## LESSON 5

In this lesson, we'll connect our video transcoding pipeline in AWS with our website so that website users can view transcoded videos.

We'll do this using an online database service called Firebase. Firebase is a no-SQL database that has rich JavaScript support and can stream data updates directly to user's connected devices using web-sockets.

We'll do this by:

- Creating a Firebase database.
- Connecting our website to Firebase to fetch the URLs of the videos.
- Modifying our existing "transcode-video" lambda function to write to Firebase, so that connected browsers can show that a transcoding operation is taking place.
- Adding a new lambda function "push-transcoded-url-to-firebase", which writes the URLs of videos in S3 to Firebase.

- Configuring the transcoded s3 bucket to trigger this lambda function when a new transcoded video arrives, so that the locations of newly transcoded videos are published to Firebase and made available to users.
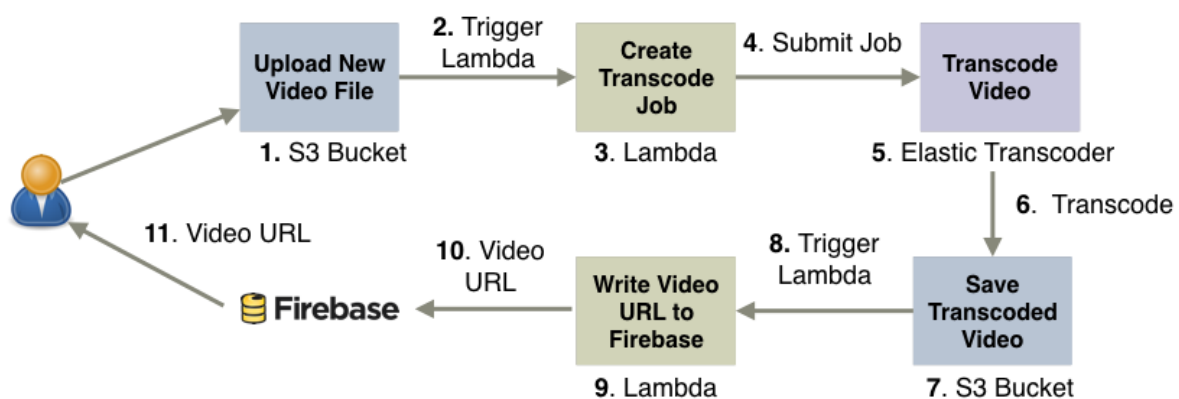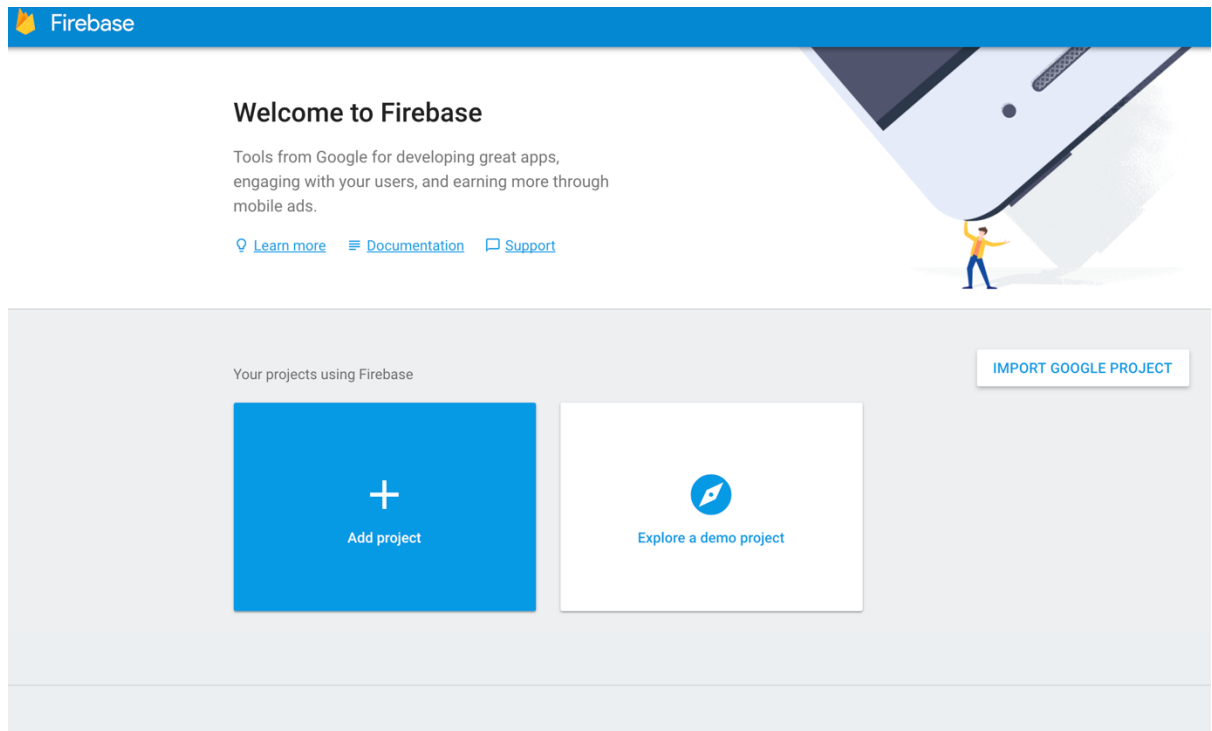

