www.cloudskillsboost.google /course_sessions/7048758/labs/448290

# Develop an app with Duet AI | Google Cloud Skills Boost

Qwiklabs ⁝ 21-27 minutes

## Overview

In this lab, you use Duet AI, an AI-powered collaborator in Google Cloud, to explore, create, modify, test, and deploy a sample app.

You use Cloud Workstations to create a development environment that uses Code OSS as the IDE. You use Duet AI in Cloud Code to understand code and build a sample inventory app that has two API methods. You also use Duet AI to generate the steps for deploying your app to Cloud Run.

This lab is intended for developers of any experience level who contribute to building apps but might not be familiar with cloud application development. It helps to have some experience using VS Code or Code OSS as your IDE and to be familiar with Python and the Flask framework.

**Note:** As an early-stage technology, Duet AI can generate output that seems plausible but is factually incorrect. We recommend that you validate all output from Duet AI before you use it. For more information, see Duet AI in Google Cloud and responsible AI.

## Objectives

In this lab, you learn how to perform the following tasks:

- Create a cloud-based application development environment by using Cloud Workstations.
- Explore various Google services that you can use to deploy an app by asking Duet AI context-based questions.
- Prompt Duet AI to provide templates that you can use to develop a basic app in Cloud Run.
- Create, explore, and modify the app by using Duet AI to explain and generate the code.
- Run and test the app locally, and then deploy it to Google Cloud by using Duet AI to generate the steps.

## Setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time.
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.

4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
   If you use other credentials, you'll receive errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

## Activate Cloud Shell

Cloud Shell is a virtual machine that contains development tools. It offers a persistent 5-GB home directory and runs on Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources. `gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab completion.

1. Click the **Activate Cloud Shell** button (>_) at the top right of the console.

2. Click **Continue**.
   It takes a few moments to provision and connect to the environment. When you are connected, you are also authenticated, and the project is set to your *PROJECT_ID*.

**Sample commands**

- List the active account name:

gcloud auth list

(Output)

Credentialed accounts: - <myaccount>@<mydomain>.com (active)

(Example output)

Credentialed accounts: - google1623327_student@qwiklabs.net

- List the project ID:

gcloud config list project

(Output)

[core] project = <project_ID>

(Example output)

[core] project = qwiklabs-gcp-44776a13dea667a6 **Note:** Full documentation of **gcloud** is available in the gcloud CLI overview guide.

## Task 1. Configure your environment and account

1. Sign in to the Google Cloud console with your lab credentials, and open the **Cloud Shell** terminal window.

2. To set your project ID and region environment variables, in Cloud Shell, run the following commands:

   ```
   PROJECT_ID=$(gcloud config get-value project) REGION={{{project_0.default_region|set at lab start}}}
   echo "PROJECT_ID=${PROJECT_ID}" echo "REGION=${REGION}"
   ```

3. To store the signed-in Google user account in an environment variable, run the following command:

   ```
   USER=$(gcloud config get-value account 2> /dev/null) echo "USER=${USER}"
   ```

4. Enable the Cloud AI Companion API for Duet AI:

   ```
   gcloud services enable cloudaicompanion.googleapis.com --project ${PROJECT_ID}
   ```

5. To use Duet AI, grant the necessary IAM roles to your Google Cloud Qwiklabs user account:

   ```
   gcloud projects add-iam-policy-binding ${PROJECT_ID} --member user:${USER} --
   role=roles/cloudaicompanion.user gcloud projects add-iam-policy-binding ${PROJECT_ID} --member
   user:${USER} --role=roles/serviceusage.serviceUsageViewer
   ```

   Adding these roles lets the user use Duet AI assistance.

To verify the objective, click **Check my progress**. Enable relevant APIs, and set IAM roles.

## Task 2. Create a Cloud Workstation

This lab uses Duet AI assistance to develop an app with the Cloud Code plugin for Cloud Workstations IDE. Cloud Workstations is a fully managed integrated development environment that includes native integration with Duet AI.

In this task, you configure and provision your Cloud Workstation environment, and you enable the Cloud Code plugin for Duet AI.

### View the workstation cluster

A workstation cluster named `my-cluster` has been pre-created for this lab. This cluster is used to configure and create a workstation.

