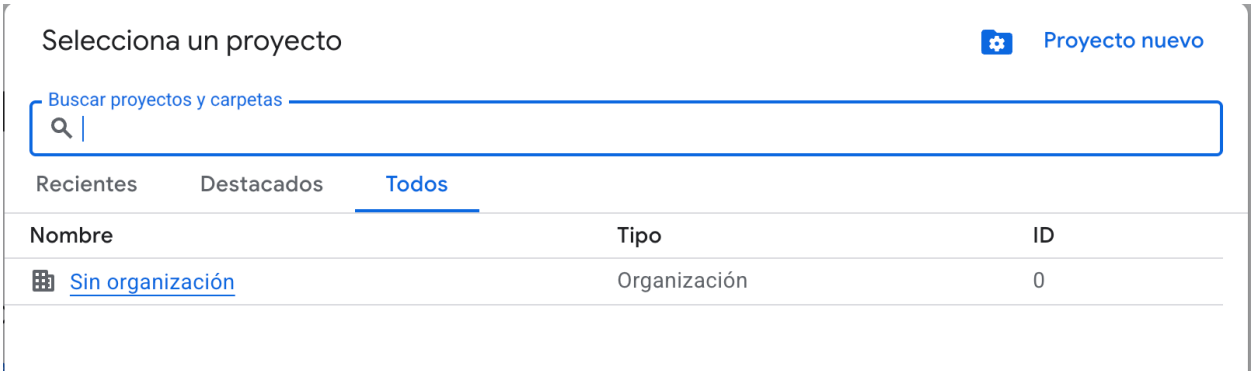
DIA 1. CONFIGURACION AVANZADA DE LA INFRAESTRUCTURA DE GCP.

1. Creación y configuracion del Proyecto:

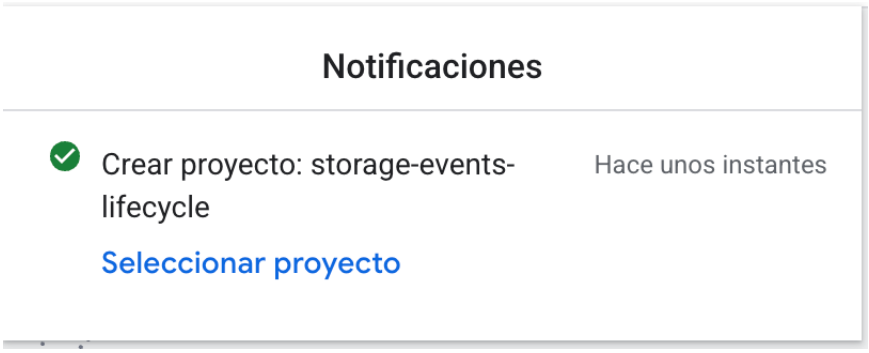
Dirigirse a la URL console.cloud.google.com/.

Para crear un proyecto seleccionar el boton Crear o selecciona un proyecto.

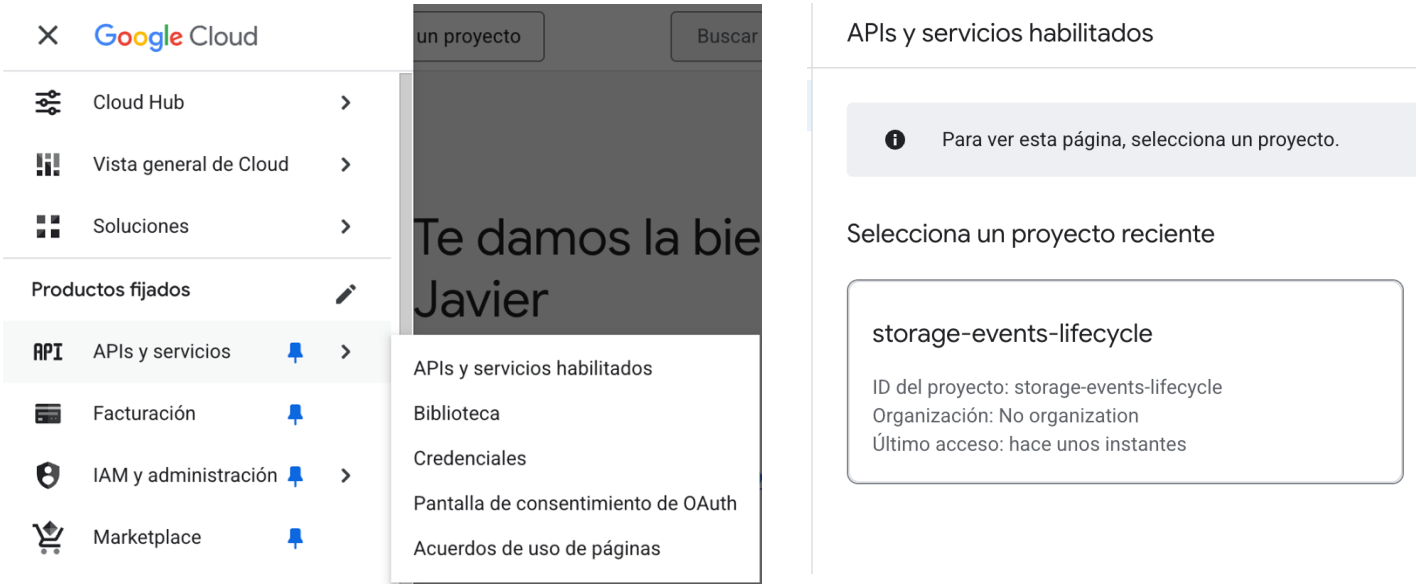
Esto mostra una pestaña en la cual es posible visualizar los proyectos existentes o crear uno nuevo, para crear el nuevo proyecto dar click en proyecto nuevo, siendo el boton mostrado en la esquina superior izquierda.



Al seleccionar ese boton se mostraran las siguientes opciones donde se elegira el nombre del proyecto, al hacerlo y dar clic en el boton de crear proyecto mostrara la siguiente notificacion indicando el tiempo de espera de creacion del proyecto



Ahora seleccionar el menu de navegacion, APIs y servicios, APIs y servicios habilitados; al hacer esto mostrara los proyectos que se tienen disponibles.



