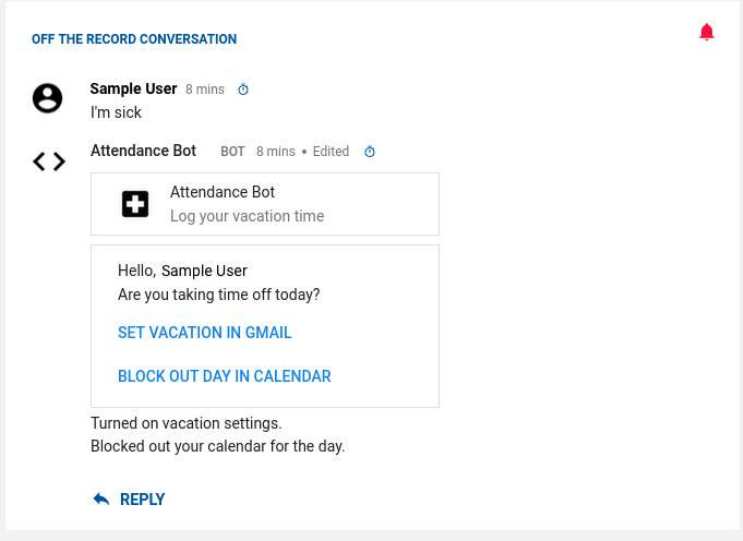
Hangouts Chat bot - Apps Script

GSP250

Overview

Hangouts Chat bots provide easy-to-use access points to your organization's data and services. Users can interact with bots conversationally within a chat experience.

Among other implementation options, you can create a Hangouts Chat bot using Google Apps Script. By building your bot in Apps Script, you can easily access other Google services like Drive, Gmail, Calendar, Docs, Sheets, and much more.
In this lab, you learn how to create a simple Hangouts Chat bot—"Attendance Bot"—using Google Apps Script. The bot integrates with Gmail to set a user's vacation responder and integrates with Calendar to put a meeting on the user's calendar.



What you'll learn

- How to add handlers in events raised in Hangouts Chat
- How to parse event objects sent from Hangouts Chat
- How to respond back to Hangouts Chat with card-formatted responses
- How to define and react to custom actions for button clicks in cards

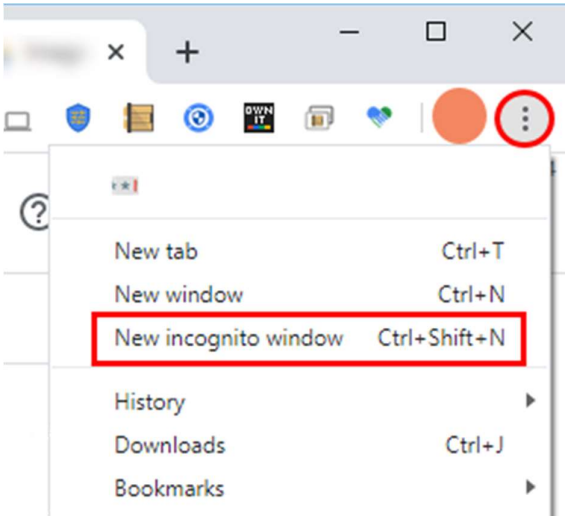
Setup and requirements

What you need

- An internet browser. Chrome browser is recommended.
- Time. 10 minutes to read through steps in the lab. 30 minutes to step through the hands-on practice lab. Once you start this lab, you cannot pause and return later.
  All work for this lab is done in your Qwiklab G Suite account. If you have a different G Suite or Google account open, **sign out** before you start this lab. If you are signed into more than one account, you get the message "Server error is not authorized to access Admin Console of .... Note: Multi-Login is not supported for resellers in Admin Console`".

Note: If you are using a Pixelbook, run this lab in an Incognito window. To do this:

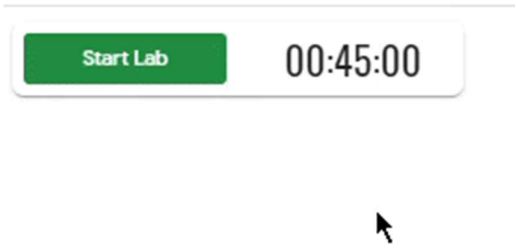
In the Chrome browser, click on the three dots next to your user picture, then select **New incognito window**.



Start your lab

When you are ready, click **Start Lab**. If you need to pay for the lab, a pop-up opens for you to select your payment method.

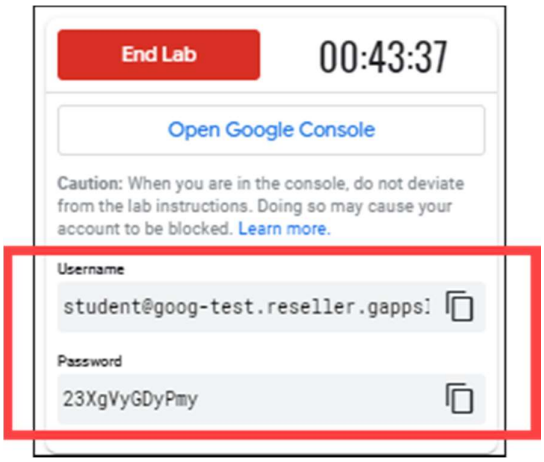
You can track your lab's setup progress with the status message in the blue box.



When the lab is set up, you will see the button to **Open Google Console** and the temporary credentials you must use for this lab.

