create a storage account using python script and upload the conetnet to bucket and access it using the url - index .html - in gcp

**ChatGPT said:**

To **create a GCP Storage Bucket**, **upload content (like index.html)**, and **access it using the public URL**, you can use Python and the google-cloud-storage library.

**✅ Prerequisites:**

1. **Install the GCP Storage library:**

bash

CopyEdit

pip install google-cloud-storage

1. **Authenticate your application:**
   * Set up a service account in GCP.
   * Download the JSON key.
   * Set the environment variable:

bash

CopyEdit

export GOOGLE\_APPLICATION\_CREDENTIALS="path/to/your/service-account.json"

**✅ Python Script: Create Bucket, Upload File, Make Public**

python

CopyEdit

from google.cloud import storage

def create\_bucket(bucket\_name):

"""Creates a new GCS bucket"""

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

new\_bucket = storage\_client.create\_bucket(bucket, location="us")

print(f"Bucket {bucket.name} created.")

return new\_bucket

def upload\_file(bucket\_name, source\_file\_name, destination\_blob\_name):

"""Uploads a file to the bucket"""

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_name)

print(f"File {source\_file\_name} uploaded to {destination\_blob\_name}.")

def make\_blob\_public(bucket\_name, blob\_name):

"""Makes a blob publicly accessible"""

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(blob\_name)

blob.make\_public()

print(f"Blob {blob\_name} is publicly accessible at {blob.public\_url}")

return blob.public\_url

# === MAIN ===

bucket\_name = "your-unique-bucket-name-123" # must be globally unique

source\_file = "index.html"

destination\_blob = "index.html"

# Create the bucket

create\_bucket(bucket\_name)

# Upload the file

upload\_file(bucket\_name, source\_file, destination\_blob)

# Make it public and print the URL

url = make\_blob\_public(bucket\_name, destination\_blob)

print("Access your file at:", url)

**📂 index.html Sample (for testing)**

html

CopyEdit

<!DOCTYPE html>

<html>

<head>

<title>Welcome</title>

</head>

<body>

<h1>Hello from GCP Bucket!</h1>

</body>

</html>

**🌍 Public Access URL**

Once uploaded and made public, your file will be accessible at:

pgsql

CopyEdit

https://storage.googleapis.com/[your-bucket-name]/index.html

If you'd like, I can also help you **configure the bucket to serve as a static website**. Want to do that too?

4o

**You said:**

write contnet for this file only pleas

**ChatGPT said:**

Sure! Here's a simple and elegant HTML content for your index.html file that you can use to test the GCP bucket static hosting:

html

CopyEdit

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>GCP Bucket Hosting</title>

<style>

body {

font-family: 'Segoe UI', sans-serif;

background-color: #f2f2f2;

color: #333;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

text-align: center;

}

.container {

background: white;

padding: 2rem;

border-radius: 12px;

box-shadow: 0 0 20px rgba(0,0,0,0.1);

}

h1 {

color: #4285f4;

}

</style>

</head>

<body>

<div class="container">

<h1>🚀 Welcome to GCP Bucket Hosting</h1>

<p>Your <strong>index.html</strong> file is live and publicly accessible!</p>

<p>Hosted on Google Cloud Storage 📦</p>

</div>

</body>

</html>

This page will look clean and modern when accessed via the GCP public URL. Let me know if you want to add a logo, image, or additional content!

4o

**You said:**

no add this isnide script

**ChatGPT said:**

Got it! Here's your **complete Python script**, including the creation of a GCP bucket, writing the index.html content dynamically, uploading it to the bucket, and making it publicly accessible:

**✅ Full Python Script with Embedded index.html Content**

python

CopyEdit

from google.cloud import storage

import os

# Step 1: Write index.html content to a file

html\_content = """<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>GCP Bucket Hosting</title>

<style>

body {

font-family: 'Segoe UI', sans-serif;

background-color: #f2f2f2;

color: #333;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

text-align: center;

}

.container {

background: white;

padding: 2rem;

border-radius: 12px;

box-shadow: 0 0 20px rgba(0,0,0,0.1);

}

h1 {

color: #4285f4;

}

</style>

</head>

<body>

<div class="container">

<h1>🚀 Welcome to GCP Bucket Hosting</h1>

<p>Your <strong>index.html</strong> file is live and publicly accessible!</p>

<p>Hosted on Google Cloud Storage 📦</p>

</div>

</body>

</html>

"""

with open("index.html", "w") as f:

f.write(html\_content)

# Step 2: GCP bucket and upload configuration

bucket\_name = "your-unique-bucket-name-12345" # change this to a globally unique name

source\_file = "index.html"

destination\_blob = "index.html"

def create\_bucket(bucket\_name):

client = storage.Client()

bucket = client.bucket(bucket\_name)

bucket.storage\_class = "STANDARD"

new\_bucket = client.create\_bucket(bucket, location="us")

print(f"✅ Bucket created: {new\_bucket.name}")

return new\_bucket

def upload\_file(bucket\_name, source\_file\_name, destination\_blob\_name):

client = storage.Client()

bucket = client.bucket(bucket\_name)

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_name)

print(f"✅ File uploaded: {source\_file\_name} ➜ {destination\_blob\_name}")

def make\_blob\_public(bucket\_name, blob\_name):

client = storage.Client()

bucket = client.bucket(bucket\_name)

blob = bucket.blob(blob\_name)

blob.make\_public()

print(f"🌐 Public URL: {blob.public\_url}")

return blob.public\_url

# Step 3: Execute steps

create\_bucket(bucket\_name)

upload\_file(bucket\_name, source\_file, destination\_blob)

public\_url = make\_blob\_public(bucket\_name, destination\_blob)

print("\n🎉 You can now access your page at:", public\_url)

**💡 Final Output**

This script will:

* Create index.html with modern content.
* Upload it to GCP bucket.
* Make it publicly accessible.
* Print the public URL to open in your browser.

Let me know if you also want to configure the bucket to serve as a **static website** (i.e. default object like index.html on root path).