1. In the Google Cloud console, select the **Navigation menu** (☰), and then select **More Products > Tools > Cloud Workstations**.

2. In the **Navigation pane**, click **Cluster management**.

3. Check the **Status** of the cluster. If the status of the cluster is `Reconciling` or `Updating`, periodically refresh and wait until it becomes `Ready` before moving to the next step.

## Create a workstation configuration

Before creating a workstation, you must create a workstation configuration in Cloud Workstations.

1. In the **Navigation pane**, click **Workstation configurations**, and then click **Create Workstation Configuration**.

2. Specify the following values:

| **Property** | **Value** |
|---|---|
| Name | **my-config** |
| Workstation cluster | *select* **my-cluster** |

3. Click **Create**.

4. Click **Refresh**.

5. Check the **Status** of the configuration being created. If the status of the configuration is `Reconciling` or `Updating`, periodically refresh and wait until the status becomes `Ready` before moving to the next step.

## Create a workstation

1. In the **Navigation pane**, click **Workstations**, and then click **Create Workstation**.

2. Specify the following values:

| **Property** | **Value** |
|---|---|
| Name | **my-workstation** |
| Configuration | *select* **my-config** |

3. Click **Create**.

   After the workstation is created, it is listed under **My workstations** with a status of `Stopped`.

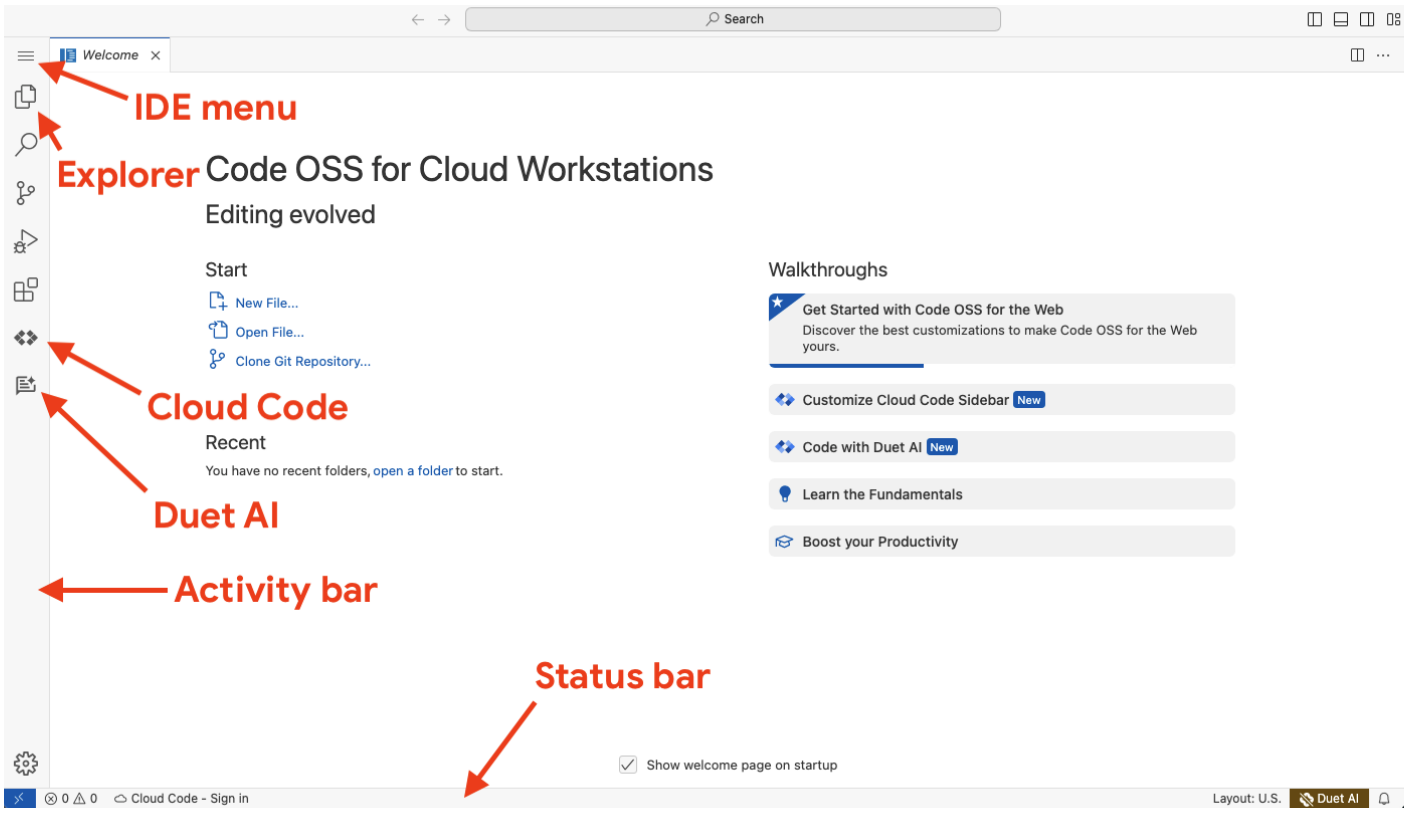
4. To start the workstation, click **Start**.

