How to set up Guide
Video Analysis via Video Intelligence API

Group 9

Chathura Lakmal H P

Harshkumar Vyas

Jiazhou Ye

Rupa Chakrabarti

Devanshi patel

CPSC – 5207 Intro to cloud technologies

Lecturer: Jaspreet Bhatia

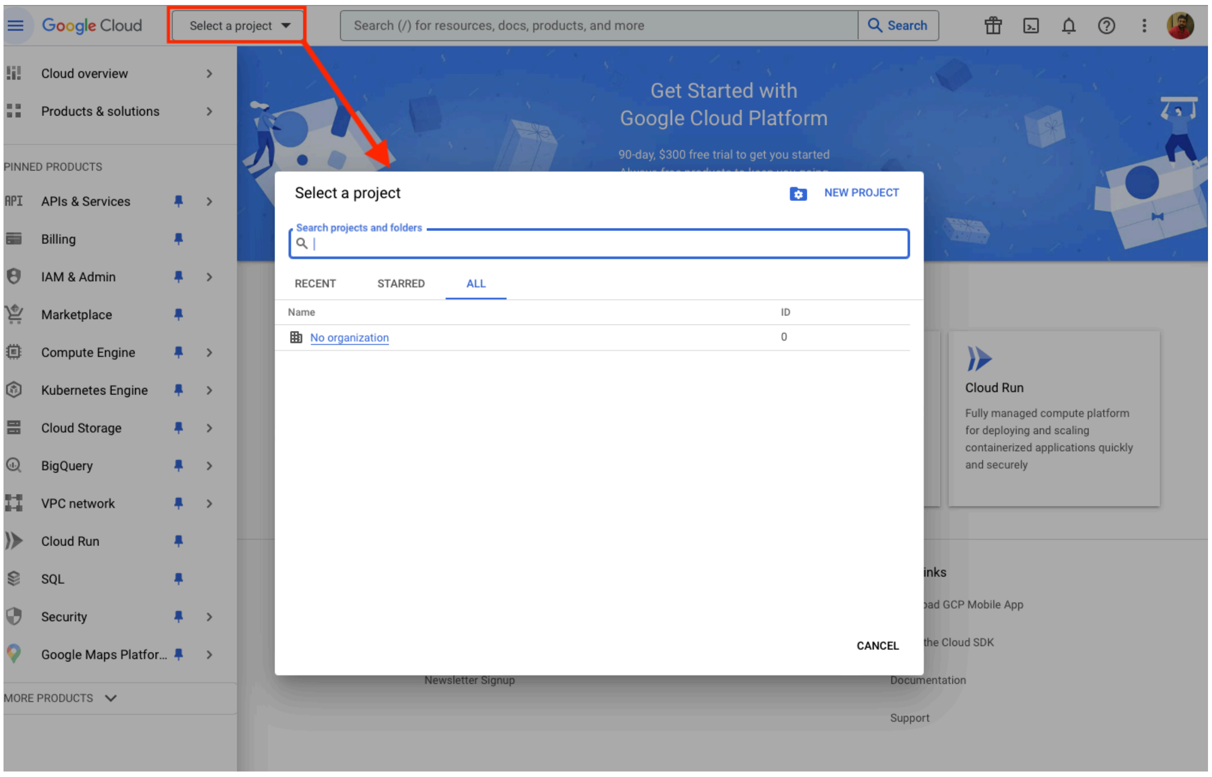
Overview

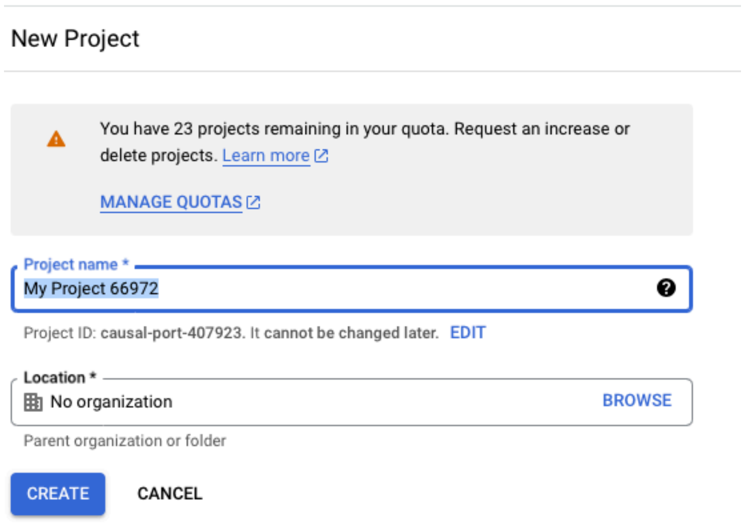
For this project, we need to set up a Google Cloud project. Inside, we have two cloud storage buckets, one cloud function, and a BigQuery instance. Then connect those together to make a pipeline to process the videos uploaded into the input bucket.

Step 1: Set up the cloud project.

Go to Google Cloud Console create an account and add your credit or debit card to initiate an account. Google will not charge you for setting up an account. They will provide you with USD 300 in free credits over 90 days to try out their services.

Once you initiate an account, you can navigate to select a project and give a name for the new project.




New Project

You have 23 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

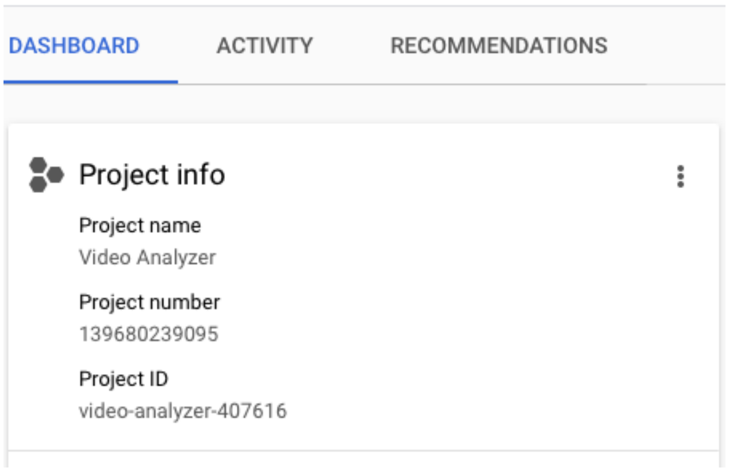
Project name *
My Project 66972

Project ID: causal-port-407923. It cannot be changed later. EDIT

Location *
No organization BROWSE

Parent organization or folder

We gave name “Video Analyzer” Project ID is generated as “video-analyzer-407616”



Project info

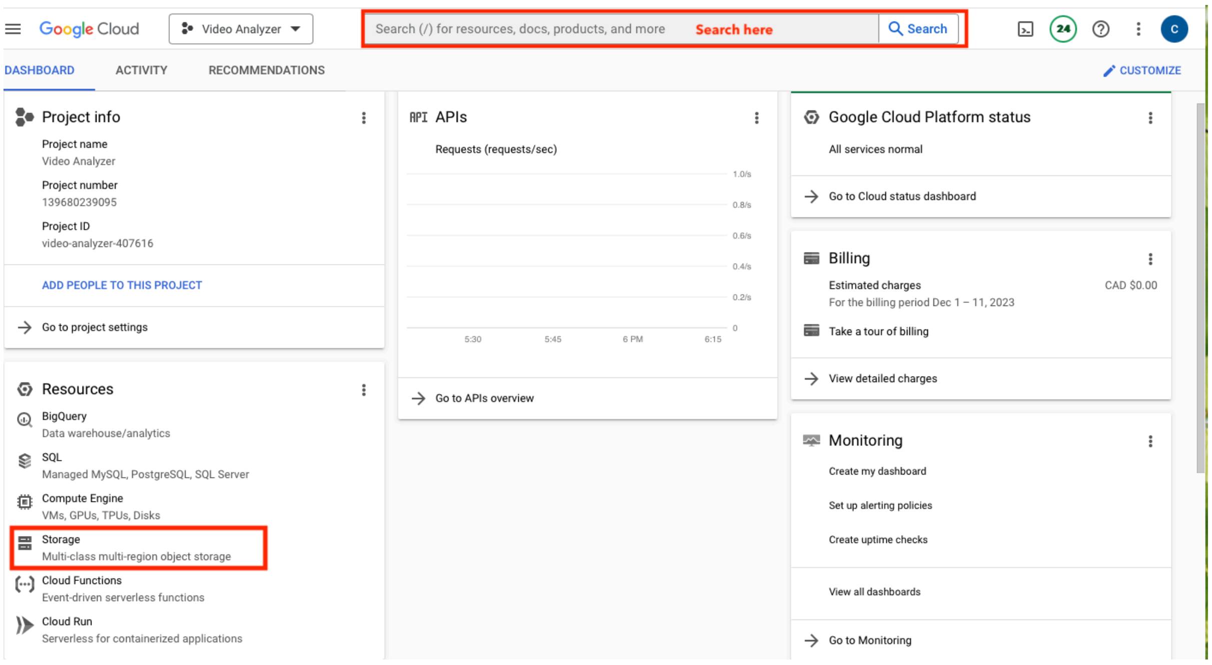
Project name
Video Analyzer

Project number
139680239095

Project ID
video-analyzer-407616

Step 2: Create Cloud Storage Buckets.

Once you select the project, go to Storage by clicking on the icon or searching on the top bar to open the Storage Management Dashboard.



Project info

Project name
Video Analyzer

Project number
139680239095

Project ID
video-analyzer-407616

ADD PEOPLE TO THIS PROJECT

Go to project settings

APIs

Requests (requests/sec)

Go to APIs overview

Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Billing

Estimated charges CAD $0.00
For the billing period Dec 1 – 11, 2023

Take a tour of billing

View detailed charges

Resources

BigQuery
Data warehouse/analytics

SQL
Managed MySQL, PostgreSQL, SQL Server

Compute Engine
VMs, GPUs, TPUs, Disks

Storage
Multi-class multi-region object storage

Cloud Functions
Event-driven serverless functions

Cloud Run
Serverless for containerized applications

Monitoring

Create my dashboard

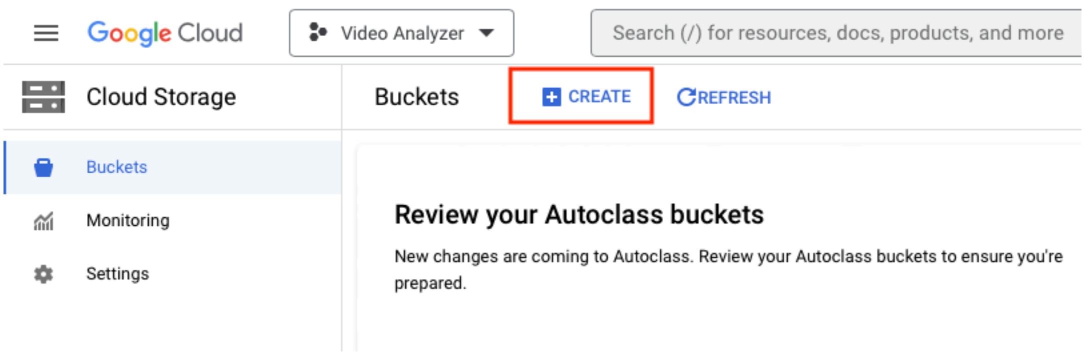
Set up alerting policies

Create uptime checks

View all dashboards

Go to Monitoring

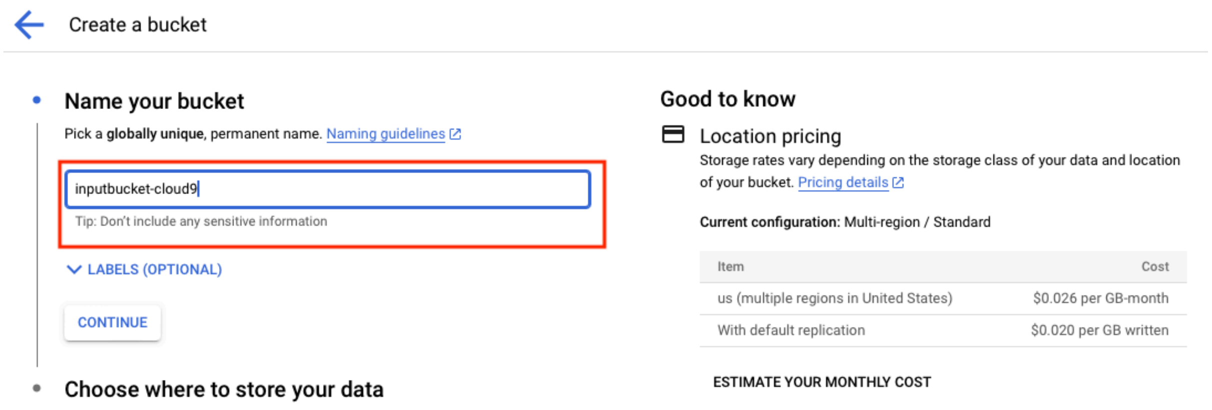
In the storage manage dashboard click create and follow the steps to create a storage bucket.



Review your Autoclass buckets

New changes are coming to Autoclass. Review your Autoclass buckets to ensure you're prepared.