4o

**You said:**

from google.cloud import storage import os # Step 1: Write index.html content to a file html\_content = """<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1"> <title>GCP Bucket Hosting</title> <style> body { font-family: 'Segoe UI', sans-serif; background-color: #f2f2f2; color: #333; display: flex; justify-content: center; align-items: center; height: 100vh; margin: 0; text-align: center; } .container { background: white; padding: 2rem; border-radius: 12px; box-shadow: 0 0 20px rgba(0,0,0,0.1); } h1 { color: #4285f4; } </style> </head> <body> <div class="container"> <h1>🚀 Welcome to GCP Bucket Hosting</h1> <p>Your <strong>index.html</strong> file is live and publicly accessible!</p> <p>Hosted on Google Cloud Storage 📦</p> </div> </body> </html> """ with open("index.html", "w") as f: f.write(html\_content) # Step 2: GCP bucket and upload configuration bucket\_name = "your-unique-bucket-name-12345" # change this to a globally unique name source\_file = "index.html" destination\_blob = "index.html" def create\_bucket(bucket\_name): client = storage.Client() bucket = client.bucket(bucket\_name) bucket.storage\_class = "STANDARD" new\_bucket = client.create\_bucket(bucket, location="us") print(f"✅ Bucket created: {new\_bucket.name}") return new\_bucket def upload\_file(bucket\_name, source\_file\_name, destination\_blob\_name): client = storage.Client() bucket = client.bucket(bucket\_name) blob = bucket.blob(destination\_blob\_name) blob.upload\_from\_filename(source\_file\_name) print(f"✅ File uploaded: {source\_file\_name} ➜ {destination\_blob\_name}") def make\_blob\_public(bucket\_name, blob\_name): client = storage.Client() bucket = client.bucket(bucket\_name) blob = bucket.blob(blob\_name) blob.make\_public() print(f"🌐 Public URL: {blob.public\_url}") return blob.public\_url # Step 3: Execute steps create\_bucket(bucket\_name) upload\_file(bucket\_name, source\_file, destination\_blob) public\_url = make\_blob\_public(bucket\_name, destination\_blob) print("\n🎉 You can now access your page at:", public\_url) user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python storage.py Traceback (most recent call last): File "D:\agcp\storage.py", line 1, in <module> from google.cloud import storage ImportError: cannot import name 'storage' from 'google.cloud' (unknown location) user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $

**ChatGPT said:**

The error you're seeing:

pgsql

CopyEdit

ImportError: cannot import name 'storage' from 'google.cloud' (unknown location)

means that Python is unable to properly import the google.cloud.storage module. This usually happens due to one of the following:

**✅ Fix 1: Install the Required Library**

First, make sure you have installed the **Google Cloud Storage** library:

bash

CopyEdit

pip install google-cloud-storage

To confirm it's installed and available:

bash

CopyEdit

pip show google-cloud-storage

**✅ Fix 2: Avoid File Naming Conflicts**

If your script is named google.py or there's a file/folder named google in the same directory, it will shadow the actual Google Cloud package.

Check this:

bash

CopyEdit

ls

If you see a file or folder named google.py, google/, or cloud/ — **rename or delete** it.

For example, if you have:

text

CopyEdit

google/

run:

bash

CopyEdit

rm -r google

Or if you named your script google.py, rename it:

bash

CopyEdit

mv google.py storage\_script.py

**✅ Fix 3: Use the Correct Python Interpreter**

Sometimes, multiple Python installations can cause the wrong one to run. Run:

bash

CopyEdit

python -m pip install google-cloud-storage

Or if you're using Python 3 explicitly:

bash

CopyEdit

python3 -m pip install google-cloud-storage

Then try running your script again:

bash

CopyEdit

python storage.py

Let me know what ls or dir outputs in your working directory, and I can help you further if something's shadowing the import!

4o

**You said:**

from google.cloud import storage def create\_bucket(bucket\_name): """Creates a new GCS bucket""" storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) new\_bucket = storage\_client.create\_bucket(bucket, location="us") print(f"Bucket {bucket.name} created.") return new\_bucket def upload\_file(bucket\_name, source\_file\_name, destination\_blob\_name): """Uploads a file to the bucket""" storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) blob = bucket.blob(destination\_blob\_name) blob.upload\_from\_filename(source\_file\_name) print(f"File {source\_file\_name} uploaded to {destination\_blob\_name}.") def make\_blob\_public(bucket\_name, blob\_name): """Makes a blob publicly accessible""" storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) blob = bucket.blob(blob\_name) blob.make\_public() print(f"Blob {blob\_name} is publicly accessible at {blob.public\_url}") return blob.public\_url # === MAIN === bucket\_name = "your-unique-bucket-name-123" # must be globally unique source\_file = "index.html" destination\_blob = "index.html" # Create the bucket create\_bucket(bucket\_name) # Upload the file upload\_file(bucket\_name, source\_file, destination\_blob) # Make it public and print the URL url = make\_blob\_public(bucket\_name, destination\_blob) print("Access your file at:", url) user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python storage.py Traceback (most recent call last): File "D:\agcp\storage.py", line 36, in <module> create\_bucket(bucket\_name) ~~~~~~~~~~~~~^^^^^^^^^^^^^ File "D:\agcp\storage.py", line 5, in create\_bucket storage\_client = storage.Client() File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\client.py", line 247, in \_\_init\_\_ super(Client, self).\_\_init\_\_( ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^ project=project, ^^^^^^^^^^^^^^^^ ...<2 lines>... \_http=\_http, ^^^^^^^^^^^^ ) ^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 338, in \_\_init\_\_ \_ClientProjectMixin.\_\_init\_\_(self, project=project, credentials=credentials) ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 286, in \_\_init\_\_ project = self.\_determine\_default(project) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 305, in \_determine\_default return \_determine\_default\_project(project) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\\_helpers\\_\_init\_\_.py", line 152, in \_determine\_default\_project \_, project = google.auth.default() ~~~~~~~~~~~~~~~~~~~^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\auth\\_default.py", line 685, in default raise exceptions.DefaultCredentialsError(\_CLOUD\_SDK\_MISSING\_CREDENTIALS) google.auth.exceptions.DefaultCredentialsError: Your default credentials were not found. To set up Application Default Credentials, see https://cloud.google.com/docs/authentication/external/se t-up-adc for more information. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

**ChatGPT said:**

The error you're seeing:

pgsql

CopyEdit

google.auth.exceptions.DefaultCredentialsError: Your default credentials were not found.

means that Google Cloud SDK (gcloud) doesn't know **which credentials to use** to authenticate with GCP.

**✅ How to Fix It (Set Up Authentication)**

To resolve this, you need to **authenticate your local machine** with Google Cloud using **Application Default Credentials (ADC)**.

**Step 1: Install gcloud CLI (if not already)**

If you haven’t installed the Google Cloud SDK, download it here and follow the installation instructions.

