App Dev: Storing Image and Video Files in Cloud Storage - Python

GSP185

Overview

Cloud Storage allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download.

In this lab you'll configure an application to use Cloud Storage to store and retrieve application data. The application is an online Quiz, the data is the form data, including an image you upload from your local machine.

Objectives

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

- Set up Cloud Shell as your development environment
- Update the application code to integrate Cloud Datastore
- Use the Quiz application to upload an image file into Cloud Storage and view the image in the Quiz

Qwiklabs setup

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

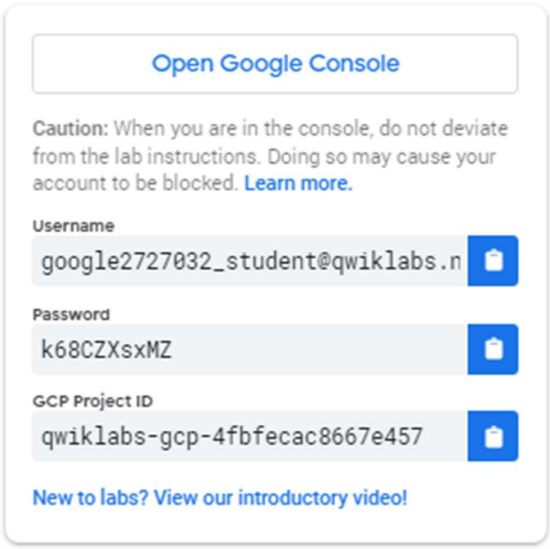
To complete this lab, you need:

- 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.

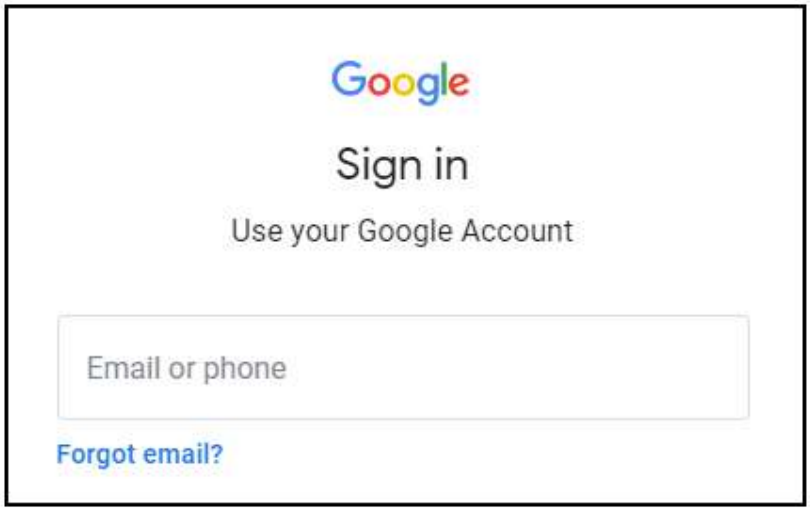
Note: If you are using a Pixelbook, open an Incognito window to run this lab.

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.

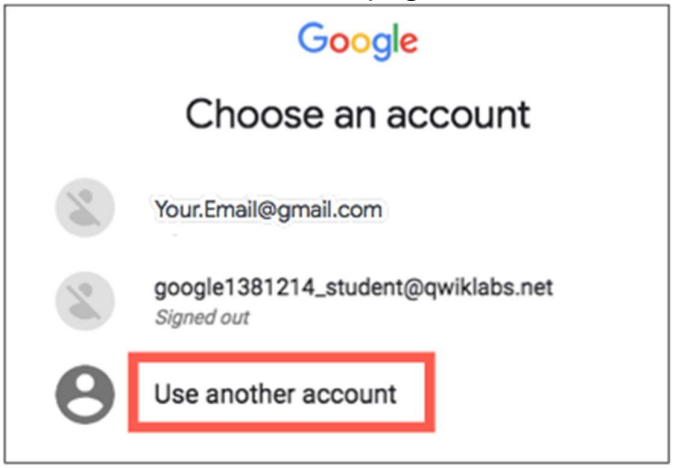


2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



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

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



3. In the **Sign in** page, paste the username that you copied from the 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).

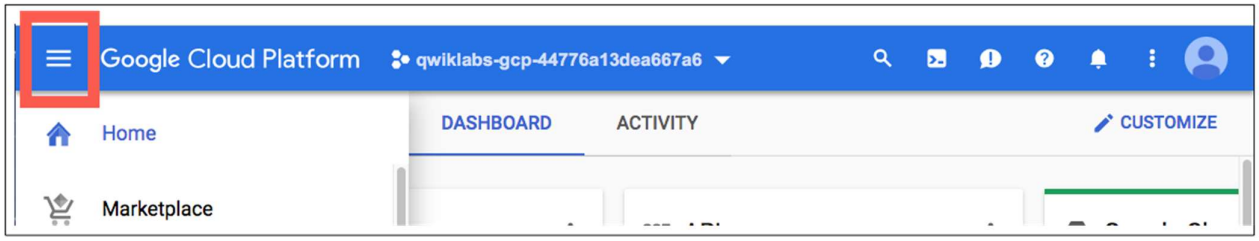
4. Click through the subsequent pages:

   - 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-

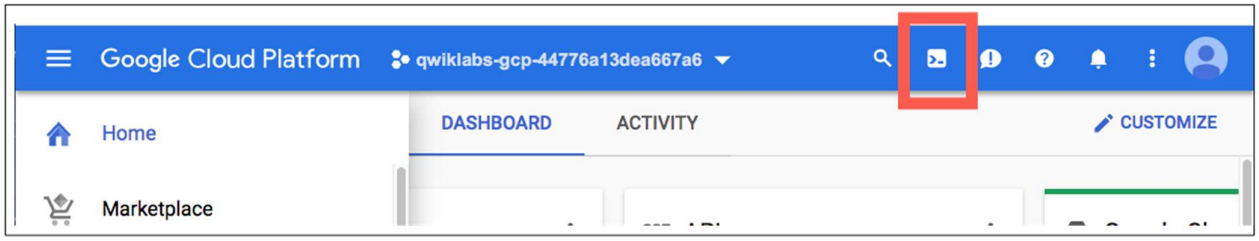
left.



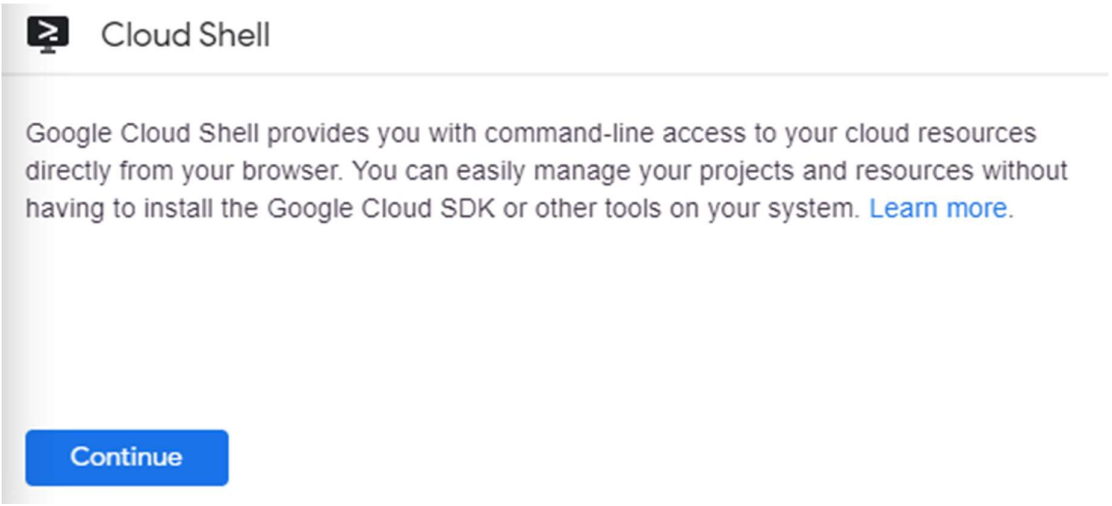
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

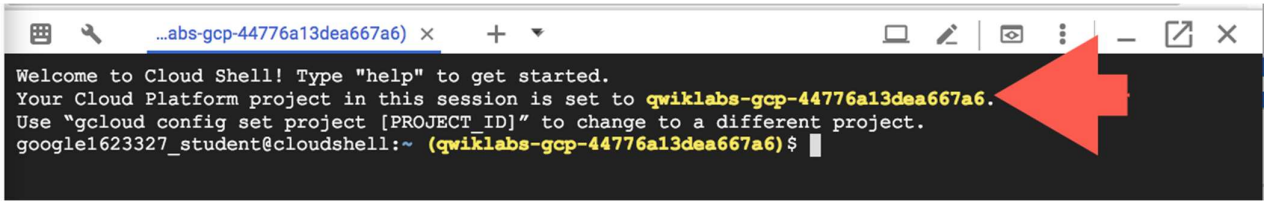
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

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

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

(Example output)

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the gcloud command-line tool overview.