Seleccionar el proyecto STORAGE-EVENTS-LIFECYCLE para ahora poder habilitar las APIs: Cloud Storage, Cloud Functions, Cloud Pub/Su, Cloud Logging y Cloud Build. Seleccionar cloud shell (opcion superior izquierda de la pagina) para introducir lo siguiente. Seguido de eso, mostrara un mensaje de “Finalizado exitosamente” con la linea de la operacion

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud services enable storage.googleapis.com \
cloudfunctions.googleapis.com \
pubsub.googleapis.com \
logging.googleapis.com \
cloudbuild.googleapis.com
Operation "operations/acf.p2-196136658496-0ceb1ca5-90af-4724-88e0-79859c8d49ec" finished successfully.
tutosvorago@cloudshell:~ (storage-events-lifecycle)$
```

Dentro del mismo cloud shell se creara el usuario con permisos minimos realizando lo siguiente:

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud iam service-accounts create usu
ario-pruebas \
> --display-name="Usuario de pruebas"
```

Con el usuario ya creado se procede a añadir sus respectivos permisos de la siguiente manera:

Utilizar el siguiente comando para crear una variable donde se guardara el ID del proyecto

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ PROJECT_ID=$(gcloud config get-value p
roject)
Your active configuration is: [cloudshell-22157]
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ echo $PROJECT_ID
storage-events-lifecycle
```

Escribir el siguiente comando para dar los permisos de subir archivos al bucket

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud projects add-iam-policy-binding
 $PROJECT_ID \
> --member="serviceAccount:usuario-pruebas@$PROJECT_ID.iam.gserviceaccount.com" \
> --role="roles/storage.objectCreator"
Updated IAM policy for project [storage-events-lifecycle].
bindings:
```

Escribir el siguiente comando para dar el permiso de leer metadatos dentro del bucket

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud projects add-iam-policy-binding
 $PROJECT_ID --member="serviceAccount:usuario-pruebas@$PROJECT_ID.iam.gserviceaccount.com"
--role="roles/storage.objectViewer"
Updated IAM policy for project [storage-events-lifecycle].
```

Escribir el siguiente comando para dar el permiso de las funciones sean ejecutadas.

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud projects add-iam-policy-binding
 $PROJECT_ID --member="serviceAccount:usuario-pruebas@$PROJECT_ID.iam.gserviceaccount.com"
--role="roles/cloudfunctions.invoker"
Updated IAM policy for project [storage-events-lifecycle].
```

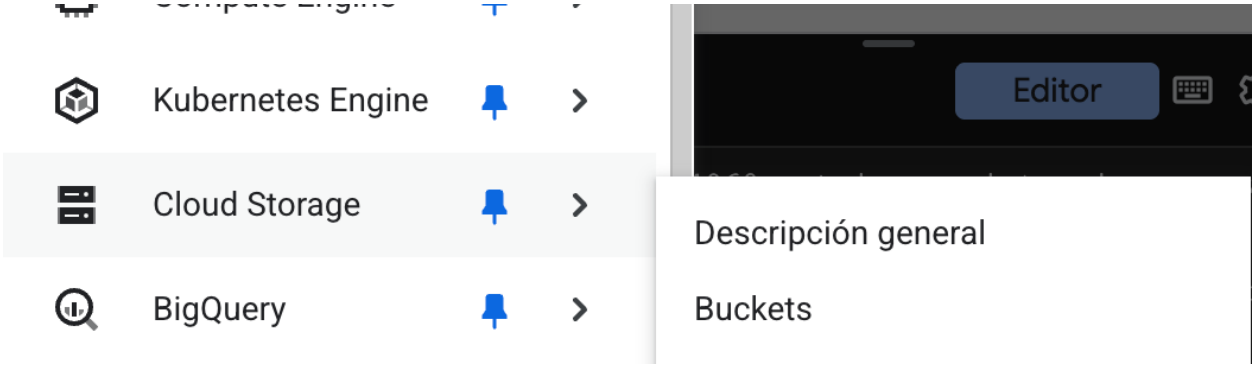
Escribir el siguiente comando para dar el permiso de que la Cloud Function escriba logs.