Add a name to the bucket we used “inputbucket-cloud9”



Name your bucket

Pick a **globally unique**, permanent name. Naming guidelines

inputbucket-cloud9

Tip: Don't include any sensitive information

Choose where to store your data

Good to know

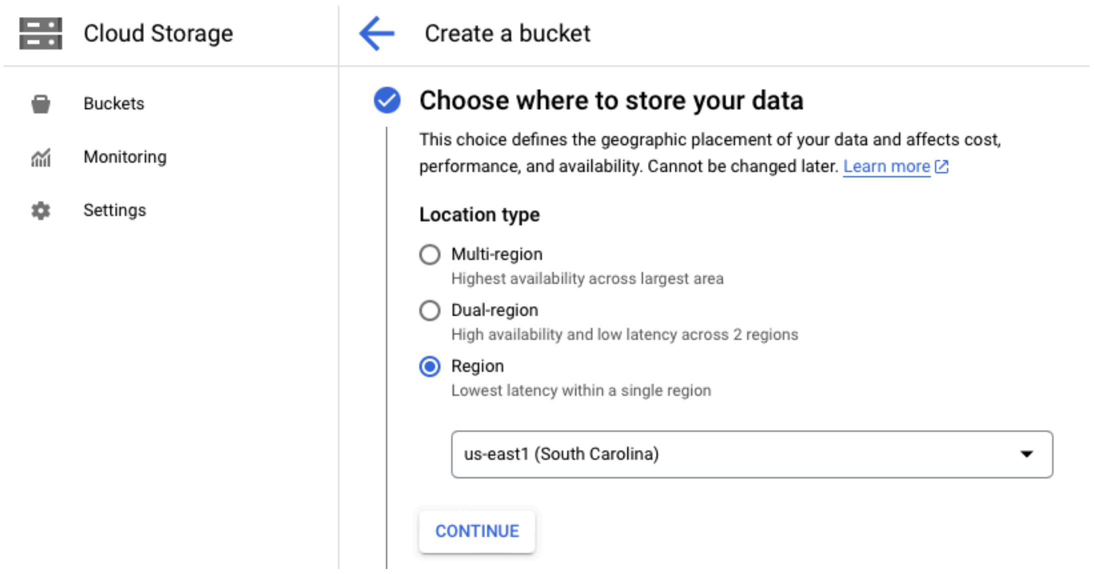
Location pricing

Storage rates vary depending on the storage class of your data and location of your bucket. Pricing details

Current configuration: Multi-region / Standard

Item	Cost
us (multiple regions in United States)	$0.026 per GB-month
With default replication	$0.020 per GB written

Since we did a try-out and our requirements were very low, we selected the single-region, cheaper option "US-East1 (South Carolina)."



Choose where to store your data

This choice defines the geographic placement of your data and affects cost, performance, and availability. Cannot be changed later. Learn more

Location type

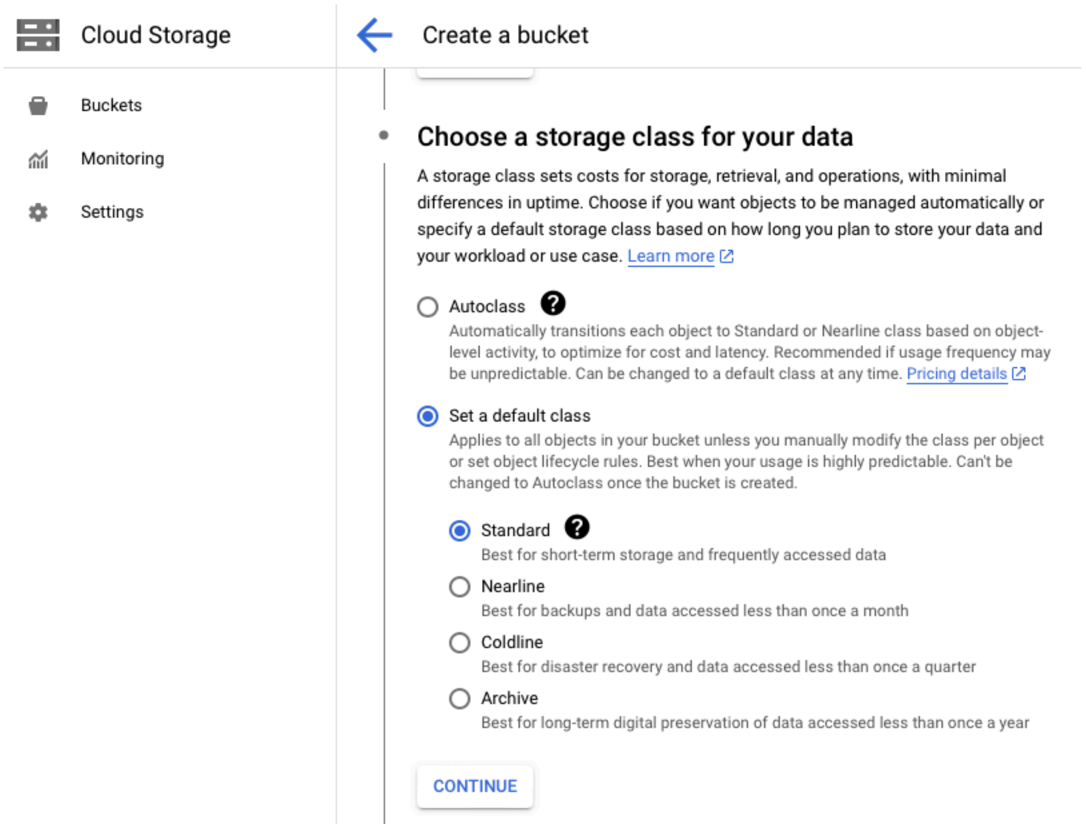
Multi-region
Highest availability across largest area

Dual-region
High availability and low latency across 2 regions

[✓ Region]
Lowest latency within a single region

us-east1 (South Carolina)

Then we choose the storage class as “standard” because we need frequent access to write and read from the bucket.



Choose a storage class for your data

A storage class sets costs for storage, retrieval, and operations, with minimal differences in uptime. Choose if you want objects to be managed automatically or specify a default storage class based on how long you plan to store your data and your workload or use case. Learn more

Autoclass
Automatically transitions each object to Standard or Nearline class based on object-level activity, to optimize for cost and latency. Recommended if usage frequency may be unpredictable. Can be changed to a default class at any time. Pricing details

[✓ Set a default class]
Applies to all objects in your bucket unless you manually modify the class per object or set object lifecycle rules. Best when your usage is highly predictable. Can't be changed to Autoclass once the bucket is created.

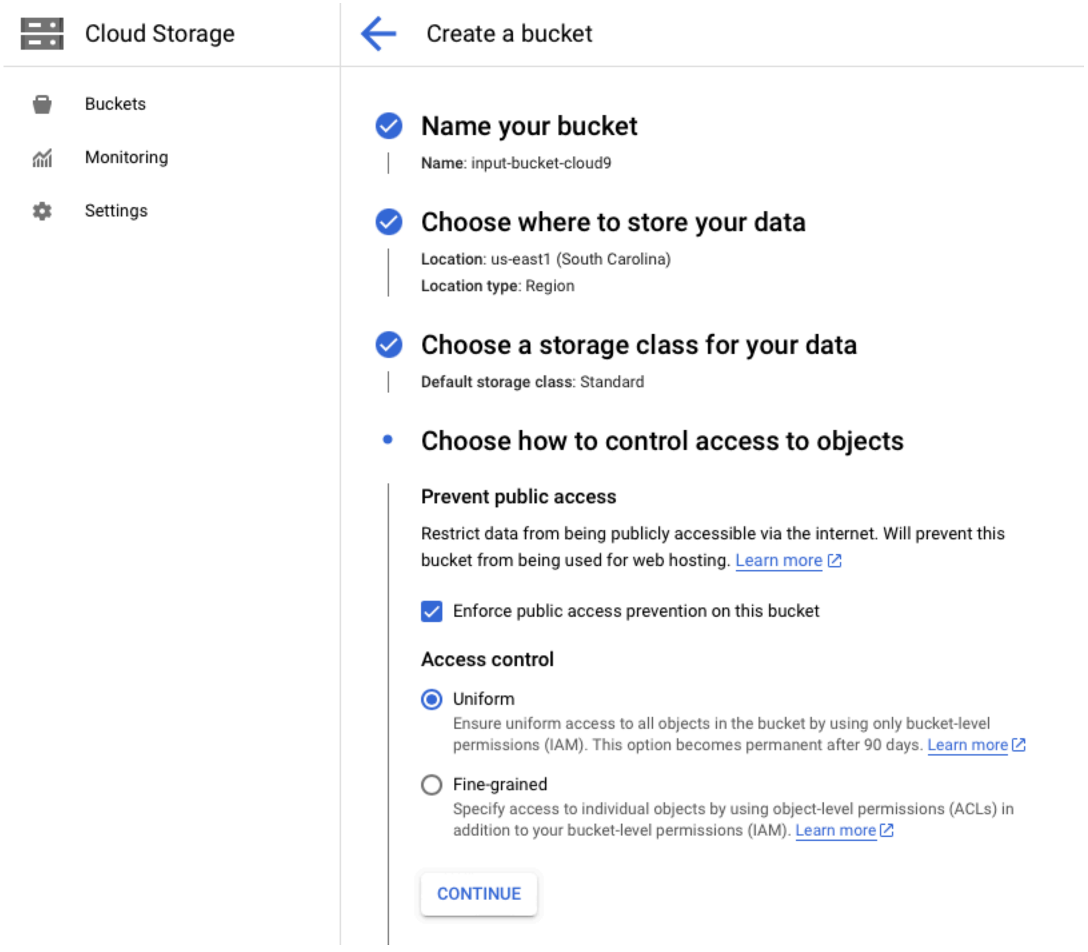
[✓ Standard]
Best for short-term storage and frequently accessed data

Nearline
Best for backups and data accessed less than once a month

Coldline
Best for disaster recovery and data accessed less than once a quarter

Archive
Best for long-term digital preservation of data accessed less than once a year

Then we keep access control “Uniform” since this is a pipeline and once set up, there is no need for manual intervention.


Cloud Storage

Buckets

Monitoring

Settings

Create a bucket

Name your bucket

Name: input-bucket-cloud9

Choose where to store your data

Location: us-east1 (South Carolina)

Location type: Region

Choose a storage class for your data

Default storage class: Standard

Choose how to control access to objects

Prevent public access

Restrict data from being publicly accessible via the internet. Will prevent this bucket from being used for web hosting. Learn more

[✓] Enforce public access prevention on this bucket

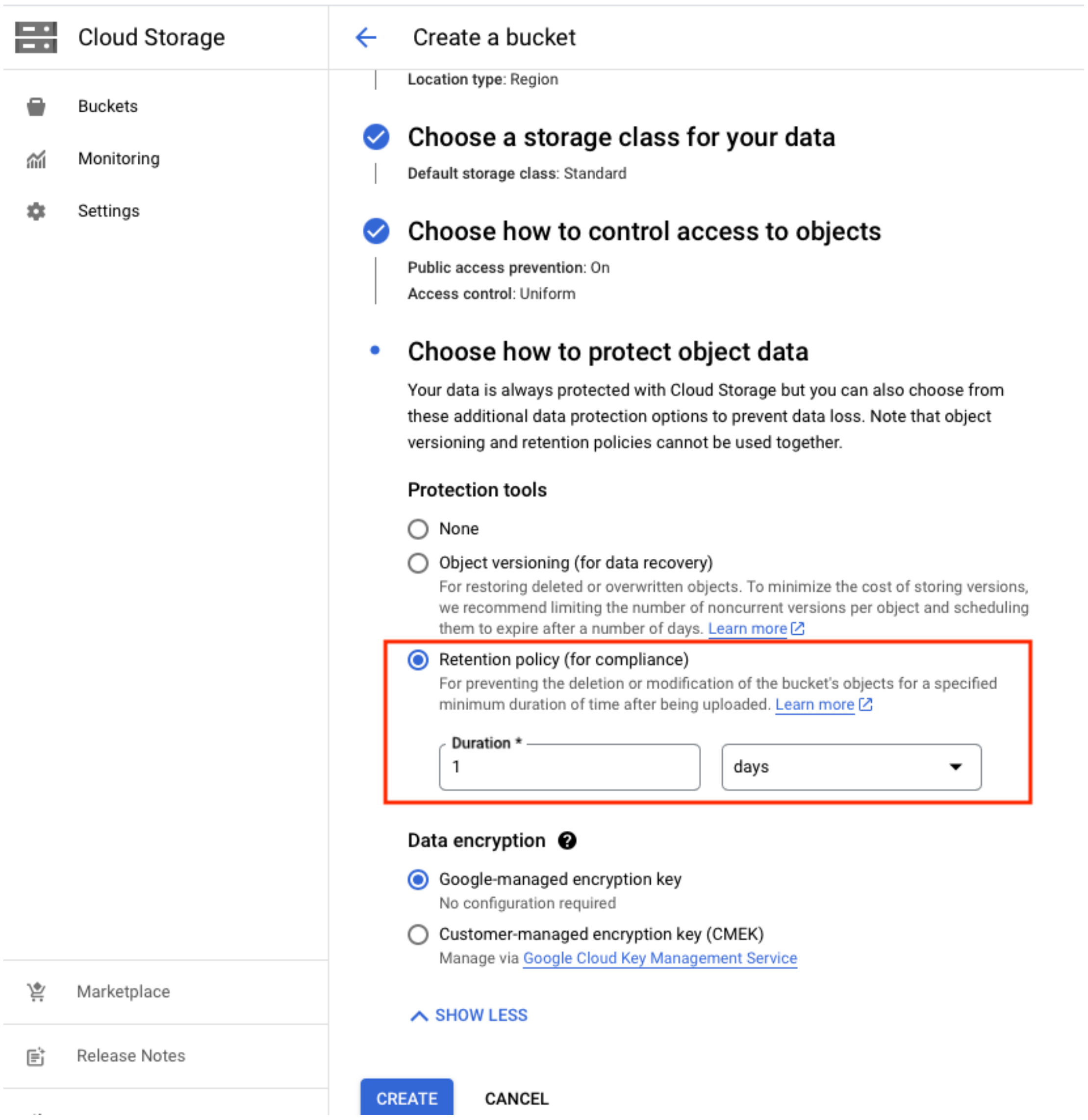
Access control

(●) Uniform
Ensure uniform access to all objects in the bucket by using only bucket-level permissions (IAM). This option becomes permanent after 90 days. Learn more

( ) Fine-grained
Specify access to individual objects by using object-level permissions (ACLs) in addition to your bucket-level permissions (IAM). Learn more

CONTINUE

Once we use the video for processing, we no longer need that data, so we set a data retention policy to delete the file after 1 day.



Location type: Region

Choose a storage class for your data

Default storage class: Standard

Choose how to control access to objects

Public access prevention: On

Access control: Uniform

Choose how to protect object data

Your data is always protected with Cloud Storage but you can also choose from these additional data protection options to prevent data loss. Note that object versioning and retention policies cannot be used together.

