## 🌩️ 1. Overview of Google Cloud Storage (GCS)

**Google Cloud Storage (GCS)** is Google's object storage service. It allows you to:

- Store and retrieve **any type of data (images, videos, backups, logs, etc.)**.
- Access it **anytime, from anywhere in the world**.
- Use it for **hosting websites, backups, data lakes, or distributing large files**.

Each GCS resource is organized into:

- **Buckets** → Top-level containers for your data.
- **Objects** → The actual files stored inside buckets.
- **Folders** (optional structure) → Logical organization (like directories, but virtual).

---

## ⚙️ 2. What is the Google APIs Explorer?

The **APIs Explorer** is an interactive Google tool that lets you:

- Browse and test Google APIs (like Cloud Storage, Vision API, YouTube API, etc.)
- Execute live API requests right from your browser.
- Authenticate via OAuth and view real-time API responses.
- It's perfect for developers to understand how an API behaves **before coding**.

In your lab, you used the **Cloud Storage JSON API** through the APIs Explorer.

---

## 🗄️ 3. Cloud Storage Operations (via APIs Explorer)

### ✅ Task 1: Create Cloud Storage Buckets

- You navigated to **APIs & Services → Library → Cloud Storage JSON API**.
- Opened the **Buckets: insert** method in APIs Explorer.
- Provided:
    - project: your Qwiklabs project ID.
    - name: a **unique bucket name** following naming rules.
- Clicked **Execute** to create the bucket.

### 📘 Response Example:


```
{
 "kind": "storage#bucket",
 "name": "qwiklabs-bucket01",
 "location": "US",
```

"storageClass": "STANDARD"

}

This confirmed the bucket was created successfully.

---

✅ **Task 2: Create a Second Bucket**

You repeated the process to make a **second bucket**, useful for file transfer tasks.

---

✅ **Task 3: Upload Files**

- You uploaded images (demo-image1.png, demo-image2.png) via the **Cloud Console UI**.
- These images became **objects** inside the bucket.
- You could view them under **Cloud Storage → Buckets → Objects tab**.

---

✅ **Task 4: Copy Files Between Buckets**

Used **Objects: copy** method in the APIs Explorer:

- sourceBucket: your first bucket name.
- sourceObject: demo-image1.png.
- destinationBucket: your second bucket name.
- destinationObject: demo-image1-copy.png.

📘 **Result:** The file was successfully copied using the Cloud Storage API without manually downloading or re-uploading.

---

✅ **Task 5: Delete Files**

Used the **Objects: delete** API method to remove objects from a bucket:

- Specified bucket and object name.
- Got a **204 No Content** response → success.

---

✅ **Task 6: Delete a Bucket**

Finally, used **Buckets: delete** method:

- Provided bucket name.
- Executed and got **204** → bucket successfully deleted.

---

### 💻 4. Same Operations via Cloud Shell / CLI

Now you repeated similar steps using **Cloud Shell (command line)** with gcloud and gsutil.

---

### ⚙️ Setup

- Activated **Cloud Shell** (a pre-configured Linux VM).
- Verified authentication and project with:
- gcloud auth list
- gcloud config list project
- gcloud config set compute/region REGION

---

### ✅ Task 1: Create a Bucket

gcloud storage buckets create gs://<YOUR-BUCKET-NAME>

This created a new bucket with **default settings** (region, storage class, permissions).

---

### ✅ Task 2: Upload an Object

1. Download an image using curl:
2. curl --output ada.jpg https://storage.googleapis.com/gweb-cloudblog-publish/images/Ada_Lovelace.2e16d0ba.fill-1200x630-c100.jpg
3. Upload it to your bucket:
4. gcloud storage cp ada.jpg gs://YOUR-BUCKET-NAME

---

### ✅ Task 3: Download from the Bucket

gcloud storage cp -r gs://YOUR-BUCKET-NAME/ada.jpg .

This downloads the image locally.

---

### ✅ Task 4: Copy File into Folder (Inside Bucket)

gcloud storage cp gs://YOUR-BUCKET-NAME/ada.jpg gs://YOUR-BUCKET-NAME/image-folder/

Creates a **virtual folder** called image-folder and copies the file there.

