## LESSON 5

In this lesson, we'll connect our video transcoding pipeline in AWS with our web site so that web site 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 web site to fetch the URLs of videos from Firebase.
- 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.
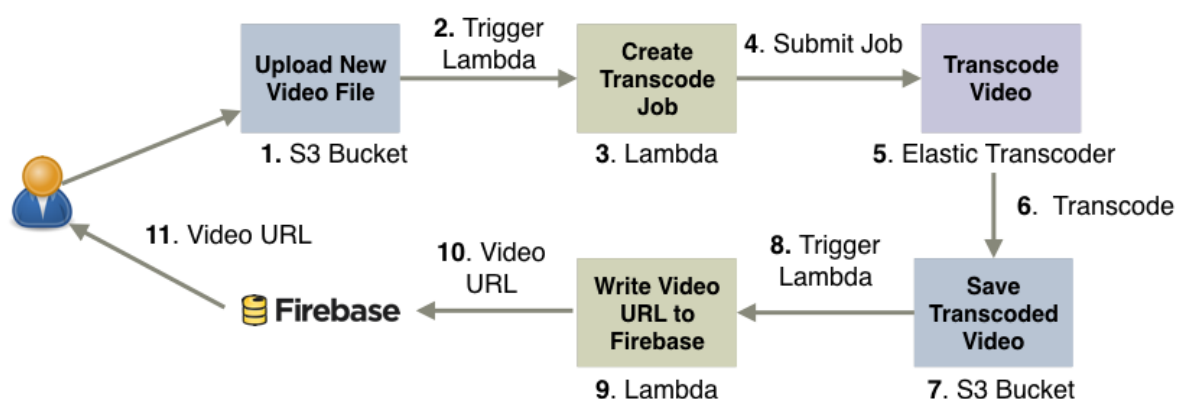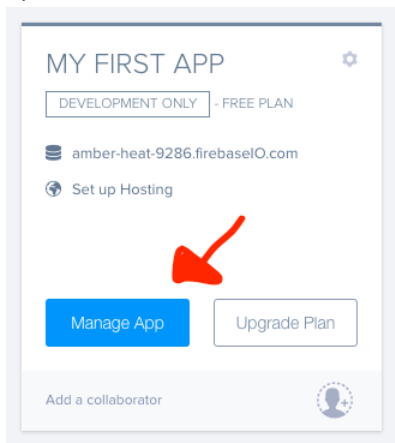

NOTE: PLEASE CREATE ALL YOUR RESOURCES IN THE N. VIRGINIA REGION (US-EAST-1)

1.   CREATE A FIREBASE DATABASE

First, you need to create a Firebase account.

- Visit https://www.firebase.com/ to create a free account.

- When you create an account, Firebase will automatically create an app for you. Click **Manage App** to open the database.

- You'll see you have an empty database. That's OK, we'll add some data later. For now, take note of the URL in your browser, this is your Firebase database URL:
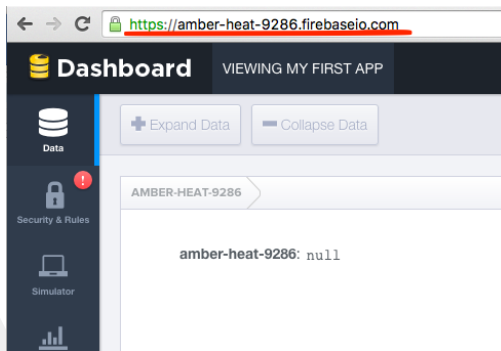
2.   MODIFY WEBSITE TO ACCESS FIREBASE

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 lesson-4/website/js/config.js to lesson-5/website/js/config.js
- Now edit the copied config file to add the following line:
  firebaseUrl: 'https://YOUR-FIREBASE_APP.firebaseio.com'
  Match the value to the firebase URL you read from your browser address bar in the last section.
- In your terminal or command-prompt, change to the following folder:

lesson-5/website

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

npm install

- Now run:

npm start

- Open the web-site 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 URL. You'll see an empty database.
- Press the **Import Data** button in the top right-hand corner of the screen.
- Upload the json file from the following location:
  lesson-5/data/firebase-sample-data.json
- Your firebase data will now be populated:

- Your web site will have 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 web-site update automatically:
  - **Delete** an entry, and watch it disappear from the web-site
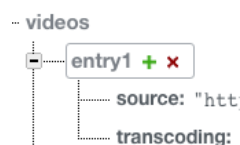
  - Change the **transcoding** flag on one of the entries from false to **true**.
    The transcoding flag is used to indicate that a file has been uploaded, but is currently being transcoded. This allows the UI to show an entry for it with an animation showing that something is in progress. Change the flag and watch the UI update to show the transcoding

indicator:
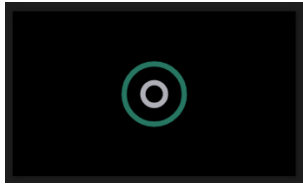
4. MODIFY VIDEO TRANSCODE LAMBDA FUNCTION FOR FIREBASE

Now we're going to modify the existing video-transcode lambda function, to have it push a new entry into Firebase with transcoding: true. With this in place, the user interface will be able to 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 recognize this key and use it as the unique key for the video in Firebase.

- **Copy** the config.js file containing your account specific settings from lesson 1
  Copy **lesson-1/lambda/video-transcoder/config.js** to **lesson-5/lambda/transcode-video-firebase-enabled/config.js**

- Edit the copied config file to add the following lines:
  env.FIREBASE_URL = 'YOUR_FIREBASE_URL_HERE';
  env.FIREBASE_SECRET = 'YOUR_FIREBASE_SECRET_HERE';
  Enter your firebase url, and secret key. You can get the secret key by clicking on the **Secrets** button in the left menu in Firebase:
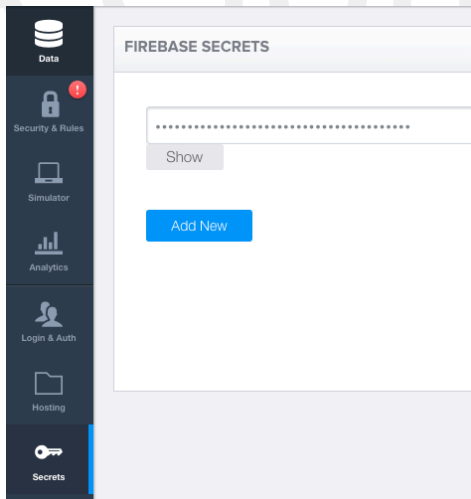
- ZIP up your lambda function

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

  cd lesson-5/lambda/video-transcoder-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 **lesson-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 click **Lambda**, select your '**transcode-video**' function, and click **Upload**:

- Select the ZIP file of the Lambda function you just created.
- Click the **Save** button to upload the function.

We are also going to use nodejs 0.10 for this lambda function to show you how to it works
- Click **Configuration**
- Select **Node.js 0.10** from the Runtime dropdown
- Click **Save**

- Test that this 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 the transcoding has completed.

- **Open** the config.js file in your favourite text editor:
  lesson-5/lambda/push-transcoded-url-to-firebase /config.js

  You'll need to modify the **S3** value to use the name of your transcoded S3 bucket, in the format of:
  https://s3.amazonaws.com/YOUR_TRANSCODED_BUCKET_NAME

  You'll also need to set your **FIREBASE_URL** and **FIREBASE_SECRET**.

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

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

- Install npm packages by typing:

  npm install

- **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 **lesson-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, click **Lambda**, and then click *Create a Lambda Function*.
- Skip over the blueprint.
- **Name** the function **push-transcoded-url-to-firebase**
- Make sure that <u>**Node.js 0.10**</u> is selected in the *Runtime* dropdown.
- Select **Upload a ZIP file.** Choose the zip file you just created:
  /lesson-5/lambda/push-transcoded-url-to-firebase/Lambda-Deployment.zip
- Under Role select *lambda-s3-execution-role*.
- Click *Next* to go the Review screen and from there click *Create function* to finish.

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:

- Open the transcoded bucket (e.g. serverless-video-transcoded) in the AWS console, select **Properties**, expand **Events**, and click *Add Notification*.
- Give your event a **name**, such as **Video Transcoded**, and in the **Events** dropdown select *ObjectCreated (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.
- Click the *Lambda function* radio button, right below it select the **push-transcoded-url-to-firebase** Lambda function from the dropdown and save:

Name	Video Transcoded
Events	ObjectCreated (All) ✕
Prefix	e.g. images/
Suffix	.mp4
Send To	○ SNS topic ○ SQS queue ● Lambda function
Lambda function	push-transcoded-url-to-firebase ▾

S3 will add the necessary permissions to invoke your Lambda function from this source bucket. See the Developer Guide.

Save   **Cancel**

7.   TEST SYSTEM END-TO-END

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

- Open the 24 hour video web site 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 web-site 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.