Implementation

In this exercise, you will have to implement three new features in the image sharing application:

- A custom authorizer that verifies JWT tokens signed using RS256 (asymmetric) algorithm
- A function that extracts a user id from a JWT token
- Update the `CreateGroup` function and store ID of a user with each new group object stored in DynamoDB

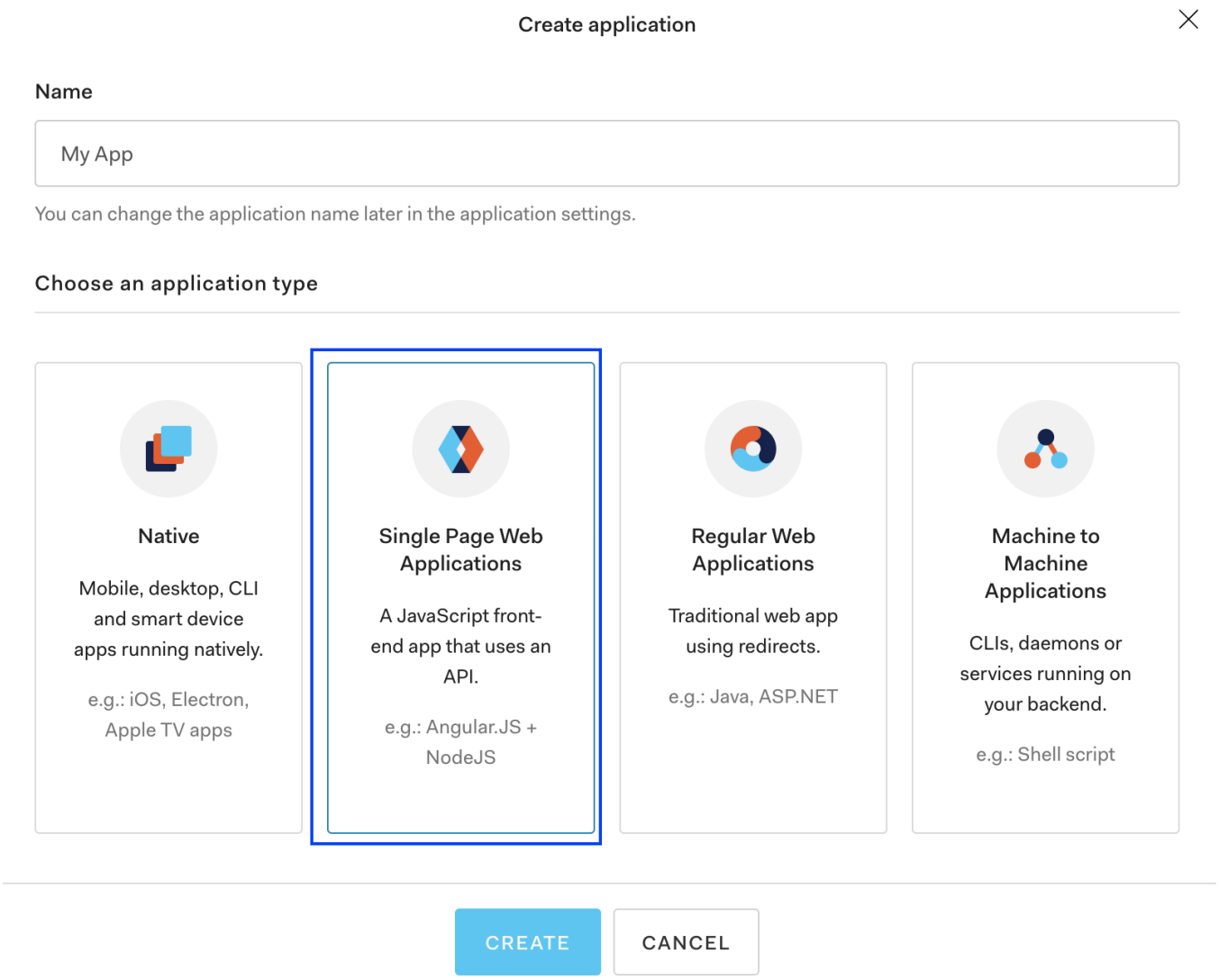
Here are the steps that you need to follow to implement this application.