Protection tools

None

Object versioning (for data recovery)
For restoring deleted or overwritten objects. To minimize the cost of storing versions, we recommend limiting the number of noncurrent versions per object and scheduling them to expire after a number of days. Learn more

[✓ Retention policy (for compliance)]
For preventing the deletion or modification of the bucket's objects for a specified minimum duration of time after being uploaded. Learn more

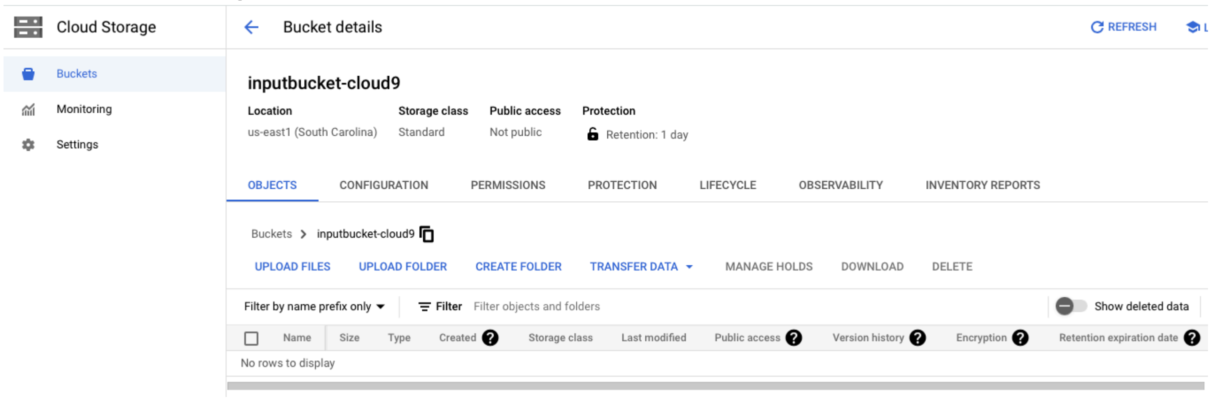
Duration *: 1 days

Data encryption

[✓ Google-managed encryption key]
No configuration required

Customer-managed encryption key (CMEK)
Manage via Google Cloud Key Management Service

Then finish creating the bucket. You can see it like this:



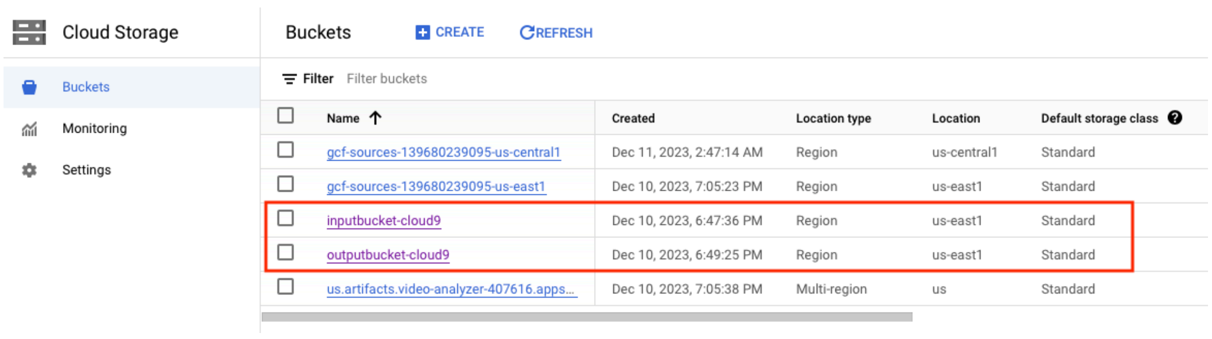
inputbucket-cloud9

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	Retention: 1 day

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date

No rows to display

Follow the same steps to create another bucket for the JSON output file to save. We named it "outputbucket-cloud9.".

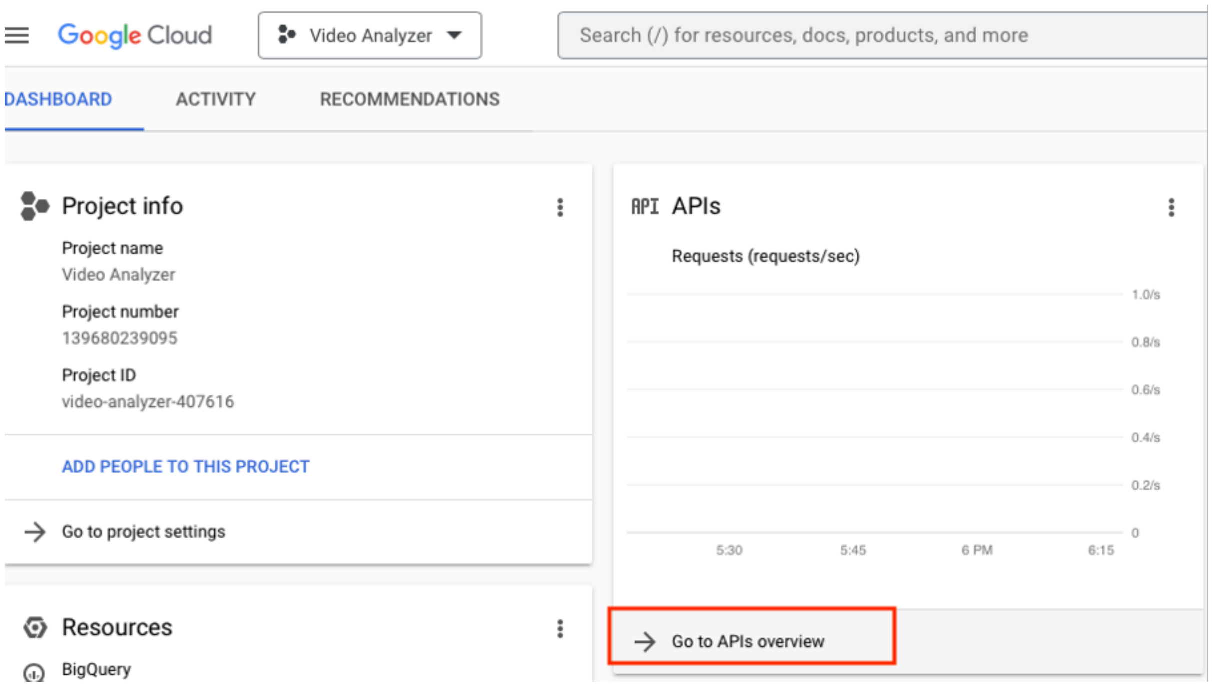


Name	Created	Location type	Location	Default storage class
gcf-sources-139680239095-us-central1	Dec 11, 2023, 2:47:14 AM	Region	us-central1	Standard
gcf-sources-139680239095-us-east1	Dec 10, 2023, 7:05:23 PM	Region	us-east1	Standard
inputbucket-cloud9	Dec 10, 2023, 6:47:36 PM	Region	us-east1	Standard
outputbucket-cloud9	Dec 10, 2023, 6:49:25 PM	Region	us-east1	Standard
us.artifacts.video-analyzer-407616.apps...	Dec 10, 2023, 7:05:38 PM	Multi-region	us	Standard

Above are the two buckets we created. The other three buckets are automatically created by Google to support project infrastructure.

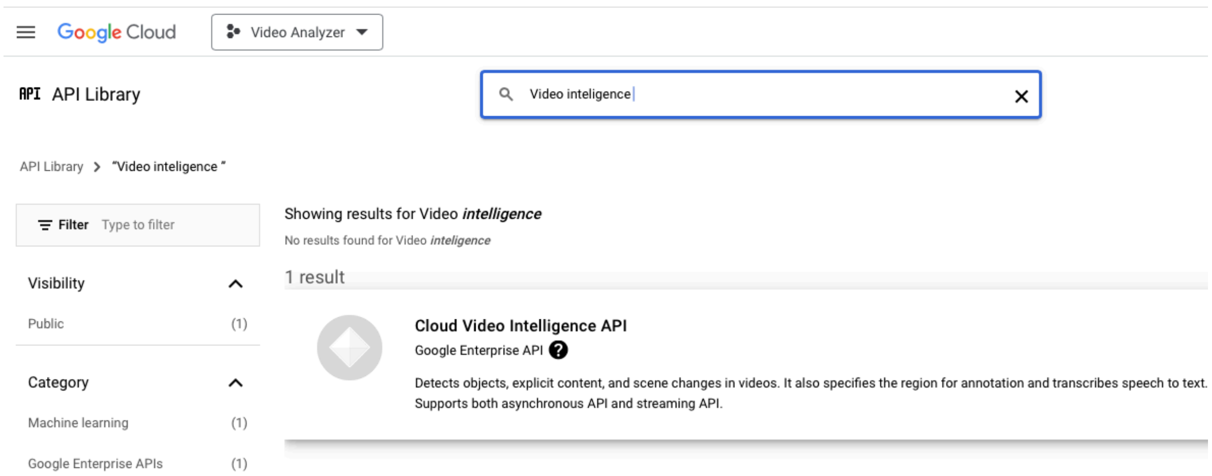
Step 3: Enable Necessary API s

Go the API dash board.



Project name
Video Analyzer
Project number
139680239095
Project ID
video-analyzer-407616

From the API library, select and enable the below APIs if they are not already enabled.



API Library > "Video inteligence "

Filter Type to filter

Showing results for Video ***intelligence***

No results found for Video ***inteligence***

Visibility

Public	(1)

Category

Machine learning	(1)
Google Enterprise APIs	(1)

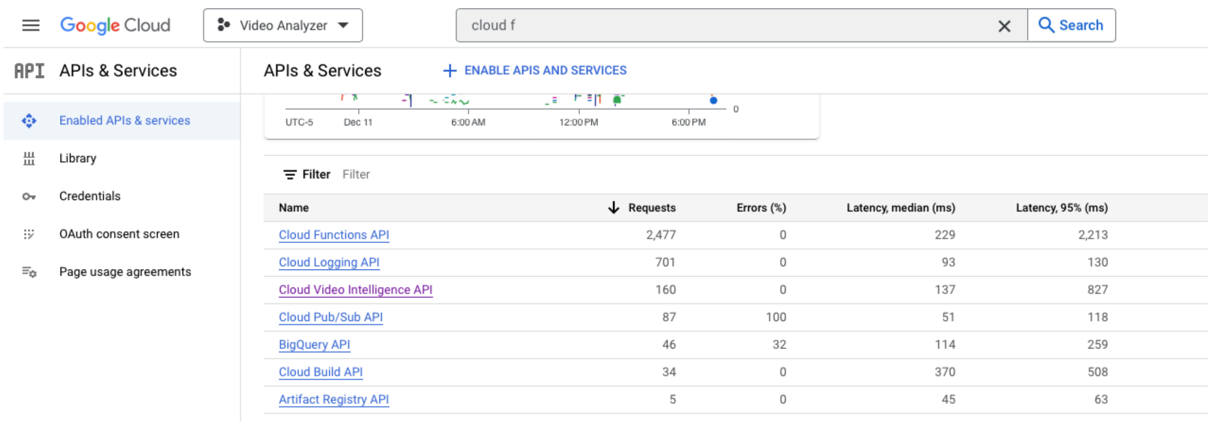
1 result

Cloud Video Intelligence API

Google Enterprise API

Detects objects, explicit content, and scene changes in videos. It also specifies the region for annotation and transcribes speech to text. Supports both asynchronous API and streaming API.

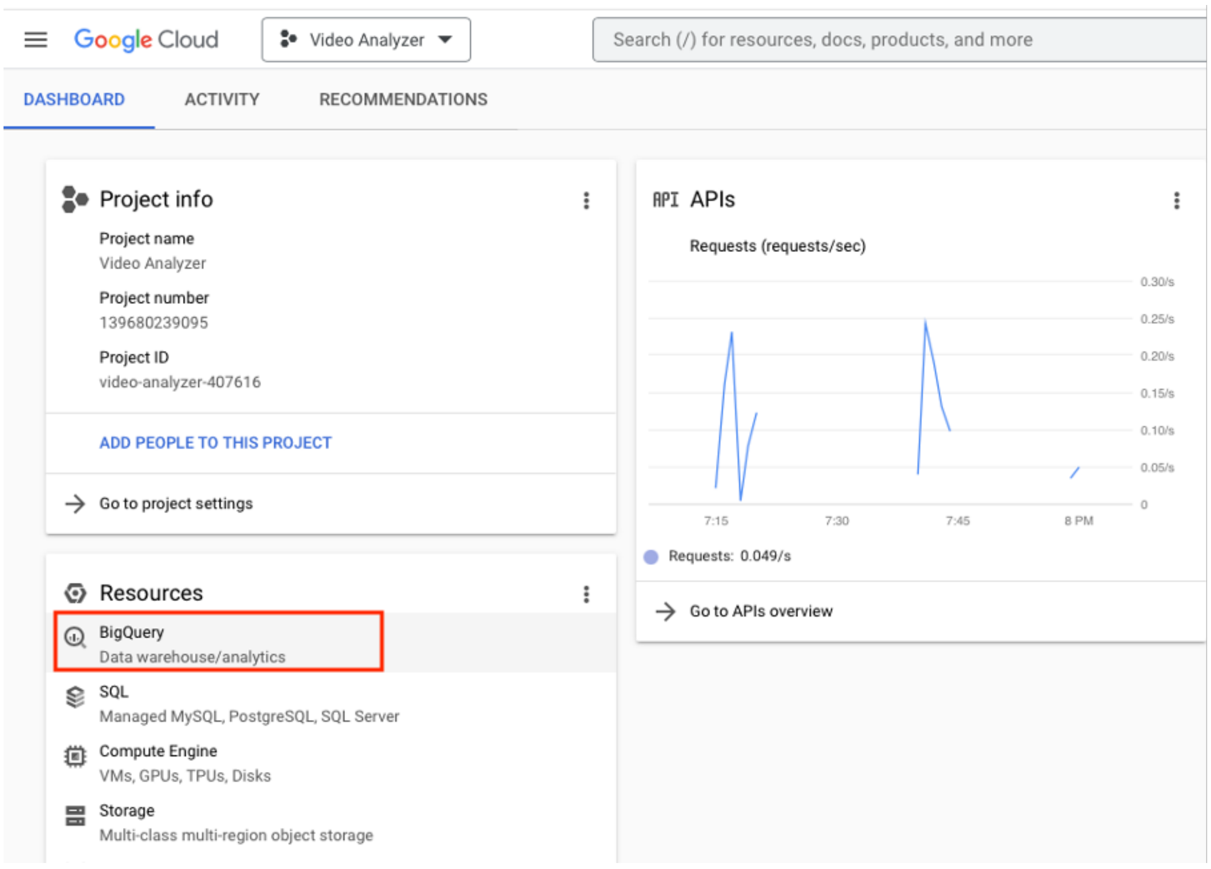
Make sure the below APIs are enabled in your API services dashboard.



