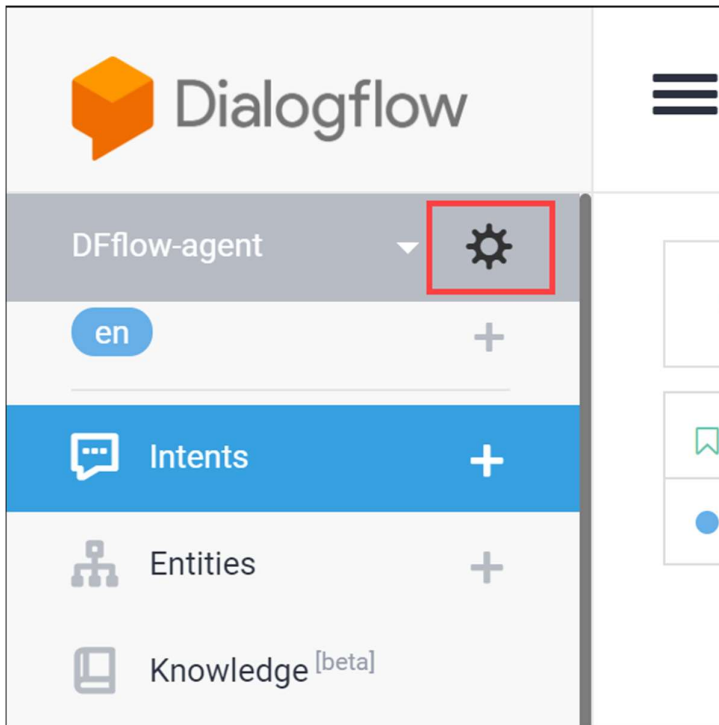
To save time, instead of configuring an agent from scratch, you are going to import the intents and entities for an existing helpdesk agent.

## Import an IT helpdesk agent

- Download settings file from <https://github.com/googlecodecloudlabs/cloud-dialogflow-bqml/raw/master/ml-helpdesk-agent.zip> and save it to your local computer.
- Still in the Dialogflow console, click the hamburger icon in the top left to open the left panel.



- Click on the gear icon  next to your existing agent.



- Click on **Export and Import** tab from the list of settings.
- Select **Import from zip**.
- Select the `ml-helpdesk-agent.zip` file from your local computer.
- Type `IMPORT` in the given textbox and click **Import**.
- Click **Done**.

Once the import completes, use the left panel to navigate to **Intents**. Examine the intents that were imported. **Submit Ticket** is the main intent, which has the follow-up intents **Submit Ticket - Email** and **Submit Ticket - Issue Category**. **Submit Ticket - Email** is used to collect a user's email address and **Submit Ticket - Issue Category** is used to collect the issue description from the user and automatically infer a support category.

Use the left panel to navigate to **Entities** and look at the **@category** entity. This entity will be used to map a request description that the user provides to support category. The support category will be used for predicting response time.

### Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

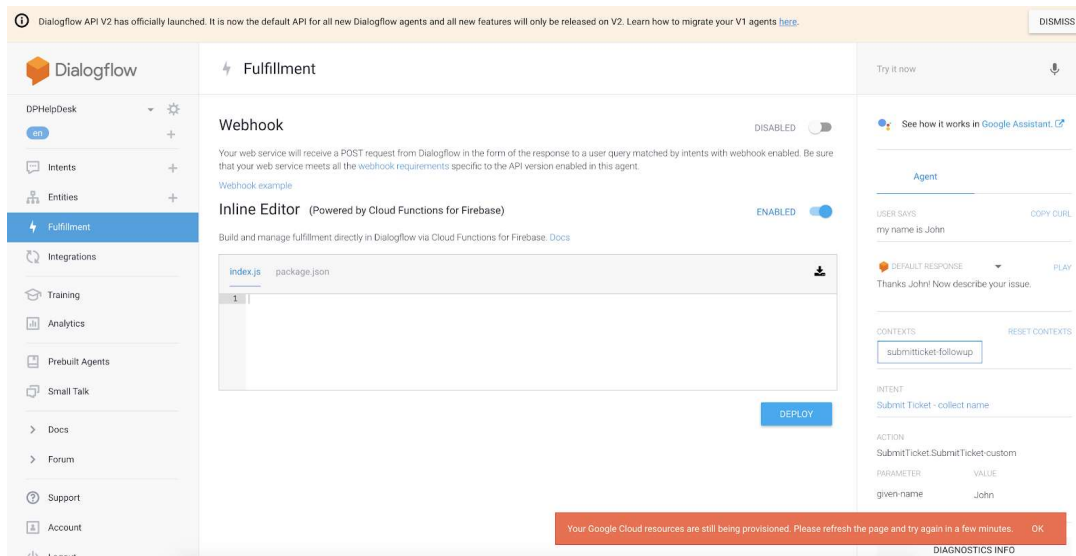
Import an IT Helpdesk Agent

Check my progress

# Use the Inline Editor to create a fulfillment that integrates with BigQuery

**Note:** The code doesn't store the ticket. It only sends a decorated message back to the user to demonstrate the possibilities.

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



**Note:** 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:

```
/**
 * Copyright 2020 Google Inc. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

"use strict";

const functions = require("firebase-functions");
const { WebhookClient } = require("dialogflow-fulfillment");
const { Card, Payload } = require("dialogflow-fulfillment");
const { BigQuery } = require("@google-cloud/bigquery");

const bigquery = new BigQuery({
  projectId: "your-project-id" // ** CHANGE THIS **
});

process.env.DEBUG = "dialogflow:debug";
```

```

function welcome(agent) {
  agent.add(`Welcome to my agent!`);
}

function fallback(agent) {
  agent.add(`I didn't understand`);
  agent.add(`I'm sorry, can you try again?`);
}

async function etaPredictionFunction(agent) {

  const issueCategory = agent.getContext('submitticket-email-followup').parameters.category;
  const sqlQuery = `WITH pred_table AS (SELECT 5 as seniority, "3-Advanced" as experience,
    @category as category, "Request" as type)
  SELECT cast(predicted_label as INT64) as predicted_label
  FROM ML.PREDICT(MODEL helpdesk.predict_eta, TABLE pred_table)`;
  const options = {
    query: sqlQuery,
    location: "US",
    params: {
      category: issueCategory
    }
  };

  const [rows] = await bigquery.query(options);

  return rows;
}

async function ticketCollection(agent) {

  const email = agent.getContext('submitticket-email-followup').parameters.email;
  const issueCategory = agent.getContext('submitticket-email-followup').parameters.category;

  let etaPrediction = await etaPredictionFunction(agent);

  agent.setContext({
    name: "submitticket-collectname-followup",
    lifespan: 2
  });

  agent.add(`Your ticket has been created. Someone will contact you shortly at ${email}.
  The estimated response time is ${etaPrediction[0].predicted_label} days.`);
}

exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response)
=> {
  const agent = new WebhookClient({ request, response });
  console.log('Dialogflow Request headers: ' + JSON.stringify(request.headers));
  console.log('Dialogflow Request body: ' + JSON.stringify(request.body));
  let intentMap = new Map();
  intentMap.set("Default Welcome Intent", welcome);
  intentMap.set("Default Fallback Intent", fallback);
  intentMap.set("Submit Ticket - Issue Category", ticketCollection);
  agent.handleRequest(intentMap);
});

```

While in this file update the BIGQUERY\_CLIENT variable. Replace your-project-id with your Project ID:

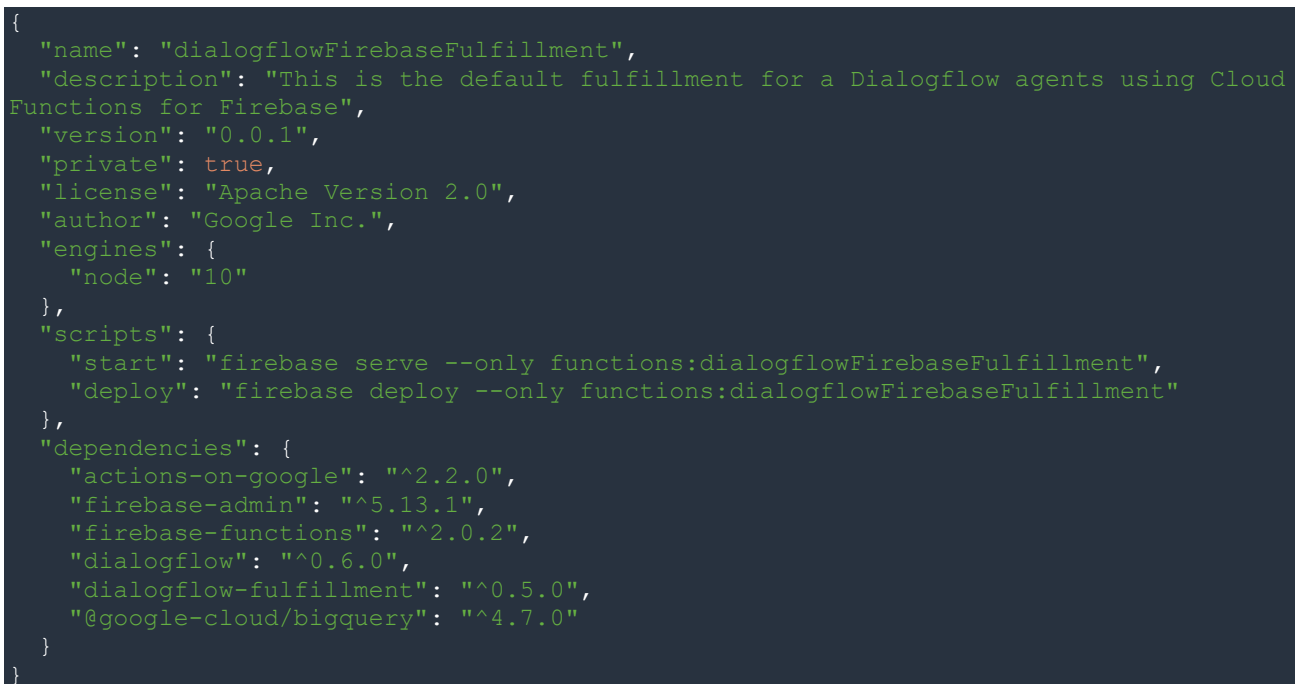


Notice that your fulfillment script contains the **ML.PREDICT** function in Query statement. This is what is going to return a prediction of response time back to the client. Dialogflow will automatically categorize the ticket description and send the category to **BigQuery ML** for predicting issue response time.



```
index.js package.json
41
42 const issueCategory = agent.getContext('submitticket-email-followup').parameters.category;
43 const sqlQuery = `WITH pred_table AS (SELECT 5 as seniority, "3-Advanced" as experience,
44 @category as category, "Request" as type)
45 SELECT cast(predicted_label as INT64) as predicted_label
46 FROM ML.PREDICT(MODEL helpdesk.predict_eta, TABLE pred_table)`;
47 const options = {
48   query: sqlQuery,
49   location: "US",
50   params: {
51     category: issueCategory
52   }
53 };
54
55 const [rows] = await bigquery.query(options);
```

Copy the following code and paste it in the **package.json** tab, replacing the existing content:



```
{
  "name": "dialogflowFirebaseFulfillment",
  "description": "This is the default fulfillment for a Dialogflow agents using Cloud Functions for Firebase",
  "version": "0.0.1",
  "private": true,
  "license": "Apache Version 2.0",
  "author": "Google Inc.",
  "engines": {
    "node": "10"
  },
  "scripts": {
    "start": "firebase serve --only functions:dialogflowFirebaseFulfillment",
    "deploy": "firebase deploy --only functions:dialogflowFirebaseFulfillment"
  },
  "dependencies": {
    "actions-on-google": "^2.2.0",
    "firebase-admin": "^5.13.1",
    "firebase-functions": "^2.0.2",
    "dialogflow": "^0.6.0",
    "dialogflow-fulfillment": "^0.5.0",
    "@google-cloud/bigquery": "^4.7.0"
  }
}
```

Click the **Deploy** button. Wait until you see a message that the deployment was successful. This may take a few minutes.

If the deployment fails, try refreshing and clicking Deploy one more time.