Username and Password

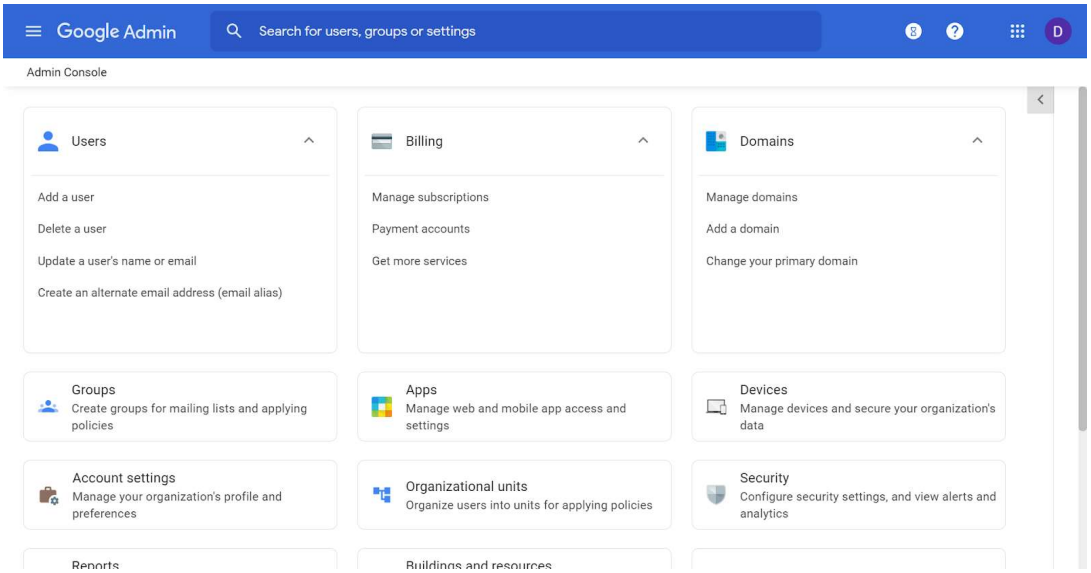
Find the Username and Password you need to access your Qwiklab's Google account under the button to **Open Google Console**.



Sign in to Google Workspace Admin Console

Visit admin.google.com or click **Open Google Console** in the upper-left panel and enter the username and password located in the connection details panel
Accept the Google Workspace via Reseller Agreement and **Accept Terms of Service** to accept the Supplemental Terms and Conditions For the Google Workspace Enterprise Plus free trial.

The **Admin console** opens.



If you get the server error "We are unable to process your request at this time, please try again later", the Admin console is still spinning up. Close the browser window and click **Open Google Console** again.

The sample code

The code used in this lab is in [https://github.com/googlecodelabs/hangouts-chat-apps-script]. You can view the code at GitHub or download it to your computer.
The `step-NN` folders contain the desired end state of each step of this lab. They are there for reference.

Create the handlers for Hangouts Chat events

To implement your bot, create a new Google Apps Script script:

1. Click this Google Apps Script editor link to open Google Apps Script online editor.
2. Click *Untitled project* (the current name). In the **Edit project name** dialog, rename the new script "Attendance Bot" then click **OK.**

Events in Hangouts Chat

Most Apps Script bot interactions with Hangouts Chat are event-driven. The interaction between the user, the bot, and Hangouts chat typically follows this sequence:

1. A user initiates an action, like adding a bot to a room, starting a direct message (DM) with a bot, or removing the bot from a room.
2. The action raises an event aimed at the bot in Hangouts Chat.
3. Hangouts Chat calls the corresponding event handler defined in the bot's script.

Hangouts Chat raises four events that your Apps Script bot can listen for:

- `ADDED_TO_SPACE`: This event occurs when a human user adds a bot to a room or a DM. In Apps Script, you define an `onAddToSpace()` function to handle this event.
- `REMOVED_FROM_SPACE`: This event occurs when a user removes the bot from a room or DM. This event does not post a response back to Hangouts Chat. In Apps Script, you define an `onRemoveFromSpace()` function to handle this event.
- `MESSAGE`: This event occurs when a user messages the bot, either directly in a DM or as an @mention in a room. In Apps Script, you define an `onMessage()` function to respond to this event.
- `CARD_CLICKED`: This event occurs when the user clicks a button with a custom action assigned to it. In Apps Script, you define an `onCardClick()` function to respond to this event.

To define the handlers for the for the `ADDED_TO_SPACE` and `REMOVE_FROM_SPACE` events, replace the contents of the `Code.gs` file with the following code. (You add handlers for the `MESSAGE` and `CARD_CLICKED` events later in this lab.)

Code.gs

```
/**
 * Responds to an ADDED_TO_SPACE event in Hangouts Chat.
 * @param {object} event the event object from Hangouts Chat
 * @return {object} JSON-formatted response
 * @see https://developers.google.com/hangouts/chat/reference/message-formats/events
 */
function onAddToSpace(event) {
  console.info(event);
  var message = 'Thank you for adding me to ';
  if (event.space.type === 'DM') {
    message += 'a DM, ' + event.user.displayName + '!';
  } else {
    message += event.space.displayName;
  }
  return { text: message };
}

/**
 * Responds to a REMOVED_FROM_SPACE event in Hangouts Chat.
 * @param {object} event the event object from Hangouts Chat
 * @see https://developers.google.com/hangouts/chat/reference/message-formats/events
 */
function onRemoveFromSpace(event) {
  console.info(event);
  console.log('Bot removed from ', event.space.name);
}
```

Save `Code.gs`.

Publish the bot

Before you run and test the bot, you must publish your bot to your Workspace organization through the Cloud Console. To publish your bot, you

- Update the manifest file
- Enable the and configure the Hangouts Chat bot

Update the script manifest

Before you publish your bot to Hangouts Chat, you must update the script manifest.

1. In the Apps Script editor, click **View > Show manifest file**.

The `appsscript.json` file opens in the editor.

2. Add a comma after the line: `"exceptionLogging": "STACKDRIVER"` and then add a new line `"chat": {}` to your manifest file.