**Step 2: Log in with gcloud**

Run this command in your terminal:

bash

CopyEdit

gcloud auth application-default login

* This will open a browser window to let you log into your Google account.
* After successful login, it will generate the required credentials (adc.json) at:

makefile

CopyEdit

C:\Users\<YourUsername>\AppData\Roaming\gcloud\application\_default\_credentials.json

**Step 3: Set your GCP project (optional but recommended)**

Replace <YOUR\_PROJECT\_ID> with your actual project ID:

bash

CopyEdit

gcloud config set project <YOUR\_PROJECT\_ID>

**Step 4: Run your Python script again**

Once the credentials are set, try again:

bash

CopyEdit

python storage.py

Let me know if you want to automate this process inside your script (not recommended for production, but useful for testing/dev environments).

4o

**You said:**

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ gcloud init Welcome! This command will take you through the configuration of gcloud. Settings from your current configuration [default] are: accessibility: screen\_reader: 'True' compute: region: us-central1 zone: us-central1-b core: account: dk1078451@gmail.com disable\_usage\_reporting: 'True' project: manifest-zephyr-457115-v2 Pick configuration to use: [1] Re-initialize this configuration [default] with new settings [2] Create a new configuration Please enter your numeric choice: 1 Your current configuration has been set to: [default] You can skip diagnostics next time by using the following flag: gcloud init --skip-diagnostics Network diagnostic detects and fixes local network connection issues. Checking network connection...done. Reachability Check passed. Network diagnostic passed (1/1 checks passed). Choose the account you want to use for this configuration. To use a federated user account, exit this command and sign in to the gcloud CLI with your login configuration file, then run this command again. Select an account: [1] dk1078451@gmail.com [2] Sign in with a new Google Account [3] Skip this step Please enter your numeric choice: 1 You are signed in as: [dk1078451@gmail.com]. Pick cloud project to use: [1] manifest-zephyr-457115-v2 [2] skilled-chalice-435007-t5 [3] temporal-clover-441306-k4 [4] Enter a project ID [5] Create a new project Please enter numeric choice or text value (must exactly match list item): 1 Your current project has been set to: [manifest-zephyr-457115-v2]. Do you want to configure a default Compute Region and Zone? (Y/n)? y Which Google Compute Engine zone would you like to use as project default? If you do not specify a zone via a command line flag while working with Compute Engine resources, the default is assumed. [1] us-east1-b [2] us-east1-c [3] us-east1-d [4] us-east4-c [5] us-east4-b [6] us-east4-a [7] us-central1-c [8] us-central1-a [9] us-central1-f [10] us-central1-b [11] us-west1-b [12] us-west1-c [13] us-west1-a [14] europe-west4-a [15] europe-west4-b [16] europe-west4-c [17] europe-west1-b [18] europe-west1-d [19] europe-west1-c [20] europe-west3-c [21] europe-west3-a [22] europe-west3-b [23] europe-west2-c [24] europe-west2-b [25] europe-west2-a [26] asia-east1-b [27] asia-east1-a [28] asia-east1-c [29] asia-southeast1-b [30] asia-southeast1-a [31] asia-southeast1-c [32] asia-northeast1-b [33] asia-northeast1-c [34] asia-northeast1-a [35] asia-south1-c [36] asia-south1-b [37] asia-south1-a [38] australia-southeast1-b [39] australia-southeast1-c [40] australia-southeast1-a [41] southamerica-east1-b [42] southamerica-east1-c [43] southamerica-east1-a [44] africa-south1-a [45] africa-south1-b [46] africa-south1-c [47] asia-east2-a [48] asia-east2-b [49] asia-east2-c [50] asia-northeast2-a Did not print [78] options. Too many options [128]. Enter "list" at prompt to print choices fully. Please enter numeric choice or text value (must exactly match list item): 8 Your project default Compute Engine zone has been set to [us-central1-a]. You can change it by running [gcloud config set compute/zone NAME]. Your project default Compute Engine region has been set to [us-central1]. You can change it by running [gcloud config set compute/region NAME]. Error creating a default .boto configuration file. Please run [gsutil config -n] if you would like to create this file. The Google Cloud CLI is configured and ready to use! \* Commands that require authentication will use dk1078451@gmail.com by default \* Commands will reference project manifest-zephyr-457115-v2 by default \* Compute Engine commands will use region us-central1 by default \* Compute Engine commands will use zone us-central1-a by default Run gcloud help config to learn how to change individual settings This gcloud configuration is called [default]. You can create additional configurations if you work with multiple accounts and/or projects. Run gcloud topic configurations to learn more. Some things to try next: \* Run gcloud --help to see the Cloud Platform services you can interact with. And run gcloud help COMMAND to get help on any gcloud command. \* Run gcloud topic --help to learn about advanced features of the CLI like arg files and output formatting \* Run gcloud cheat-sheet to see a roster of go-to gcloud commands. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python storage.py Traceback (most recent call last): File "D:\agcp\storage.py", line 36, in <module> create\_bucket(bucket\_name) ~~~~~~~~~~~~~^^^^^^^^^^^^^ File "D:\agcp\storage.py", line 5, in create\_bucket storage\_client = storage.Client() File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\client.py", line 247, in \_\_init\_\_ super(Client, self).\_\_init\_\_( ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^ project=project, ^^^^^^^^^^^^^^^^ ...<2 lines>... \_http=\_http, ^^^^^^^^^^^^ ) ^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 338, in \_\_init\_\_ \_ClientProjectMixin.\_\_init\_\_(self, project=project, credentials=credentials) ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 286, in \_\_init\_\_ project = self.\_determine\_default(project) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 305, in \_determine\_default return \_determine\_default\_project(project) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\\_helpers\\_\_init\_\_.py", line 152, in \_determine\_default\_project \_, project = google.auth.default() ~~~~~~~~~~~~~~~~~~~^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\auth\\_default.py", line 685, in default raise exceptions.DefaultCredentialsError(\_CLOUD\_SDK\_MISSING\_CREDENTIALS) google.auth.exceptions.DefaultCredentialsError: Your default credentials were not found. To set up Application Default Credentials, see https://cloud.google.com/docs/authentication/external/se t-up-adc for more information. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $

**ChatGPT said:**

You're almost there, Deepanshu! The error you're seeing:

pgsql

CopyEdit

google.auth.exceptions.DefaultCredentialsError: Your default credentials were not found.

means that your **Application Default Credentials (ADC)** are not yet set up — even though you're authenticated with the gcloud CLI, Python libraries like google-cloud-storage use ADC, which requires an explicit setup step.