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud projects add-iam-policy-binding
 $PROJECT_ID --member="serviceAccount:usuario-pruebas@$PROJECT_ID.iam.gserviceaccount.com"
--role="roles/logging.logWriter"
Updated IAM policy for project [storage-events-lifecycle].
```

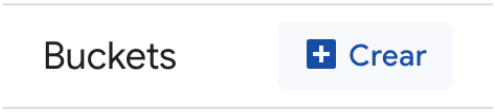
2. Diseño de Almacenamiento y Automatización Inicial:

Crear un bucket en cloud storage.

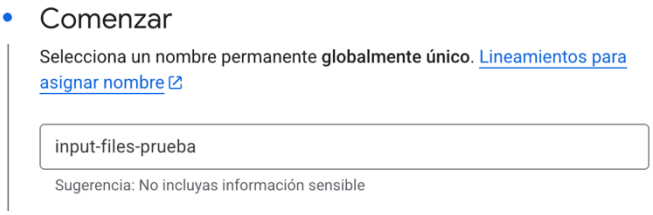
Se creara un bucket, para ello ir a Menu de navegacion -> Cloud Storage -> Buckets



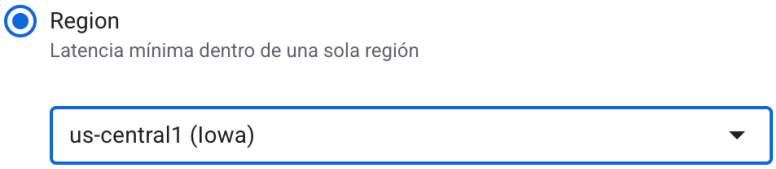
Dar click en crear buckets.



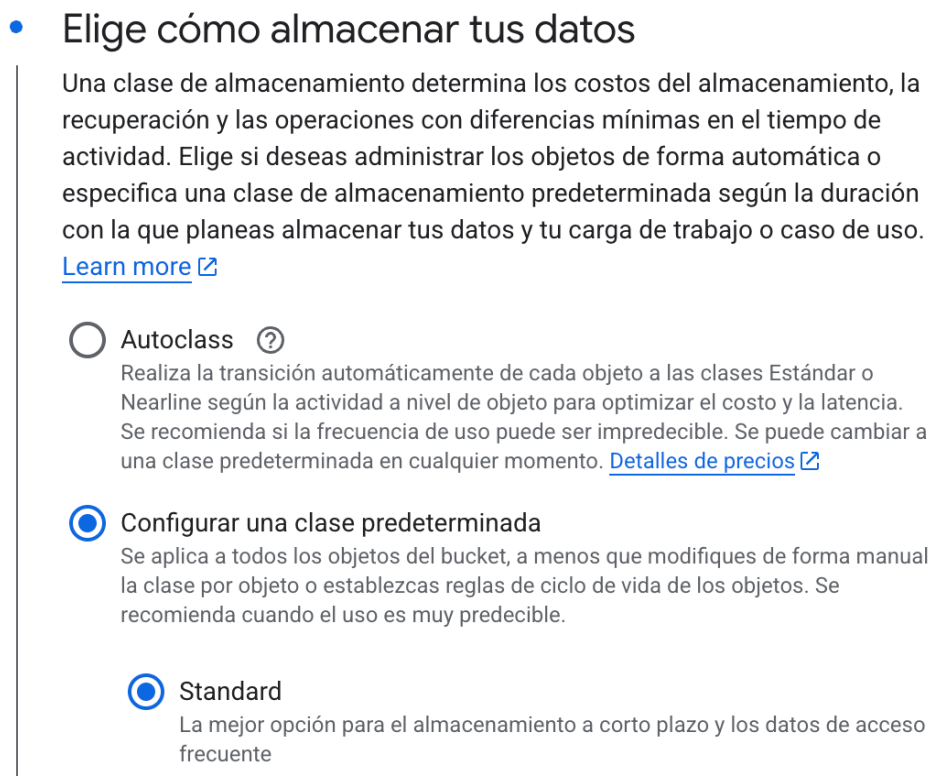
Ahora introducir el nombre, en este caso INPUT-FILES-PRUEBA



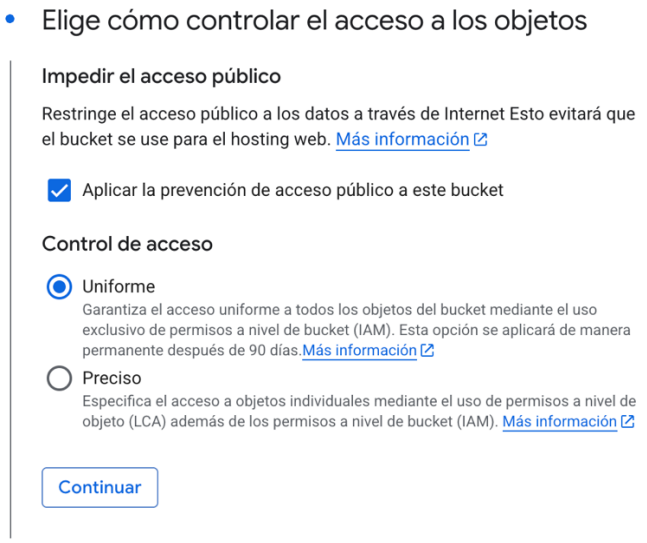
Seleccionar la region US-CENTRAL1 del bucket



Seleccionar la manera de almacenar en estandar

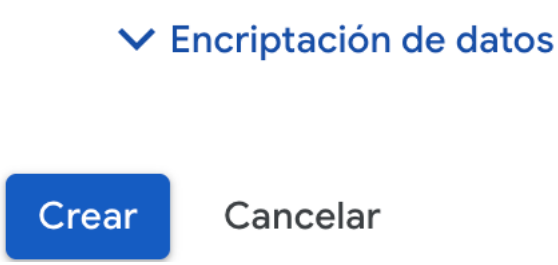


Dejar la configuracion como muestra en la opcion de como controlar el acceso a los objetos



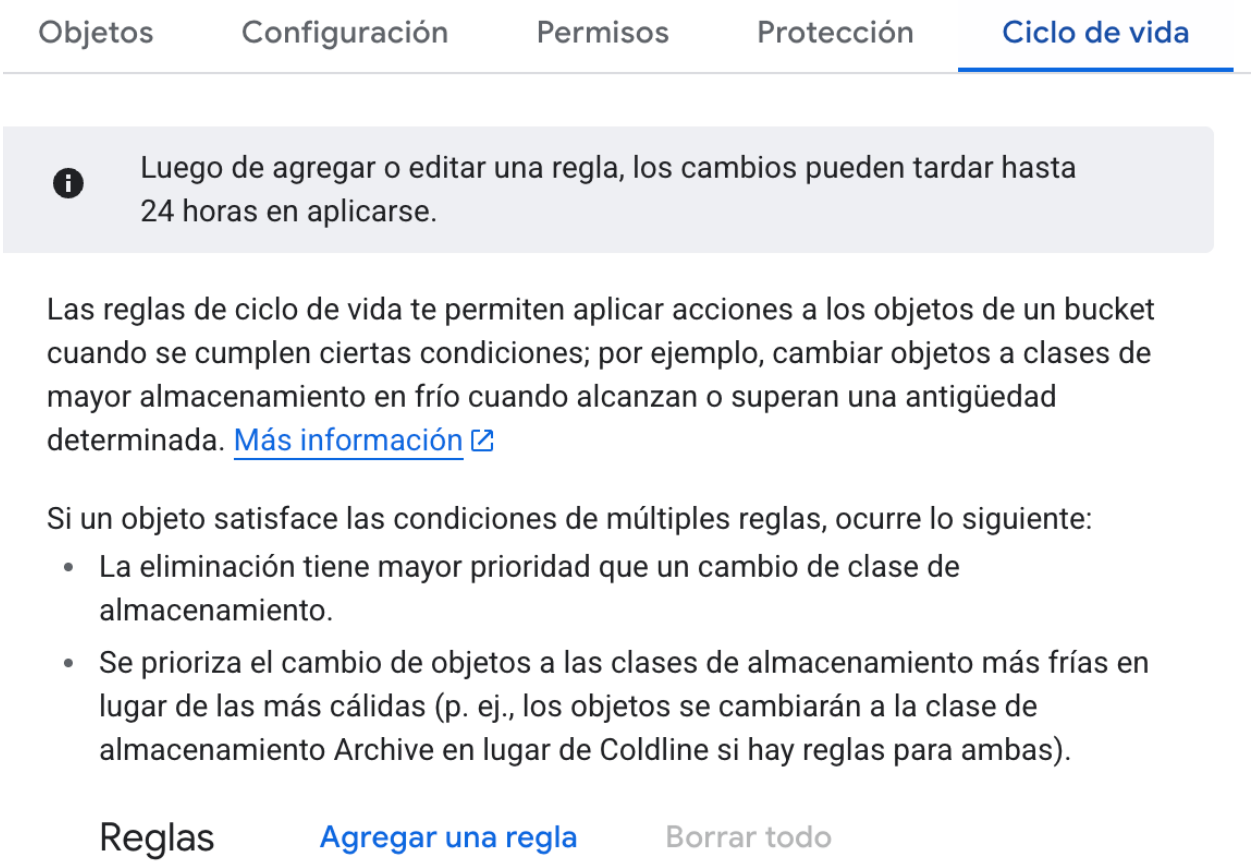
Y ahora desmarcar la POLITICA DE ELIMINACION NO DEFINITIVA