Your manifest file should look similar to the following example.

appsscript.json

```
{
  "timeZone": "America/New_York",
  "dependencies": {
  },
  "exceptionLogging": "STACKDRIVER",
  "chat": {}
}
```

3. Save `appsscript.json`

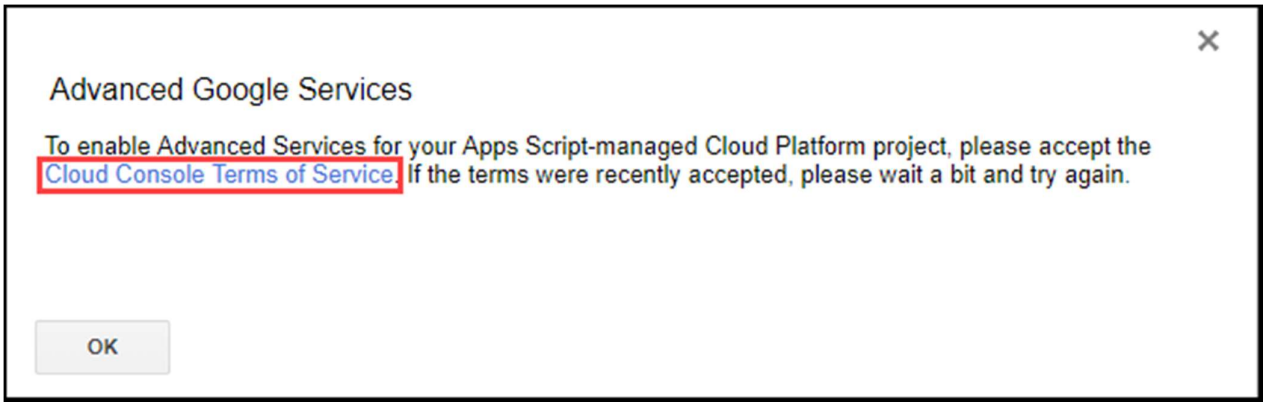
To configure and publish your bot to Hangouts Chat, you need the **Deployment ID**:

1. From the Apps Script editor, get the deployment ID for the script by clicking **Publish** > **Deploy from manifest**.
2. In the **Deployments** dialog box, click **Get ID**.
3. In the **Deployment ID** dialog box, copy the value listed for the **Deployment ID** to use later in this lab. Click **Close** and **Close** to dismiss the dialog boxes.

Enable and configure the Hangouts Chat API

1. Still in the App Script editor, click **Resources** > **Advanced Google services**.

2. In the Advanced Google Services window, click **Cloud Console Terms of Service**.



3. The Google Cloud opens. If necessary, enter your Username and Password (from the upper left panel) as required.

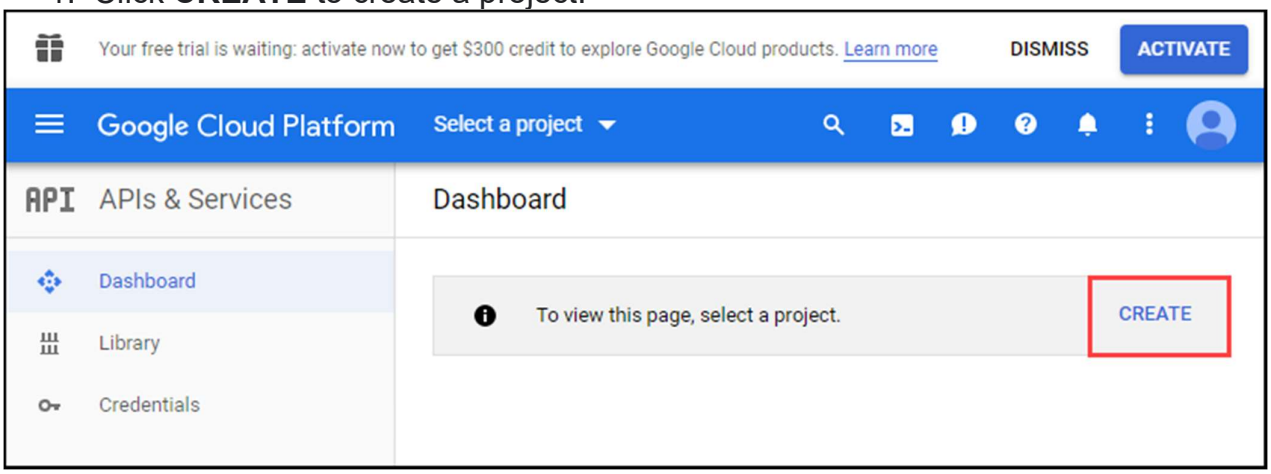
4. In the Welcome! dialog, review, then check that you agree to **Terms of Service**. Click **AGREE AND CONTINUE**.

The Cloud Console, API & Services window opens.