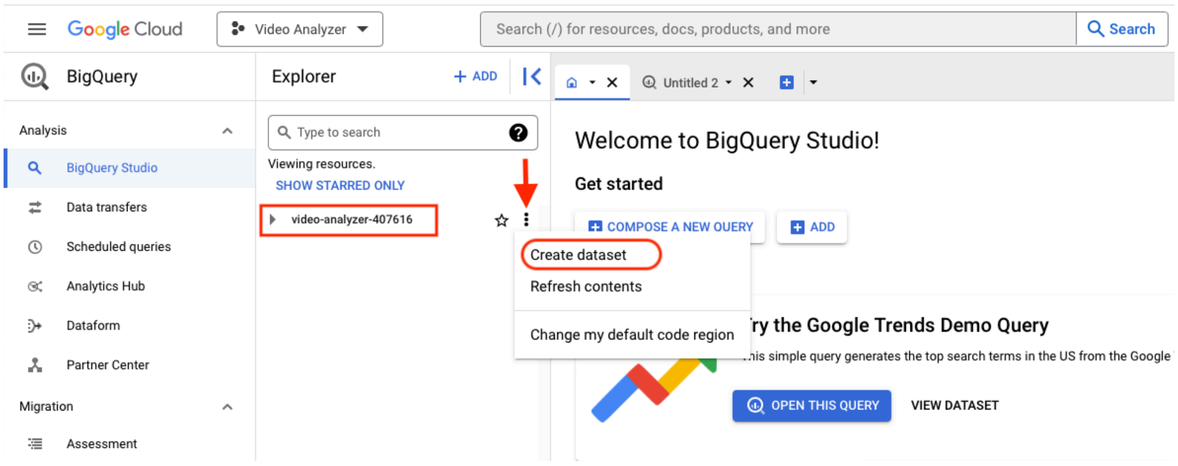
Name	Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)
Cloud Functions API	2,477	0	229	2,213
Cloud Logging API	701	0	93	130
Cloud Video Intelligence API	160	0	137	827
Cloud Pub/Sub API	87	100	51	118
BigQuery API	46	32	114	259
Cloud Build API	34	0	370	508
Artifact Registry API	5	0	45	63

Step 4: Create BigQuery Data Set

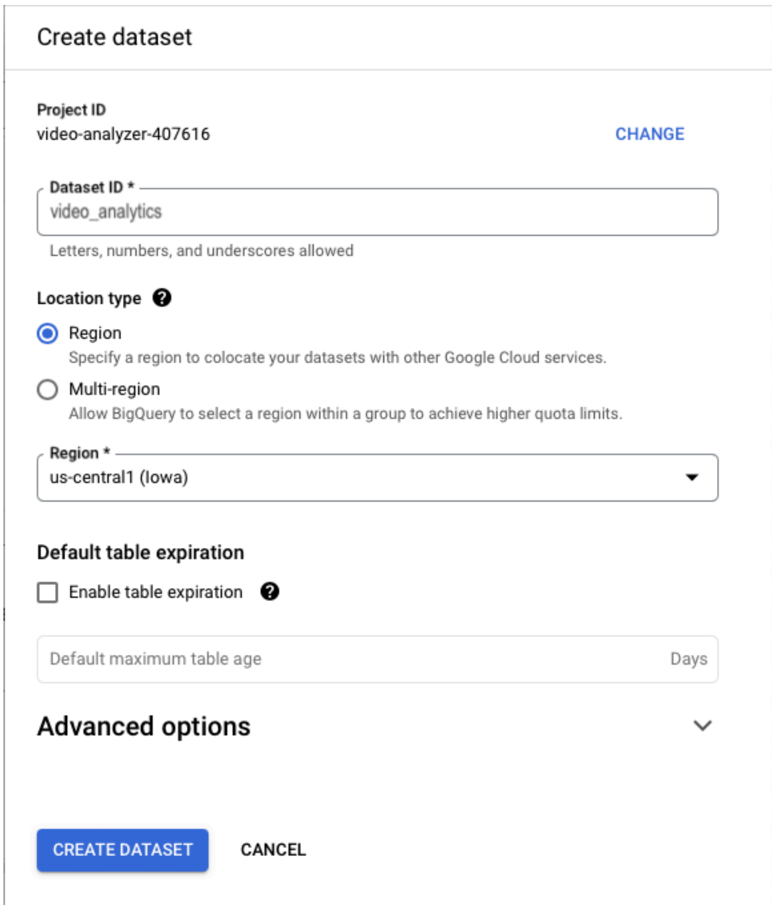
Go to the BigQuery dashboard via the project dashboard.



You can see your project ID on the side panel. Click on the three dots near it to open the drop-down menu. select create dataset.



Give a name to the dataset; we use "video_analytics," which we are going to use in the cloud function in the next step. Select the same region as the one in which we created the bucket to minimize the latency. Since the project does not have demand to be highly available, we did not select multiple regions.



Create dataset

Project ID
video-analyzer-407616

Dataset ID *
video_analytics
Letters, numbers, and underscores allowed

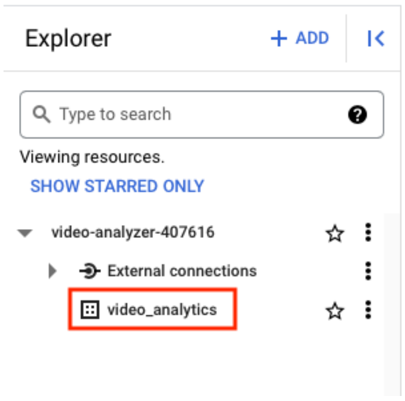
Location type
[✓ Region]
Specify a region to colocate your datasets with other Google Cloud services.
Multi-region
Allow BigQuery to select a region within a group to achieve higher quota limits.

Region *
us-central1 (Iowa)

Default table expiration
Enable table expiration
Default maximum table age — Days

Advanced options

Once you complete creating the data set, you can see it in the Explorer window as below. No need to create tables since a function will create them if they do not exist.



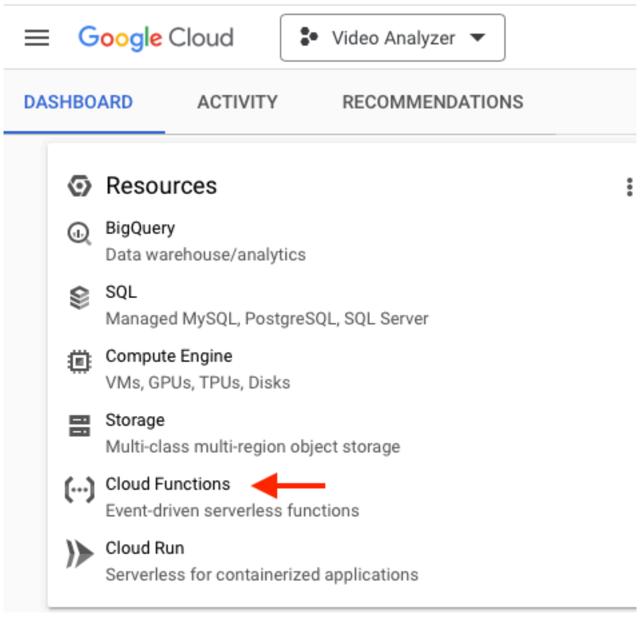
Explorer

Type to search
Viewing resources.

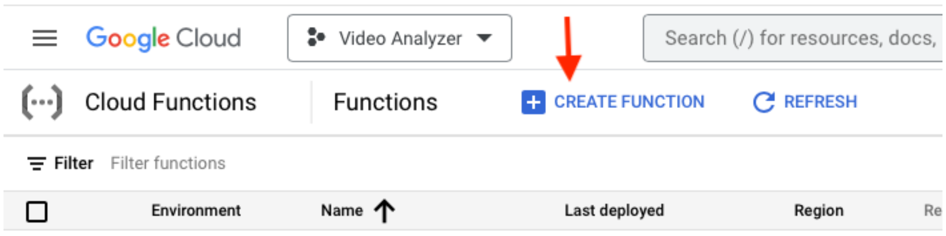
video-analyzer-407616
External connections
video_analytics

Step 5: Create Cloud function.

This is the main and vital step that connects other components and enables the pipeline for the data flow. Go to the project dashboard and open the cloud functions dashboard.



Click on the create button to open the cloud creation wizard.

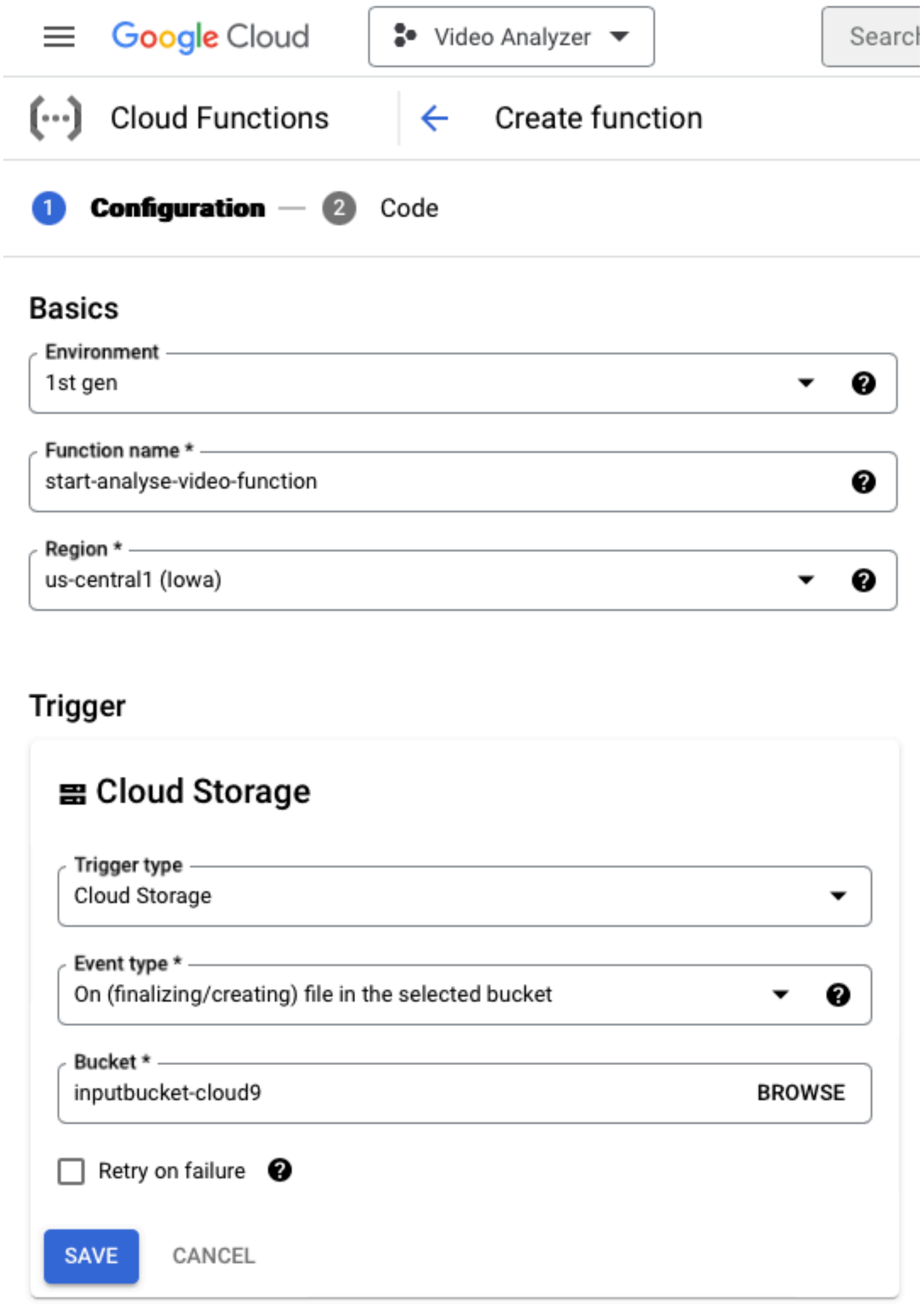


In the cloud function, create a wizard. We have selected the environment as the first generation, provided a function name we have given, "start-analyse-video-function," and selected the region as the same area as when we create cloud buckets and BigQuery data bases to minimize the latency.

In the Trigger section, select trigger type as “Cloud Storage” and select event type as "On (finalizing or creating) file in the selected bucket.”.

In the select bucket section, click on browse to open the selection window, which will show all the previously created buckets. Here you need to carefully select the video_input bucket; in our case, it is "inputbucket-cloud9.”

This is the triggering point of the cloud function once an image upload finishes this could function will start running on that object. The image reference will receive to the cloud function as an event. Your configuration should look like below. After that scroll down a little.