Y por ultimo dar click en el boton Crear

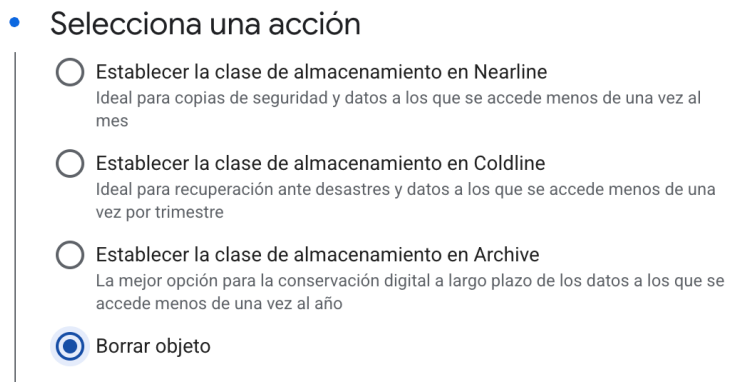


Al seleccionar el boton crear se mostrara la siguiente ventana, sin necesidad de seleccionar algo mas, dar click en confirmar.

En la interfaz ahí mismo en el bucket, seleccionar Ciclo de vida (Lifecycle) -> Agregar regla (Add rule) -> Borrar objeto (Delete) -> Edad (Age) = 30

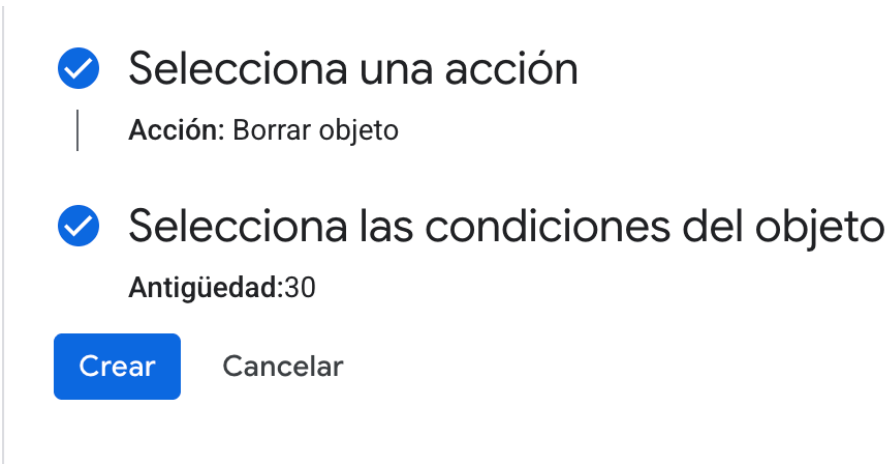


Al seleccionar agregar una regla, se mostrara lo siguiente, para siguiente seleccionar la opcion borrar objeto

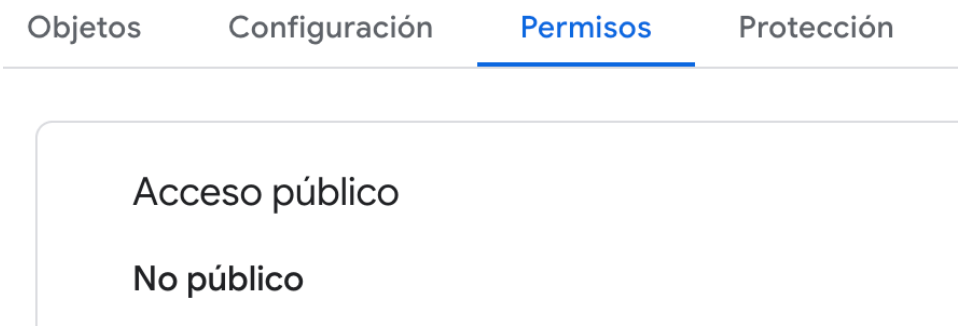


En las condiciones del objeto que sera la opcion siguiente a seleccionar, marcar la casilla antigüedad y añadir 30

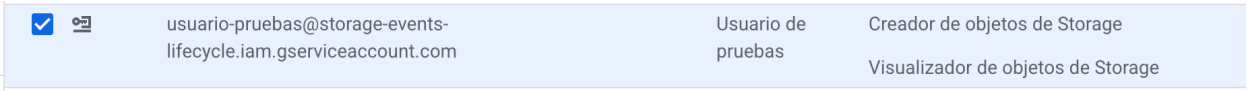
Seleccionar el boton de crear, basicamente borrara archivos despues de 30 días.



Ahora, despues de crear la regla, seleccionar la opcion Permisos en los detalles de bucket con el objetivo de aplicar minimo privilegio al bucket.

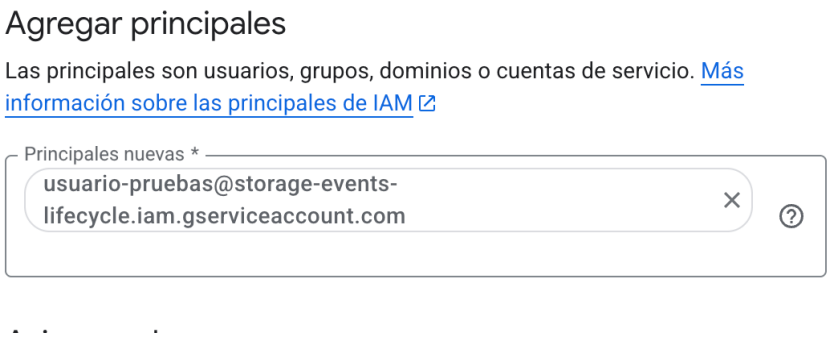


Marcar al usuario de prubeas que aparece en la seccion de permisos