**✅ To fix it, run this command:**

bash

CopyEdit

gcloud auth application-default login

This will open a browser window to authenticate and will save the credentials that Python client libraries like google-cloud-storage rely on.

**After this:**

You can re-run your script:

bash

CopyEdit

python storage.py

and it should work fine, using the credentials and project you've already configured with gcloud init.

Let me know if you want help verifying your storage.py script too, or if you're trying to do something specific like **create a bucket, upload a file, or list objects** — I can help with that!

4o

**You said:**

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ gcloud auth application-default login Your browser has been opened to visit: https://accounts.google.com/o/oauth2/auth?response\_type=code&client\_id=764086051850-6qr4p6gpi6hn506pt8ejuq83di341hur.apps.googleusercontent.com&redirect\_uri=http%3A%2F%2Flocalhost%3A8085%2 F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login&stat e=Orhn6eTAKFlmnfqcJ5ID0EBcYo0XXz&access\_type=offline&code\_challenge=\_\_FNF9T1IS\_a-uoo3vM7ovM1Zn9kT0jbUeTVNunLC0g&code\_challenge\_method=S256 Credentials saved to file: [C:\Users\user121\AppData\Roaming\gcloud\application\_default\_credentials.json] These credentials will be used by any library that requests Application Default Credentials (ADC). Quota project "manifest-zephyr-457115-v2" was added to ADC which can be used by Google client libraries for billing and quota. Note that some services may still bill the project owning the res ource. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python storage.py Traceback (most recent call last): File "D:\agcp\storage.py", line 36, in <module> create\_bucket(bucket\_name) ~~~~~~~~~~~~~^^^^^^^^^^^^^ File "D:\agcp\storage.py", line 7, in create\_bucket new\_bucket = storage\_client.create\_bucket(bucket, location="us") File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\client.py", line 1118, in create\_bucket api\_response = self.\_post\_resource( "/b", ...<4 lines>... \_target\_object=bucket, ) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\client.py", line 744, in \_post\_resource return self.\_connection.api\_request( ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^ method="POST", ^^^^^^^^^^^^^^ ...<6 lines>... \_target\_object=\_target\_object, ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ) ^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\\_http.py", line 90, in api\_request return call() File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\api\_core\retry\retry\_unary.py", line 293, in retry\_wrapped\_func return retry\_target( target, ...<3 lines>... on\_error=on\_error, ) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\api\_core\retry\retry\_unary.py", line 153, in retry\_target \_retry\_error\_helper( ~~~~~~~~~~~~~~~~~~~^ exc, ^^^^ ...<6 lines>... timeout, ^^^^^^^^ ) ^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\api\_core\retry\retry\_base.py", line 212, in \_retry\_error\_helper raise final\_exc from source\_exc File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\api\_core\retry\retry\_unary.py", line 144, in retry\_target result = target() File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\\_http\\_\_init\_\_.py", line 494, in api\_request raise exceptions.from\_http\_response(response) google.api\_core.exceptions.Conflict: 409 POST https://storage.googleapis.com/storage/v1/b?project=manifest-zephyr-457115-v2&prettyPrint=false: The requested bucket name is not available. The b ucket namespace is shared by all users of the system. Please select a different name and try again. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $

**ChatGPT said:**

You're almost there! The error message you're seeing:

vbnet

CopyEdit

google.api\_core.exceptions.Conflict: 409 ... The requested bucket name is not available.

The bucket namespace is shared by all users of the system. Please select a different name and try again.

**✅ Problem:**

The bucket name you're trying to create **already exists globally**. Google Cloud Storage bucket names are **unique across the entire platform**, not just within your project.

**✅ Solution:**

Just choose a globally unique bucket name.

You can do something like:

python

CopyEdit

import uuid

bucket\_name = f"my-unique-bucket-{uuid.uuid4().hex[:6]}"

Or manually update the bucket\_name variable in your script to something like:

python

CopyEdit

bucket\_name = "deepanshu-storage-bucket-457115-v2"

Make sure it:

* Contains only lowercase letters, numbers, dashes (-)
* Starts and ends with a letter or number
* Is between 3 and 63 characters

**🔁 What to do now:**

1. **Update the bucket name** in your Python script.
2. Run the script again:

bash

CopyEdit

python storage.py

Let me know if you want me to review your storage.py script to ensure it's all good.

4o

**You said:**

create a private buucekt not public bucket please only private bucket -using python script please and access it please with url

**ChatGPT said:**

To create a **private bucket** using Python and then access it **securely via a signed URL**, follow this guide step-by-step.

**✅ 1. Install required libraries (if not already installed)**

bash

CopyEdit

pip install google-cloud-storage

**✅ 2. Python script: Create a private bucket and access a file using a signed URL**

python

CopyEdit

from google.cloud import storage

from datetime import timedelta

import uuid

def create\_private\_bucket(bucket\_name):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

# Create the bucket in US region

new\_bucket = storage\_client.create\_bucket(bucket, location="US")

# Ensure bucket is private by disabling public access

new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True

new\_bucket.patch()

print(f"Private bucket {bucket.name} created.")

return new\_bucket

def upload\_file(bucket, source\_file\_path, destination\_blob\_name):

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_path)

print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.")

def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(blob\_name)

url = blob.generate\_signed\_url(

version="v4",

expiration=timedelta(minutes=expiration\_minutes),

method="GET"

)

print(f"Signed URL (valid for {expiration\_minutes} mins): {url}")

return url

if \_\_name\_\_ == "\_\_main\_\_":

# UNIQUE bucket name

bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}"

file\_path = "demo.txt" # Local file path

blob\_name = "demo.txt" # Name to store in GCS

# 1. Create private bucket

bucket = create\_private\_bucket(bucket\_name)

# 2. Upload file to bucket

upload\_file(bucket, file\_path, blob\_name)

# 3. Generate secure signed URL

generate\_signed\_url(bucket.name, blob\_name)

**✅ 3. How it works**

* The bucket is made **private** by enabling **uniform bucket-level access**.
* A file (demo.txt) is uploaded.
* A **signed URL** is generated, allowing **temporary access** without making the bucket public.

**📎 Example output:**

arduino

CopyEdit

Private bucket private-bucket-a1b2c3 created.

File demo.txt uploaded as demo.txt.