---

### ✅ Task 5: List Contents

gcloud storage ls gs://YOUR-BUCKET-NAME

To get detailed info:

gcloud storage ls -l gs://YOUR-BUCKET-NAME/ada.jpg

---

✅ **Task 6: Make Object Public**

To make the image accessible publicly:

gsutil acl ch -u AllUsers:R gs://YOUR-BUCKET-NAME/ada.jpg

✅ This gives **read-only access** to everyone (AllUsers).

Now, you could open the **Public URL** to see Ada Lovelace's image live on the web.

---

✅ **Task 7: Remove Public Access**

gsutil acl ch -d AllUsers gs://YOUR-BUCKET-NAME/ada.jpg

Removes the public-read permission from the object.

---

✅ **Task 8: Delete the Object**

gcloud storage rm gs://YOUR-BUCKET-NAME/ada.jpg

Deletes the file permanently.

---

🧠 **5. Concepts You Learned**

| Concept | Meaning |
|---|---|
| **Bucket** | Top-level container in Cloud Storage |
| **Object** | A single file (image, data, etc.) stored in a bucket |
| **Folder** | Logical grouping of objects (not real directories) |
| **JSON API** | RESTful API to access Cloud Storage programmatically |
| **OAuth 2.0** | Authentication method for secure API access |
| **IAM & ACL** | Access control systems for managing permissions |
| **Storage Class** | Determines cost and performance tier (Standard, Nearline, etc.) |
| **204 Status Code** | Indicates successful deletion (no content returned) |

---

🚀 **6. Real-World Application**

You now understand how **APIs** and **CLI tools** can both manage cloud storage.
This skill is foundational for:

- Automating storage management using **Python or Node.js scripts**.
- Integrating storage actions in **machine learning workflows** (e.g., uploading models, datasets).
- Building **AI pipelines** that use GCS as data storage for training and deployment.

☁️ **Google Cloud Storage – Bucket Lock (GSP297)**

🧩 **Overview**

This lab is part of the **Google Cloud Study Jams** series and focuses on the **Cloud Storage Bucket Lock feature**.
It demonstrates how to implement **data retention policies**, **temporary holds**, and **event-based holds** to meet **compliance and regulatory requirements** in real-world scenarios.

**Bucket Lock** allows administrators to:

- Define retention policies that control how long objects in a Cloud Storage bucket must be kept.
- Permanently lock these policies to prevent accidental or intentional modifications.

This is crucial for organizations dealing with **financial**, **legal**, or **healthcare data** where retention compliance is mandatory (e.g., **FINRA**, **SEC**, **CFTC** regulations).

---

🎯 **Learning Objectives**

By completing this lab, you'll learn how to:

1. Create a Cloud Storage bucket.
2. Define and configure an **Object Retention Policy**.
3. Lock a retention policy to make it immutable.
4. Apply **Temporary Holds** on sensitive objects during audits or investigations.
5. Configure **Event-Based Holds** for time-dependent data retention workflows.

---

🛠️ **Prerequisites**

Before starting:

- Access to **Google Cloud Console** through **Qwiklabs temporary credentials**.
- Recommended browser: **Google Chrome (Incognito Mode)**.
- Familiarity with **Cloud Shell** and **gsutil** command-line tools.

⚠️ Use only the provided temporary credentials. Using your personal account may incur real charges.

---

⚙️ **Step-by-Step Implementation**

**Task 1: Create a New Bucket**

Define a variable and create a bucket:

export BUCKET=$(gcloud config get-value project)

gsutil mb "gs://$BUCKET"

✅ *Progress Check:* Bucket created successfully.

---

**Task 2: Define a Retention Policy**

Example: Retain financial records for **10 seconds** (simulating a 6-year retention requirement).

gsutil retention set 10s "gs://$BUCKET"

gsutil retention get "gs://$BUCKET"

Upload a dummy transaction record:

gsutil cp gs://spls/gsp297/dummy_transactions "gs://$BUCKET/"

Check its retention expiration:

gsutil ls -L "gs://$BUCKET/dummy_transactions"

✅ *Progress Check:* Retention Policy applied successfully.

---

**Task 3: Lock the Retention Policy**

Locking ensures that the retention duration can never be reduced or removed.

gsutil retention lock "gs://$BUCKET/"

Confirm:

gsutil retention get "gs://$BUCKET/"

Once locked, this policy **cannot be unlocked or shortened** — only extended.

