## A Cloud Run function in Python can download a file and then upload it to Cloud Storage by utilizing the google-cloud-storage library.

1. Setup:

* requirements.txt: Include google-cloud-storage and Flask (or functions-framework if using Cloud Functions style).

Code

google-cloud-storage  
 Flask

* main.py: Your function code.

2. Python Code Example:

Python

import os  
from flask import Flask, request  
from google.cloud import storage  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/', methods=['GET', 'POST'])  
def handle\_request():  
 *# Get parameters from the request (e.g., source\_url, destination\_bucket, destination\_blob\_name)*  
 *# For simplicity, let's assume these are hardcoded or passed as query parameters/JSON body*  
 source\_url = request.args.get('source\_url', 'http://example.com/some\_file.txt')  
 destination\_bucket\_name = os.environ.get('DESTINATION\_BUCKET', 'your-destination-bucket-name')  
 destination\_blob\_name = request.args.get('destination\_blob\_name', 'downloaded\_file.txt')  
  
 storage\_client = storage.Client()  
 bucket = storage\_client.bucket(destination\_bucket\_name)  
 blob = bucket.blob(destination\_blob\_name)  
  
 try:  
 *# 1. Download the file*  
 *# For a simple HTTP download, use requests library (add 'requests' to requirements.txt)*  
 import requests  
 response = requests.get(source\_url, stream=True)  
 response.raise\_for\_status() *# Raise an exception for bad status codes*  
  
 *# 2. Upload to Cloud Storage*  
 *# Upload directly from the stream to avoid saving to local disk (which is ephemeral in Cloud Run)*  
 blob.upload\_from\_file(response.raw, content\_type=response.headers['Content-Type'])  
  
 return f"File '{source\_url}' downloaded and uploaded to gs://{destination\_bucket\_name}/{destination\_blob\_name}", 200  
  
 except Exception as e:  
 return f"Error processing file: {e}", 500  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True, host='0.0.0.0', port=int(os.environ.get('PORT', 8080)))

Explanation:

* **Dependencies**:

The google-cloud-storage library is used for interacting with Cloud Storage, and Flask is used to create a simple web server for the Cloud Run service. requests is used for downloading the external file.

* **Download**:

The requests.get(source\_url, stream=True) initiates a stream-based download. response.raw provides a file-like object directly from the HTTP response, which is efficient for large files as it avoids loading the entire file into memory.

* **Upload**:

blob.upload\_from\_file(response.raw, ...) directly uploads the content from the downloaded stream to the specified Cloud Storage blob. This avoids writing the file to the ephemeral /tmp directory on the Cloud Run instance, which can be inefficient and potentially exceed memory limits for large files.

* **Error Handling**:

Basic try-except block is included for error management during the process.

* **Deployment**:

Deploy this code as a Cloud Run service, ensuring the service account has the necessary permissions to write to the target Cloud Storage bucket. Environment variables can be used to configure bucket names or other parameters.

## Environment variables for Cloud Run functions can be set during deployment using the Google Cloud console or the gcloud command-line tool.

Using the Google Cloud Console:

* Navigate to the Cloud Run service in the Google Cloud console.
* Select the desired service or job.
* Click "Edit and deploy new revision."
* Go to the "Container" tab.
* In the "Variables and secrets" section, add or modify environment variables by specifying their Name and Value.
* Click "Done" and then "Deploy" to apply the changes.

Using the gcloud Command-Line Tool:

Environment variables can be specified during deployment using the gcloud run deploy command. To set all environment variables.

Code

gcloud run deploy SERVICE\_NAME --image IMAGE\_URL --set-env-vars KEY1=VALUE1,KEY2=VALUE2

* **To update or add specific environment variables without affecting others:**

Code

gcloud run deploy SERVICE\_NAME --image IMAGE\_URL --update-env-vars KEY\_TO\_UPDATE=NEW\_VALUE,NEW\_KEY=NEW\_VALUE

Accessing Environment Variables in Code:

Within the Cloud Run function code, environment variables are typically accessed through the operating system's environment variable mechanism.

* **Node.js:** process.env.YOUR\_VARIABLE\_NAME
* **Python:** os.environ.get("YOUR\_VARIABLE\_NAME")

Note: For local development, tools like env-cmd or dotenv can be used to manage environment variables from a .env file, allowing for consistent configuration between local and deployed environments.