Signed URL (valid for 15 mins): https://storage.googleapis.com/private-bucket-a1b2c3/demo.txt?X-Goog-Algorithm=GOOG4-RSA-SHA256&...

Let me know if you want to:

* Upload folders instead of a file
* Use service account keys
* Set specific IAM policies (like allowing only your user or app to access)

I'll tailor the script for you!

4o

**You said:**

write <h1>hello from deepanshu </h1> in index.html in python script include it

**ChatGPT said:**

Got it! Here's an updated Python script that:

1. **Creates a private bucket**
2. **Creates an index.html file with <h1>Hello from Deepanshu</h1>**
3. **Uploads the file to the bucket**
4. **Generates a signed URL to access index.html**

**✅ Full Python Script (upload\_index.py)**

python

CopyEdit

from google.cloud import storage

from datetime import timedelta

import uuid

def create\_html\_file():

html\_content = "<h1>Hello from Deepanshu</h1>"

file\_path = "index.html"

with open(file\_path, "w") as f:

f.write(html\_content)

print(f"Created local file: {file\_path}")

return file\_path

def create\_private\_bucket(bucket\_name):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

new\_bucket = storage\_client.create\_bucket(bucket, location="US")

new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True

new\_bucket.patch()

print(f"Private bucket {bucket.name} created.")

return new\_bucket

def upload\_file(bucket, source\_file\_path, destination\_blob\_name):

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_path)

print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.")

def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(blob\_name)

url = blob.generate\_signed\_url(

version="v4",

expiration=timedelta(minutes=expiration\_minutes),

method="GET"

)

print(f"\nAccess it securely using this signed URL (valid for {expiration\_minutes} minutes):\n{url}")

return url

if \_\_name\_\_ == "\_\_main\_\_":

# Generate unique bucket name

bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}"

blob\_name = "index.html"

# 1. Create HTML file

file\_path = create\_html\_file()

# 2. Create private GCS bucket

bucket = create\_private\_bucket(bucket\_name)

# 3. Upload HTML file

upload\_file(bucket, file\_path, blob\_name)

# 4. Generate signed URL to access index.html

generate\_signed\_url(bucket.name, blob\_name)

**📁 Output**

html

CopyEdit

<h1>Hello from Deepanshu</h1>

**🔗 Example Signed URL Output**

cpp

CopyEdit

Access it securely using this signed URL (valid for 15 minutes):

https://storage.googleapis.com/private-bucket-ab12cd/index.html?X-Goog-Algorithm=...

Let me know if you'd like to serve a full website or auto-open the URL in your browser too!

4o

**You said:**

from google.cloud import storage from datetime import timedelta import uuid def create\_html\_file(): html\_content = "<h1>Hello from Deepanshu</h1>" file\_path = "index.html" with open(file\_path, "w") as f: f.write(html\_content) print(f"Created local file: {file\_path}") return file\_path def create\_private\_bucket(bucket\_name): storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) new\_bucket = storage\_client.create\_bucket(bucket, location="US") new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True new\_bucket.patch() print(f"Private bucket {bucket.name} created.") return new\_bucket def upload\_file(bucket, source\_file\_path, destination\_blob\_name): blob = bucket.blob(destination\_blob\_name) blob.upload\_from\_filename(source\_file\_path) print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.") def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15): storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) blob = bucket.blob(blob\_name) url = blob.generate\_signed\_url( version="v4", expiration=timedelta(minutes=expiration\_minutes), method="GET" ) print(f"\nAccess it securely using this signed URL (valid for {expiration\_minutes} minutes):\n{url}") return url if \_\_name\_\_ == "\_\_main\_\_": # Generate unique bucket name bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}" blob\_name = "index.html" # 1. Create HTML file file\_path = create\_html\_file() # 2. Create private GCS bucket bucket = create\_private\_bucket(bucket\_name) # 3. Upload HTML file upload\_file(bucket, file\_path, blob\_name) # 4. Generate signed URL to access index.html generate\_signed\_url(bucket.name, blob\_name) - user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python private\_bucket.py Created local file: index.html Private bucket private-bucket-1c0496 created. File index.html uploaded as index.html. Traceback (most recent call last): File "D:\agcp\private\_bucket.py", line 58, in <module> generate\_signed\_url(bucket.name, blob\_name) ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^ File "D:\agcp\private\_bucket.py", line 34, in generate\_signed\_url url = blob.generate\_signed\_url( version="v4", expiration=timedelta(minutes=expiration\_minutes), method="GET" ) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\blob.py", line 660, in generate\_signed\_url return helper( credentials, ...<13 lines>... universe\_domain=universe\_domain, ) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\\_signing.py", line 551, in generate\_signed\_url\_v4 ensure\_signed\_credentials(credentials) ~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\\_signing.py", line 58, in ensure\_signed\_credentials raise AttributeError( ...<4 lines>... ) AttributeError: you need a private key to sign credentials.the credentials you are currently using <class 'google.oauth2.credentials.Credentials'> just contains a token. see https://googleapis .dev/python/google-api-core/latest/auth.html#setting-up-a-service-account for more details.

**ChatGPT said:**

Thanks for sharing the error — it’s a **common one** when trying to generate signed URLs using **user credentials (like gcloud login)** instead of a **service account with a private key**.

**❌ Problem**

You're using **user credentials (google.oauth2.credentials.Credentials)**, which **don’t have a private key**, and therefore **cannot sign URLs**.

**✅ Solution: Use a Service Account Key File**

Here’s what to do:

**🔧 1. Create a Service Account with Access to Storage**

1. Go to: Google Cloud Console - IAM & Admin > Service Accounts
2. Click **Create Service Account**
3. Name it like signed-url-service
4. Assign role: Storage Admin
5. Click **Done**
6. Click the new service account → **Keys** tab → **Add Key → JSON**
7. Save the JSON file as service-account-key.json (put it in your script directory)

**🛠️ 2. Modify Your Script to Use This Service Account**

Add this line at the top **before any storage.Client() is called**:

python

CopyEdit

import os

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "service-account-key.json"

**✅ Updated Full Working Script**

python

CopyEdit

import os

from google.cloud import storage

from datetime import timedelta

import uuid

# Use service account key

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "service-account-key.json"

def create\_html\_file():

html\_content = "<h1>Hello from Deepanshu</h1>"

file\_path = "index.html"

with open(file\_path, "w") as f:

f.write(html\_content)

print(f"Created local file: {file\_path}")

return file\_path

def create\_private\_bucket(bucket\_name):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

new\_bucket = storage\_client.create\_bucket(bucket, location="US")

new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True

new\_bucket.patch()

print(f"Private bucket {bucket.name} created.")

return new\_bucket

def upload\_file(bucket, source\_file\_path, destination\_blob\_name):

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_path)

print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.")

def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(blob\_name)

url = blob.generate\_signed\_url(

version="v4",

expiration=timedelta(minutes=expiration\_minutes),

method="GET"

)

print(f"\nAccess it securely using this signed URL (valid for {expiration\_minutes} minutes):\n{url}")

return url

if \_\_name\_\_ == "\_\_main\_\_":

bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}"

blob\_name = "index.html"

file\_path = create\_html\_file()

bucket = create\_private\_bucket(bucket\_name)

upload\_file(bucket, file\_path, blob\_name)

generate\_signed\_url(bucket.name, blob\_name)

**✅ Final Result**

Run this again after you set your service account key:

bash

CopyEdit

python private\_bucket.py

You'll see the signed URL printed, and it’ll work in the browser even for a private bucket 🎯

Let me know if you want to add CORS or make this part of a full CI/CD flow.

4o

**You said:**

1. Create a Service Account with Access to Storage Go to: Google Cloud Console - IAM & Admin > Service Accounts Click Create Service Account Name it like signed-url-service Assign role: Storage Admin Click Done Click the new service account → Keys tab → Add Key → JSON Save the JSON file as service-account-key.json (put it in your script directory) - is it compute storage admin

**ChatGPT said:**

No, the correct role to assign to the service account is **Storage Admin**, not **Compute Storage Admin**.

Here’s the clarification:

* **Storage Admin**: This role allows the service account to manage **Cloud Storage resources**, including creating buckets, uploading objects, generating signed URLs, and managing access to Cloud Storage.
* **Compute Storage Admin**: This role would be for managing **Compute Engine** instances and managing storage resources for compute-related tasks.

So, please use the **Storage Admin** role for managing Google Cloud Storage.

Let me know if you need further help!

4o mini

**You said:**

deepanshu-storage.json in download folder downloaded

**ChatGPT said:**

Great! Now that you have the deepanshu-storage.json service account key file, follow these steps to use it in your Python script to authenticate and generate signed URLs:

**Steps:**

1. **Move the JSON file** (if it's not already in the script directory): Move the deepanshu-storage.json file from your Downloads folder to the directory where your Python script is located, or provide the full path to it.
2. **Set the environment variable** to authenticate using the service account: Before running your script, set the GOOGLE\_APPLICATION\_CREDENTIALS environment variable to the path of your deepanshu-storage.json file. This will allow your script to authenticate using the service account.

On **Windows** (in your terminal):

bash

CopyEdit

set GOOGLE\_APPLICATION\_CREDENTIALS=D:\path\to\deepanshu-storage.json

Replace D:\path\to\deepanshu-storage.json with the actual path to your file.

If you are using a **bash terminal (like Git Bash or WSL)**, use the export command:

bash

CopyEdit

export GOOGLE\_APPLICATION\_CREDENTIALS="/path/to/deepanshu-storage.json"

1. **Modify your script** (if not already set up) to authenticate using this service account: Your script should automatically use the service account if you've set up the environment variable correctly, as the Google Cloud SDK and libraries look for this variable to authenticate.
2. **Run your Python script**: Now you can run your Python script, and it should authenticate properly using the deepanshu-storage.json service account key.

If you encounter any issues during these steps, feel free to ask!

4o mini

**You said:**

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ export GOOGLE\_APPLICATION\_CREDENTIALS="/path/to/deepanshu-storage.json" = its' in downlaod folder

**ChatGPT said:**

Since your deepanshu-storage.json file is in the Downloads folder, you need to set the correct path for it.

Here's how you can do it on **Windows**:

1. **Find the path to the Downloads folder**:
   * The typical path for Downloads on Windows is:

pgsql

CopyEdit

C:\Users\<YourUsername>\Downloads\deepanshu-storage.json

1. **Set the environment variable in Git Bash**: Use the export command with the full path to your service account JSON file. For example:

bash

CopyEdit

export GOOGLE\_APPLICATION\_CREDENTIALS="/c/Users/user121/Downloads/deepanshu-storage.json"

In this case, replace user121 with your actual Windows username if it's different.

1. **Run your Python script**: Now that you've set the environment variable, your Python script should be able to authenticate with the Google Cloud API using the service account.

Let me know if this works!

4o mini

**You said:**

file:///C:/Users/user121/Downloads/ - inside this

**ChatGPT said:**

To set the GOOGLE\_APPLICATION\_CREDENTIALS environment variable for your service account key file located in the Downloads folder, follow these steps:

1. **Locate the correct path**: Your deepanshu-storage.json file is located in C:\Users\user121\Downloads\.
2. **Set the environment variable**:

In Git Bash, run the following command:

bash

CopyEdit

export GOOGLE\_APPLICATION\_CREDENTIALS="/c/Users/user121/Downloads/deepanshu-storage.json"

This uses the Unix-style path format, where C:\ is converted to /c/, and the rest of the path follows the same pattern.

1. **Verify that the environment variable is set**:

You can check if the environment variable is set correctly by running:

bash

CopyEdit

echo $GOOGLE\_APPLICATION\_CREDENTIALS

This should output the path to your JSON key file.

1. **Run your script**:

Now, you should be able to run your Python script, and it will authenticate using the service account specified in deepanshu-storage.json.

Let me know how it goes!