Launch the Cloud Shell code editor

From Cloud Shell, click **Launch the code editor** icon (looks like a pencil) to launch the code editor.

The code editor launches in a separate tab of your browser, along with Cloud Shell.

Run the following command to configure your Project ID, replacing `YOUR-PROJECT-ID` with your Project ID:

```
gcloud config set project <YOUR-PROJECT-ID>
```

Prepare the Quiz application

In this section, you access Cloud Shell, clone the git repository containing the Quiz application, and run the application.

Clone source code in Cloud Shell

To clone the repository for the class, execute the following command:

```
git clone https://github.com/GoogleCloudPlatform/training-data-analyst
```

Configure and run the Quiz application

1. Change the working directory:

   ```
   cd ~/training-data-analyst/courses/developingapps/python/cloudstorage/start
   ```
2. Configure the application:

   ```
   . prepare_environment.sh
   ```

   Ignore any warnings.

   This script file:

   - Creates an App Engine application.
   - Exports an environment variable, `GCLOUD_PROJECT`.
   - Updates pip, then runs `pip install -r requirements.txt`.
   - Creates entities in Cloud Datastore.
   - Prints out the Project ID.
3. Run the application:

   ```
   python run_server.py
   ```

   The application is running when you see the following output:

   ```
   * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
   * Restarting with stat
   * Debugger is active!
   * Debugger PIN: 502-577-323
   ```

Review the Quiz application

1. To view the application, click **Web preview** > **Preview on port 8080**.

2. Click the **Create Question** link in the toolbar.

You should see a simple form that contains textboxes for the question and answers and radio buttons to select the correct answer

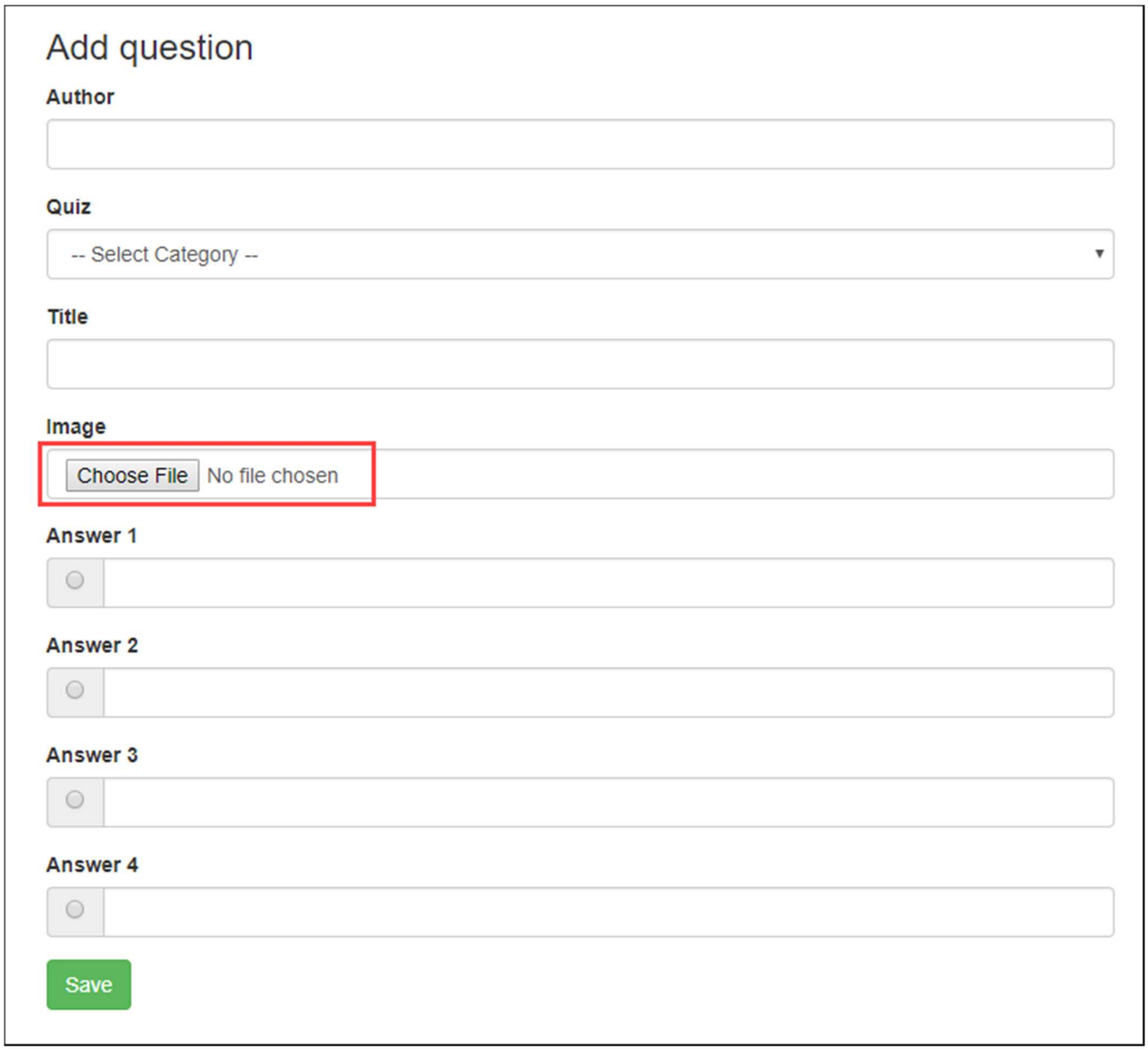
The form has a new file upload field that will be used to upload either image or video files. In this lab you upload an image file; you use the same process to upload video file.

