LESSON 4

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

1. Create a Lambda function to grant us credentials/policy to upload files to the 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 your users upload videos to S3. To create the lambda function:

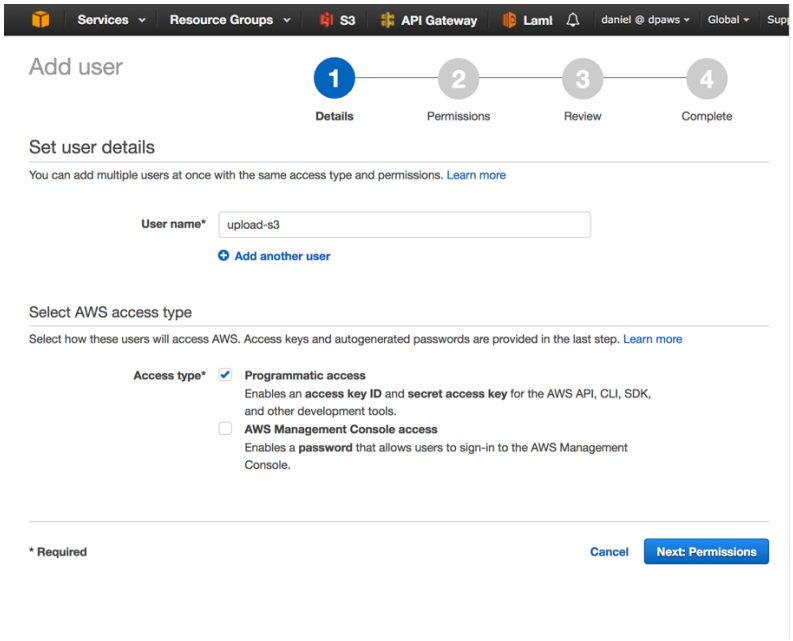
- Go to **Lambda** in the AWS Console.
- Create a new function as you did for your other Lambda functions.
- Name the function **get-upload-policy** and select **Node.js 4.**3 as the Runtime.
- Assign the **lambda-s3-execution-role** policy to it (the same policy created in lesson 1).
- Set the **Timeout** to 30 seconds.
- Leave all other settings as their default values and **Save** the function.

2. CREATE IAM USER

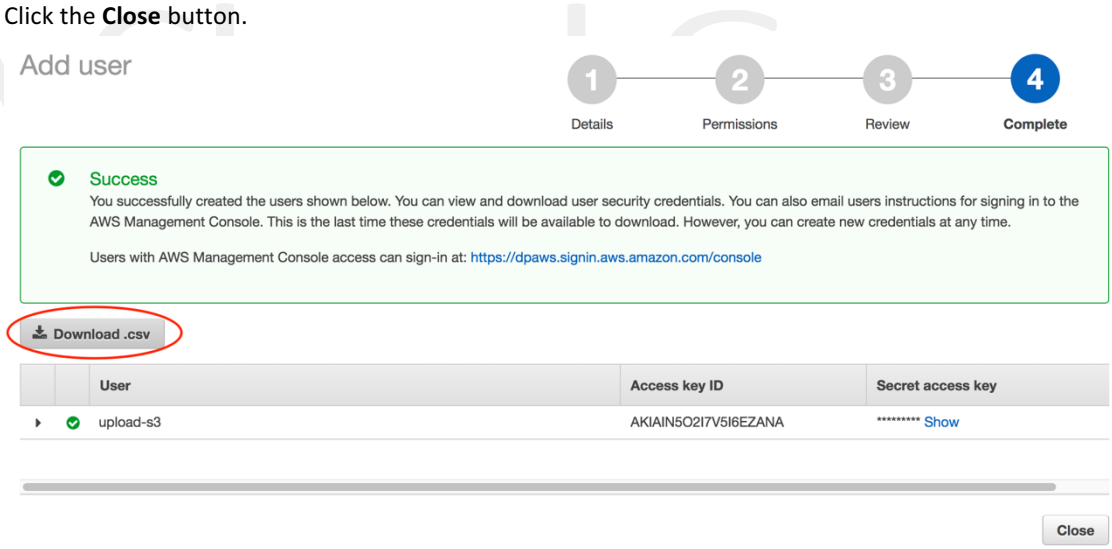
The policy and credentials that we are going to generate in the Lambda function need to be signed by an IAM user that has permissions to upload files to S3. Let's create this user now.

- Go to **IAM** in the AWS console.
- Click **Users** in the left navigation menu, then click the **Add user** button in the top left.