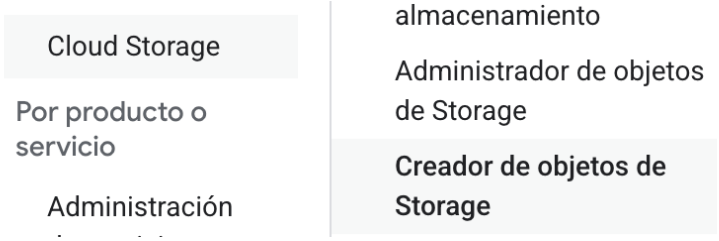
Seleccionar la opcion otorgar acceso

Seleccionar el service account

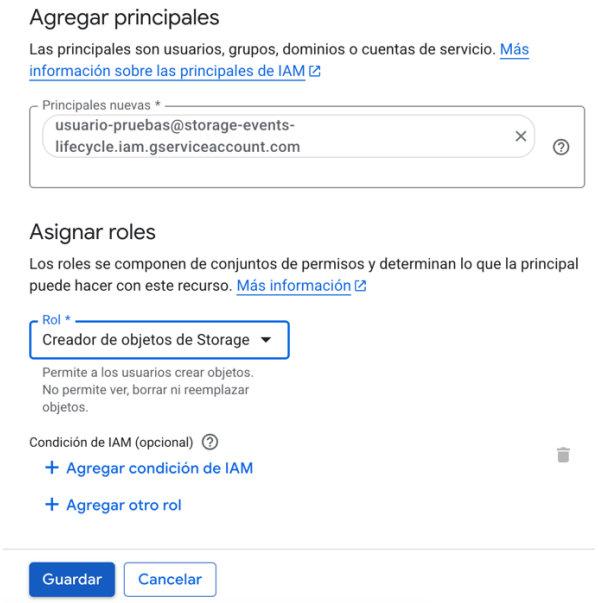


Ahora sigue el momento de asignar los roles siendo 2 los que se buscan, storage object creator y storage object viewer

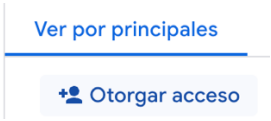
Seleccionar Cloud Storage -> Creador de objetos de Storage



Despues de hacer eso, se mostrara lo siguiente, solo sera necesario dar click en guardar



Nuevamente otorgamos al mismo usuario acceso para agregar otro rol



Seleccionar el mismo principal de la anterior con el rol Cloud Storage -> Visualizador de objetos de Storage y click en guardar.

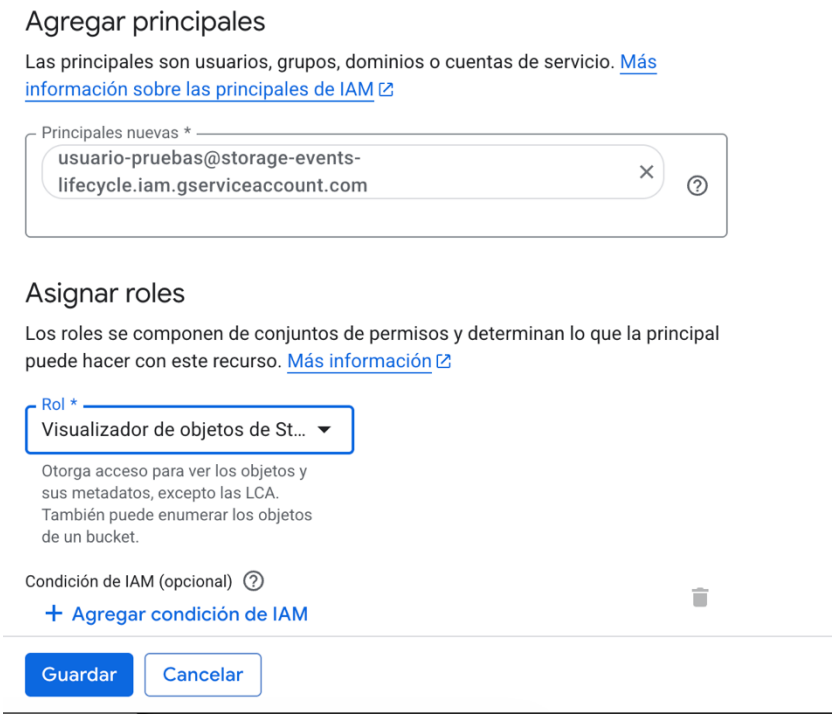
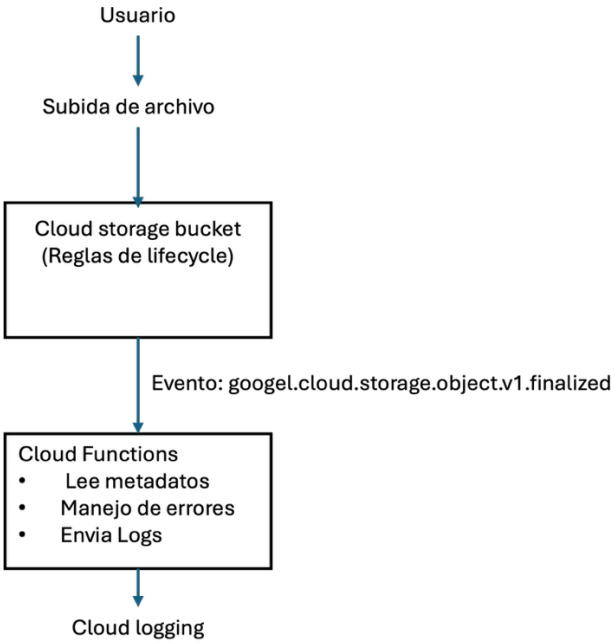


DIAGRAMA REPRESENTATIVO DEL FLUJO COMPLETO



Desarrollar una Cloud Function (en node.js) que se active automaticamente al subir un archivo al bucket.

Para esto, crear un archivo index.js en el Cloud Shell

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ nano index.js
```