Basics

Environment: 1st gen

Function name *: start-analyse-video-function

Region *: us-central1 (Iowa)

Trigger

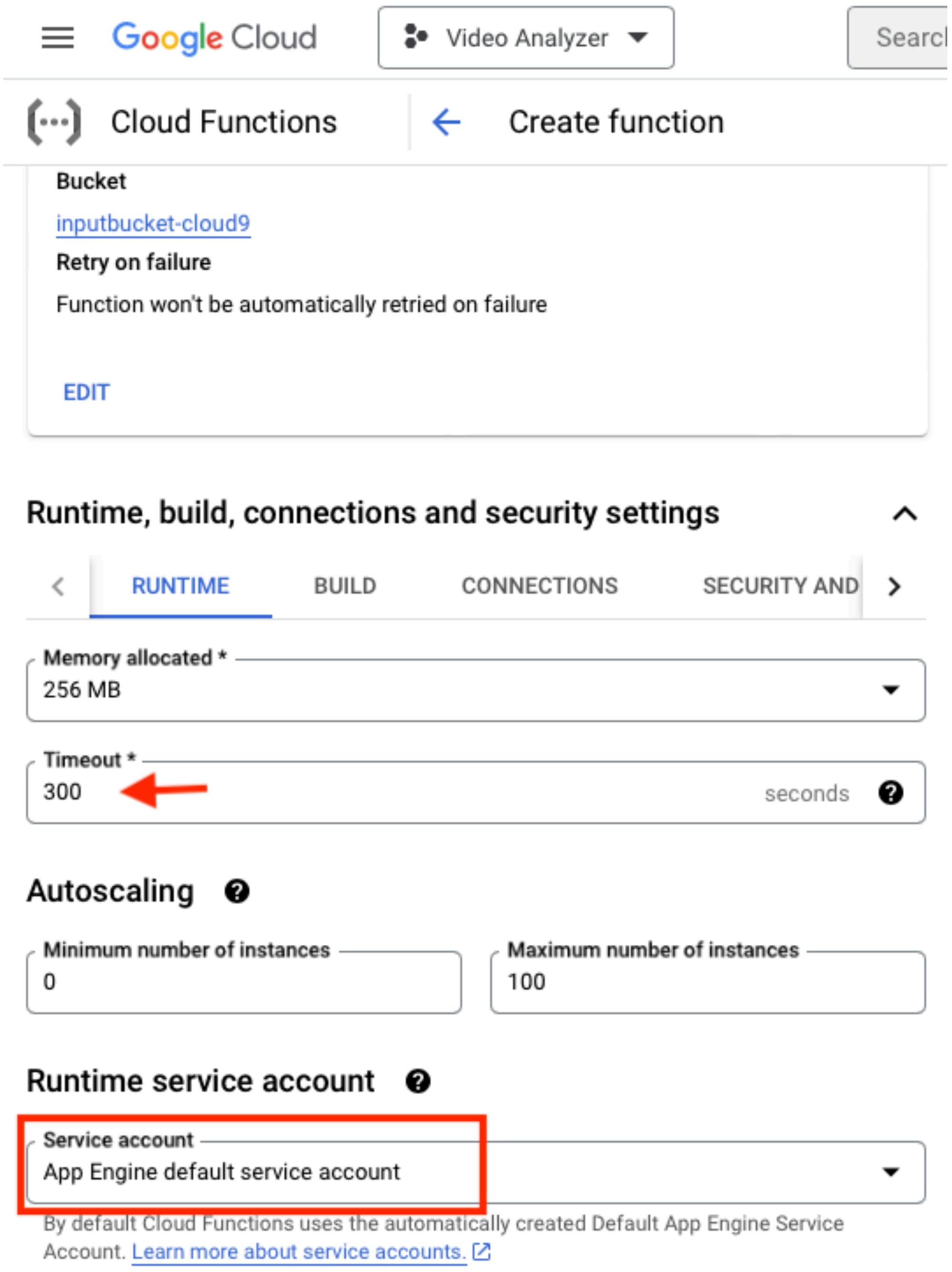
Cloud Storage

Trigger type: Cloud Storage

Event type *: On (finalizing/creating) file in the selected bucket

Bucket *: inputbucket-cloud9

Into the section on runtime settings. Here we update the memory and timeout length based on our expected video length and could function complexity. In our case, we are going to use smaller video clips from online sources that are less than 30 seconds for demonstration purposes. So we keep the default 256MB limit and update the timeout to 300s in case longer processing is needed. Also, please note that the service account for the could function is kept as the default App Engine service account. However, we can create a new service account and provide specific permissions in IAM to ensure more secure execution.



Bucket
inputbucket-cloud9
Retry on failure
Function won't be automatically retried on failure

Runtime, build, connections and security settings

Memory allocated * 256 MB

Timeout * 300 seconds

Autoscaling

Minimum number of instances 0

Maximum number of instances 100

Runtime service account

Service account: App Engine default service account

By default Cloud Functions uses the automatically created Default App Engine Service Account. Learn more about service accounts.

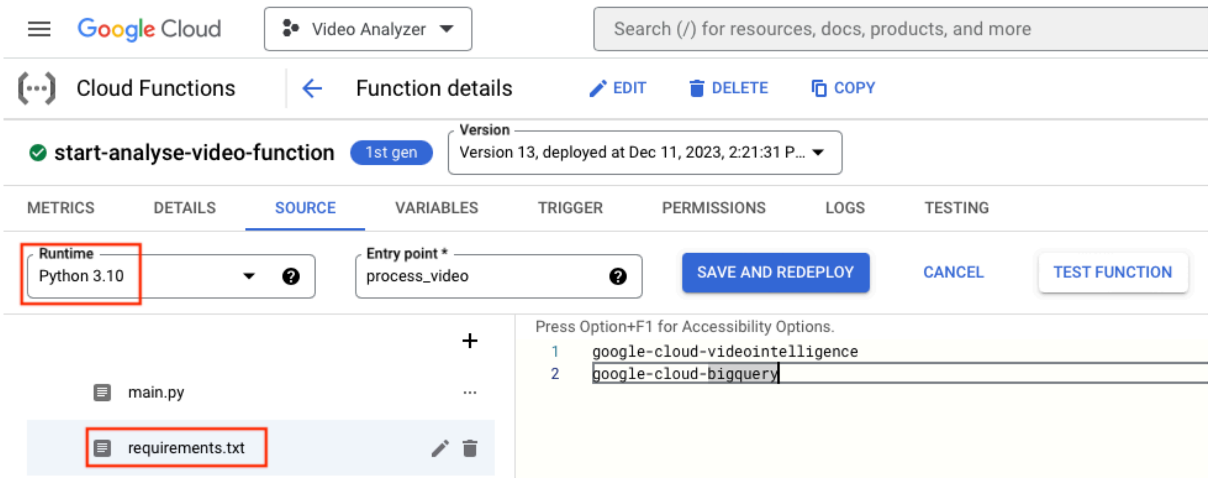
Click next,

In the source code section, select Runtime as the "Python 3.10" environment and select the "requirements.txt" file to add the lines.

google-cloud-videointelligence

google-cloud-bigquery

to it in the inline editor.



Version: Version 13, deployed at Dec 11, 2023, 2:21:31 P...

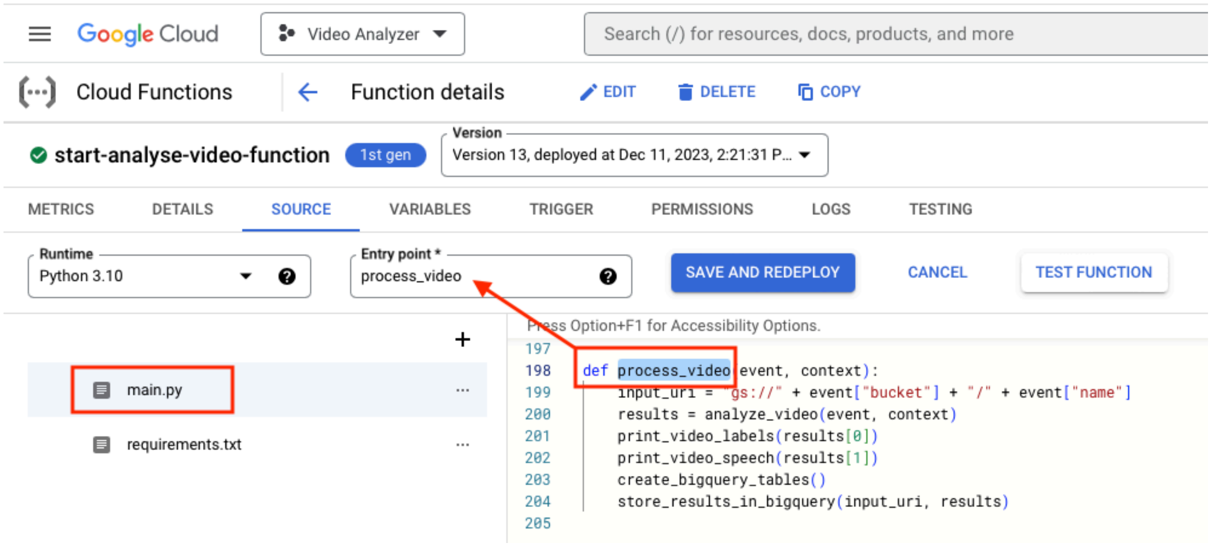
Runtime: Python 3.10

Entry point *: process_video

Press Option+F1 for Accessibility Options.

```
google-cloud-videointelligence
google-cloud-bigquery
```

Then select the next file, "main.py," and copy and paste the provided cloud function code from the end of this document. Make sure the main starting function name matches the entry point, as shown below.



Version: Version 13, deployed at Dec 11, 2023, 2:21:31 P...

Runtime: Python 3.10

Entry point *: process_video

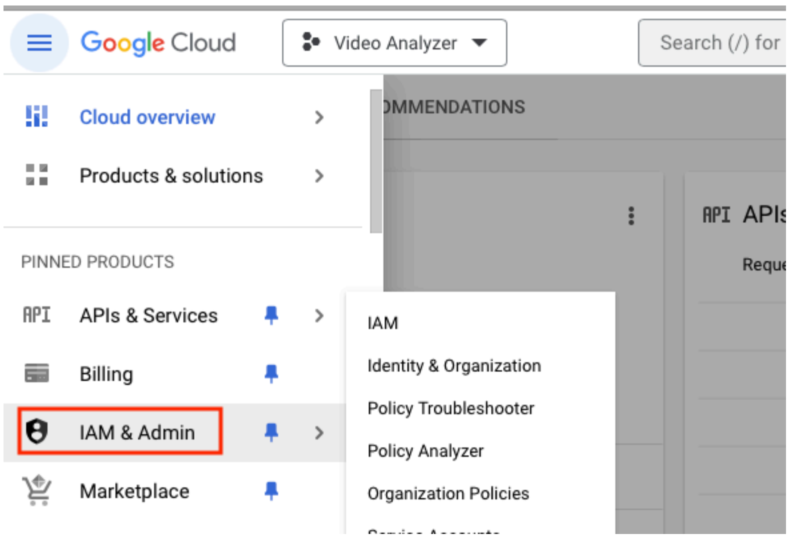
Press Option+F1 for Accessibility Options.

```

def process_video(event, context):
    input_uri = "gs://" + event["bucket"] + "/" + event["name"]
    results = analyze_video(event, context)
    print_video_labels(results[0])
    print_video_speech(results[1])
    create_bigquery_tables()
    store_results_in_bigquery(input_uri, results)

```

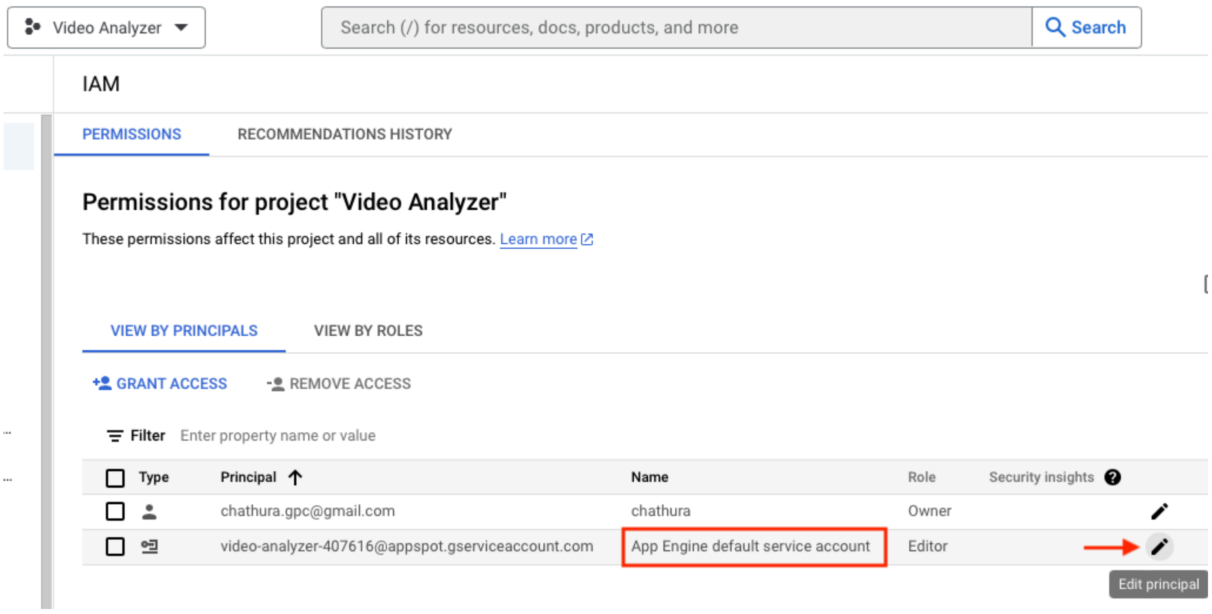
Once you are done, click on the “Save and Deploy” button to deploy the cloud function.

Step 6: Set permission for the service account

One last important step is to give permission to the “App Engine default service account” to access BigQuery. For this from the project dashboard, go to the IAM console.



Go to the “default App Engine default service account” line. Click on the edit icon in the right corner.

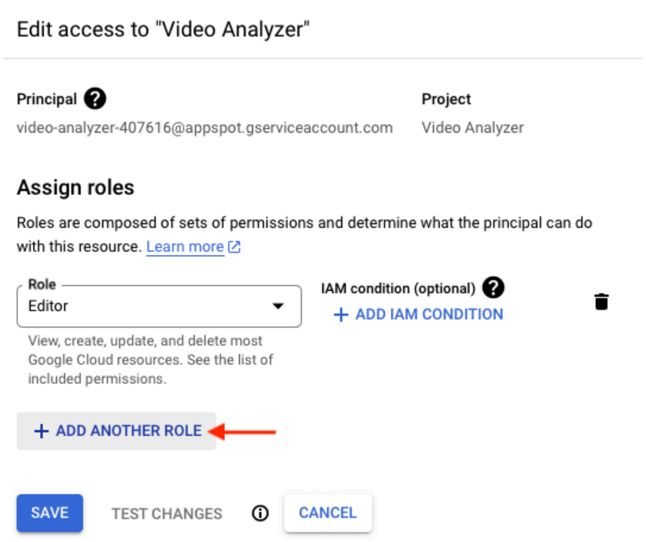


Permissions for project "Video Analyzer"

These permissions affect this project and all of its resources. Learn more

Type	Principal	Name	Role	Security insights
	chathura.gpc@gmail.com	chathura	Owner	
	video-analyzer-407616@appspot.gserviceaccount.com	App Engine default service account	Editor	

Then click on add another role.