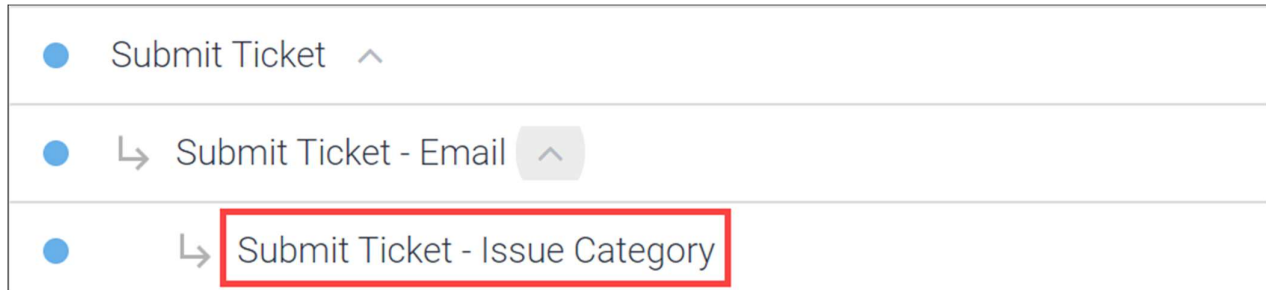
## Test Completed Task

To check your progress in this lab, click **Check my progress** below. A checkmark means you're on track.

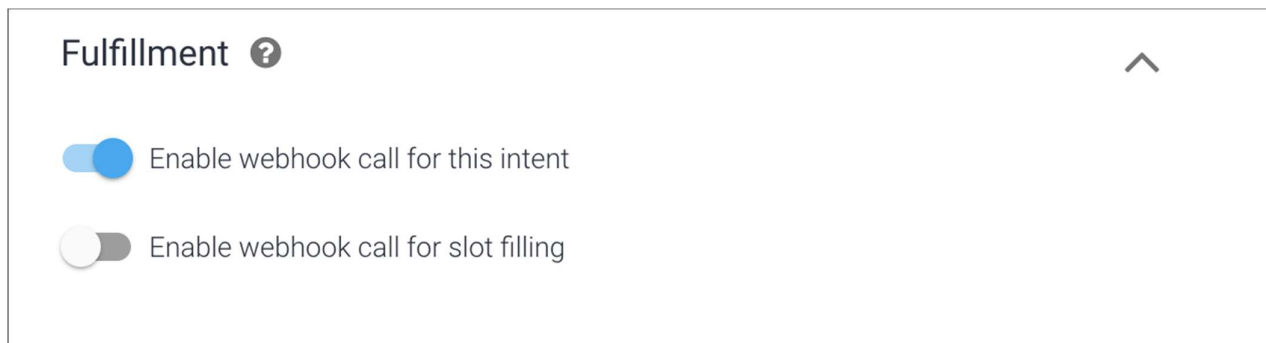
# Enable webhook for fulfillment

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

Click on the last follow-up intent **Submit Ticket - Issue Category** to open it for editing.



Scroll to the bottom, in the **Fulfillment** section, verify that **Enable webhook call for this intent** is turned on.



# Test your chatbot

At this point, the Dialogflow should be set up.

Test it in the **Try it now** panel on the right by typing in the following conversation, one line at a time:

1. Hi
2. I would like to submit a ticket
3. My email is student@qwiklabs.net
4. I can't login

Try it now

Please use test console above to try a sentence.

See how it works in [Google Assistant](#). [↗](#)

The output from #4 should look like:

Try it now

Agent

USER SAYS

COPY CURL

I can't login

DEFAULT RESPONSE

▼

Your ticket has been created. Someone will you contact shortly at student@qwiklabs.net. The estimated response time is 4 days.

**Note:** If you don't see the expected output depicted above, go back to the Fulfillment section of the agent and redeploy the code you copied into the Inline Editor. Also verify that you updated the **projectId** specified in the code to match your project.

## Understanding the BigQuery integration

Remember that your BigQuery ML model required other fields to return a prediction: **seniority** and **experience**. If this were a real world app, you could programmatically get the user's seniority & experience from a company database using the name or company ID they provide. For this example assume **5** as seniority and **3-Advanced** as experience.



```
index.js package.json
41
42 const issueCategory = agent.getContext('submitticket-email-followup').parameters.category;
43 const sqlQuery = `WITH pred_table AS (SELECT 5 as seniority, "3-Advanced" as experience,
44 @category as category, "Request" as type)
45 SELECT cast(predicted_label as INT64) as predicted_label
46 FROM ML.PREDICT(MODEL helpdesk.predict_eta, TABLE pred_table)`;
47 const options = {
48   query: sqlQuery,
49   location: "US",
50   params: {
51     category: issueCategory
52   }
53 };
54
55 const [rows] = await bigquery.query(options);
```

**Extra Credit:** Change the fulfillment script so that your response time prediction is based on a user with 2 as seniority instead of 5. Then retest the bot. How does this change the Estimated ETA?