En este archivo se ingreso el siguiente codigo

```
exports.onFileUpload = async (event, context) => {
  try {
    // El parámetro "event" contiene los metadatos del objeto que activó la función.
    const file = event;

    // Extraer metadatos principales del archivo subido.
    // En caso de que algún valor no exista, se asigna un valor por defecto.
    const name = file.name || "desconocido";          // Nombre del archivo
    const size = file.size || 0;                      // Tamaño en bytes
    const contentType = file.contentType || "sin-tipo"; // Tipo MIME del archivo

    // Registrar los metadatos en Cloud Logging para auditoría y depuración.
    console.log(`
      Archivo recibido:
      - Nombre: ${name}
      - Tamaño: ${size} bytes
      - Tipo: ${contentType}
    `);

    // La función retorna un objeto para confirmar su ejecución correcta.
    return {
      status: "OK",
      fileName: name,
    };

  } catch (error) {
    // En caso de error, se registra un mensaje en Cloud Logging.
    console.error("Error procesando archivo:", error);

    // Se lanza una excepción para indicar que la ejecución falló.
    throw new Error("Cloud Function failed.");
  }
};
```

Lo siguiente es, en el mismo Cloud Shell se crea un archivo package.json

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ nano package.son
```

E ingresar lo siguiente

package.json > ...

```
{
  "name": "storage-file-processor",
  "version": "1.0.0",
  "main": "index.js",
  "dependencies": {}
}
```

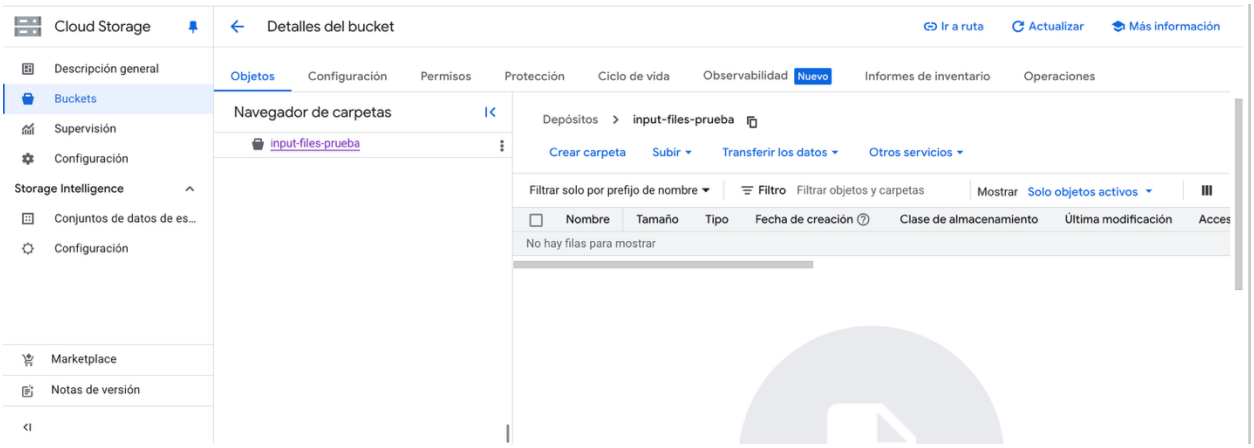
Ahora se procese a realizar el despliegue la funcion

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ gcloud functions deploy onFileUpload \

  --no-gen2 \
  --runtime=nodejs20 \
  --trigger-resource=input-files-prueba \
  --trigger-event=google.storage.object.finalize \
  --entry-point=onFileUpload \
  --region=us-central1

Deploying function (may take a while - up to 2 minutes)...working
```

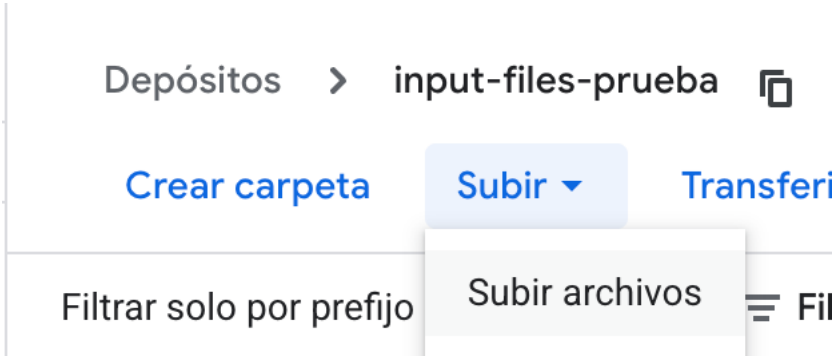
Para saber si se desplego correctamente solo es mirar el status

```
sourceUploadUrl: https://storage.googleapis.com/uploa
ions.appspot.com/5c31b4bd-a1cd-4f3d-b7f0-e2e27b55648
status: ACTIVE
timeout: 60s
updateTime: '2025-11-29T22:07:43.880761854Z'
versionId: '1'
tutosvorago@cloudshell:~ (storage-events-lifecycle)$
```