✅ *Progress Check:* Retention Policy locked permanently.

---

**Task 4: Apply a Temporary Hold**

When a regulator requests an audit, temporary holds can prevent deletion of critical data.

Set a temporary hold:

gsutil retention temp set "gs://$BUCKET/dummy_transactions"

Try deleting:

gsutil rm "gs://$BUCKET/dummy_transactions"

Expected error:

AccessDeniedException: 403 Object is under active Temporary hold.

Release the hold after audit completion:

gsutil retention temp release "gs://$BUCKET/dummy_transactions"

✅ *Progress Check:* Temporary Hold applied and released.

---

**Task 5: Event-Based Holds**

Used for time-based workflows — e.g., retain loan records for 10 seconds **after a loan is paid off**.

Enable event-based holds:

gsutil retention event-default set "gs://$BUCKET/"

Upload a loan record:

gsutil cp gs://spls/gsp297/dummy_loan "gs://$BUCKET/"

Check its hold status:

gsutil ls -L "gs://$BUCKET/dummy_loan"

Output:

Event-Based Hold: Enabled

Release the event-based hold once the “event” (e.g., loan payoff) occurs:

gsutil retention event release "gs://$BUCKET/dummy_loan"

Then verify retention expiration:

gsutil ls -L "gs://$BUCKET/dummy_loan"

✅ *Progress Check:* Event-Based Hold workflow configured.

---

🧠 **Key Concepts**

| Feature | Description | Example Use Case |
|---|---|---|
| **Retention Policy** | Minimum time objects must be stored before deletion. | Financial transaction data retention. |
| **Lock Retention Policy** | Permanently prevent retention time reduction/removal. | Compliance with FINRA, SEC regulations. |
| **Temporary Hold** | Protect objects from deletion during audits/investigations. | Regulatory audit data protection. |
| **Event-Based Hold** | Delay retention countdown until a triggering event. | Loan data retained after repayment. |

---

### 📊 Real-World Relevance

- **Finance:** Ensure compliance with record-keeping laws.
- **Healthcare:** Protect patient data for required retention periods.
- **Legal:** Prevent premature deletion of case-related evidence.
- **Enterprise:** Automate compliance and retention strategies using Cloud Storage.

---

### 🗒️ Commands Summary Table

| Task | Command | Purpose |
|---|---|---|
| Create Bucket | export BUCKET=$(gcloud config get-value project)<br>gsutil mb "gs://$BUCKET" | Creates a new Cloud Storage bucket. |
| Set Retention Policy | gsutil retention set 10s "gs://$BUCKET" | Defines a 10-second retention policy. |
| Get Retention Policy | gsutil retention get "gs://$BUCKET" | Displays current retention configuration. |
| Upload File | gsutil cp gs://spls/gsp297/dummy_transactions "gs://$BUCKET/" | Uploads a sample file to test policy. |
| Check Object Info | gsutil ls -L "gs://$BUCKET/dummy_transactions" | Displays metadata and retention info. |
| Lock Retention Policy | gsutil retention lock "gs://$BUCKET/" | Permanently locks retention settings. |
| Temporary Hold (Set) | gsutil retention temp set "gs://$BUCKET/dummy_transactions" | Enables a temporary hold. |
| Temporary Hold (Release) | gsutil retention temp release "gs://$BUCKET/dummy_transactions" | Removes temporary hold. |
| Event-Based Hold (Enable Default) | gsutil retention event-default set "gs://$BUCKET/" | Enables event-based hold for new objects. |
| Event-Based Hold (Release) | gsutil retention event release "gs://$BUCKET/dummy_loan" | Releases the event-based hold. |

---

### 🏁 Conclusion

This lab provided a complete understanding of how **Google Cloud Storage's Bucket Lock** feature enforces **data immutability**, **compliance**, and **audit-readiness**.
Through hands-on experience, you learned to:

- Create and configure retention policies

- Permanently lock policies
- Manage temporary and event-based holds

These capabilities are essential for any organization managing regulated or time-sensitive data in the cloud.

---

### 📚 References

- Google Cloud Storage Documentation
- Bucket Lock Overview
- Lab ID: **GSP297**
- Part of: **Google Cloud Study Jams – Cloud Engineer Learning Path**