Examine the Quiz application code

In this section, you review the case study application code.

In this lab you'll view and edit files. You can use the shell editors that are installed on Cloud Shell, such as `nano` or `vim` or the Cloud Shell code editor.

This lab uses the Cloud Shell code editor to review the Quiz application code.



Examine the application code

1. Navigate to the `/training-data-analyst/courses/developingapps/python/cloudstorage/start` folder using the file browser panel on the left side of the editor.

2. Select the `add.html file` in the `...quiz/webapp/templates/` folder.

   This file contains the template for the Create Question form.

   Notice how the form has been modified to use `multipart/form-data` as the `enc-type`, and there are two new form controls:

   - A file upload control called `image`
   - A hidden field called `imageUrl`

3. Select the `routes.py` file in the `...quiz/webapp` folder.

   This file contains the route for the POST handler that receives the form data. It has been modified to get the image file from the form.

4. Select the `questions.py` file in the `...quiz/webapp` folder.

   This file contains the handler that processes the form data extracted in the `routes.py` file. You will modify this file to use a new module that is a client for Cloud Storage.

5. Select the `...quiz/gcp/storage.py` file.

   This is the file where you will write code to save image file data into Cloud Storage.

Create a Cloud Storage Bucket

In this section, you create a Cloud Storage bucket and export an environment variable that references it.

1. Return to the Cloud Shell command line. Stop the application by pressing **Ctrl+c**.

2. Create a Cloud Storage bucket named `<Project ID>-media`:

   ```
   gsutil mb gs://$DEVSHELL_PROJECT_ID-media
   ```

   You can create a bucket using the gsutil mb command, passing through the name of the bucket as gs://BUCKET_NAME

   You can use $DEVSHELL_PROJECT_ID as the bucket name prefix followed by -media

3. To export the Cloud Storage bucket name as an environment variable named `GCLOUD_BUCKET`, execute the following command:

```
export GCLOUD_BUCKET=$DEVSHELL_PROJECT_ID-media
```

Recall that the application makes use of environment variables for configuration. This allows the development team to deploy the application into development, test, staging, and production just by changing these variables.

Adding objects to Cloud Storage

In this section, you write code to save uploaded files into Cloud Storage.

Important: Update code within the sections marked as follows: # TODO # END TODO To maximize your learning, review the code, inline comments, and related API documentation.
See APIs & Reference for information on API documentation for Cloud Storage.

Import and use the Python Cloud Storage module

1. In code editor, move to the top of the `...quiz/gcp/storage.py` file.
2. Get the bucket name from the `GCLOUD_BUCKET` environment variable.
3. Import the storage module from the `google.client` package.

4. Create a Cloud Storage client.
5. Get a reference to the Cloud Storage bucket.

quiz/gcp/storage.py

```
# TODO: Get the Bucket name from the
# GCLOUD_BUCKET environment variable

bucket_name = os.getenv('GCLOUD_BUCKET')

# END TODO

# TODO: Import the storage module

from google.cloud import storage

# END TODO

# TODO: Create a client for Cloud Storage

storage_client = storage.Client()

# END TODO

# TODO: Use the client to get the Cloud Storage bucket

bucket = storage_client.get_bucket(bucket_name)

# END TODO
```