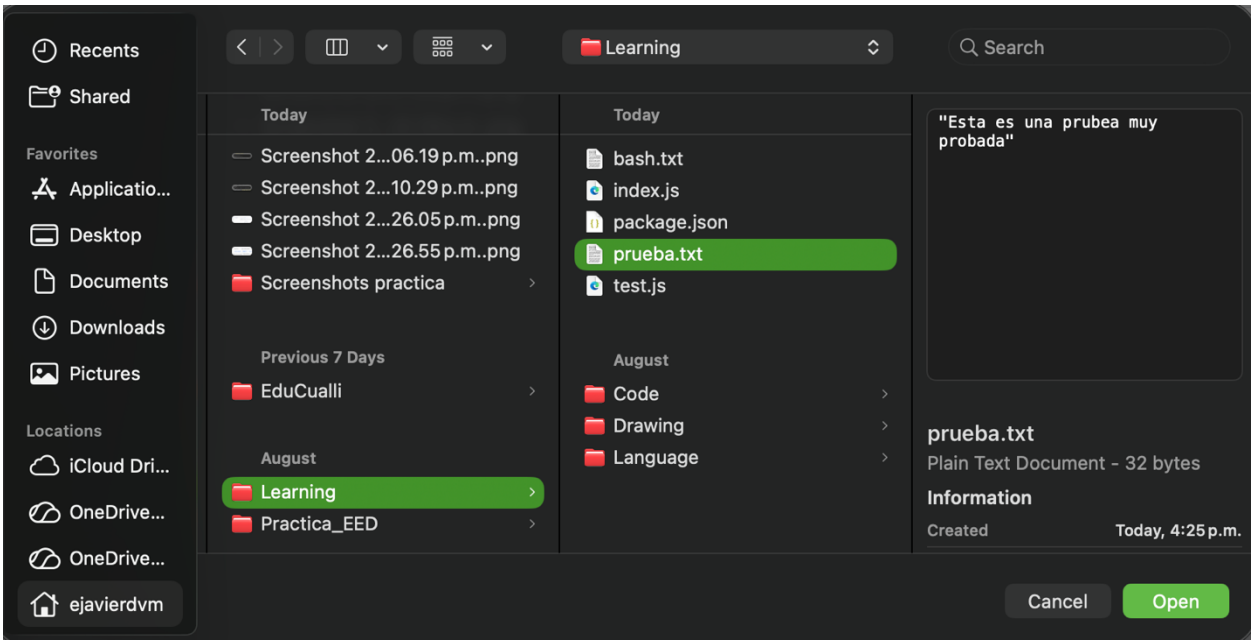
Para probar la funcion, sera subir un archivo llamado prueba.txt siendo dentro del bucket



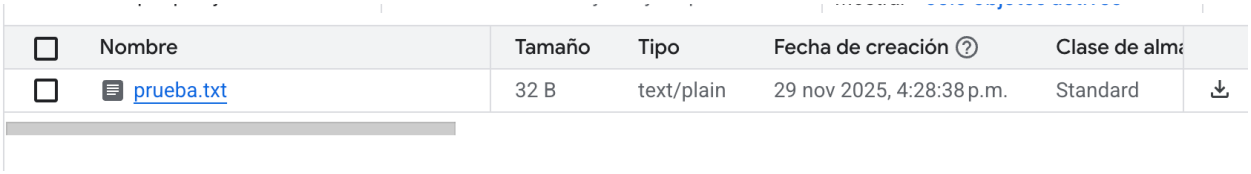
Seleccionar la opcion subir -> subir archivos



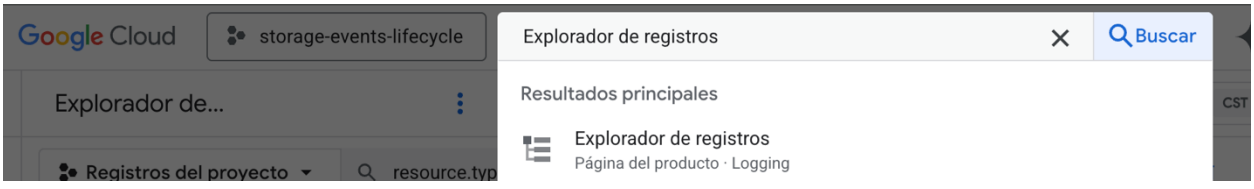
Seleccionar el archivo a utilizar en la computadora, en este caso prueba.txt



El archivo seleccionado ahora esta dentro del bucket



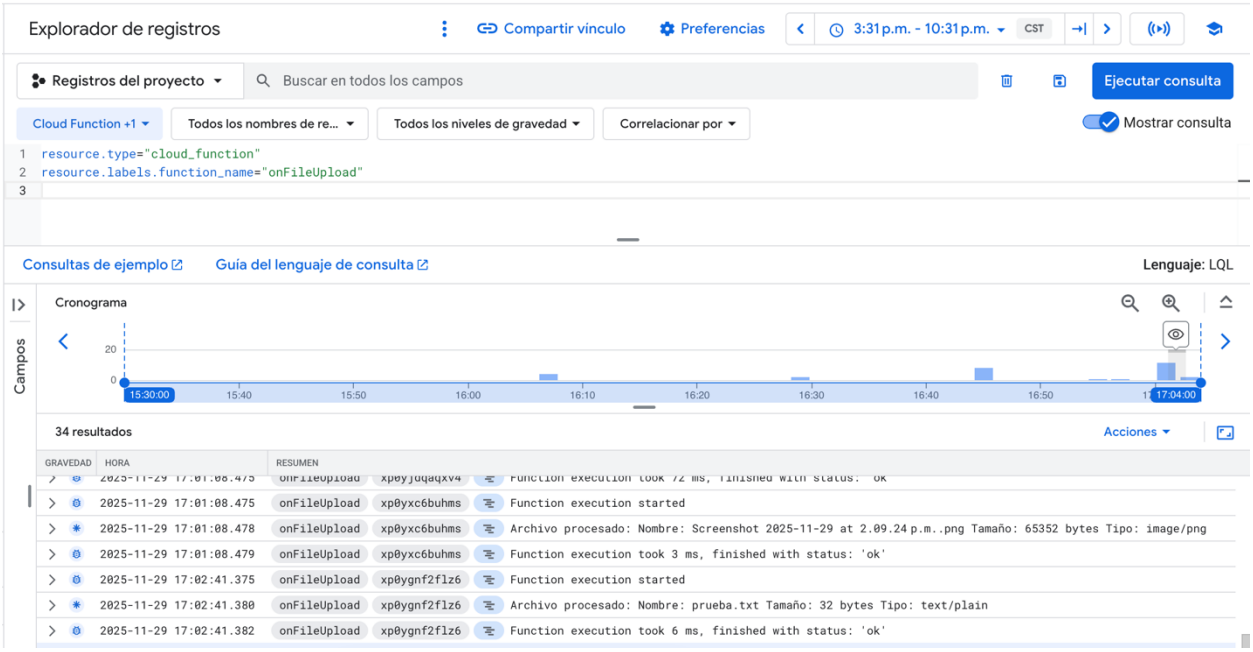
Ahora en el explorador, buscar lo siguiente “Explorador de registros”