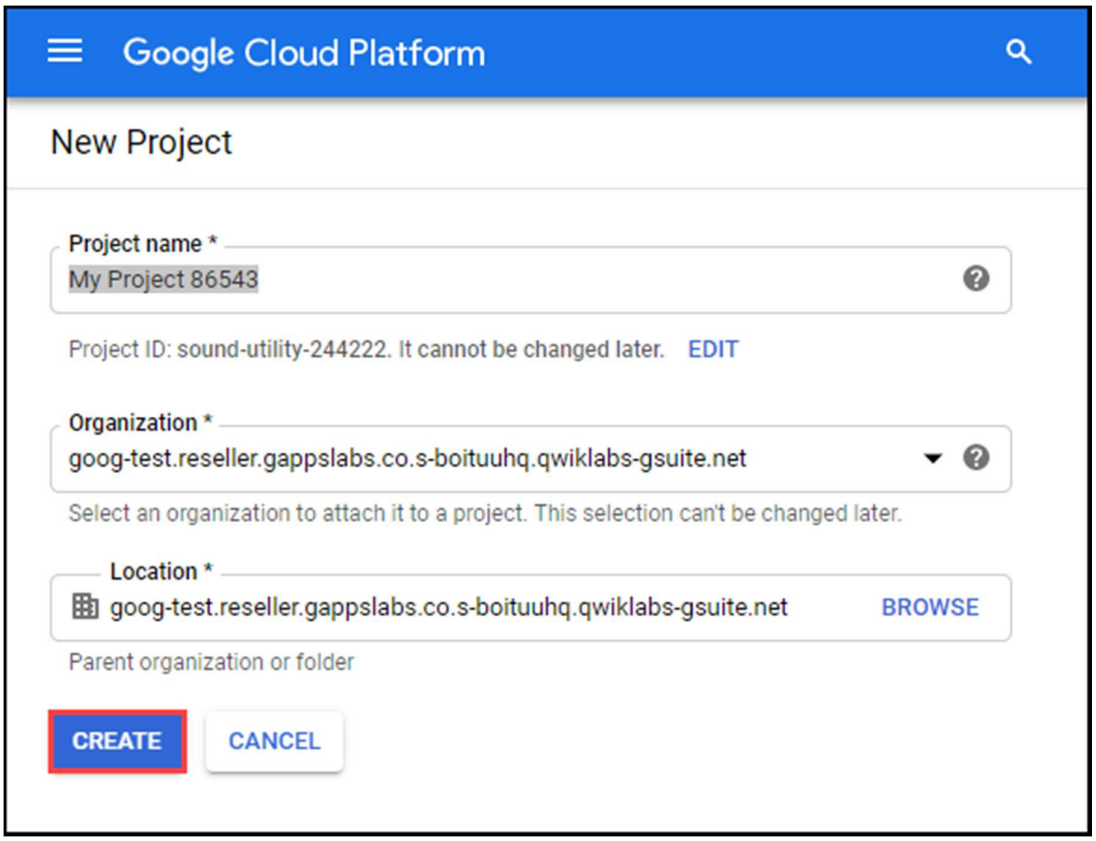
Create a Google Cloud Project

You must create a Google Cloud Project before you can enable an API.

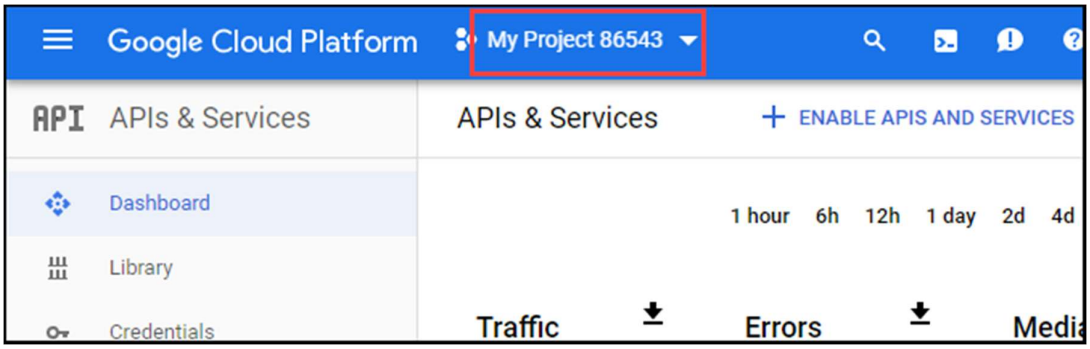
1. Click **CREATE** to create a project.



2. Leave all fields at the default values and click **CREATE**.



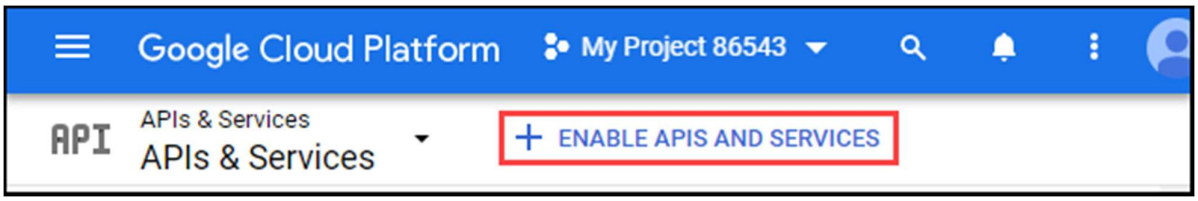
The project you created is now selected in the Cloud Console.



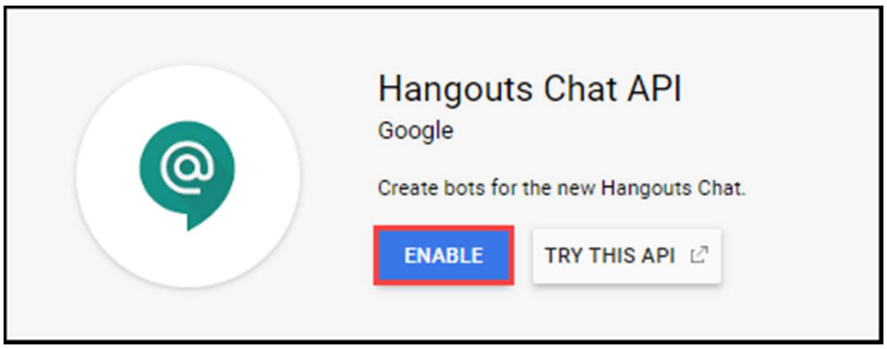
Enable the Hangouts Chat API

Enable the Hangouts Chat API from Cloud Console.