   As the workstation starts up, the status changes to `Starting`. Wait for the status to change to `Running` which indicates that it is ready to be used. It might take several minutes for the workstation to fully start up.

## Launch the IDE

- To launch the Code OSS IDE on the workstation, from the **Workstations** page in the Google Cloud console, click **Launch**.

The IDE opens in a separate browser tab.



To verify the objective, click **Check my progress**. Create and start a Cloud Workstation.

## Task 3. Update the Cloud Code extension to enable Duet AI

In this task, you enable Duet AI in Cloud Code for your Workstation IDE.

### Connect to Google Cloud

To connect to Google Cloud in your workstation, perform these steps:

1. At the bottom of the window, on the status bar, click **Cloud Code - Sign In**.
2. When you're prompted to sign in, click **Proceed to sign in**.

   A link is displayed in the terminal.

3. To launch the Cloud Cloud sign-in flow, press Control (for Windows and Linux) or Command (for MacOS) and click the link in the terminal.
4. If you are asked to confirm the opening of the external website, click **Open**.
5. Click the student email address.
6. When you're prompted to let the Google Cloud SDK access your Google Account and agree to the terms, click **Allow**.

Your verification code is displayed in the browser tab.

**Note:** You may see a warning that you ran a gcloud auth login command. This process is normal. The IDE ran this command on your behalf.