NOTE: PLEASE CREATE ALL YOUR RESOURCES IN THE N. VIRGINIA REGION (US-EAST-1)

1. CREATE A FIREBASE DATABASE

First, you need to create a Firebase account.

- Visit https://www.firebase.com/ to create a free account.
- Click the **Get Started** button.
- Register with your Google Account.
- You will be taken to the console. In the console click **Add Project**.

- Give your project a name like, "serverless-workshop", and click **Create Project**.

- Your project will be created, which comes with a database. Let's check it out. Click **Database** in the left navigation menu. Then click the **Getting started** button in the **Realtime Database** tile

- You'll see you have an empty database. That's OK, we'll add some data later. For now, take note of the database URL. You will need it.
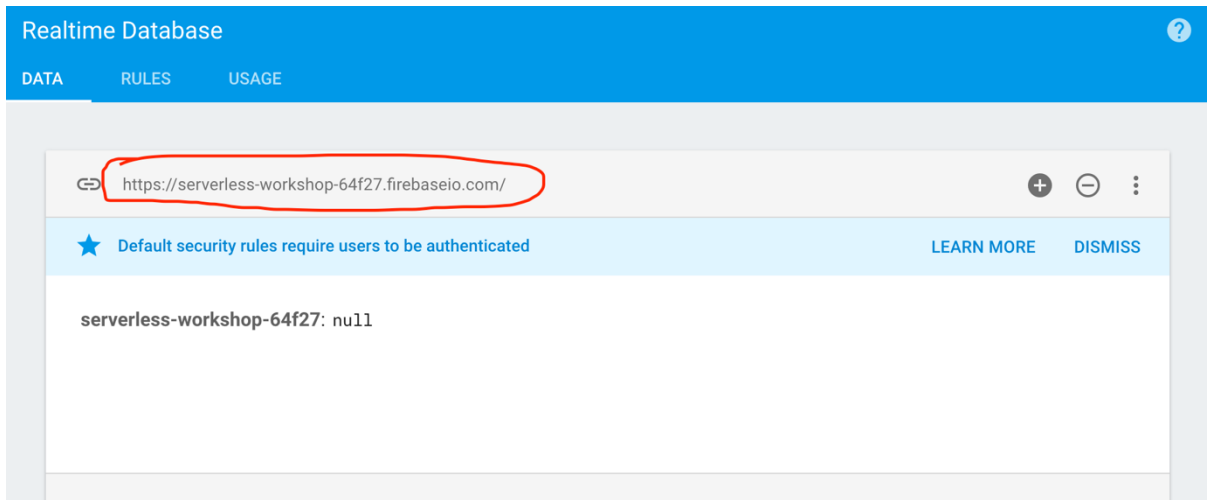
- Click on the **Rules** tab.
- Set ".read" to "true".
- Click the **PUBLISH** button to save.

2.   MODIFY WEBSITE TO ACCESS FIREBASE

Now we're going to connect our web site to Firebase.