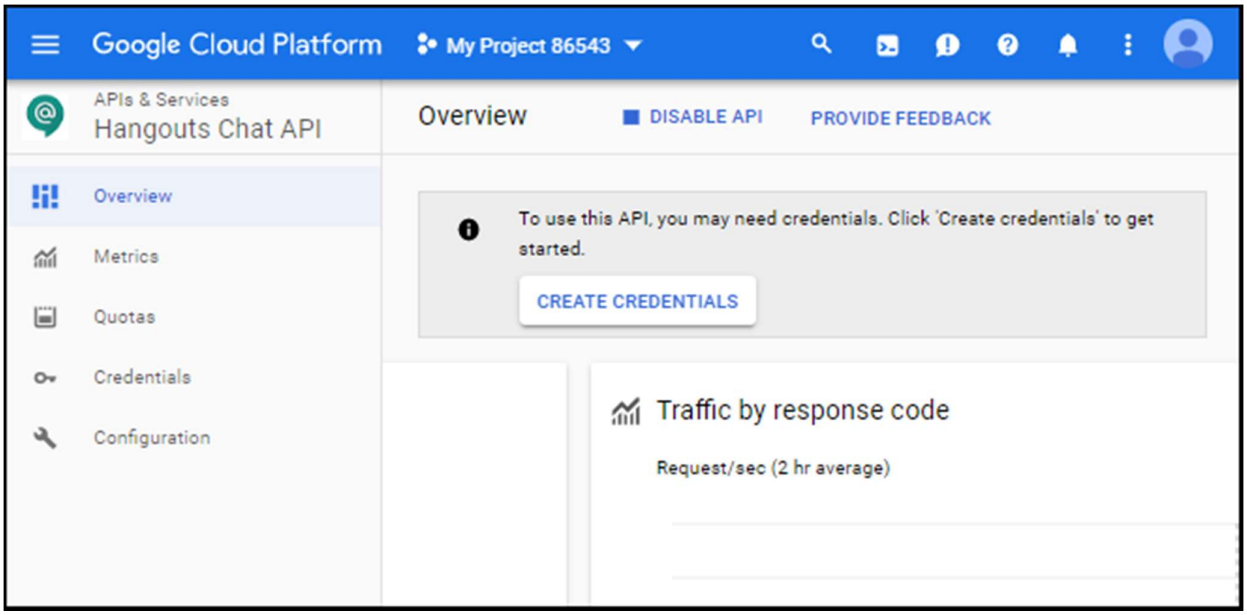
1. Click **ENABLE APIS AND SERVICES**.



2. In the Library, search for "Hangouts Chat". Select the **Hangouts Chat API** from the list of results.
3. On the Hangouts Chat API dialog, click **Enable**.



The Cloud Console displays a progress indicator while enabling the API. Once enabled, Hangouts Chat API window opens in the Cloud Console.



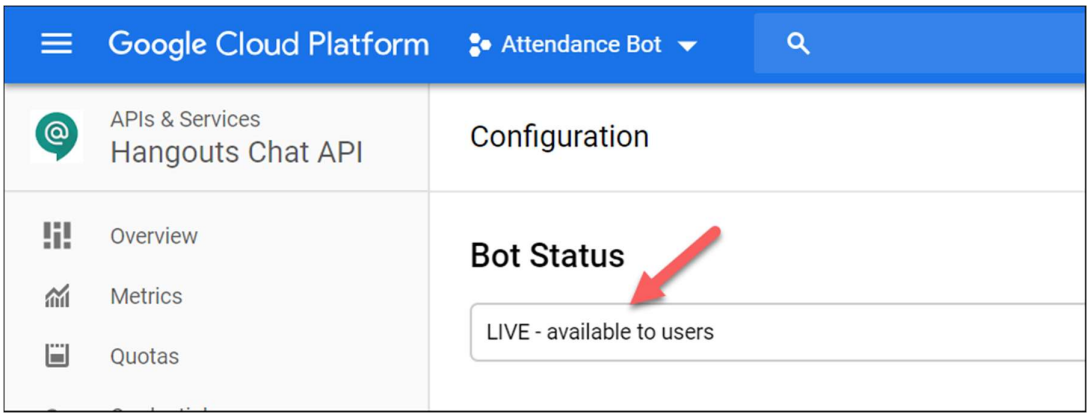
Configure and publish the Hangouts Chat bot

1. Click **Configuration** in the left pane.
2. In the **Configuration** dialog, set the fields with the following values:

Field	Value
Bot name	Attendance Bot
Avatar URL	https://goo.gl/kv2ENA
Description	Apps Script lab bot
Functionality	Bot works in direct messages
Connection settings	check **Apps Script project**, and then paste your script's Deployment ID into the Deployment ID field
Permissions	select **Specific people and group in your domain**, then in the text box, enter your Username (from the **Connection Details** panel of your lab)

3. Click **SAVE**.

After you save your changes, verify that the status in the **Configuration** section shows the **Bot Status** to be **LIVE – available to users**.

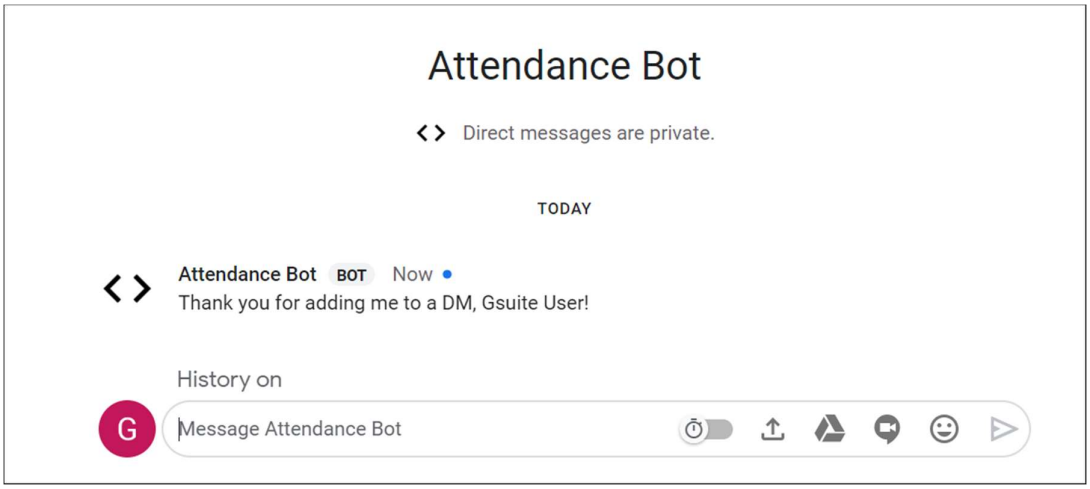


Test the bot

To test your bot in Hangouts Chat, do the following:

1. Click this Hangouts Chat link to open Hangouts Chat.
2. Click **Find people, rooms, bots > Message a bot**.
3. From the list, select the "*Attendance Bot, Apps Script lab bot*" that you created.