Write code to send a file to Cloud Storage

1. Still in `storage.py`, in the the `upload_file(...)` function, remove the existing pass statement, then use the Cloud Storage client to upload a file to your Cloud Storage bucket and make it publicly available.
2. Get a reference to a Cloud Storage blob object in the bucket.
3. Use the blob object to upload the image.
4. Make the file public.
5. Return the blob's public URL.

quiz/gcp/storage.py - `upload)file(...)` function

```
"""
Uploads a file to a given Cloud Storage bucket and returns the public url
to the new object.
"""
def upload_file(image_file, public):

    # TODO: Use the bucket to get a blob object

    blob = bucket.blob(image_file.filename)

    # END TODO

    # TODO: Use the blob to upload the file

    blob.upload_from_string(
        image_file.read(),
        content_type=image_file.content_type)

    # END TODO

    # TODO: Make the object public

    if public:
        blob.make_public()

    # END TODO


    # TODO: Modify to return the blob's Public URL

    return blob.public_url

    # END TODO
```

Save `storage.py`.

Write code to use the Cloud Storage functionality

1. In the editor, move to the top of the `...quiz/webapp/questions.py` file.
2. Modify the import statement to use your storage client as well as the datastore client.
3. Move to the `upload_file(...)` function. Use your storage client to upload a file, and assign the returned public URL to a variable.
4. Modify the return statement to return the public URL.
5. Move to the `save_question(...)` function. Write an if test to see if the `image_file` is present.
6. If it is, then call the `upload_file(...)` function, and assign the public URL to a entity property named imageUrl.
7. If not, then assign an empty string to the entity imageUrl property.

quiz/webapp/questions.py

```
# TODO: Import the storage module

from quiz.gcp import storage, datastore

# END TODO

"""
uploads file into google cloud storage
- upload file
- return public_url
"""
def upload_file(image_file, public):
    if not image_file:
        return None

    # TODO: Use the storage client to Upload the file
    # The second argument is a boolean

    public_url = storage.upload_file(
       image_file,
       public
    )

    # END TODO

    # TODO: Return the public URL
    # for the object

    return public_url

    # END TODO

"""
uploads file into google cloud storage
- call method to upload file (public=true)
- call datastore helper method to save question
"""
def save_question(data, image_file):

    # TODO: If there is an image file, then upload it
    # And assign the result to a new Datastore
    # property imageUrl
    # If there isn't, assign an empty string

    if image_file:
        data['imageUrl'] = str(
                  upload_file(image_file, True))
    else:
        data['imageUrl'] = u''

    # END TODO

    data['correctAnswer'] = int(data['correctAnswer'])
    datastore.save_question(data)
    return
```

Save `questions.py`.

Run the application and create a Cloud Storage object

1. Save the `...gcp/storage.py` and `...webapp/questions.py` files, and then return to the Cloud Shell command.
2. Return to Cloud Shell to run the application:

```
python run_server.py
```

3. Download an image file to your local machine from here.
4. In Cloud Shell, click **Web preview** > **Preview on port 8080** to preview the Quiz application.

5. Click the **Create Question** link.

6. Complete the form with the following values, and then click **Save**.

Form Field	Value
Author	Your name
Quiz	`Google Cloud`
Title	`Which product does this logo relate to?`
Image	`Upload the Google_Cloud_Storage_logo.png file you previously downloaded`
Answer 1	`App Engine`
Answer 2	`Cloud Storage` (Select the Answer 2 radio button)
Answer 3	`Compute Engine`
Answer 4	`Container Engine`

7. Return to the Cloud Console and navigate to **Navigation menu** > **Storage**.

8. On the **Storage** > **Browser** page, click the correct bucket (named `<Project ID>-media`).

You should see your new object named `Google_Cloud_Storage_logo.png`.

Run the client application and test the Cloud Storage public URL

1. Add `/api/quizzes/gcp` to the end of the application's URL.

   You should see that JSON data has been returned to the client corresponding to the Question you added in the web application.

   The imageUrl property should have a value corresponding to the object in Cloud Storage.

2. Return to the application home page and click the **Take Test** link. Click **GCP**, and answer each question.

   When you get to the question you just added, you should see the image has been formatted inside the client-side web application!

Congratulations!

This concluded the lab, App Dev: Storing Image and Video Files in Cloud Storage - Python. You used Cloud Storage to store and retrieve application data.


Finish your Quest

This self-paced lab is part of the Application Development - Python and Cloud Development 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 (or badges) 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

See what else you can do with images and video:

- Extract, Analyze, and Translate Text from Images with the Cloud ML APIs.
- Video Intelligence: Qwik Start

Learn more about Python on the Google Cloud
Manual last updated May 11, 2020
Lab last tested October 15, 2018