- Copy the config.js file containing your account specific settings, from the last lesson.
  Copy lab-4/website/js/config.js to lab-5/website/js/config.js
- Go the **Firebase** console and click on **Project Overview** in the upper left hand corner.
- Click on **Add Firebase to your web app** circle and copy the **config** object.
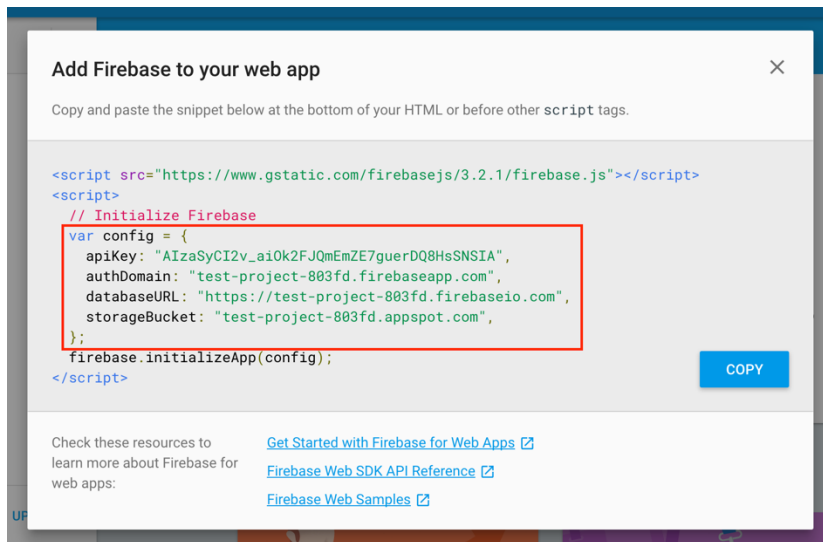
- Paste the config object to **lab-5/website/js/video-controller.js** where it says: **/* PASTE CONFIG HERE */** in connect your website to Firebase.
- In your terminal or command-prompt, go to the following folder:

lab-5/website

- Run the following command to make sure that required npm components are installed:

npm install

- Now run:

npm start

- Open the website in your browser:
  http://localhost:8100

  You should see a blue spinner in the center of the page. This will continue spinning until some videos are found in Firebase, which we'll add next…

3.   TEST WITH SOME SAMPLE DATA

To validate that our web site is connected to Firebase, we'll load some sample data into the Firebase database. This data points to some videos we've already transcoded for you.