When the direct message thread opens, you should see a message from the bot thanking you for adding it to a DM, as shown in the following image.



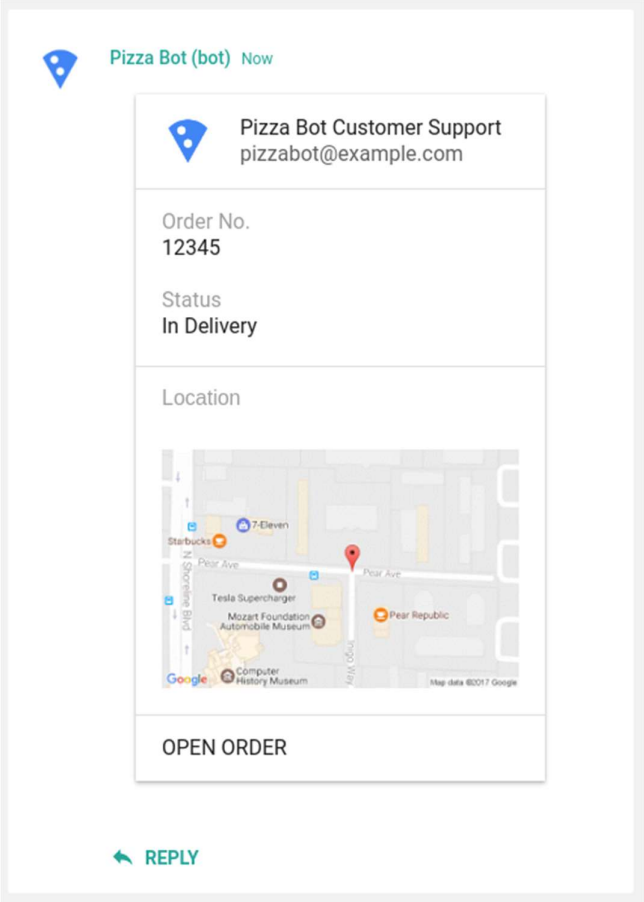
Note: The code for this bot logs the event objects received from Hangouts Chat. You can look at the structure of the event objects by viewing the logs. To open the logs, in the Apps Script editor click **View > Cloud Logging**.

Define a card-formatted response

In the previous step, your bot responded to Hangouts Chat events with a simple text response. In this step, you will update your bot to respond with *cards*.

Card responses

Hangouts Chat supports the use of cards for responses. Cards are visual containers that allow you to group sets of user interface widgets together. Cards can display headers, text paragraphs, sets of buttons, images, and key/value text. Your bot can define one or many cards in its JSON response to Hangouts Chat, which then translates your response into the corresponding UI elements.
The following image shows a card response with three sections, that includes a header, a key/value widget, an image widget, and a text button.



To respond to user messages with a card response, in Apps Script editor, add the following code to your bot's `Code.gs` file.

Code.gs

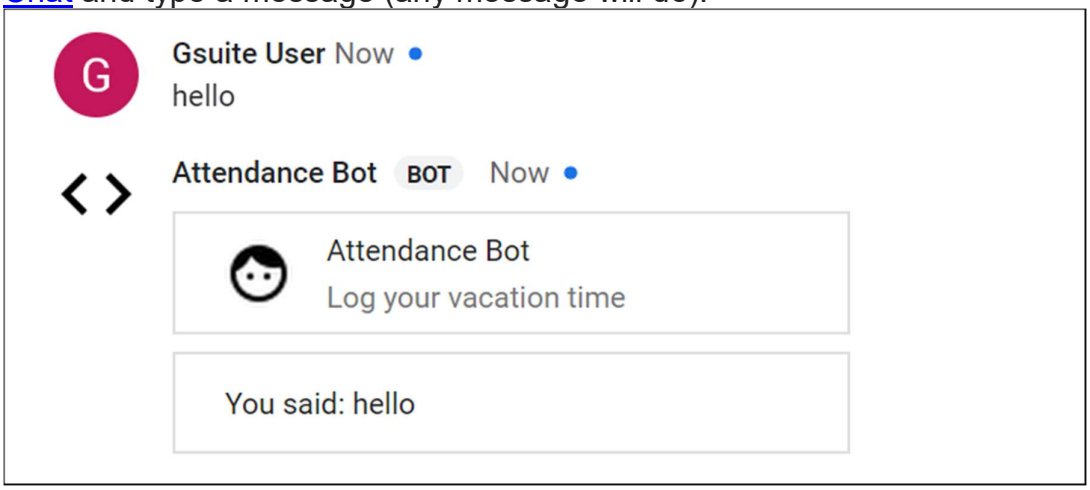
```
var DEFAULT_IMAGE_URL = 'https://goo.gl/bMqzYS';
var HEADER = {
  header: {
    title : 'Attendance Bot',
    subtitle : 'Log your vacation time',
    imageUrl : DEFAULT_IMAGE_URL
  }
};

/**
 * Creates a card-formatted response.
 * @param {object} widgets the UI components to send
 * @return {object} JSON-formatted response
 */
function createCardResponse(widgets) {
  return {
    cards: [HEADER, {
      sections: [{
        widgets: widgets
      }]
    }]
  };
}

/**
 * Responds to a MESSAGE event triggered
 * in Hangouts Chat.
 *
 * @param event the event object from Hangouts Chat
 * @return JSON-formatted response
 */
function onMessage(event) {
  var userMessage = event.message.text;

  var widgets = [{
    "textParagraph": {
      "text": "You said: " + userMessage
    }
  }];

  return createCardResponse(widgets);
}
```

Save `Code.gs`.

The `onMessage()` function, added in this step, reads the user's original message and constructs a response as a simple TextParagragh widget. The `onMessage()` function then calls `createCardResponse()`, which places the TextParagraph widget within a section of a single card. The bot returns the JavaScript object constructed with the card response back to Hangouts Chat.

Test the bot

To retest this bot, simply go back to your direct message with the bot in Hangouts Chat and type a message (any message will do).



Note that the `onMessage()` event handler parses the event object passed to it by Hangouts Chat to extract the user's original message. You can also get other types of information about the event, including the name of the user that initiated the event, their email address, the name of the room that the event occurred in, and much more.

For more information about the structure of the event objects sent by Hangouts Chat, see the Event formats reference.

React to button clicks in cards

In the previous step, your bot responded to a message from a user—a `MESSAGE` event—with a simple card that contained a TextParagragh widget. In this step, you will create a response that includes buttons, where each button has a custom action defined for it.

Interactive cards

Card responses can contain one of two types of buttons: TextButton widgets, which display text-only buttons; and ImageButton widgets, which display a button with a simple icon or image without text. Both TextButton and ImageButton widgets support one of two `onClick` behaviors (as defined in the JSON response sent back to Hangouts Chat):

either `openLink` or `action`. As the name implies, `openLink` opens a specified link in a new browser tab.

The `action` object, however, specifies a custom action for the button to perform. You can specify several arbitrary values in the action object, including a unique `actionMethodName` and a set of key / value parameter pairs.

Specifying an `action` object for the button creates an interactive card. When the user clicks the button in the message, Hangouts Chat raises a `CARD_CLICKED` event and sends a request back to the bot that sent the original message. The bot then needs to handle the event raised from Hangouts Chat and return a response back to the space.

1. Replace the `onMessage()` function in `Code.gs` with the following code. This code creates two buttons, a **Set vacation in Gmail** and a **Block out day in Calendar** button in the card sent to Hangouts Chat.

Code.gs

```
var REASON = {
  SICK: 'Out sick',
  OTHER: 'Out of office'
};
/**
 * Responds to a MESSAGE event triggered in Hangouts Chat.
 * @param {object} event the event object from Hangouts Chat
 * @return {object} JSON-formatted response
 */
function onMessage(event) {
  console.info(event);
  var reason = REASON.OTHER;
  var name = event.user.displayName;
  var userMessage = event.message.text;

  // If the user said that they were 'sick', adjust the image in the
  // header sent in response.
  if (userMessage.indexOf('sick') > -1) {
    // Hospital material icon
    HEADER.header.imageUrl = 'https://goo.gl/mnZ37b';
    reason = REASON.SICK;
  } else if (userMessage.indexOf('vacation') > -1) {
    // Spa material icon
    HEADER.header.imageUrl = 'https://goo.gl/EbgHuc';
  }

  var widgets = [{
    textParagraph: {
      text: 'Hello, ' + name + '.<br/>Are you taking time off today?'
    }
  }, {
    buttons: [{
      textButton: {
        text: 'Set vacation in Gmail',
        onClick: {
          action: {
            actionMethodName: 'turnOnAutoResponder',
            parameters: [{
              key: 'reason',
              value: reason
            }]
          }
        }
      }
```

```
    }, {
      textButton: {
        text: 'Block out day in Calendar',
        onClick: {
          action: {
            actionMethodName: 'blockOutCalendar',
            parameters: [{
              key: 'reason',
              value: reason
            }]
          }
        }
      }
    }]
  }];
  return createCardResponse(widgets);
}
```

2. To handle the `CARD_CLICKED` event, you need to add the `onCardClick()` function to your bot's script. Add the following code that defines the `onCardClick()` function `Code.gs`.

Code.gs

```
/**
 * Responds to a CARD_CLICKED event triggered in Hangouts Chat.
 * @param {object} event the event object from Hangouts Chat
 * @return {object} JSON-formatted response
 * @see https://developers.google.com/hangouts/chat/reference/message-formats/events
 */
function onCardClick(event) {
  console.info(event);
  var message = '';
  var reason = event.action.parameters[0].value;
  if (event.action.actionMethodName == 'turnOnAutoResponder') {
    turnOnAutoResponder(reason);
    message = 'Turned on vacation settings.';
  } else if (event.action.actionMethodName == 'blockOutCalendar') {
    blockOutCalendar(reason);
    message = 'Blocked out your calendar for the day.';
  } else {
    message = "I'm sorry; I'm not sure which button you clicked.";
  }
  return { text: message };
}
```

In responding to user clicks, now the bot does one of two things: it sets the user's vacation responder in Gmail to an "out of office" message; or it schedules an all-day meeting on the user's Calendar. To accomplish these tasks, the bot calls the Gmail advanced service and the Calendar Apps Script API.

3. Add the following code to your script to integrate the bot with Gmail and Calendar.

Code.gs

```
var ONE_DAY_MILLIS = 24 * 60 * 60 * 1000;
/**
 * Turns on the user's vacation response for today in Gmail.
 * @param {string} reason the reason for vacation, either REASON.SICK or REASON.OTHER
 */
function turnOnAutoResponder(reason) {
  var currentTime = (new Date()).getTime();
  Gmail.Users.Settings.updateVacation({
    enableAutoReply: true,
```

```
    responseSubject: reason,
    responseBodyHtml: "I'm out of the office today; will be back on the next business
day.<br><br><i>Created by Attendance Bot!</i>",
    restrictToContacts: true,
    restrictToDomain: true,
    startTime: currentTime,
    endTime: currentTime + ONE_DAY_MILLIS
  }, 'me');
}

/**
 * Places an all-day meeting on the user's Calendar.
 * @param {string} reason the reason for vacation, either REASON.SICK or REASON.OTHER
 */
function blockOutCalendar(reason) {
  CalendarApp.createAllDayEvent(reason, new Date(), new Date(Date.now() +
ONE_DAY_MILLIS));
}
```

4. Save `Code.gs`.

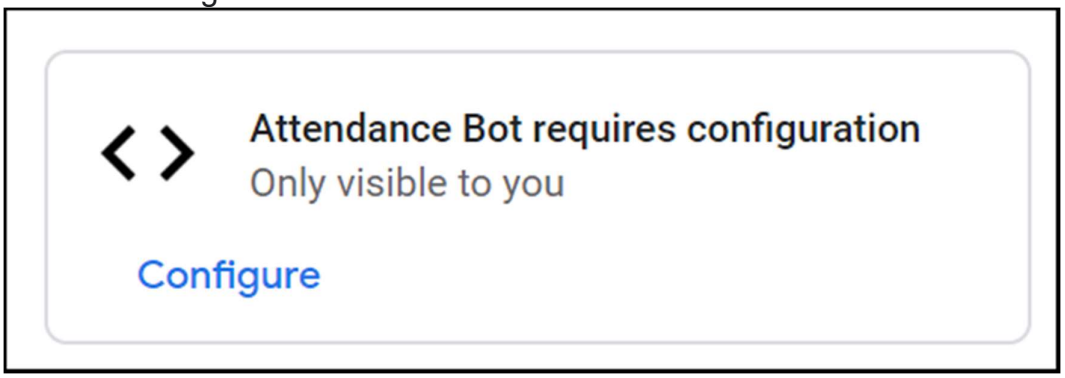
Finally, you need to enable the Gmail Advanced Service in the project. To enable the Gmail API, do the following:

1. Click **Resources > Advanced Google Services**.
2. In the list, find **Gmail API** and select **on**.
3. Still in the dialog box, click to open this Google API Console link.
4. Click **Enable APIs and Services**.
5. Search for 'Gmail API' and click the Gmail API card.
6. On the **Gmail API** page, click **Enable**.

Back in the Apps Script editor, if you get the message "Unable to load file" or "Unexpected error", click **Reload**.__

Test the bot

1. To test this version of your bot, open the DM that you started in previous steps in Hangouts Chat and type 'I'm sick'. The bot should respond with a card similar to the image below.

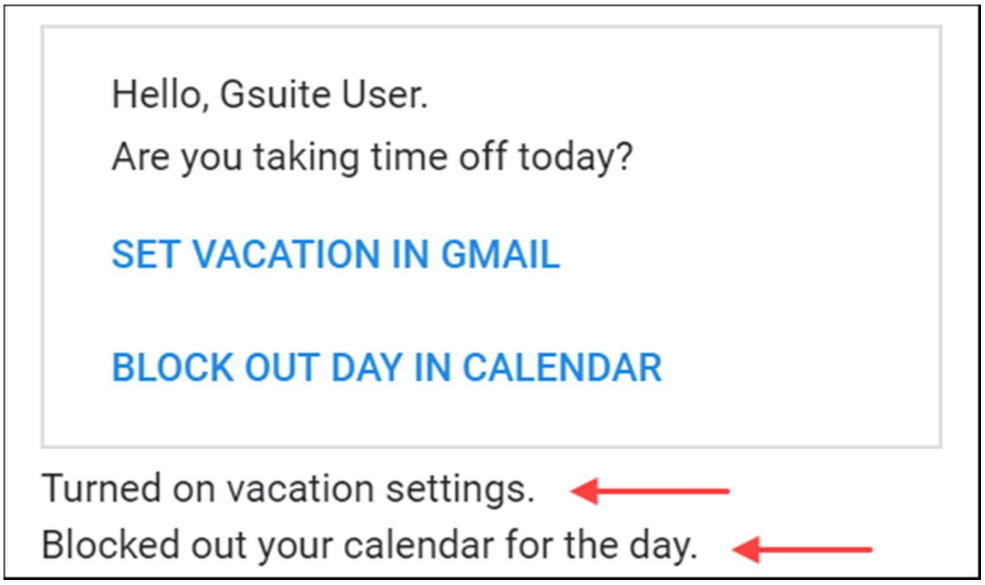


2. To configure the Attendance Bot, click **Configure**, choose your Workspace User account, click **ALLOW**, and then close the page when you see the message "You may close this page now".

3. Type your message a second time.

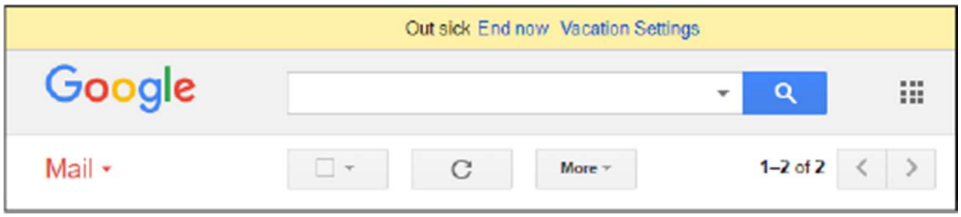
4. Click **SET VACATION IN GMAIL**. You should see the message "Turned on vacation settings."

5. Click **BLOCK OUT DAY IN CALENDAR**. You should see the message "Blocked out your calendar for the day."



6. Click the **Google Apps** icon to access then check the Gmail and Calendar associated with this account.

You should see the Vacation Setting in Gmail:



You should also see a day blocked out in the Calendar.

Congratulations!

Your bot can now respond to user messages, set their vacation responder in Gmail, and put an all-day event on their Calendar.

Finish Your Quest

This self-paced lab is part of the Workspace Integrations Quest. 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

Check out what else you can do with Workspace:

- Google Apps Script: Access Google Sheets, Maps & Gmail in 4 Lines of Code
- Big Data Analysis to Slide Presentation

Learn more

- Hangouts Chat documentation site
- Apps Script bot quickstart
- Create new bots
- Publish bots
- Create interactive cards
- Cards reference
- Event object reference
- Gmail advanced service reference
- Calendar Apps Script API reference
- Google Workspace Learning Center

Manual Last Updated Febuary 16, 2021
Lab Last Tested June 20, 2019