

## LESSON 4

In this lesson are going to add the ability to upload videos from the browser to an S3 bucket. To do this we are going to:

1. Create a Lambda function to grant us credentials/policy to upload a files to an S3 bucket.
2. Configure API Gateway to allow our website to access this Lambda function and retrieve the necessary policy document.
3. Update the website to request the policy document and upload the file to S3.

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

### 1. CREATE A LAMBDA FUNCTION

You will need to create a new Lambda function in the AWS console. This Lambda function will generate a policy document to allow our users upload videos to S3. Step through the following:

- Click **Lambda** in the AWS Console
- Create a blank new function and skip over configure triggers
- Name the function **get-upload-policy**
- Assign the **lambda-s3-execution-role** policy to it (the same policy created in lesson 1)
- Leave all other settings on default and save

### 2. CREATE IAM USER

The policy and the credentials that we are going to generate in the Lambda function need to be signed by an IAM user that has permissions to upload files to S3. Let's create this user now.

- Open IAM console
- Click **Users** and create a new user called **upload-s3**
- Download the user's access & secret keys. You will need these.
- Click the **upload-s3** user and click the **Permissions** tab
- Expand **Inline Policies**
- Create to create new **Inline Policy**, select **Custom Policy**, and click **Select**
- Set the name of the policy as **upload-policy**
- Copy the following to the Policy Document and save (make sure to specify your upload bucket name in the policy).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::YOUR_UPLOAD_BUCKET_NAME"
      ]
    }
  ],
}
```

```

    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:s3:::YOUR_UPLOAD_BUCKET_NAME/*"
    ]
}
]
}

```

## Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the [IAM Policy Simulator](#).

### Policy Name

### Policy Document

```

5      "Effect": "ALLOW",
6      "Action": [
7          "s3:ListBucket"
8      ],
9      "Resource": [
10         "arn:aws:s3:::serverless-video-upload"
11     ]
12 },
13 {
14     "Effect": "Allow",
15     "Action": [
16         "s3:PutObject"
17     ],
18     "Resource": [
19         "arn:aws:s3:::serverless-video-upload/*"

```

☒ Use autoformatting for policy editing

[Cancel](#)

[Validate Policy](#)

[Apply Policy](#)

## 3. CONFIGURE FUNCTION

Open the Lambda function provided in Lesson 4 on your computer. It's located in **lab-4/lambda/create-s3-upload-policy-document**.

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

```
lab-4/lambda/create-s3-upload-policy-document
```

- Install npm packages by typing:

```
npm install
```

- Zip Lambda function**

For OS X / Linux Users

Now 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-4/lambda/create-s3-upload-policy-document** folder via the Windows Explorer GUI, or using a utility such as 7zip. (**Note: don't zip the create-s3-upload-policy-document folder. Zip up the files inside of it**).

## 4. DEPLOY FUNCTION

Now we need to deploy the function to AWS.

- In the AWS console click **Lambda**.
- Click **get-upload-policy** in the function list.
- Click **Upload** to upload the function, select the ZIP file created in the previous step and then click **Save**.
- Create two environment variables **ACCESS\_KEY\_ID** and **SECRET\_ACCESS\_KEY** with values to match the user created in step 2.
- Create a third environment variable **UPLOAD\_BUCKET** and set it to your video upload s3 bucket.

## 5. CREATE RESOURCE & METHOD IN THE API GATEWAY

In this step we will create a resource and a method in the API Gateway. We will use it to invoke the Lambda function we deployed in the previous step.

- Click **API Gateway** in the AWS console
- Select **24-hour-video**
- Click **Actions** and then click **Create Resource**
- Set the Resource Name to **s3-policy-document**

### New Child Resource

Use this page to create a new child resource for your resource.

Configure as [proxy resource](#)

☐ ⓘ

Resource Name\*

s3-policy-document

Resource Path\*

/ s3-policy-document

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring **/{proxy+}** as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS

☒ ⓘ

\* Required

[Cancel](#)

[Create Resource](#)

- Click **Create Resource**
- Make sure that **s3-policy-document** is selected under **Resources** and click **Actions**
- Click **Create Method**
- From the dropdown box under the resource name, select **GET** and click the tick button to save
- In the screen that immediately appears:
  - Select **Lambda Function** radio
  - Check the checkbox with the label **Use Lambda Proxy Integration**
  - Set **us-east-1** as the Lambda Region
  - Type **get-upload-policy** in Lambda Function textbox
  - Click **Save**. Click **OK** in the dialog box that appears.



Choose the integration point for your new method.

Integration type ☒ Lambda Function ⓘ

☐ HTTP ⓘ

☐ Mock ⓘ

☐ AWS Service ⓘ

Use Lambda Proxy integration ☒ ⓘ

Lambda Region

Lambda Function  ⓘ

Save

## 6. DEPLOY API GATEWAY

Finally, we need to deploy the API so that our changes go live.

- Click **Actions**
- Select **Deploy API**
- In the popup select **dev** as the **Deployment stage**
- Click **Deploy** to provision the API

## 7. ENABLE CORS FOR THE S3 BUCKET

To be able to upload direct to an S3 bucket we also need to enable CORS for the bucket.

- Click **S3** in the AWS console.
- Click on the **upload** bucket (e.g. severless-video-upload)
- Click **Properties**
- Expand **Permissions**
- Click **Add CORS Configuration**
- Paste in the following CORS configuration and click **Save**

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <CORSRule>
    <AllowedOrigin>*/AllowedOrigin>
    <AllowedMethod>GET/AllowedMethod>
    <AllowedMethod>POST/AllowedMethod>
    <MaxAgeSeconds>3000/MaxAgeSeconds>
    <AllowedHeader>*/AllowedHeader>
  </CORSRule>
</CORSConfiguration>
```

## 8. TESTING

Now we are ready to test our upload functionality via the website.

- Copy the config.js file containing your account specific settings, from the last lesson.  
Copy lab-3/website/js/config.js to lab-4/website/js/config.js
- Open a terminal / command-prompt and navigate to the following folder:

*lab-4/website*

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

*npm install*

- Run the following command to start the website:

*npm start*

- Open the website and sign in. Click on the **plus** button to upload a movie file. You will see a progress bar while the upload takes place.



## All videos. All the time.

Guaranteed 100% server free.



Jump in to the AWS console and have a look at the buckets. Did the file upload to the upload S3 bucket? Are there new files in the transcoded S3 bucket?

If something didn't work make sure to check that:

1. The **config.js** file in your website contains the right Auth0 credentials and API Gateway URL.
2. You have followed steps 1-7 exactly and copied everything exactly as specified in this lesson plan.

**We are nearly there! There's one more lesson left and you'll have your full YouTube clone 😊**