Create a new Auth0 app

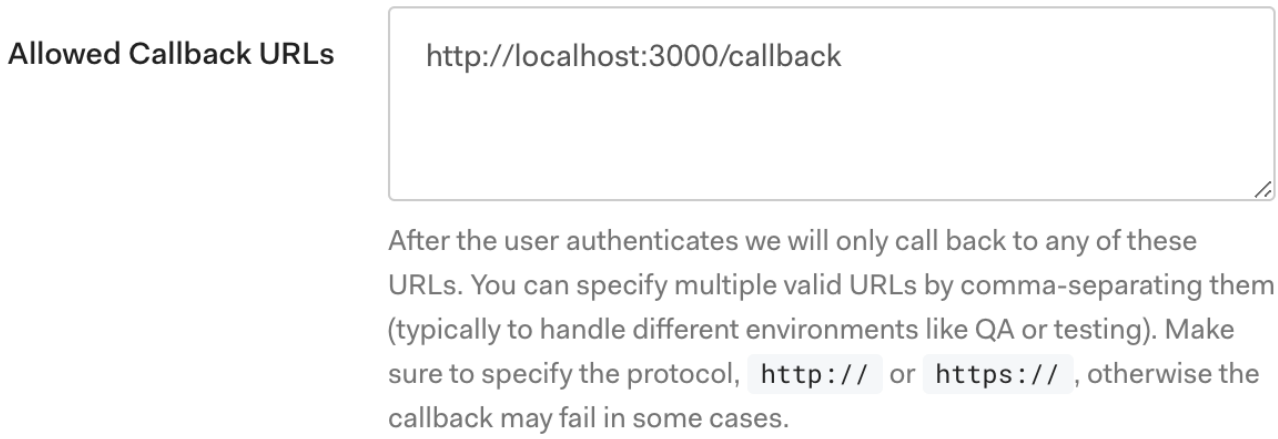
Go to the Auth0 website and log in.

Then go to the `Applications` section and create a new application:

Provide a name for your application and select "Single Page Web Applications" type. Then click "Create".



Then configure "Allowed Callback URLs" and "Allowed Web Origins" as we did in this lesson:

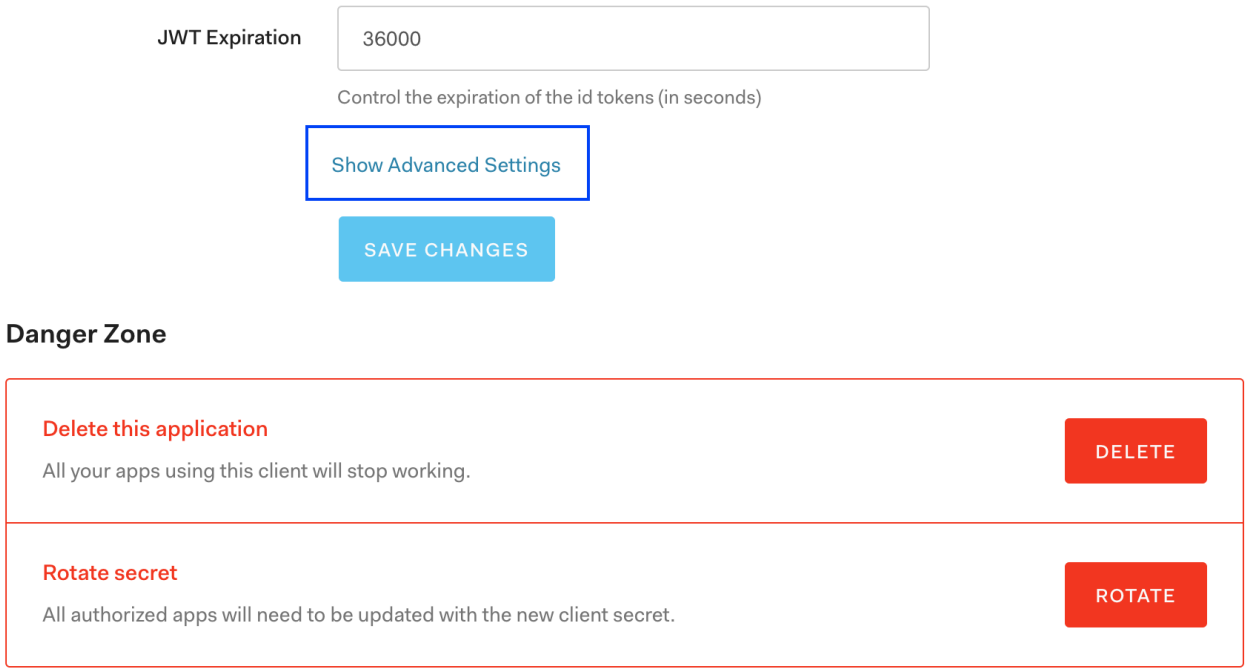


Copy a certificate that can be used to validate a JWT token

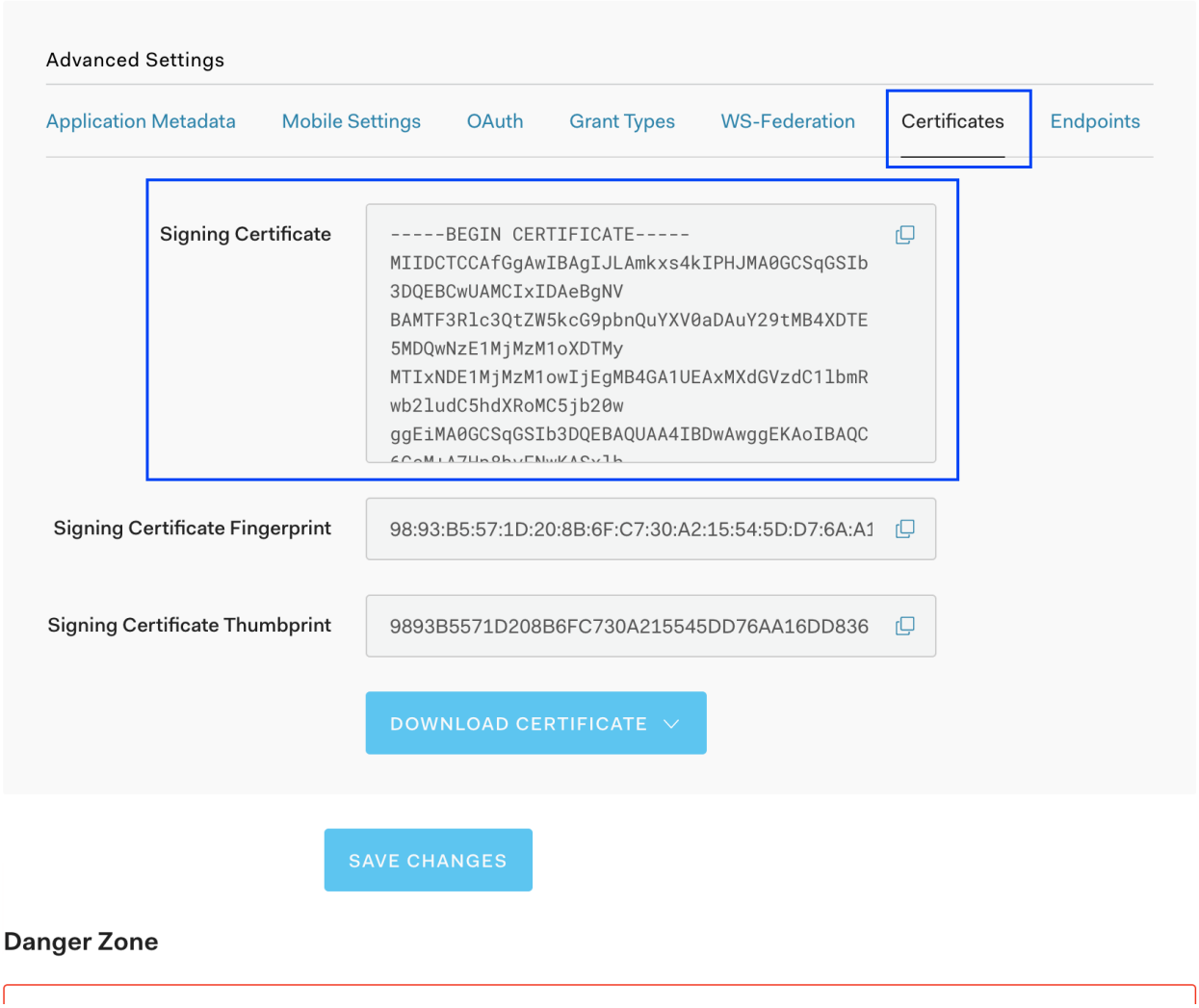
We need to get a certificate that can be used to verify a JWT token. We can programmatically fetch it from Auth0 when we validate a token, but to keep the exercise