- To show off the push-based update features of firebase, make sure the 24-hour video site is open on your screen.
- In another browser window, open your Firebase project's database and go to the **Data** tab, just like you did in Step 1.
- Press the **Hamburger** button in the top right-hand corner of the database section.
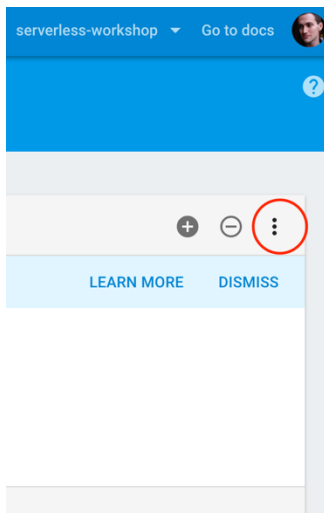
- From the menu select **Import JSON**.
- Upload the json file from the following location:
  lab-5/data/firebase-sample-data.json
- Your Firebase data will now be populated:
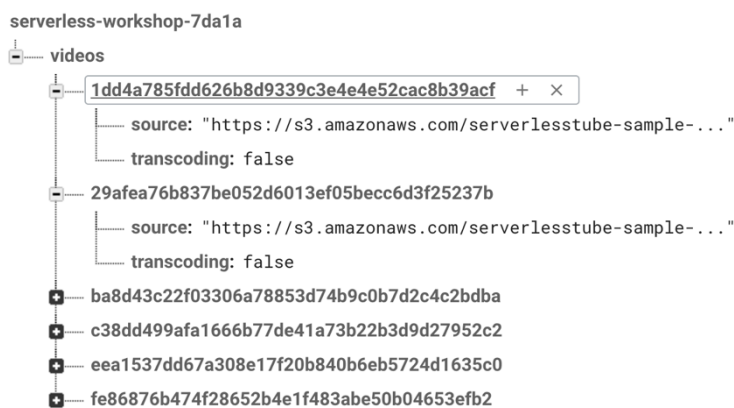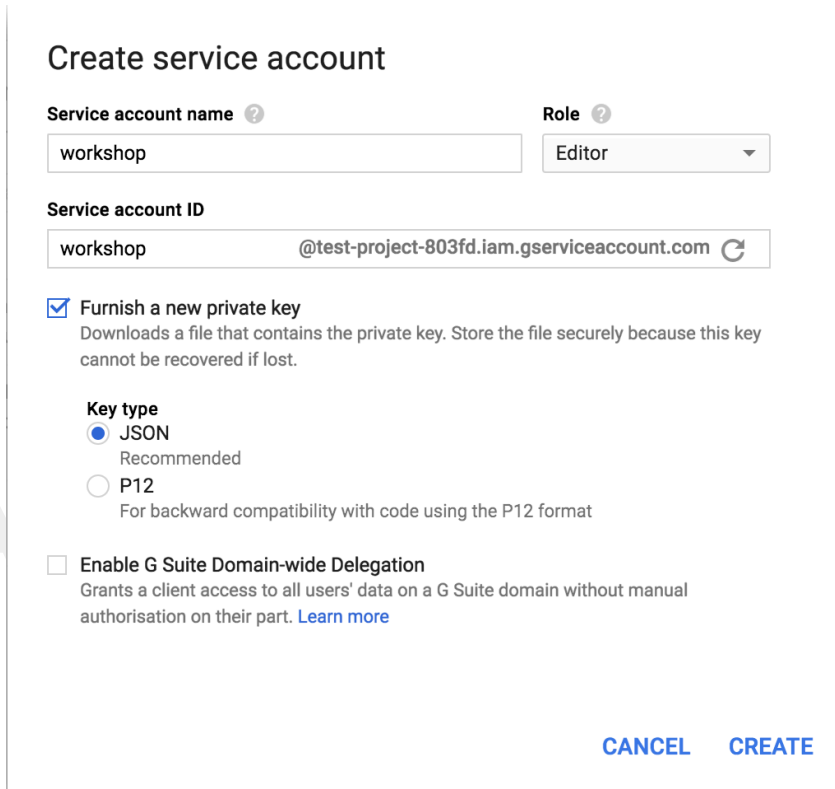
- Your web site will be automatically updated, as Firebase pushed the new data directly to your web browser via web sockets.
- Have a play, modifying the data in Firebase and watching the website update automatically:
  - **Delete** an entry by clicking the **x** icon to the right of the id, and watch it disappear from the website.
  - Change the **transcoding** flag on one of the entries from false to **true**.
    The transcoding flag is used to indicate that a file has been uploaded, but is currently being transcoded. This allows the UI to show an entry for it with an animation showing that something is in progress. Change the flag and watch the UI update to show the transcoding indicator:
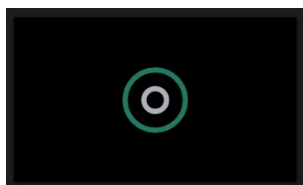
4.	MODIFY VIDEO TRANSCODE LAMBDA FUNCTION FOR FIREBASE

Before we proceed to modify Lambda functions we need to create service accounts in Firebase.

- In **Firebase** click on the **Gear** icon next to **Project Overview** in the left navigation menu, and click **Users and permissions**.
- This will open a different website in a new tab.
- Select **Service Accounts** from the left nav.
- Click the **Create Service Account** button.
- Set a service account name like "workshop".
- From the **Role** dropdown, select **Project** and then **Editor**.
- Click "Furnish a new private key" and make sure that JSON is selected.
- Leave everything else as is and click **create**.

- Click **ok** on the popup that appears
- Copy the JSON file that was downloaded to lab-5/lambda/transcode-video-firebase-enabled/
  Make a note of the filename

Now we're going to modify the existing video-transcode Lambda function, to have it push a new entry into Firebase with **transcoding:** set to **true**. With this in place, the user interface will show a placeholder of a video showing an animation while the video transcodes.

In Lesson 4 we configured file uploads, and this upload system created a unique key for each file that was uploaded (this key was used in the path when the file was stored in S3). We'll have our Lambda function use this key as the unique key for the video in Firebase.