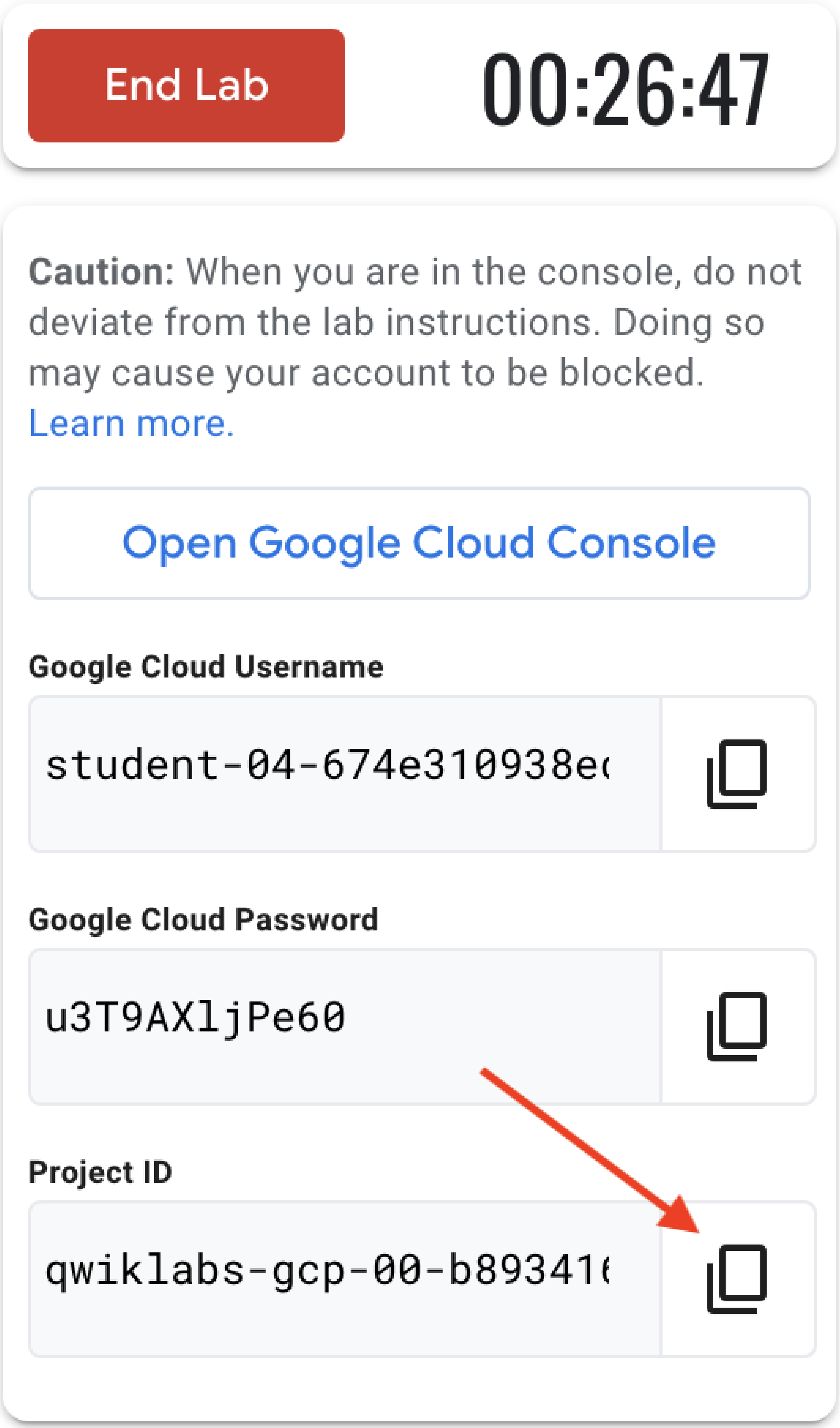
7. Click **Copy**.
8. Back in the IDE, in the terminal, where it says **Enter authorization code**, paste the code.
9. If asked to approve copying from the clipboard, click **Allow**.
10. Click Enter, and then wait for the status bar to show **Cloud Code - No Project**.

    You're now connected to Google Cloud.

## Enable Duet AI in Cloud Code

To enable Duet AI in Cloud Code for your workstation IDE, perform these steps:

1. In your workstation IDE, click the menu (☰), and then navigate to **File > Preferences > Settings**.
2. On the **User** tab of the Settings dialog, select **Extensions > Google Cloud Code**.
3. In **Search settings**, enter `Duet AI`.
4. On the Qwiklabs lab credentials panel, to copy the Project ID, click **Copy**.



00:26:47

**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.

**Google Cloud Username**

```
student-04-674e310938e
```

**Google Cloud Password**

```
u3T9AXljPe60
```

**Project ID**

```
qwiklabs-gcp-00-b89341
```

5. On the Cloud Code settings page, for **Cloudcode > Duet AI: Project**, paste the Google Cloud project ID.

6. Confirm that **Cloud Code > Duet AI: Enable** is enabled.

7. In the IDE status bar, click **Cloud Code - No Project**.

8. Click **Select a Google Cloud Project**, and then click your project ID.

   The project ID is now shown in the status bar. Duet AI is now ready to use.

## Task 4. Chat with Duet AI

Duet AI can help you choose the Google Cloud services that meet the requirements of your application architecture. If you want to develop and test your app in your local IDE, and then deploy it to Google Cloud, you can chat with Duet AI to get help.

In this task, you use the **Duet AI pane** to enter prompts and view the responses from Duet AI.

Prompts are questions or statements that describe the help that you need. Prompts can include context from existing code that Google Cloud analyzes to provide more useful or complete responses. For more information on writing prompts to generate good responses, see Write better prompts for Duet AI.