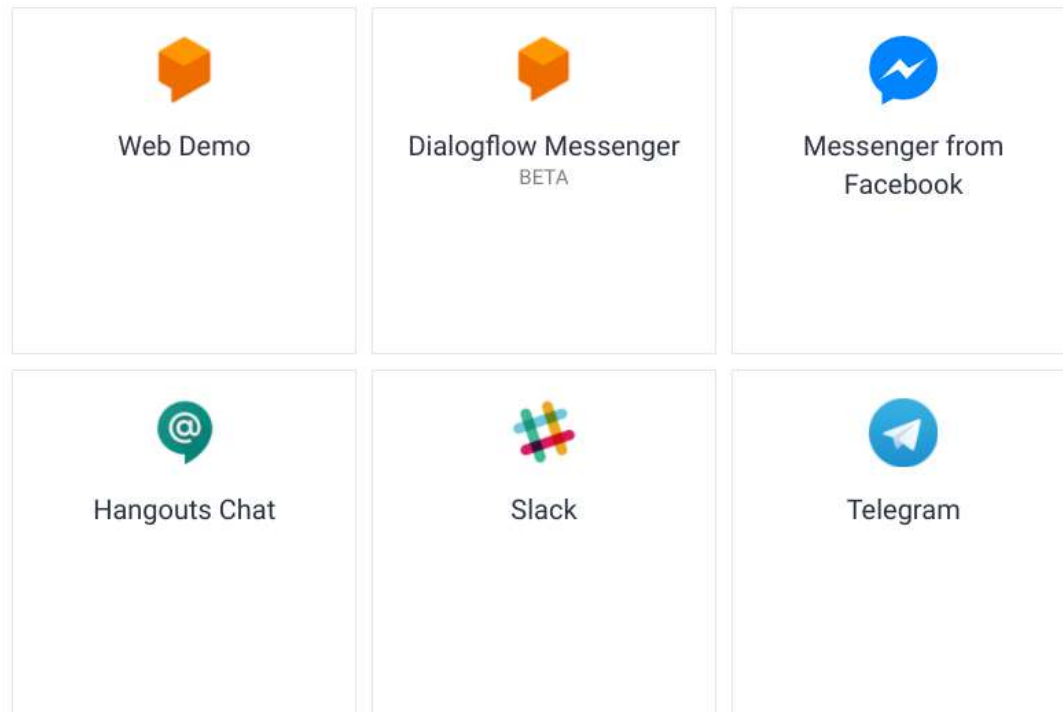
To extend this example further, you could also collect the **seniority** and **experience** information from the chatbot user by leveraging the Slot Filling functionality in Dialogflow. [Click here](#) to learn more about Slot Filling in Dialogflow.

# Test using a one-click integration

Dialogflow provides many types of integrations for your chatbot. Take a look now at a sample web user interface.

Click on **Integrations** in the Dialogflow left panel.

Click the **Web Demo** integration, and then 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/2df3d627-f6af-4245-85dd-6a48bd442819> 

CLOSE

ENABLE

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 your page to see all conversations going. [More in documentation.](#)

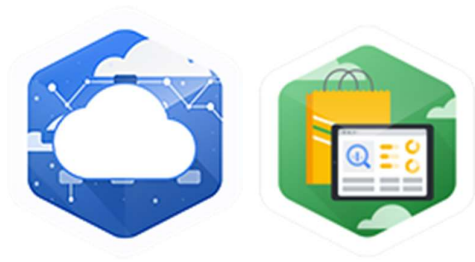
<https://bot.dialogflow.com/a3673ccf-1736-462d-a986-a9c1dcb993a2> 

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 email is student@qwiklabs.net"
- Provide the ticket details of "My printer is broken"

# Congratulations!

You built a custom machine learning model, and you're now a chatbot developer!



## Finish your Quest

This self-paced lab is part of the Qwiklabs [BigQuery for Machine Learning](#) and [Applying BQML's Classification, Regression, and Demand Forecasting for Retail Applications](#) Quests. 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 public and link to them in your online resume or social media account. Enroll in a Quest and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](#).

## Next steps / learn more:

- Check out the code samples at the [Dialogflow Github](#) page.
- [Exploring NCAA Data with BigQuery](#)

Manual Last Updated April 29, 2021

Lab Last Tested April 29, 2021

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.