Edit access to "Video Analyzer"

Principal
video-analyzer-407616@appspot.gserviceaccount.com

Project
Video Analyzer

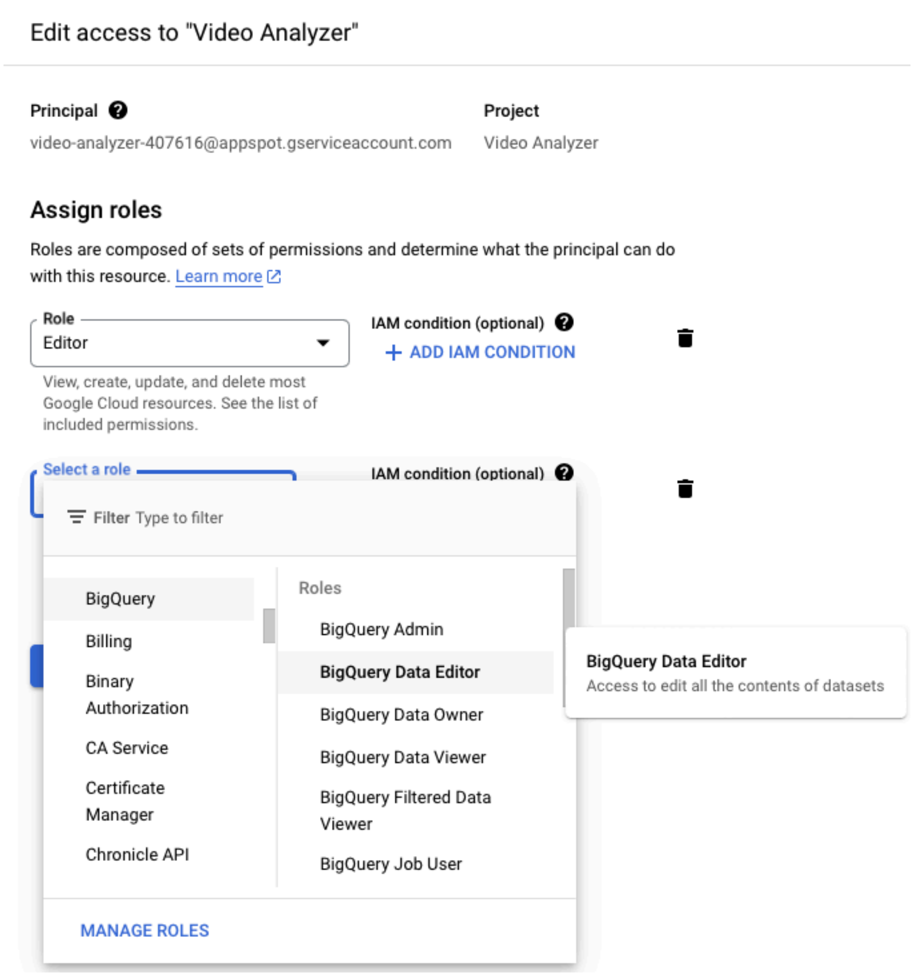
Assign roles

Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more

Role: Editor
View, create, update, and delete most Google Cloud resources. See the list of included permissions.

IAM condition (optional)

Then select “BigQuery” >> “BigQuery Data Editor” role for the “default App Engine default service account” service account. And press save.


Edit access to "Video Analyzer"

Principal
video-analyzer-407616@appspot.gserviceaccount.com

Project
Video Analyzer

Assign roles

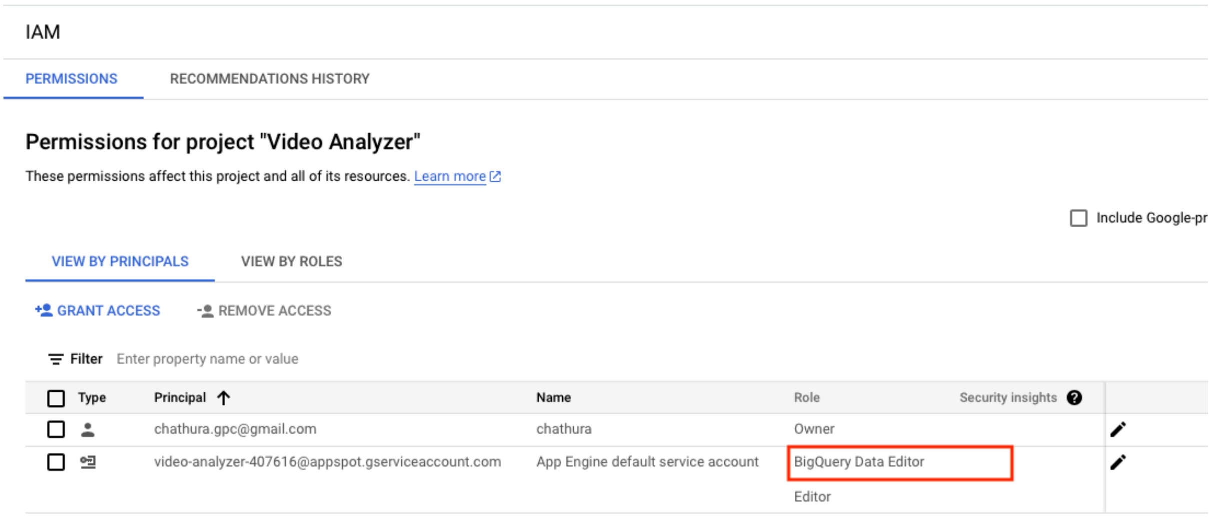
Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more

Role: Editor
View, create, update, and delete most Google Cloud resources. See the list of included permissions.

IAM condition (optional)

BigQuery Data Editor
Access to edit all the contents of datasets

You can see the new role is added to the service account as below.



IAM

PERMISSIONS RECOMMENDATIONS HISTORY

Permissions for project "Video Analyzer"

These permissions affect this project and all of its resources. Learn more

☐ Include Google-pr

VIEW BY PRINCIPALS VIEW BY ROLES

GRANT ACCESS REMOVE ACCESS

Filter Enter property name or value

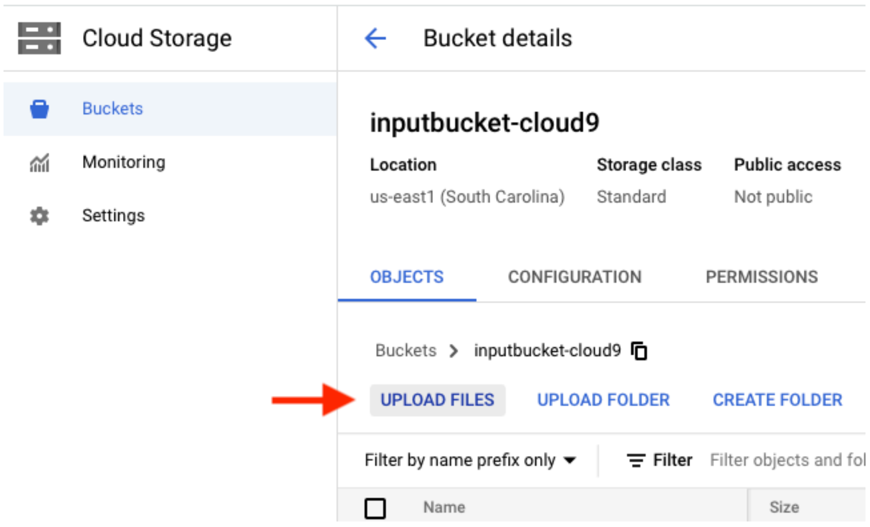
Type	Principal ↑	Name	Role	Security insights
☐	chathura.gpc@gmail.com	chathura	Owner	
☐	video-analyzer-407616@appspot.gserviceaccount.com	App Engine default service account	BigQuery Data Editor Editor	

Now all the setting-up steps are done.

Step 7. Run the project

Make sure you have selected your project from the Dashboard. Open 3 browser tabs (easy to switch quickly).

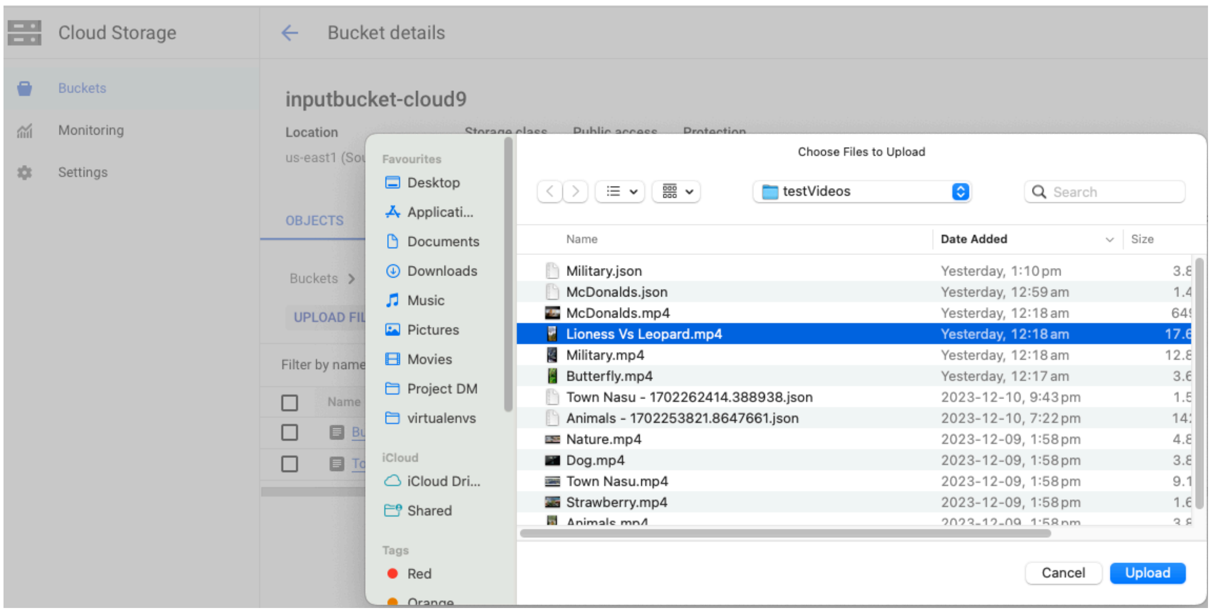
1. Open Storage Buckets >> Our input bucket in the first tab
2. Open Cloud Functions >> Start-analyse-video-function >> select logs tab
3. Open BigQuery Studio >> Select your dataset.
4. In the input bucket tab, click on upload files.



inputbucket-cloud9

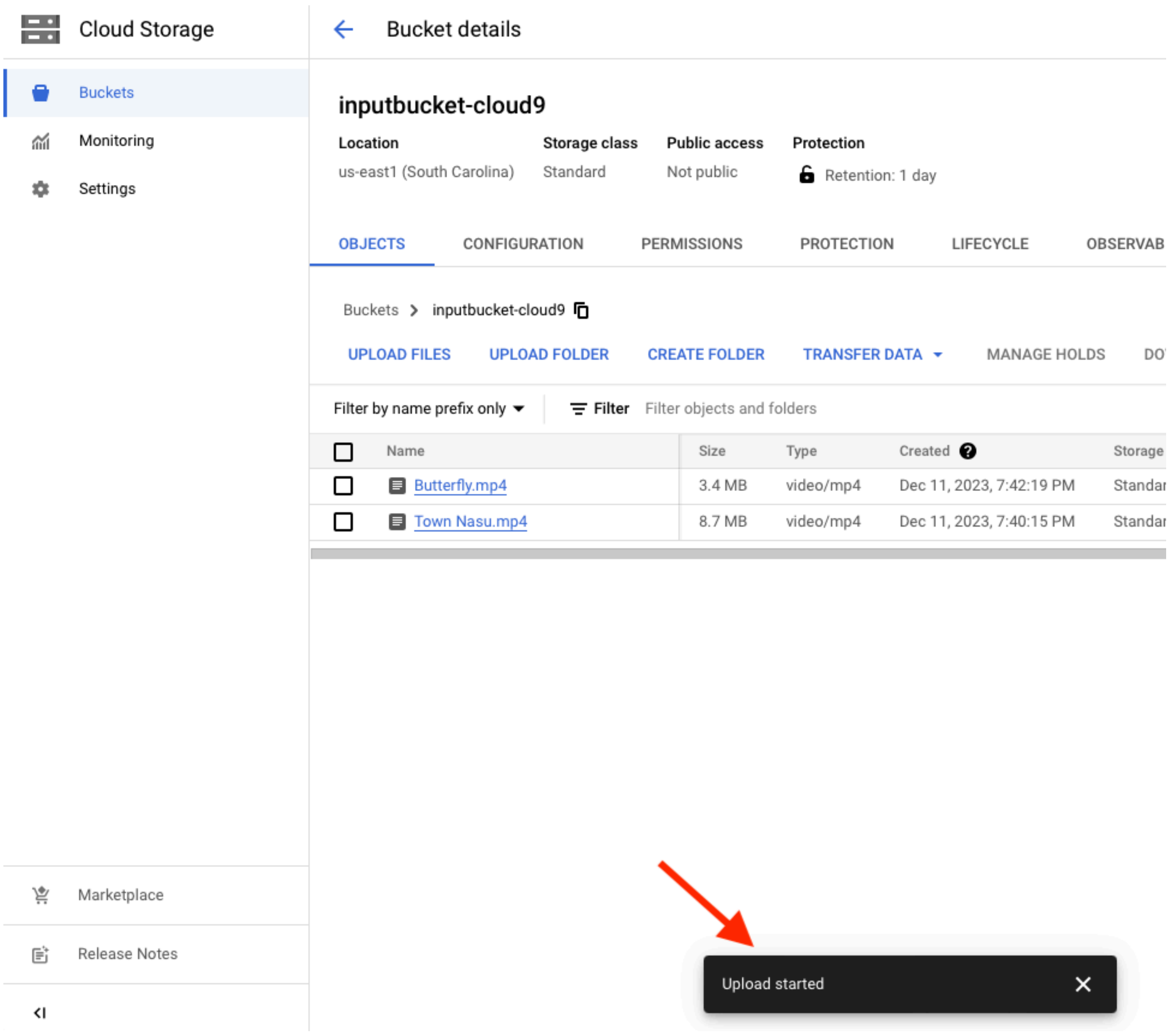
Location	Storage class	Public access
us-east1 (South Carolina)	Standard	Not public