### Prompt Duet AI

To prompt Duet AI about Google Cloud services, perform these steps:

1. To open the Duet AI chat pane, in the IDE activity bar, click **Duet AI** ().

2. If an error occurs when trying to open the Duet AI chat pane, refresh the browser window.

3. In the **Duet AI** pane, type the following prompt, and then click **Send** ():

   I am new to Google Cloud and I want to use the Cloud Code extension. Give me some examples of Google services that I can use to build and deploy a sample app.

   Duet AI responds with a list of Google Cloud services and descriptions.

   In this example, assume that Duet AI suggests both Cloud Run and Cloud Functions as two Google Cloud services that can help you build and deploy a sample app. You can ask for more information about those services.

4. To provide a follow-up question or prompt, in the **Duet AI** pane type the text below, and then click **Send**:

   What is the difference between Cloud Run and Cloud Functions?

   Duet AI responds with key differences between the two Google Cloud services.

5. To reset your chat history, in the **Duet AI** pane, click **Reset Chat** ().

   **Note:** Chat history state is kept in memory only, and doesn't persist when you switch to another workspace or when you close your IDE. Duet AI doesn't use your prompts or its responses as data to train its model. For more information, see How Duet AI in Google Cloud uses your data.

## Task 5. Develop a Python app

Let's now use Cloud Run to create and deploy a basic Python app. Because you're new to Cloud Run and Cloud Code, you need help with the steps for creating the app.

In this task, you prompt Duet AI for help to build a Hello World Python app in Cloud Run.

### Get help from Duet AI

1. In the **Duet AI** pane, to learn how to create a Cloud Run app with Cloud Code, type the following prompt, and then click **Send** (➤):

   How do I create a new Cloud Run app in Cloud Code using the command palette? What languages are supported?

2. In the response from Duet AI, view the set of steps to create an app. Duet AI also displays the supported languages for the Cloud Run app.

   **Note:** The command palette in VS Code provides a list of all the commands, including the commands for Cloud Code.

### Create a Python app by using the steps from Duet AI

1. Click the menu (≡), and then navigate to **View > Command Palette**.

2. Type `Cloud Code New`, and then select **Cloud Code: New Application**.

3. Select **Cloud Run application**.

4. Select **Python (Flask): Cloud Run**.

5. Update the name of the app and top-level folder to `/home/user/hello-world`, and then click **Ok**.

   Cloud Code downloads the template and creates the application files in the folder in your IDE.

### Explore the app with Duet AI

Now that you've created your `Hello World` app in Cloud Run, you can use Duet AI to explain the files and code snippets that are deployed in your IDE.

1. If the files are not visible, in the IDE activity bar, click **Explorer** (⧉).

2. In the Explorer pane, select **Dockerfile**.

3. Select the entire contents of the Dockerfile, click the bulb (💡), and from the **More Actions** menu, click **Explain this**.

Duet AI generates a natural-language explanation about the contents and function of the `Dockerfile`. You can also select portions of the file contents, click the bulb ( 💡 ), and then click **Explain this**.

4. Select the line that begins with **ENTRYPOINT**, click the bulb ( 💡 ), and then click **Explain this**.

   Duet AI responds with details about the ENTRYPOINT instruction. You learn that, with this instruction, Docker will run the app.py file when the container launches.

5. To view the contents of the `app.py` file, in the activity bar, click **Explorer** (⧉), and then select `app.py`.

6. In the `hello()` function definition, select the lines that contain the `K_SERVICE`, and `K_REVISION` environment variables. Click the bulb ( 💡 ), then click **Explain this**.

   Duet AI responds with a detailed explanation of these two Cloud Run environment variables and how they are used in the application code.

## Run the app locally

You can run your app locally from your IDE by using the Cloud Run emulator. In this case, *locally* means on the workstation machine.