Ya dentro del explorador de registros, en la barra de busqueda agregar

y despues dar click en el boton Ejecutar consulta

para seguido mostrar los siguientes datos, donde indica el archivo, su peso y la informacion que se requeria.



Aquí con mas detalle el archivo de prueba.txt que se subio.

3. Pruebas y documentacion

Implementar pruebas basicas unitarias para la Cloud Function.

Instalar Jest como dependencia usando NPM –save-dev jest

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ npm install --save-dev jest
```

Abrir el package.json que se crearon en los pasos anteriores.

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ nano package.json
```

Y colocar lo siguiente dentro del archivo

```
  GNU nano 7.2
{
  "name": "storage-file-processor",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "jest"
  },
  "devDependencies": {
    "jest": "^29.0.0"
  }
}
```

Crear una carpeta para las pruebas

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ mkdir tests
```

Ahora crear el archivo dentro de la carpeta tests

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ nano tests/onFileUpload.test.js
```

agregar lo siguiente dentro del archivo. Lo que se busca es tener un flujo correcto y un flujo de error bien manejado y logueado

```
// Grupo de pruebas para la función "onFileUpload".
describe("Cloud Function onFileUpload", () => {

  // Caso de prueba: Validar que la función procese correctamente
  // un archivo válido simulando el evento enviado por Cloud Storage.
  test("procesa correctamente un archivo válido", async () => {

    // Evento falso (mock) que simula los metadatos de un archivo subido.
    const fakeEvent = {
      name: "archivo-prueba.txt",
      size: 1500,
      contentType: "text/plain",
    };

    // Eje
    const result: any

    const result = await onFileUpload(fakeEvent, {});

    // Validamos que la función retorne un estado OK.
    expect(result.status).toBe("OK");

    // Validamos que el nombre devuelto coincida con el archivo simulado.
    expect(result.fileName).toBe("archivo-prueba.txt");
  });

  // Caso de prueba: Validar el manejo de errores.
  // Si el evento recibido es nulo, la función debe lanzar una excepción.
  test("lanza un error cuando el evento es nulo", async () => {

    // Se espera que la función falle al recibir un evento inválido.
    await expect(onFileUpload(null, {})).rejects.toThrow();
  });
});
```

En el cloud shell se veria asi

```
  GNU nano 7.2                        tests/onFileUpload.test.js *
// Importamos la función desde index.js
const { onFileUpload } = require("../index");

describe("Cloud Function onFileUpload", () => {
  test("procesa correctamente un archivo válido", async () => {
    const fakeEvent = {
      name: "archivo-prueba.txt",
      size: 1500,
      contentType: "text/plain",
    };

    const result = await onFileUpload(fakeEvent, {});

    expect(result.status).toBe("OK");
    expect(result.fileName).toBe("archivo-prueba.txt");
  });

  test("lanza un error cuando el evento es nulo", async () => {
    await expect(onFileUpload(null, {})).rejects.toThrow();
  });
});
```

Entonces, ahora para poder analizar si funciono o no todo esto, es momento de ejecutar las pruebas escribiendo npm test en la raiz donde se tiene package.json mostrando el siguiente resultado.

```
tutosvorago@cloudshell:~ (storage-events-lifecycle)$ npm test

> storage-file-processor@1.0.0 test
> jest

  console.log
    Archivo procesado: Nombre: archivo-prueba.txt Tamaño: 1500 bytes Tipo: text/plain

      at log (index.js:8:13)

  console.error
    Error rpocesando archivo:  TypeError: Cannot read properties of null (reading 'name')
        at name (/home/tutosvorago/index.js:5:23)
        at Object.onFileUpload (/home/tutosvorago/tests/onFileUpload.test.js:19:18)
        at Promise.finally.completed (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:1557:28)
        at new Promise (<anonymous>)
        at callAsyncCircusFn (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:1497:10)
        at _callCircusTest (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:1007:40)
        at processTicksAndRejections (node:internal/process/task_queues:103:5)
        at _runTest (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:947:3)
        at /home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:849:7
        at _runTestsForDescribeBlock (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:862:11)
        at _runTestsForDescribeBlock (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:857:11)
        at run (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:761:3)
        at runAndTransformResultsToJestFormat (/home/tutosvorago/node_modules/jest-circus/build/jestAdapterInit.js:1918:21)
        at jestAdapter (/home/tutosvorago/node_modules/jest-circus/build/runner.js:101:19)
        at runTestInternal (/home/tutosvorago/node_modules/jest-runner/build/index.js:275:16)
        at runTest (/home/tutosvorago/node_modules/jest-runner/build/index.js:343:7)

      13 |
      14 |   } catch (error) {
    > 15 |     console.error("Error rpocesando archivo: ", error);
```

```
      16 |       throw new Error("Cloud Function Failed.");
      17 |
      18 |     }

      at error (index.js:15:13)
      at Object.onFileUpload (tests/onFileUpload.test.js:19:18)

 PASS  tests/onFileUpload.test.js
  Cloud Function onFileUpload
    ✓ procesa correctamente un archivo válido (23 ms)
    ✓ lanza un error cuando el evento es nulo (13 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.437 s
Ran all test suites.
tutosvorago@cloudshell:~ (storage-events-lifecycle)$
```