- Set the username to **upload-s3** and check the box labelled **Programmatic access**

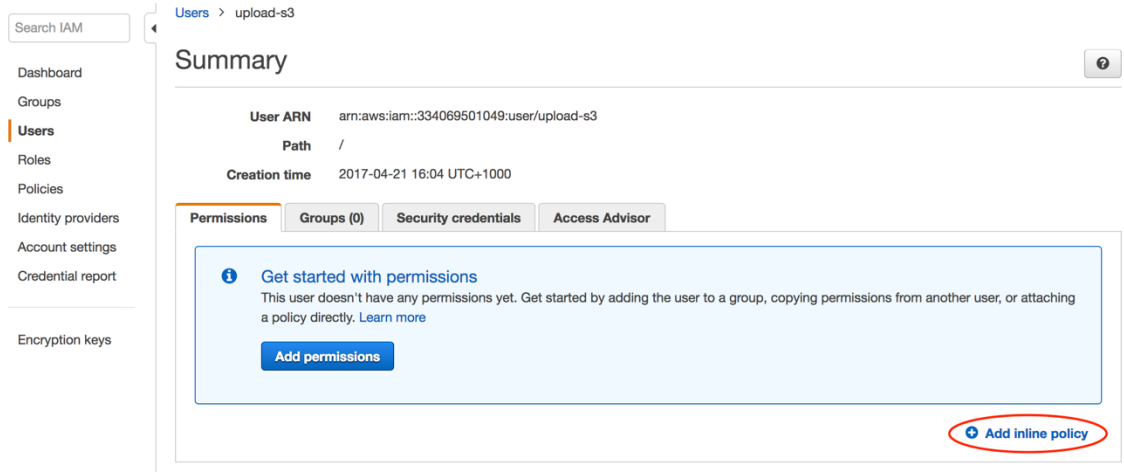


- Click **Next: Permissions** and then skip adding permissions at this time by clicking **Next: Review.**
- Ignore the warning that *This user has no permissions* and click **Create user**.
- You will then be shown the following screen where you must download the user's Access key id and Secret access key as a CSV: click the **Download .csv** button and save the **credentials.csv** file to your computer.
- Click the **Close** button.



- Click the **upload-s3** user and click the **Permissions** tab.

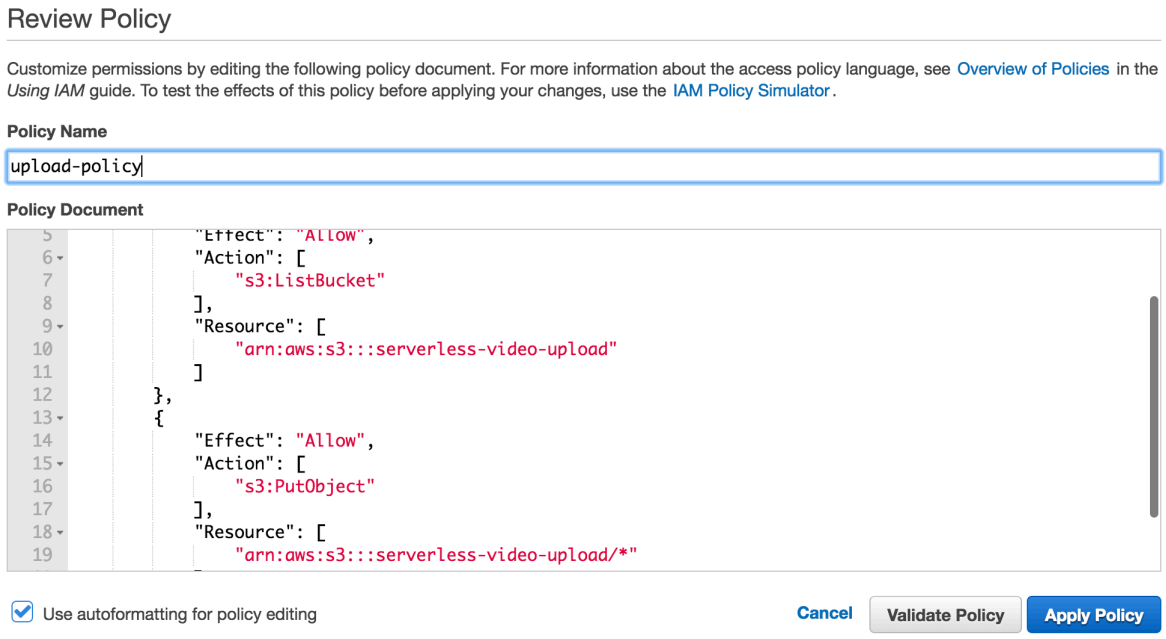
- Click **Add inline policy**



- To create a new **Inline Policy**, select **Custom Policy**, and click **Select**.
- Set the name of the policy to **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/*"
      ]
    }
  ]
}
```



3. CONFIGURE FUNCTION

Set up and zip the Lambda function provided in Lab 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 create a ZIP file of the function, by typing:

  ```
  npm run predeploy
  ```

  For Windows

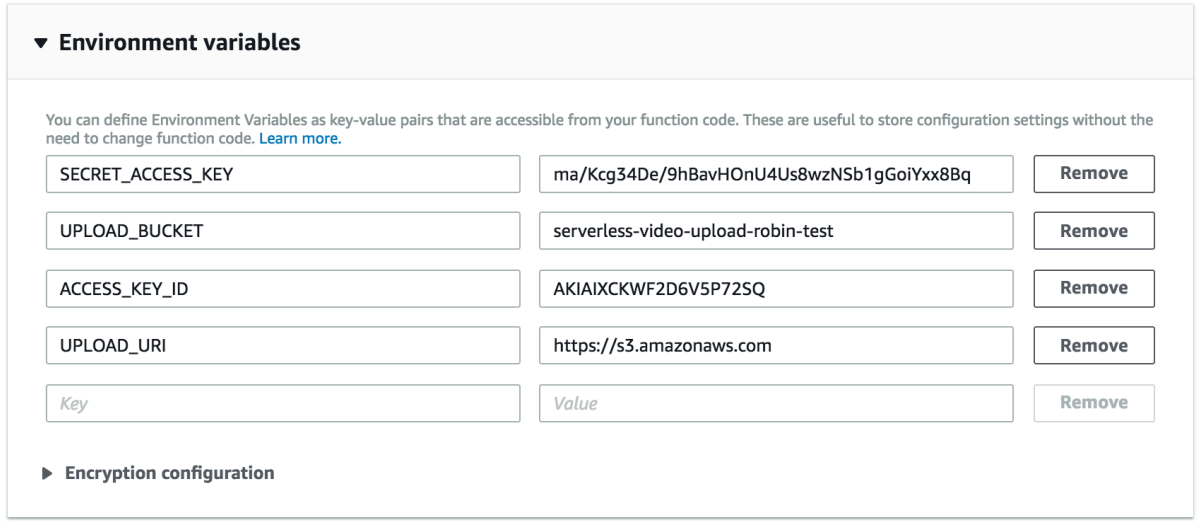
  You will need to **zip up all the files** in the **lab-4/lambda/create-s3-upload-policy-document** folder via the Windows Explorer GUI, or using a utility such as 7zip. (**Note: don't zip the create-s3-upload-policy-document folder. Zip up the files inside of it**).

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.
- Select **Code entry type** and click **Upload a .ZIP** to upload the function, select the ZIP file created in the previous step and then click **Save.**
- Create two environment variables with the keys **ACCESS_KEY_ID** and **SECRET_ACCESS_KEY**. The values of these variables will be in the .csv file you downloaded in step 2 which you need to copy and paste into this screen.
- Create a third environment variable **UPLOAD_BUCKET** and set it to your video *upload* s3 bucket.

And finally create a fourth environment variable **UPLOAD_URI** and set it to **https://s3.amazonaws.com**

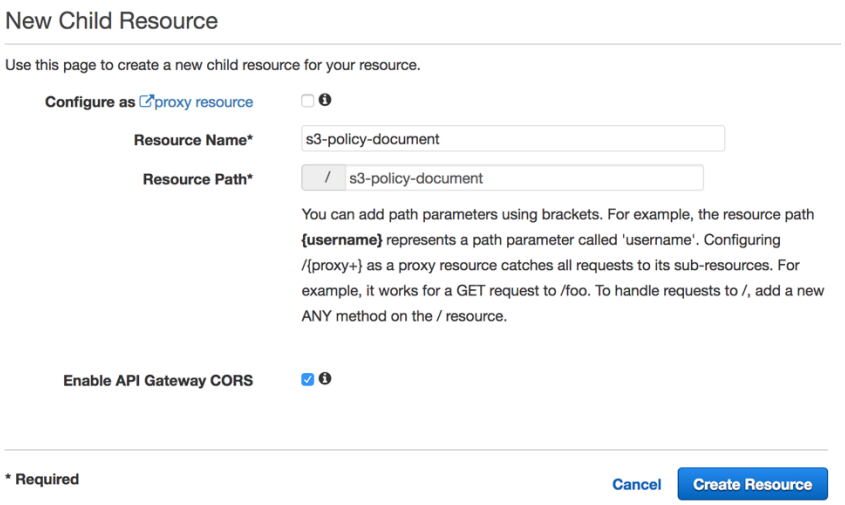


- Scroll to the top of the page and click the **Save** button.

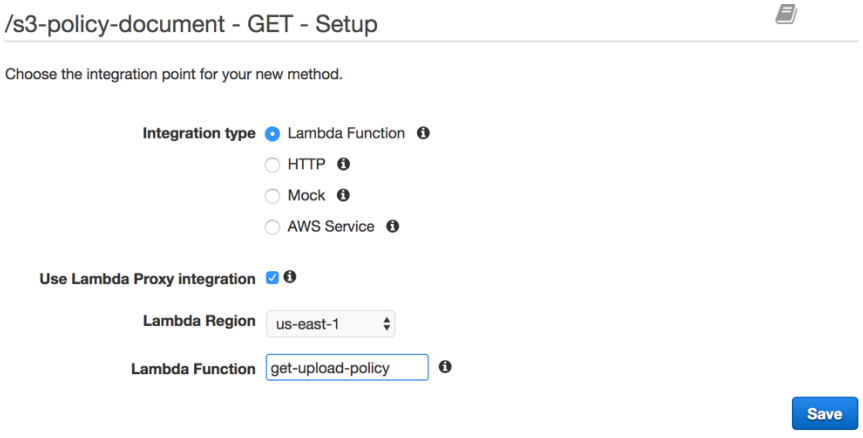
5. CREATE RESOURCE & METHOD IN THE API GATEWAY

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.

- Go to **API Gateway** in the AWS console.
- Select **24-hour-video**.
- Under **Resources** in the second column, ensure that the **/** resource is selected.
- Select **Actions** and then click **Create Resource**.
- Set the Resource Name to **s3-policy-document**.
- Ensure that the **Enable API Gateway CORS** box is checked.



- 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/check mark button to save.
- In the screen that 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.



To make the custom authorizer invoke on the GET method, follow these steps:

- Ensure you are still on the **Resources** tab in the left navigation menu.