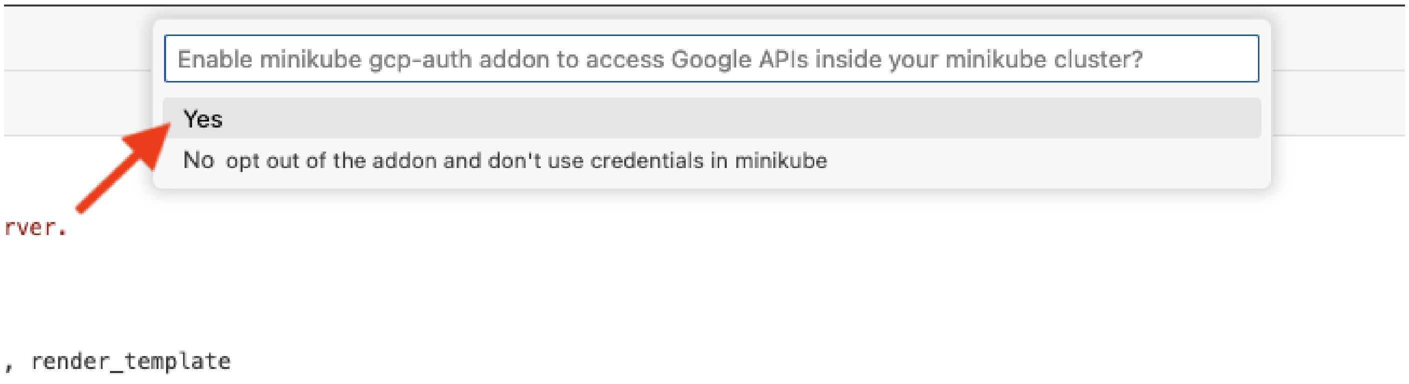
1. In the activity bar of your IDE, click **Cloud Code** (❖), and then click **Cloud Run**.

   **Note:** You will first run the app using the Cloud Run Emulator, so you won't need to enable the Cloud Run API yet.

2. In the Cloud Run activity bar, click **Run App on Local Cloud Run Emulator** ( ▷ ), and then click **Run**.

   The **Output** tab in the IDE displays the progress of the build.

3. When prompted at the top of the screen to **Enable minikube gcp-auth addon to access Google APIs**, select **Yes**.



Enable minikube gcp-auth addon to access Google APIs inside your minikube cluster?

Yes

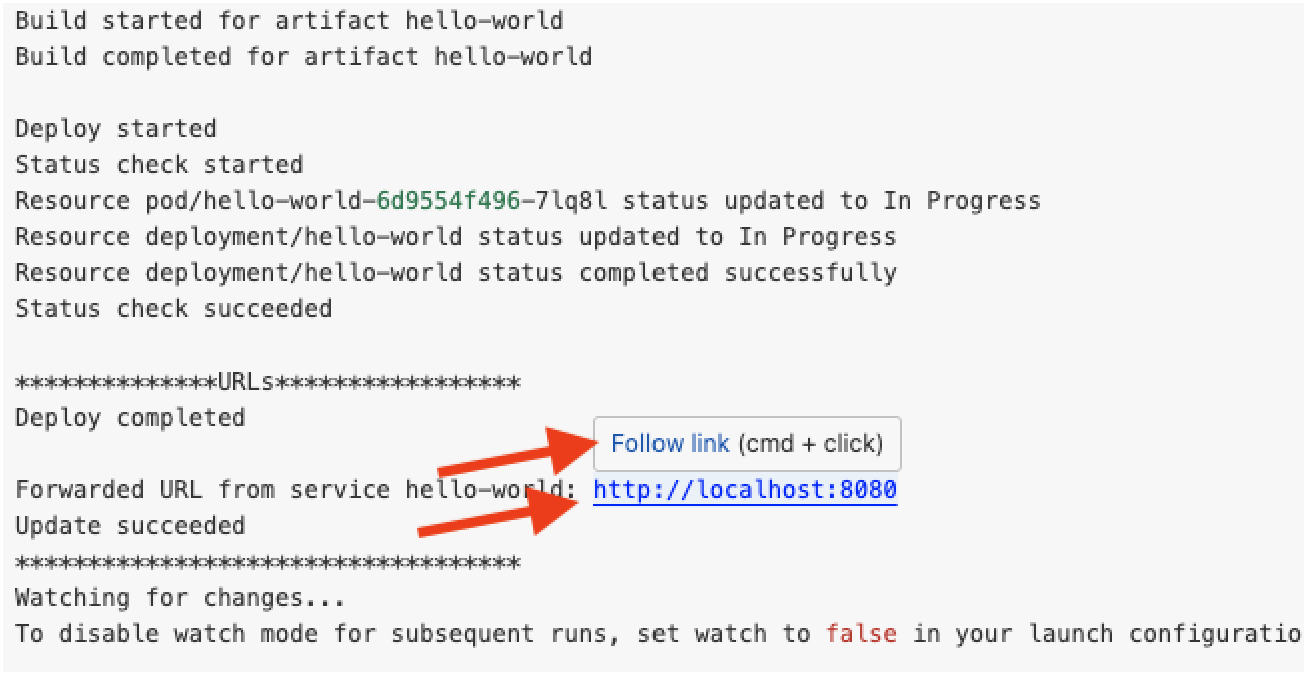
No opt out of the addon and don't use credentials in minikube

```
rver.

, render_template
```

   Wait for the build and deploy to complete.

4. Hold the pointer over the link to the hello-world service at the localhost URL, and click **Follow link**.



```
Build started for artifact hello-world
Build completed for artifact hello-world

Deploy started
Status check started
Resource pod/hello-world-6d9554f496-7lq8l status updated to In Progress
Resource deployment/hello-world status updated to In Progress
Resource deployment/hello-world status completed successfully
Status check succeeded

**************URLs*****************
Deploy completed

Forwarded URL from service hello-world: http://localhost:8080
Update succeeded
***********************************
Watching for changes...
To disable watch mode for subsequent runs, set watch to false in your launch configuratio
```

A new tab is opened in the browser that displays a page indicating that the service is running.

## Task 6. Enhance the Python app

Let's now add data and functionality to the app so that it can be used for management of inventory data.

In this task, you first add inventory data for the app.

### Generate sample data using Duet AI

1. In the activity bar of your IDE, click **Explorer** (⧉).

2. Click **New file** (🗋), and create a file named `inventory.py`.

3. To let Duet AI generate the sample data, open the **Duet AI** pane, type the following prompt, and then click **Send**:

   Create a variable called inventory which is a list of 3 JSON objects. Each JSON object has 2 attributes: productid and onhandqty. Both attributes are strings.

   Duet AI generates the `inventory` JSON array that contains 3 JSON objects.

4. To insert the sample JSON data in the inventory.py file, in the Duet AI response, click **Insert in current file** ( + ). The contents of the file is similar to:

   inventory = [ { "productid": "ABC123", "onhandqty": "10" }, { "productid": "XYZ789", "onhandqty": "20" }, { "productid": "LMN456", "onhandqty": "30" } ]

5. To save the `inventory.py` file in the `home/user/hello-world` folder, in the IDE menu (☰), click **File > Save**.

You use this sample inventory data in the next subtask.

## Add the GET /inventory list API method to the app

You now introduce API methods in the `app.py` file that can operate on the inventory data. To complete this subtask, you use the code generation feature in Duet AI.

1. In the folder and file list in **Explorer**, open the `app.py` file by selecting it.
2. Modify the flask import statement to add the `inventory.py` file and the `jsonify` library:

   from flask import Flask, render_template, jsonify from inventory import inventory

3. In the `app.py` file, position your cursor below the *app* assignment statement:

   app = Flask(__name__)

4. To let **Duet AI** generate the code for the first API method, in the `app.py` file, enter the following comment:

   # Generate an app route to display a list of inventory items in the JSON format from the inventory.py file.
   # Use the GET method.

5. Select the comment lines, including the blank line below the comment.
6. Click the bulb ( 💡 ), and then in the **More Actions** menu, select **Generate code**.

   Duet AI generates a function for the GET operation that returns a list of items from the `inventory.py` file. The function generally looks similar to this:

   @app.route('/inventory', methods=['GET']) def inventory_list(): """Return a list of inventory items in JSON format.""" return jsonify(inventory) **Note:** To learn more about the `jsonify(inventory)` function, highlight the term and prompt Duet AI to explain the code to you.

7. To accept the generated code, hold the pointer over any part of the generate code response, then click **Accept**.

   **Important:** Duet AI can generate more than one code snippet, and these snippets might differ from the snippet that is displayed above.

8. If the **app.route** and **return** statements in your generated code is different from the code shown above, replace the generated code snippet with the snippet displayed above. This should ensure that the lab works as intended.