more straightforward we will just copy it for now and store as a string in a function's source code.

To do this open the "Advanced settings" section at the bottom of the page:



And copy the certificate from the "Certificates" section:



SAVE CHANGES

Danger Zone

In the final project of this course, we will fetch it programmatically.

Implement a custom authorizer

The custom authorizer this demo will be almost the same as the authorizer implemented previously in this lesson. The main difference is how to call the `verify` function to verify a token. You need to do this in the following way:

```
verify(
    token,            // Token from an HTTP header to validate
    cert,             // A certificate copied from Auth0 website
    { algorithms: ['RS256'] } // We need to specify that we use the
) as JwtToken
```

Notice that now we don't need to store a secret, so we don't need to use AWS Secrets Manager. We also don't need to use `middy` middleware to fetch a secret.

Implement "getUserId" function

Now to the second part of the exercise!

First, we need to extract a user's ID from a JWT token. To do this, you need to implement the `getUserId` function in the `src/auth/utils.ts` file. For this, you would have to use another function called `decode` from the `jsonwebtoken` library. It does not validate a JWT token, but just parses it and returns its payload

```
const decodedJwt = decode(jwtToken) as JwtToken
```

To get an ID of a user from a JWT token, we need to use the `sub` field on the decoded token:

```
decodedJwt.sub
```

Store a user ID in a DynamoDB table

Now if we want to store an ID of a user when we create a new item, we can use `getUserId` function.

First, we need to get a JWT token in an event handler. To do this, add the following code in a handler in the `createGroup.ts` file:

```
const authorization = event.headers.Authorization
const split = authorization.split(' ')
const jwtToken = split[1]
```

Now to store a user ID we need to extract it from a JWT token using the `getUserId` function:

```
const userId = getUserId(jwtToken)
```

And store it to the DynamoDB table:

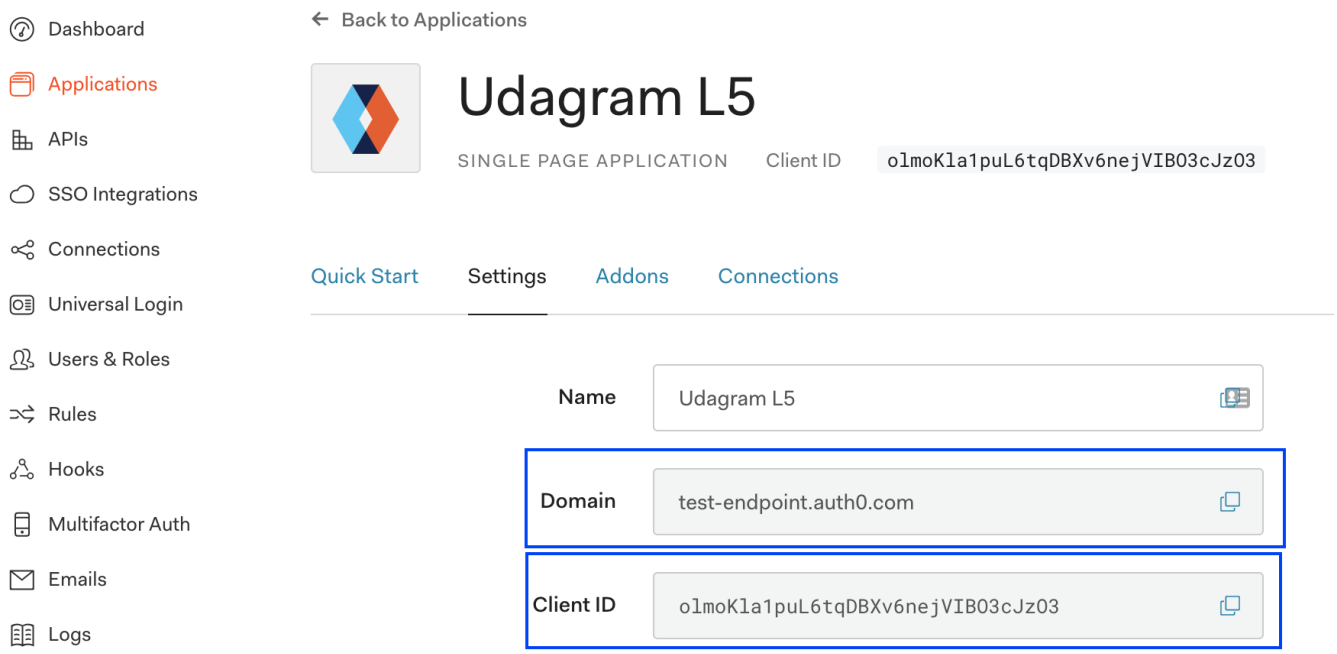
```
const newItem = {
  id: itemId,
  // A new line to add
  userId: userId, // Can be abbreviated to just "userId,"
  ...parsedBody
}
```

Configure a web application

Now the last step is to configure our web application to use the new Auth0 application that we've created. To do this, you need to change the `src/config.ts` file in the web application. You need to provide the following values:

```
export const authConfig = {
  domain: '...',    // Name of the Auth0 domain
  clientId: '...',  // Client id of a new application
  callbackUrl: 'http://localhost:3000/callback'
}
```

You can copy those values from the configuration page for your Auth0 application:



You also need to configure `apiEndpoint` to point to the API of your serverless application.

Preparations to test your function

To test your function, you should do the following:

- Deploy the serverless application
- Start the web application

Deploying the serverless application

Just as in the previous lesson, you need to ensure that S3 buckets have unique names. To ensure that your S3 buckets have unique names add a random string to the end of S3 bucket names in the `serverless.yml` file. Let's say you want to add a random string `ab4fe`. You would need to change the following section like this:

```
  environment:
    IMAGES_S3_BUCKET: serverless-udagram-images-ab4fe-${self:provide
    THUMBNAILS_S3_BUCKET: serverless-udagram-thumbnail-ab4fe-${self:
```

To deploy the whole project, you need to run the following commands:

```
npm install
serverless deploy -v
```

Make sure that the `serverless` command is installed and configured to use correct IAM credentials.

Start the web application

To start the web application, you need to run the following commands:

```
npm install
npm run start
```

Testing the result application

To test the result application, go to the `localhost:3000`. Click the "Log in" button to log in using the new Auth0 application that you have created.

Now you can create a new group with any name you like.

Expected result

Once a group is created, you should be able to see it in the `Groups-dev` DynamoDB table, but now it should contain an attribute containing a user's id: