15 - Add Upload Video Function

00:15

Create Upload Video Function

In this lesson we will create a function that will create a signed URL for uploading a video to Cloud Storage. This is required because our bucket is private, so we need to authenticate the user before they can upload a video.

In our firebase function we can confirm that the user is signed in before generating the URL. Also, we can make the URL short-lived, so that it expires after a certain amount of time.

> Technically the name of this lesson is misleading, but I think it's a better name than "Create Signed URL Function". =)

1. Create Upload Video Function

Install the Google Cloud Storage library (just like in the video processing service):

```
npm install @google-cloud/storage
```

Add the following code to `index.ts`:

```
import {Storage} from "@google-cloud/storage";
import {onCall} from "firebase-functions/v2/https";

const storage = new Storage();

const rawVideoBucketName = "nc-yt-raw-videos";

export const generateUploadUrl = onCall({maxInstances: 1},
async (request) => {
  // Check if the user is authentication
  if (!request.auth) {
    throw new functions.https.HttpsError(
      "failed-precondition",
      "The function must be called while authenticated."
    );
  }

  const auth = request.auth;
  const data = request.data;
  const bucket = storage.bucket(rawVideoBucketName);

  // Generate a unique filename for upload
  const fileName =
`${auth.uid}-${Date.now()}.${data.fileExtension}`;

  // Get a v4 signed URL for uploading file
  const [url] = await bucket.file(fileName).getSignedUrl({
    version: "v4",
    action: "write",
    expires: Date.now() + 15 * 60 * 1000, // 15 minutes
  });
```

```
  return {url, fileName};
});
```

2. Deploy `generateUploadUrl` Function

```
npm install
firebase deploy --only functions:generateUploadUrl
```

> Make sure to run this within the `functions` directory.

With the above command we only deploy the `generateUploadUrl` function, instead of all of our functions.

3. Grant `generateUploadUrl` function access to Cloud Storage

We need our Firebase Functions' Service Account to have access to our raw videos Cloud Storage bucket, so that it can generate signed URLs.

Within the GCP console, navigate to the Firebase Functions. Open the `generateUploadUrl` function and navigate to the `Details` page.

Copy the `service account` email address. It should look something like this: `1234-compute@developer.gserviceaccount.com`.

Navigate to the Cloud Storage console and open the `raw-videos` bucket. Click on the `Permissions` tab and click `Grant Access`.

Paste the service account email address into the `principals` field and grant it the `Cloud Storage` > `Storage Object Admin` role.

Now our Firebase Functions' Service Account has access to our raw videos bucket.

4. Add `Token Creator` Role to `generateUploadUrl` Function

We need to add the `Service Account Token Creator` role to our `generateUploadUrl` function, so that it can generate signed URLs.

Navigate to the GCP IAM console:
https://console.cloud.google.com/iam-admin/iam

Find the same service account that we used in the previous step. Click the edit button on the right side.

Click `Add Another Role` and search for `Token Creator`. Select the

Mark Lesson Complete

by Firebase Authentication, so we will have to test it via the Next.js app.

> The hardest part is always figuring out the permissions issues =)

References

1. Signed URLs:
https://cloud.google.com/storage/docs/access-control/signed-urls
2. Create Signed URL:
https://cloud.google.com/storage/docs/samples/storage-generate-upload-signed-url-v4#storage_generate_upload_signed_url_v4-nodejs
3. Firebase Functions v1 vs v2:
https://firebase.google.com/docs/functions/version-comparison
4. Cloud Functions v1 vs v2:
https://cloud.google.com/functions/docs/concepts/version-comparison