- ZIP up your lambda function

  For OS X / Linux Users
  In the terminal / command-prompt, change to the directory of the function:

  cd lab-5/lambda/transcode-video-firebase-enabled

Install npm packages by typing:

npm install

Now create create a ZIP file of the function, by typing:

npm run predeploy

For Windows

You will need to zip up all the files in the **lab-5/lambda/video-transcoder-firebase-enabled** folder via the Windows Explorer GUI, or using a utility such as 7zip. (**Note: don't zip the video-transcoder-firebase-enabled folder. Zip up the files inside of it**).

- In the AWS console, go to **Lambda**.
- Select the **transcode-video** function.
- Choose **Upload a .ZIP file** from **Code entry type** and click **Upload**.
- Select the ZIP file of the Lambda function you just created.
- Create two more environment variables:
    - **SERVICE_ACCOUNT**: The name of the **JSON Service Account** file you created earlier in this step.
    - **DATABASE_URL:** The URL from the **Database** tab in the Firebase.
- Click the **Save** button at the top of the page to upload the function and save your environment variables.

- Test that your function works by opening the 24-hour video web-site and uploading a video. Within a few seconds of the upload completing, you should see a new entry appear in the video list, showing the transcoding animation. This animation will remain forever, because we haven't yet connected anything to update firebase once the transcoding has completed.

5. CREATE NEW LAMBDA FUNCTION: PUSH-TRANSCODED-URL-TO-FIREBASE

Now we're going to complete the final piece of the system: We're going to add a new lambda function, that will trigger every time a newly transcoded video arrives in the second, transcoded S3 bucket. This lambda function will write the public URL of the video to Firebase (so that the browser can play the video). It will also set **transcoding: false**, indicating that transcoding has completed.

- Open a terminal / command-prompt and navigate to the following folder:

lab-5/lambda/push-transcoded-url-to-firebase

- Install npm packages by typing:

npm install

- **Copy the Firebase service account JSON file**

Copy the Firebase service account file you created in the previous step to */lab-5/lambda/push-transcoded-url-to-firebase*. The JSON file must be included with the Lambda function for it to work.

- **Now ZIP up your lambda function**

For OS X / Linux Users

Now create create a ZIP file of the function, by typing:

npm run predeploy

For Windows

You will need to zip up all the files in the **lab-5/lambda/push-transcoded-url-to-firebase** folder via the Windows Explorer GUI, or using a utility such as 7zip. (**Note: don't zip the push-transcoded-url-to-firebase folder. Zip up the files inside of it**).

- In the AWS console, go to **Lambda** and create a function as before, with the following settings:
    - o **Name**: push-transcoded-url-to-firebase
    - o **Runtime**: Node.js 4.3
    - o **Role**: lambda-s3-execution-role
    - o **Function package**: The .zip file you just created.
    - o **Timeout**: 30 seconds
    - o Environment variables:
        - ▪ **SERVICE_ACCOUNT**: the name of the JSON Service Account file you created earlier
        - ▪ **DATABASE_URL**: the database URL from Firebase, just as you did for the last function
        - ▪ **S3:** the URL of your second (*transcoded*) bucket, e.g. *https://s3.amazonaws.com/serverless-video-transcoded*.
        - ▪ **BUCKET_REGION**: *us-east-1*
- Don't forget to click the **Save** button at the top of the page.

6. MODIFY TRANSCODED VIDEO BUCKET TO TRIGGER NEW LAMBDA FUNCTION

Now we need to configure S3 to invoke the new push-transcoded-url-to-firebase lambda function when a newly transcoded video arrives in the destination bucket:

- In the Lambda console for the push-transcoded-url-to-firebase function you just created, scroll to the **Designer** section
- Click on **S3** in the **Add triggers** list on the left
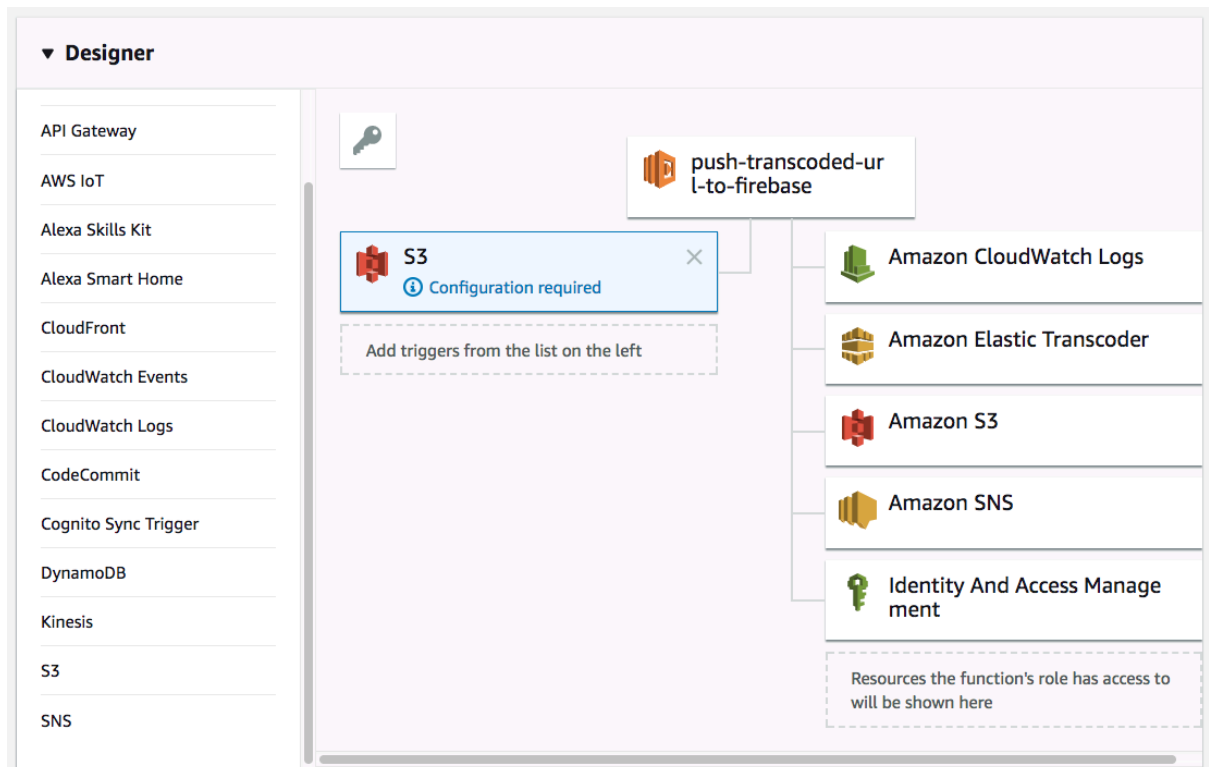
- Scroll down to the **Configure triggers** section
- Select the second bucket (e.g. *serverless-video-transcoded*).
- In the event type dropdown, select **Object Created (All)**.
- Enter a suffix of **.mp4**
  We do this to ensure that the lambda function is only called when new videos arrived. The elastic transcoder may drop other assets in the bucket (such as thumbnails, or JSON files) which should not trigger the lambda function.

Configure triggers

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

acg-sfb-transcoded-bucket ▼

Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

Object Created (All) ▼

Prefix
Enter an optional prefix to limit the notifications to objects with keys that start with matching characters.

e.g. images/

Suffix
Enter an optional suffix to limit the notifications to objects with keys that end with matching characters.

.mp4

Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).
☑

Cancel   Add

- Press **Add,** then click **save** up the top and AWS will link your S3 bucket and Lambda function.

7.   TEST SYSTEM END-TO-END

The system is complete, it's time to test it end-to-end!

- Open the 24-hour video website in your browser.
- Upload a video file. The progress bar will show as the video uploads.
- Once upload completes, a tile will appear in the user interface representing the video. It will contain an animation, indicating that the video is being transcoded.
- Once transcoding is complete, the transcoding animation will be replaced by the video.
- Click on the video to watch it play. You'll notice that this is running in a web-friendly, lower quality 480p format.

Congratulations – you now have completely serverless website that authenticates users, allows them to upload video files, transcodes these videos to a web friendly format and then makes them available to all users of the web site.

Optional Exercises:

1. In the website, move the config object from video-controller.js to the config file.
2. Your website is still just running locally. Create a new S3 bucket with Static Web Hosting enabled for your **test** environment.
3. Update CORS and your firebase database to only allow requests from the origin for your **test** environment.