## Add the GET /inventory/ method to the app

Let's add another API method to return data about a specific inventory item, given its productID. If the productID is not found, the API returns the standard HTTP status code of 404.

1. Add a few blank lines.

2. To let Duet AI generate the code for this second API method, in the `app.py` file, enter the following comment:

   # Generate an App route to get a product from the list of inventory items given the productID. # Use the GET method. # If there is an invalid productID, return a 404 error with an error message in the JSON.

3. Select the 3 comment lines and the blank line that follows the comment, click the bulb ( ), and then in the **More Actions** menu, select **Generate code**.

   Duet AI generates a function for the GET operation that returns the item from the inventory file whose productID is provided in the request, or the 404 status code if the product does not exist.

   @app.route('/inventory/<productid>', methods=['GET']) def inventory_item(productid): """Return a single inventory item in JSON format.""" for item in inventory: if item['productid'] == productid: return jsonify(item) return jsonify({'error': 'Product not found'}), 404

4. Hold the pointer over any part of the generate code response. To accept the generated code, in the toolbar, click **Accept**.

5. If the generated code is different from the code shown above, replace the generated code snippet with the snippet displayed above.

## Rebuild and redeploy the app locally

You can run your app locally from your IDE using the Cloud Run emulator. In this case, *locally* means on the workstation machine.

1. In the activity bar of your IDE, click **Cloud Code** ( ).

2. In the Cloud Run activity bar, click **Run App on Local Cloud Run Emulator** ( ), and then click **Run**.

3. When prompted at the top of the screen to **Enable minikube gcp-auth addon to access Google APIs**, select **Yes**.

   Wait for the build and deploy to complete.

4. Hold the pointer over the link to the hello-world service at the localhost URL, and click **Follow link**.

   A new tab is opened in the browser that displays a page indicating that the service is running.

## Test the API methods

1. Follow the steps in the earlier task to run the app locally.

2. After following the link to view the running app in a separate browser tab, append `/inventory` to the URL in this tab, and type Enter.

   The API returns a JSON response that contains the list of products from the `inventory.py` file.

3. Append `/{PRODUCTID}` to the URL that ends with `/inventory`, where `{PRODUCTID}` is a product ID in your inventory.
4. Type Enter.

   The API returns a JSON response that contains data about the specific product.
5. Replace the product ID with `XXXXX` and type Enter.

   XXXXX is not a valid product ID, so the API returns a JSON error response indicating that the product is not found.

## Task 7. Deploy the app to Cloud Run

You can now deploy the app to Cloud Run on Google Cloud.

1. In the activity bar main menu (☰), click **View > Command Palette**.
2. In the command palette field, type **Cloud Code Deploy**, and then select **Cloud Code: Deploy to Cloud Run** from the list.
3. To enable the Cloud Run API for your project, click **Enable API**.
4. On the **Service Settings** page, for **Region**, select .
5. Leave the remaining settings as their defaults, and then click **Deploy**.

   Cloud Code builds your image, pushes it to the registry, and deploys your service to Cloud Run. This may take a few minutes.
6. To view your running service, open the URL that is displayed in the Deploy to Cloud Run dialog.
7. Test your service by appending the `/inventory`, and `/inventory/{PRODUCTID}` paths to the URL, and verify the response.

   To get the URL for the Cloud Run service inventory page, in Cloud Shell, run the following command:

   export SVC_URL=$(gcloud run services describe hello-world \ --region {{{project_0.default_region|set at lab start}}} \ --platform managed \ --format='value(status.url)') echo ${SVC_URL}/inventory

To verify the objective, click **Check my progress**. Deploy your app to Cloud Run.

## End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

## Congratulations!

In this lab you learned how to:

- Explore various Google services that you can use to deploy an app by asking Duet AI context-based questions.
- Prompt Duet AI to provide templates that you can use to develop a basic app in Cloud Run.
- Create, explore, and modify the app by using Duet AI to explain and generate the code.
- Run and test the app locally, and then deploy it to Google Cloud by using Duet AI to generate the steps.

### Next Steps/Learn More

- Duet AI in Google Cloud overview
- Tutorial: Develop an app with Duet AI assistance