4o mini

**You said:**

import os from google.cloud import storage from datetime import timedelta import uuid # Use service account key os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "deepanshu-storage.json" def create\_html\_file(): html\_content = "<h1>Hello from Deepanshu</h1>" file\_path = "index.html" with open(file\_path, "w") as f: f.write(html\_content) print(f"Created local file: {file\_path}") return file\_path def create\_private\_bucket(bucket\_name): storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) new\_bucket = storage\_client.create\_bucket(bucket, location="US") new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True new\_bucket.patch() print(f"Private bucket {bucket.name} created.") return new\_bucket def upload\_file(bucket, source\_file\_path, destination\_blob\_name): blob = bucket.blob(destination\_blob\_name) blob.upload\_from\_filename(source\_file\_path) print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.") def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15): storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) blob = bucket.blob(blob\_name) url = blob.generate\_signed\_url( version="v4", expiration=timedelta(minutes=expiration\_minutes), method="GET" ) print(f"\nAccess it securely using this signed URL (valid for {expiration\_minutes} minutes):\n{url}") return url if \_\_name\_\_ == "\_\_main\_\_": bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}" blob\_name = "index.html" file\_path = create\_html\_file() bucket = create\_private\_bucket(bucket\_name) upload\_file(bucket, file\_path, blob\_name) generate\_signed\_url(bucket.name, blob\_name) - user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python private\_bucket.py Created local file: index.html Traceback (most recent call last): File "D:\agcp\private\_bucket.py", line 52, in <module> bucket = create\_private\_bucket(bucket\_name) File "D:\agcp\private\_bucket.py", line 18, in create\_private\_bucket storage\_client = storage.Client() File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\storage\client.py", line 247, in \_\_init\_\_ super(Client, self).\_\_init\_\_( ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^ project=project, ^^^^^^^^^^^^^^^^ ...<2 lines>... \_http=\_http, ^^^^^^^^^^^^ ) ^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 338, in \_\_init\_\_ \_ClientProjectMixin.\_\_init\_\_(self, project=project, credentials=credentials) ~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 286, in \_\_init\_\_ project = self.\_determine\_default(project) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\client\\_\_init\_\_.py", line 305, in \_determine\_default return \_determine\_default\_project(project) File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\cloud\\_helpers\\_\_init\_\_.py", line 152, in \_determine\_default\_project \_, project = google.auth.default() ~~~~~~~~~~~~~~~~~~~^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\auth\\_default.py", line 651, in default credentials, project\_id = checker() ~~~~~~~^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\auth\\_default.py", line 644, in <lambda> lambda: \_get\_explicit\_environ\_credentials(quota\_project\_id=quota\_project\_id), ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\auth\\_default.py", line 293, in \_get\_explicit\_environ\_credentials credentials, project\_id = load\_credentials\_from\_file( ~~~~~~~~~~~~~~~~~~~~~~~~~~^ os.environ[environment\_vars.CREDENTIALS], quota\_project\_id=quota\_project\_id ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ) ^ File "C:\Users\user121\AppData\Local\Programs\Python\Python313\Lib\site-packages\google\auth\\_default.py", line 125, in load\_credentials\_from\_file raise exceptions.DefaultCredentialsError( "File {} was not found.".format(filename) ) google.auth.exceptions.DefaultCredentialsError: File deepanshu-storage.json was not found. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $

**ChatGPT said:**

It looks like the script cannot find the deepanshu-storage.json file because the path is not set correctly or the file is not located in the directory where the script is being executed.

Here’s what you can do:

1. **Verify the file path**: Make sure the file deepanshu-storage.json is in the correct directory and that the path is correct. If the file is in the Downloads folder, use the full path.
2. **Update the environment variable for correct path**: Instead of setting the environment variable using a relative path, use the full path to the file in the script like this:

python

CopyEdit

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "C:/Users/user121/Downloads/deepanshu-storage.json"

1. **Alternatively, set the path via the terminal**: If you are running the script from the terminal, you can export the environment variable directly like this:

In Git Bash:

bash

CopyEdit

export GOOGLE\_APPLICATION\_CREDENTIALS="/c/Users/user121/Downloads/deepanshu-storage.json"

1. **Run the script again**: Once the environment variable is set correctly, run your Python script again.

Let me know if you face any issues!

4o mini

**You said:**

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ python private\_bucket.py Created local file: index.html Private bucket private-bucket-44de2e created. File index.html uploaded as index.html. Access it securely using this signed URL (valid for 15 minutes): https://storage.googleapis.com/private-bucket-44de2e/index.html?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=signer-url-service%40manifest-zephyr-457115-v2.iam.gserviceaccount.com%2F202 50420%2Fauto%2Fstorage%2Fgoog4\_request&X-Goog-Date=20250420T054632Z&X-Goog-Expires=900&X-Goog-SignedHeaders=host&X-Goog-Signature=25b93e62ab1417f8eb16af2495048a46e213fa5207e03e1c405080face4ec9 7b29b44aa9c04401dbe2c555831037d368ae6202a5507b71a3546d4f628c6c0e3c5f2c75690ba2de53f9bc10cde14b042713cdbaecb3e22b2c66d989006fcbb1d23ebd7a94c99ed200961b785dec0db71400e770af837d0ded01e37717262110 95d6d2b40753df99f18c81f852302a6eb83842a03f4f48fb1c721b74df0b2115c880f0d2913c166d6d90070a2af06882bf9f987c06565c9c8631e9eee3c71b06a15f67e5c6a1074a79706641f155f36cef6b30dffd8c9f62785b8891df6ec099 3d1be04ac5dd1d9999aec592309bebc7a27a86f19fc8da20a43019dd3d5bb03551

**ChatGPT said:**

It looks like your script worked successfully! You have created a private bucket, uploaded the HTML file (index.html), and generated a signed URL that grants temporary access to the file.

You can access the HTML file using the generated signed URL within the next 15 minutes. Here's your signed URL:

Access the HTML file

This link will allow you to securely access your index.html file for the next 15 minutes.

If you need further help or modifications, feel free to ask!

4o mini

**You said:**

#working import os from google.cloud import storage from datetime import timedelta import uuid # Use service account key # os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "deepanshu-storage.json" os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "C:/Users/user121/Downloads/deepanshu-storage.json" def create\_html\_file(): html\_content = "<h1>Hello from Deepanshu</h1>" file\_path = "index.html" with open(file\_path, "w") as f: f.write(html\_content) print(f"Created local file: {file\_path}") return file\_path def create\_private\_bucket(bucket\_name): storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) new\_bucket = storage\_client.create\_bucket(bucket, location="US") new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True new\_bucket.patch() print(f"Private bucket {bucket.name} created.") return new\_bucket def upload\_file(bucket, source\_file\_path, destination\_blob\_name): blob = bucket.blob(destination\_blob\_name) blob.upload\_from\_filename(source\_file\_path) print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.") def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15): storage\_client = storage.Client() bucket = storage\_client.bucket(bucket\_name) blob = bucket.blob(blob\_name) url = blob.generate\_signed\_url( version="v4", expiration=timedelta(minutes=expiration\_minutes), method="GET" ) print(f"\nAccess it securely using this signed URL (valid for {expiration\_minutes} minutes):\n{url}") return url if \_\_name\_\_ == "\_\_main\_\_": bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}" blob\_name = "index.html" file\_path = create\_html\_file() bucket = create\_private\_bucket(bucket\_name) upload\_file(bucket, file\_path, blob\_name) generate\_signed\_url(bucket.name, blob\_name) - explain the script line by line p

**ChatGPT said:**

Here's a breakdown of your script, explaining it line by line:

**1. Import Required Libraries**

python

CopyEdit

import os

from google.cloud import storage

from datetime import timedelta

import uuid

* **os**: This module provides a way to interact with the operating system, such as setting environment variables.
* **google.cloud.storage**: This is the Google Cloud Storage client library. It allows you to interact with Google Cloud's storage service programmatically.
* **timedelta**: This is used for specifying durations. It’s used here to define the expiration time of the signed URL.
* **uuid**: This module is used to generate unique identifiers. It's used to create a unique name for the bucket.