- Click **GET** under **/s3-policy-document** in the second column.
- Click **Method Request** in the **/s3-policy-document - GET - Method Execution** section.

- Click the pencil next to **Authorization**.
- From the dropdown select **custom authorizer** and save.
- Click on the **HTTP Request Headers** section to expand it.
- Click on the **Add Header** link, put "authorization" for the name, and click the tick/check mark icon to the right to save it.

6. DEPLOY API GATEWAY

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

- Click **Actions** at the top of the second column.
- Select **Deploy API**.
- In the popup select **dev** as the **Deployment stage**.
- Click **Deploy** to deploy the API.

7. ENABLE CORS FOR THE S3 BUCKET

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

- Go to **S3** in the AWS console.
- Click on the **upload** bucket (e.g. *severless-video-upload*).
- Click **Permissions** from the bucket menu.
- Click **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

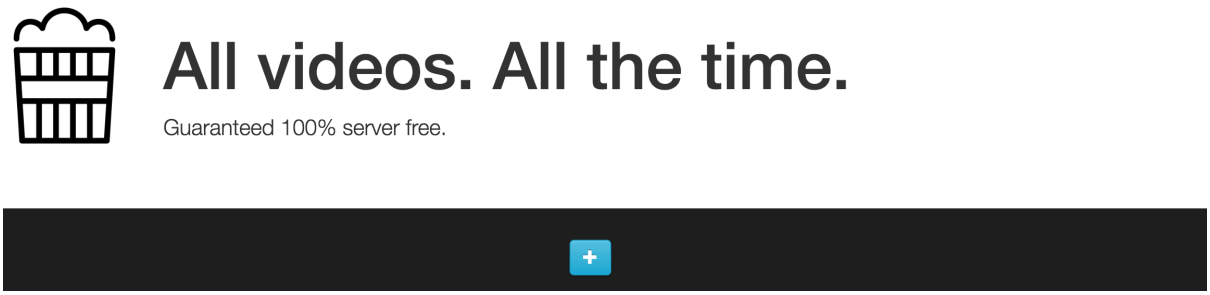
- Run the following command to make sure that required npm components are installed:

  npm install

- Run the following command to start the website:

```
npm start
```

- Open the website (http://localhost:8100) and sign in. Click on the **plus** button at the bottom of the page to upload a movie file. You can use one of the example files from the first lesson. You will see a progress bar while the upload takes place.



Go 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? The files will be inside a folder with a randomly generated name, like a70e496a579b9fb21144fb108e6bf000747a98d3.

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.

When you're done with this lab, exit the "npm start" command in your terminal by pressing <Control>-c.

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