5. Select a small video file that is already saved on your computer. (Note the file name.)



Name	Date Added	Size
Military.json	Yesterday, 1:10 pm	3.8
McDonalds.json	Yesterday, 12:59 am	1.4
McDonalds.mp4	Yesterday, 12:18 am	64
Lioness Vs Leopard.mp4	Yesterday, 12:18 am	17.6
Military.mp4	Yesterday, 12:18 am	12.8
Butterfly.mp4	Yesterday, 12:17 am	3.6
Town Nasu - 1702262414.388938.json	2023-12-10, 9:43 pm	1.5
Animals - 1702253821.8647661.json	2023-12-10, 7:22 pm	14
Nature.mp4	2023-12-09, 1:58 pm	4.8
Dog.mp4	2023-12-09, 1:58 pm	3.8
Town Nasu.mp4	2023-12-09, 1:58 pm	9.1
Strawberry.mp4	2023-12-09, 1:58 pm	1.6
Animals.mp4	2023-12-09, 1:58 pm	3.8

6. Click upload and observe the black-coloured information boxes showing you the status.

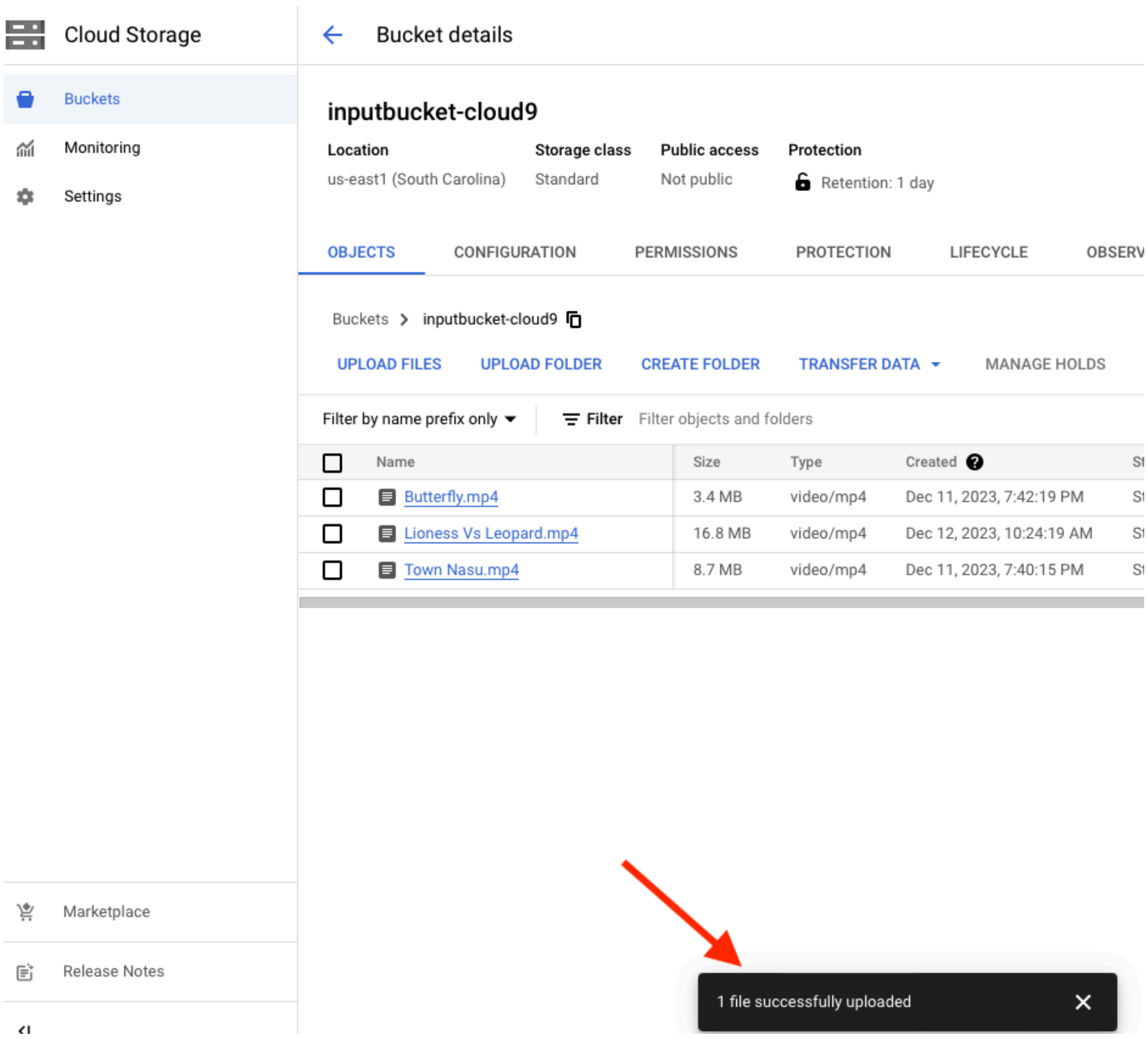


inputbucket-cloud9

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	Retention: 1 day

Name	Size	Type	Created	Storage
Butterfly.mp4	3.4 MB	video/mp4	Dec 11, 2023, 7:42:19 PM	Standar
Town Nasu.mp4	8.7 MB	video/mp4	Dec 11, 2023, 7:40:15 PM	Standar

Upload started



inputbucket-cloud9

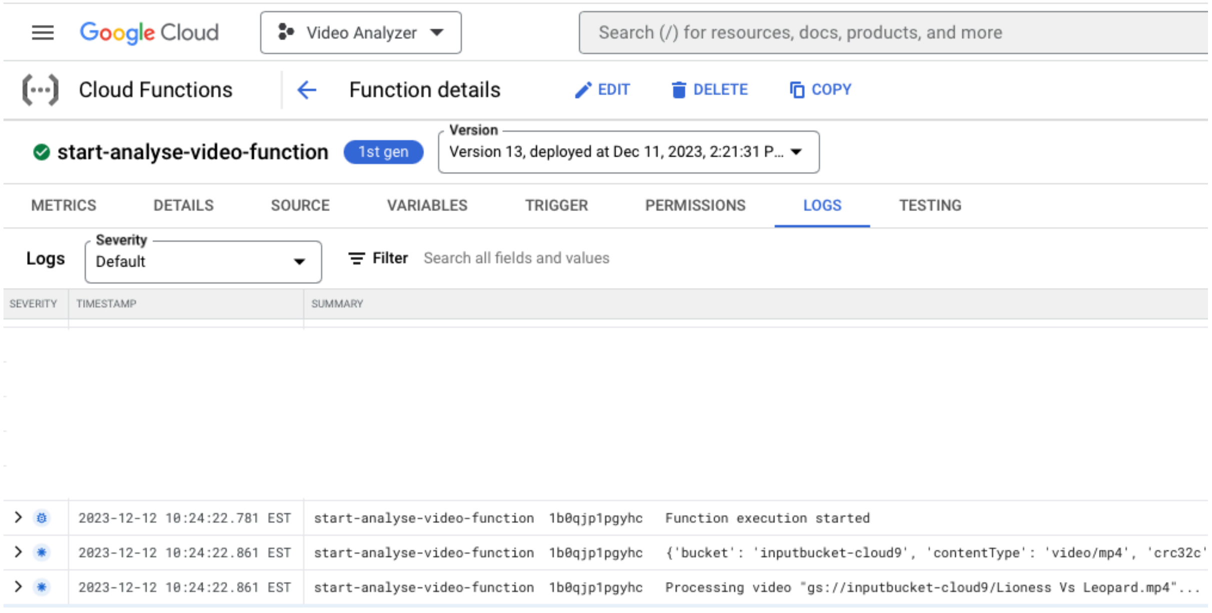
Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	Retention: 1 day

Buckets > inputbucket-cloud9

	Name	Size	Type	Created
☐	Butterfly.mp4	3.4 MB	video/mp4	Dec 11, 2023, 7:42:19 PM
☐	Lioness Vs Leopard.mp4	16.8 MB	video/mp4	Dec 12, 2023, 10:24:19 AM
☐	Town Nasu.mp4	8.7 MB	video/mp4	Dec 11, 2023, 7:40:15 PM

1 file successfully uploaded

7. Now switch to the cloud function log window. See how the function execution started.



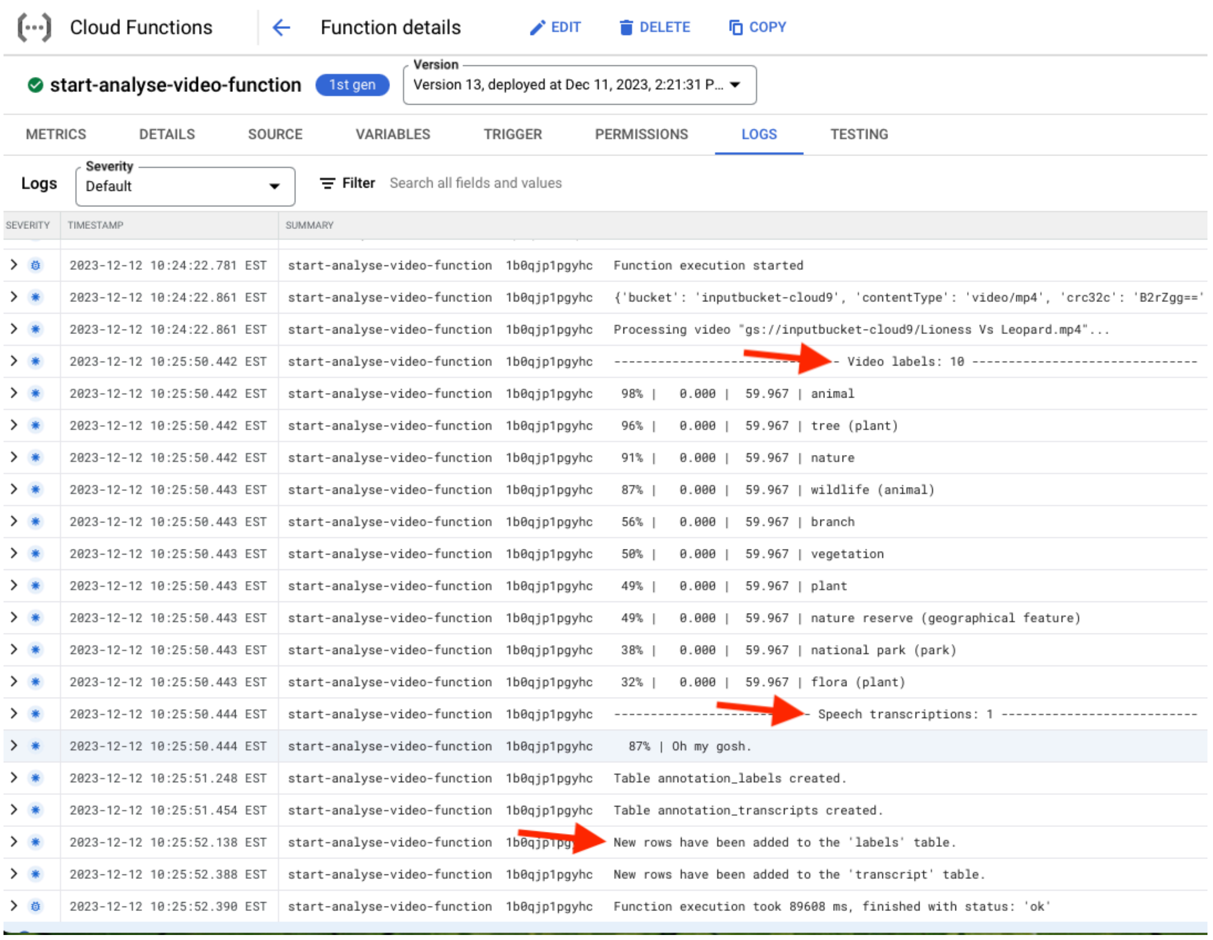
start-analyse-video-function

Version: Version 13, deployed at Dec 11, 2023, 2:21:31 P...

Severity: Default

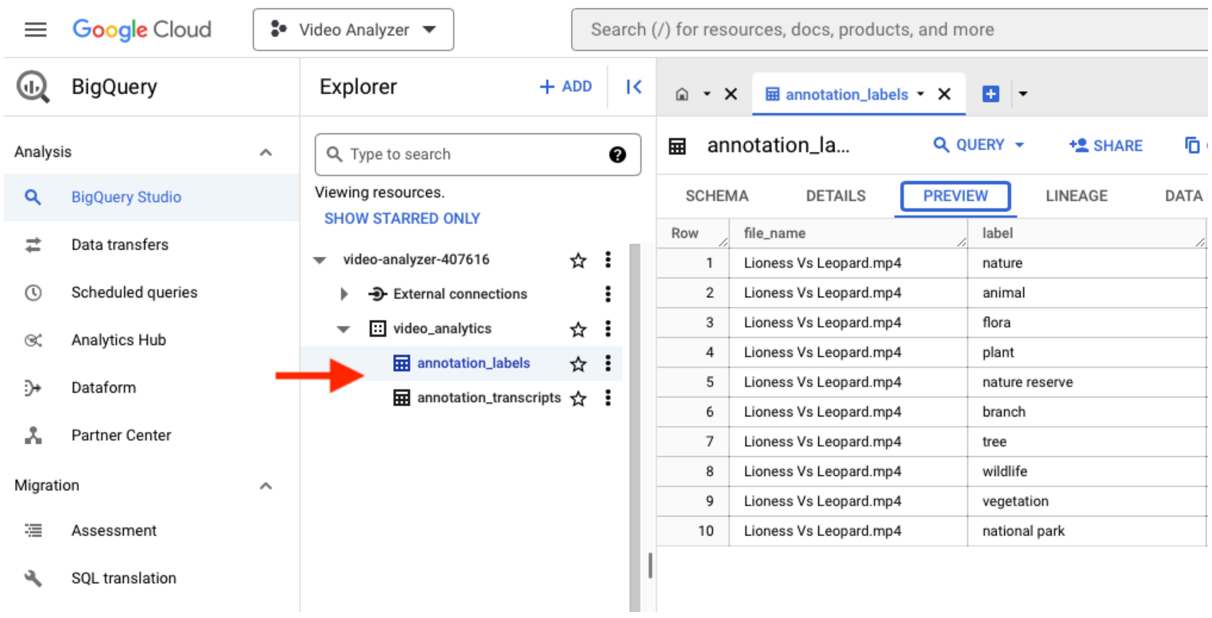
```
2023-12-12 10:24:22.781 EST   start-analyse-video-function  1b0qjp1pgyhc  Function execution started
2023-12-12 10:24:22.861 EST   start-analyse-video-function  1b0qjp1pgyhc  {'bucket': 'inputbucket-cloud9', 'contentType': 'video/mp4', 'crc32c'
2023-12-12 10:24:22.861 EST   start-analyse-video-function  1b0qjp1pgyhc  Processing video "gs://inputbucket-cloud9/Lioness Vs Leopard.mp4"...
```

8. You can observe the full log for demo purposes; in the production environment, we will remove all these detailed logs and add error-code-based complex logs for security purposes. In the below logs, you can see 10 video labels were identified as per our video, and one speech was also extracted (since it is a wild-life video, there are not many dialogs). In the latter logs, you can see the label data has been successfully saved to BigQuery from the logs.