**2. Setting Google Cloud Service Account Key**

python

CopyEdit

# os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "deepanshu-storage.json"

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "C:/Users/user121/Downloads/deepanshu-storage.json"

* This line sets the environment variable GOOGLE\_APPLICATION\_CREDENTIALS to the path of your Google Cloud service account JSON key file. The environment variable tells the Google Cloud client library which credentials to use when making requests to Google Cloud services (in this case, Cloud Storage).

**3. Function to Create an HTML File Locally**

python

CopyEdit

def create\_html\_file():

html\_content = "<h1>Hello from Deepanshu</h1>"

file\_path = "index.html"

with open(file\_path, "w") as f:

f.write(html\_content)

print(f"Created local file: {file\_path}")

return file\_path

* **create\_html\_file()**: This function creates a simple HTML file called index.html with the content <h1>Hello from Deepanshu</h1>.
* **open(file\_path, "w")**: Opens the file in write mode. If the file doesn’t exist, it is created.
* **f.write(html\_content)**: Writes the HTML content into the file.
* The function then prints a confirmation message and returns the file path (index.html).

**4. Function to Create a Private Bucket**

python

CopyEdit

def create\_private\_bucket(bucket\_name):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

new\_bucket = storage\_client.create\_bucket(bucket, location="US")

new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True

new\_bucket.patch()

print(f"Private bucket {bucket.name} created.")

return new\_bucket

* **create\_private\_bucket(bucket\_name)**: This function creates a new private bucket in Google Cloud Storage.
* **storage.Client()**: Initializes a client object that allows interaction with Google Cloud Storage.
* **storage\_client.bucket(bucket\_name)**: Creates a bucket object in your storage client based on the specified bucket name.
* **storage\_client.create\_bucket(bucket, location="US")**: This line creates the bucket in the US region. The create\_bucket() method actually creates the storage bucket.
* **new\_bucket.iam\_configuration.uniform\_bucket\_level\_access\_enabled = True**: This enables Uniform Bucket-Level Access, which means only IAM roles will control access to the bucket, and ACLs won’t be used.
* **new\_bucket.patch()**: Applies the changes to the bucket.
* **return new\_bucket**: The function returns the created bucket object.

**5. Function to Upload a File to the Bucket**

python

CopyEdit

def upload\_file(bucket, source\_file\_path, destination\_blob\_name):

blob = bucket.blob(destination\_blob\_name)

blob.upload\_from\_filename(source\_file\_path)

print(f"File {source\_file\_path} uploaded as {destination\_blob\_name}.")

* **upload\_file(bucket, source\_file\_path, destination\_blob\_name)**: This function uploads a file to the specified bucket.
* **bucket.blob(destination\_blob\_name)**: Creates a Blob object which represents a file stored in the bucket. The destination\_blob\_name is the name the file will have in the bucket.
* **blob.upload\_from\_filename(source\_file\_path)**: Uploads the local file specified by source\_file\_path to Google Cloud Storage.
* The function then prints a confirmation message that the file has been uploaded.

**6. Function to Generate a Signed URL**

python

CopyEdit

def generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15):

storage\_client = storage.Client()

bucket = storage\_client.bucket(bucket\_name)

blob = bucket.blob(blob\_name)

url = blob.generate\_signed\_url(

version="v4",

expiration=timedelta(minutes=expiration\_minutes),

method="GET"

)

print(f"\nAccess it securely using this signed URL (valid for {expiration\_minutes} minutes):\n{url}")

return url

* **generate\_signed\_url(bucket\_name, blob\_name, expiration\_minutes=15)**: This function generates a signed URL to access the file from Google Cloud Storage for a limited time.
* **storage\_client.bucket(bucket\_name)**: Gets the bucket object using the bucket name.
* **bucket.blob(blob\_name)**: Gets the blob object representing the file.
* **blob.generate\_signed\_url(...)**: This generates the signed URL with the following parameters:
  + version="v4": Specifies the version of the signed URL.
  + expiration=timedelta(minutes=expiration\_minutes): Sets the expiration time for the signed URL, which defaults to 15 minutes.
  + method="GET": Specifies the HTTP method allowed for the URL (in this case, GET, to retrieve the file).
* **print(f"\nAccess it securely using this signed URL...")**: Prints the signed URL that can be used to access the file securely.

**7. Main Block**

python

CopyEdit

if \_\_name\_\_ == "\_\_main\_\_":

bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}"

blob\_name = "index.html"

file\_path = create\_html\_file()

bucket = create\_private\_bucket(bucket\_name)

upload\_file(bucket, file\_path, blob\_name)

generate\_signed\_url(bucket.name, blob\_name)

* **if \_\_name\_\_ == "\_\_main\_\_":**: This block ensures that the script runs only when it is executed directly, not when it is imported as a module.
* **bucket\_name = f"private-bucket-{uuid.uuid4().hex[:6]}"**: This generates a unique bucket name by creating a random UUID and taking the first 6 characters.
* **blob\_name = "index.html"**: Specifies the name of the file in the bucket (the file we just created locally).
* **file\_path = create\_html\_file()**: Calls the create\_html\_file() function to create the index.html file and returns the path to it.
* **bucket = create\_private\_bucket(bucket\_name)**: Calls the create\_private\_bucket() function to create the bucket and returns the bucket object.
* **upload\_file(bucket, file\_path, blob\_name)**: Uploads the local index.html file to the bucket.
* **generate\_signed\_url(bucket.name, blob\_name)**: Generates a signed URL that allows temporary access to the uploaded file.

**Summary**

This script creates an HTML file, uploads it to a new private Google Cloud Storage bucket, and generates a signed URL that provides temporary access to the file. The signed URL is valid for 15 minutes and allows secure access to the file.