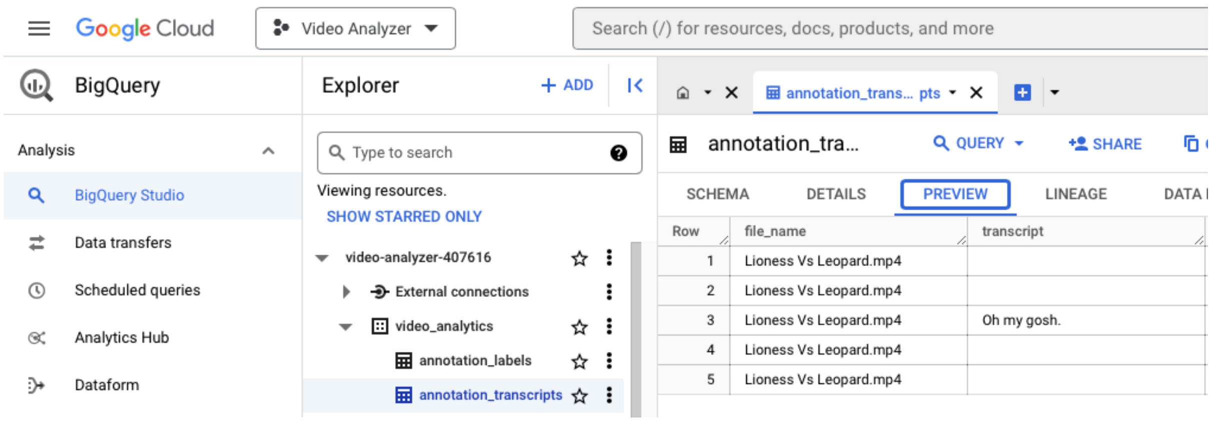


```
2023-12-12 10:24:22.781 EST  start-analyse-video-function  1b0qjp1pgyhc  Function execution started
2023-12-12 10:24:22.861 EST  start-analyse-video-function  1b0qjp1pgyhc  {'bucket': 'inputbucket-cloud9', 'contentType': 'video/mp4', 'crc32c': 'B2rZgg=='
2023-12-12 10:24:22.861 EST  start-analyse-video-function  1b0qjp1pgyhc  Processing video "gs://inputbucket-cloud9/Lioness Vs Leopard.mp4"...
2023-12-12 10:25:50.442 EST  start-analyse-video-function  1b0qjp1pgyhc  ------------------------------ Video labels: 10 ------------------------------
2023-12-12 10:25:50.442 EST  start-analyse-video-function  1b0qjp1pgyhc   98% |   0.000 |  59.967 | animal
2023-12-12 10:25:50.442 EST  start-analyse-video-function  1b0qjp1pgyhc   96% |   0.000 |  59.967 | tree (plant)
2023-12-12 10:25:50.442 EST  start-analyse-video-function  1b0qjp1pgyhc   91% |   0.000 |  59.967 | nature
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   87% |   0.000 |  59.967 | wildlife (animal)
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   56% |   0.000 |  59.967 | branch
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   50% |   0.000 |  59.967 | vegetation
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   49% |   0.000 |  59.967 | plant
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   49% |   0.000 |  59.967 | nature reserve (geographical feature)
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   38% |   0.000 |  59.967 | national park (park)
2023-12-12 10:25:50.443 EST  start-analyse-video-function  1b0qjp1pgyhc   32% |   0.000 |  59.967 | flora (plant)
2023-12-12 10:25:50.444 EST  start-analyse-video-function  1b0qjp1pgyhc  ------------------------- Speech transcriptions: 1 -------------------------
2023-12-12 10:25:50.444 EST  start-analyse-video-function  1b0qjp1pgyhc    87% | Oh my gosh.
2023-12-12 10:25:51.248 EST  start-analyse-video-function  1b0qjp1pgyhc  Table annotation_labels created.
2023-12-12 10:25:51.454 EST  start-analyse-video-function  1b0qjp1pgyhc  Table annotation_transcripts created.
2023-12-12 10:25:52.138 EST  start-analyse-video-function  1b0qjp1pg     New rows have been added to the 'labels' table.
2023-12-12 10:25:52.388 EST  start-analyse-video-function  1b0qjp1pgyhc  New rows have been added to the 'transcript' table.
2023-12-12 10:25:52.390 EST  start-analyse-video-function  1b0qjp1pgyhc  Function execution took 89608 ms, finished with status: 'ok'
```

9. Nope, open the BigQuery window and refresh the data. You can see two new tables have been created, data has been populated, and you can clearly see that the same data from the logs has been saved.



Row	file_name	label
1	Lioness Vs Leopard.mp4	nature
2	Lioness Vs Leopard.mp4	animal
3	Lioness Vs Leopard.mp4	flora
4	Lioness Vs Leopard.mp4	plant
5	Lioness Vs Leopard.mp4	nature reserve
6	Lioness Vs Leopard.mp4	branch
7	Lioness Vs Leopard.mp4	tree
8	Lioness Vs Leopard.mp4	wildlife
9	Lioness Vs Leopard.mp4	vegetation
10	Lioness Vs Leopard.mp4	national park



Row	file_name	transcript
1	Lioness Vs Leopard.mp4	
2	Lioness Vs Leopard.mp4	
3	Lioness Vs Leopard.mp4	Oh my gosh.
4	Lioness Vs Leopard.mp4	
5	Lioness Vs Leopard.mp4	

This ends the setup and running of our project. Have fun experimenting further. Below is the Python code for the cloud function.

Code

```python
import time
import os

from google.cloud import videointelligence as vi
from typing import Optional, Sequence, cast
from google.cloud import bigquery
from google.cloud.exceptions import NotFound
from google.cloud.exceptions import GoogleCloudError

OUTPUT_BUCKET = "gs://outputbucket-cloud9"
PROJECT_ID = "video-analyzer-407616"
DATASET_ID = "video_analytics"
TABLE_ID_LABELS = "annotation_labels"
TABLE_ID_TRANSCRIPT = "annotation_transcripts"

features = [
    vi.Feature.OBJECT_TRACKING,
    vi.Feature.LABEL_DETECTION,
    vi.Feature.SHOT_CHANGE_DETECTION,
    vi.Feature.SPEECH_TRANSCRIPTION,
    vi.Feature.LOGO_RECOGNITION,
    vi.Feature.EXPLICIT_CONTENT_DETECTION,
    vi.Feature.TEXT_DETECTION,
    vi.Feature.FACE_DETECTION,
    vi.Feature.PERSON_DETECTION,
]

speech_config = vi.SpeechTranscriptionConfig(
    language_code="en-US",
    enable_automatic_punctuation=True,
)

person_config = vi.PersonDetectionConfig(
    include_bounding_boxes=True,
    include_attributes=False,
    include_pose_landmarks=True,
)

face_config = vi.FaceDetectionConfig(
    include_bounding_boxes=True,
    include_attributes=True,
)

video_context = vi.VideoContext(
    speech_transcription_config=speech_config,
    person_detection_config=person_config,
    face_detection_config=face_config,
)
```

```
def analyze_video(event, context):
    print(event)
    input_uri = "gs://" + event["bucket"] + "/" + event["name"]
    file_stem = event["name"].split(".")[0]
    output_uri = f"{OUTPUT_BUCKET}/{file_stem}.json"
    request = {
        "features": features,
        "input_uri": input_uri,
        "output_uri": output_uri,
        "video_context": video_context,
    }
    print(f'Processing video "{input_uri}"...')
    video_client = vi.VideoIntelligenceServiceClient()
    operation = video_client.annotate_video(request)
    response = cast(vi.AnnotateVideoResponse, operation.result())
    results = response.annotation_results
    return results

def sorted_by_first_segment_confidence(labels: Sequence[vi.LabelAnnotation],) ->
Sequence[vi.LabelAnnotation]:
    return sorted(labels, key=lambda label: label.segments[0].confidence,
reverse=True)

def category_entities_to_str(category_entities: Sequence[vi.Entity]) -> str:
    if not category_entities:
        return ""
    entities = ", ".join([e.description for e in category_entities])
    return f" ({entities})"

def print_video_labels(results: vi.VideoAnnotationResults):
    labels = sorted_by_first_segment_confidence(results.segment_label_annotations)
    print(f" Video labels: {len(labels)} ".center(80, "-"))
    for label in labels:
        categories = category_entities_to_str(label.category_entities)
        for segment in label.segments:
            confidence = segment.confidence
            t1 = segment.segment.start_time_offset.total_seconds()
            t2 = segment.segment.end_time_offset.total_seconds()
            print(f"{confidence:4.0%} | {t1:7.3f} | {t2:7.3f} |
{label.entity.description}{categories}")

def print_video_speech(results: vi.VideoAnnotationResults, min_confidence: float =
0.8):
    def keep_transcription(transcription: vi.SpeechTranscription) -> bool:
        return min_confidence <= transcription.alternatives[0].confidence

    transcriptions = results.speech_transcriptions
    transcriptions = [t for t in transcriptions if keep_transcription(t)]

    print(f" Speech transcriptions: {len(transcriptions)} ".center(80, "-"))
    for transcription in transcriptions:
        first_alternative = transcription.alternatives[0]
        confidence = first_alternative.confidence
        transcript = first_alternative.transcript
        print(f" {confidence:4.0%} | {transcript.strip()}")

def create_bigquery_tables():
    client = bigquery.Client()
    # Define schema for the first table
    schema_labels = [
        bigquery.SchemaField("file_name", "STRING", mode="REQUIRED"),
        bigquery.SchemaField("label", "STRING", mode="REQUIRED"),
        bigquery.SchemaField("confidence", "FLOAT", mode="NULLABLE"),
        bigquery.SchemaField("start_time", "FLOAT", mode="NULLABLE"),
        bigquery.SchemaField("end_time", "FLOAT", mode="NULLABLE"),
        bigquery.SchemaField("file_uri", "STRING", mode="REQUIRED"),
    ]
    # Define schema for the second table
    schema_transcript = [
        bigquery.SchemaField("file_name", "STRING", mode="REQUIRED"),
        bigquery.SchemaField("transcript", "STRING", mode="REQUIRED"),
        bigquery.SchemaField("confidence", "FLOAT", mode="NULLABLE"),
    ]

    # Create the labels table
    table_ref_labels = client.dataset(DATASET_ID).table(TABLE_ID_LABELS)
    try:
        client.get_table(table_ref_labels)
        print("Table {} already exists. Skipping
creation.".format(table_ref_labels.table_id))
    except NotFound:
        table_labels = bigquery.Table(table_ref_labels, schema=schema_labels)
        table_labels = client.create_table(table_labels)
        print("Table {} created.".format(table_labels.table_id))

    # Create the transcript table
    table_ref_transcript = client.dataset(DATASET_ID).table(TABLE_ID_TRANSCRIPT)
    try:
        client.get_table(table_ref_transcript)
        print("Table {} already exists. Skipping
creation.".format(table_ref_transcript.table_id))
    except NotFound:
        table_transcript = bigquery.Table(table_ref_transcript,
schema=schema_transcript)
        table_transcript = client.create_table(table_transcript)
        print("Table {} created.".format(table_transcript.table_id))

def store_results_in_bigquery(video_uri, results):
    results_labels = results[0]
    results_transcript = results[1]
    url_parts = video_uri.split('/')
    file_name = url_parts[-1]
    client = bigquery.Client()
    rows_to_insert_labels = []
    rows_to_insert_transcript = []

    # Prepare data into the 'labels' table
    for label_annotation in results_labels.segment_label_annotations:
        for segment in label_annotation.segments:
            start_time = segment.segment.start_time_offset.total_seconds()
            end_time = segment.segment.end_time_offset.total_seconds()
            confidence = segment.confidence
            rows_to_insert_labels.append({
                "file_name": file_name,
                "label": label_annotation.entity.description,
                "confidence": confidence,
                "start_time": start_time,
                "end_time": end_time,
                "file_uri": video_uri,
            })

    # Prepare data into the 'transcript' table
    transcriptions = results_transcript.speech_transcriptions
    for transcription in transcriptions:
        first_alternative = transcription.alternatives[0]
        confidence = first_alternative.confidence
        transcript = first_alternative.transcript
        rows_to_insert_transcript.append({
            "file_name": file_name,
            "transcript": transcript,
            "confidence": confidence,
        })

    try:
        # Insert into the 'labels' table
        table_labels = client.dataset(DATASET_ID).table(TABLE_ID_LABELS)
        annotation_labels = client.insert_rows_json(table_labels,
rows_to_insert_labels)
        if annotation_labels == []:
            print("New rows have been added to the 'labels' table.")
        else:
            print("Encountered errors while inserting rows into 'labels' table:
{}".format(annotation_labels))

        # Insert into the 'transcript' table
        table_transcript = client.dataset(DATASET_ID).table(TABLE_ID_TRANSCRIPT)
        annotation_transcripts = client.insert_rows_json(table_transcript,
rows_to_insert_transcript)
        if annotation_transcripts == []:
            print("New rows have been added to the 'transcript' table.")
        else:
            print("Encountered errors while inserting rows into 'transcript' table:
{}".format(annotation_transcripts))
    except GoogleCloudError as e:
        print("Error inserting rows into BigQuery: {}".format(e))


def process_video(event, context):
    input_uri = "gs://" + event["bucket"] + "/" + event["name"]
    results = analyze_video(event, context)
    print_video_labels(results[0])
    print_video_speech(results[1])
    create_bigquery_tables()
    store_results_in_bigquery(input_uri, results)
```