[version_1.0.4]

Errors or corrections? Contact us at https://support.aws.amazon.com/#/contacts/aws-training

Exercise: Cognito

Story So Far

Mike and the team have been circling the kiosks since you arrived this morning and everyone seems very excited. Or at least not unimpressed that you have a Proof of Concept in place.

You *know* when you have done a good job when the assistant manager brings you a latte.

So now you're on day 3. You have a choice. Do you flush out the API and wire them up to start using real data using all that S3 select stuff. Or do you build out all the authentication ready for the POST API? 🤔.

Decisions. Decisions.

Considering you have spent the last 2 days staring at code and wrangling the SDK. You think today might be the day to do some console stuff instead and brush up on your point and click game.

You roll your sleeves up and announce to yourself (in your inside voice) that today is going to be Cognito day.

You will learn in this lab:

1. How to create a Cognito User Pool using the AWS console. How to extract Cognito tokens using localhost.
2. How to set up and test the API GW Cognito authentication integration.
3. Update Cognito to work with the website, and then test it all.

Accessing the AWS Management Console

1. At the top of these instructions, click **Start Lab** to launch your lab.
2. A Start Lab panel opens displaying the lab status.
3. Wait until you see the message "**Lab status: ready**", then click the **X** to close the Start Lab panel.
4. At the top of these instructions, click **AWS**

   This will open the AWS Management Console in a new browser tab. The system will automatically log you in.

> **TIP:** If a new browser tab does not open, there will typically be a banner or icon at the top of your browser indicating that your browser is preventing the site from opening pop-up windows. Click on the banner or icon and choose "Allow pop ups."

Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time, to make it easier to follow the lab steps.

Setup

1. Ensure you are in **Cloud9**. Choose **Services** and search for **Cloud9**. You should see an existing IDE called Building_2.0. Click the button **Open IDE**. Once the IDE has loaded, enter the following command into the terminal: *(This command will ensure that you are in the correct path)*

   ```
   cd /home/ec2-user/environment
   ```

2. You will need get the files that will be used for this exercise. Go to the Cloud9 **bash terminal** (at the bottom of the page) and run the following `wget` command:

   ```
   wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/DEV-AWS-MO-Building_2.0/lab-3-cognito.zip
   ```

3. Unzip:

   ```
   unzip lab-3-cognito.zip
   ```

4. Let's cleanup:

   ```
   rm lab-3-cognito.zip
   ```

5. Run the `resources/setup.sh` script that will grab the website contents and upload them to the S3 bucket created by our CloudFormation template.

   ```
   chmod +x ./resources/setup.sh && ./resources/setup.sh
   ```

   ⚠ **If you are using Java** you will also need to run the following script:

   ```
   chmod +x ./resources/java_setup.sh && ./resources/java_setup.sh
   ```

   ```
   export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.51.amzn1.x86_64
   ```

   ```
   source "$HOME/.sdkman/bin/sdkman-init.sh"
   ```

Lab Steps

Stage 1 - Create A Cognito User Pool Using Localhost

1. Choose **Aws Cloud9** and choose **Go To Your Dashboard**.
2. Choose **Services** and search for **Cognito**.
3. Choose **Manage User Pools**. Choose **Create a user pool**. Name it `FancyPool`. Choose **Review defaults**. Leave the current settings and choose **Create pool**.

   This should give you **"Your user pool was created successfully."** at the top of the page.
4. At the left choose **MFA and verifications**. At the top leave MFA **Off**. At the next step choose **Email only**. Finally at the bottom choose **No verification**. You will see the following warning:

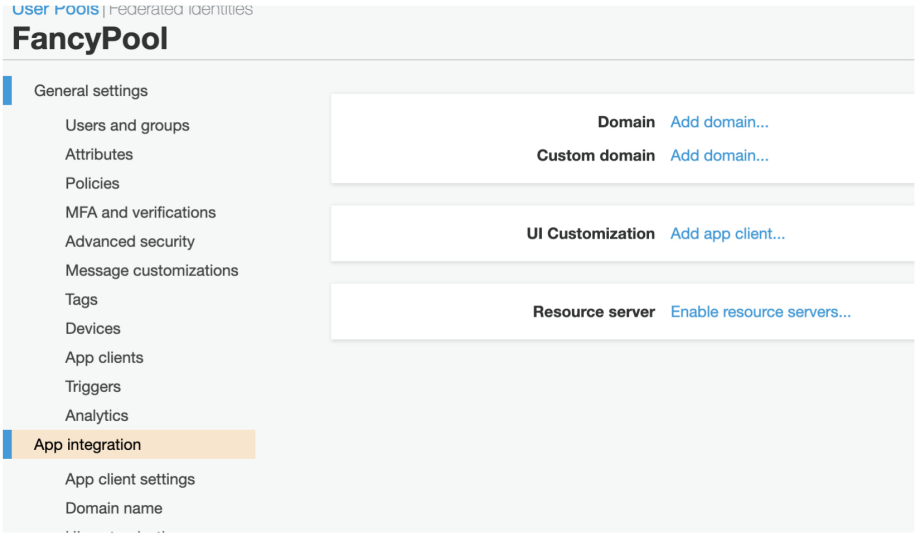
   ```
   You have not selected either email or phone number verification, so your users will not be able to
   recover their passwords without contacting you for support.
   ```

   This is expected btw.
5. An IAM role should be populated at the bottom called `FancyPool-SMS-Role`. This is not needed so there is *no need* to press create role. *Instead* choose **Save changes**.
6. At the left under **General settings** and choose **Users and groups**.
7. Choose **Create user**. Name the user `ricky`. Uncheck **Send an invitation to this new user?** For the **Temporary password** use `!FooBar55`. Enter in a valid **Phone Number**, and leave **Mark phone number as verified?** checked. *Note*: needs to be in **+1xxxxxxxxxx** format *(+1 being the USA code)*. Also enter in a valid **Email** and leave **Mark email as verified?** checked. Finally choose **Create user**.

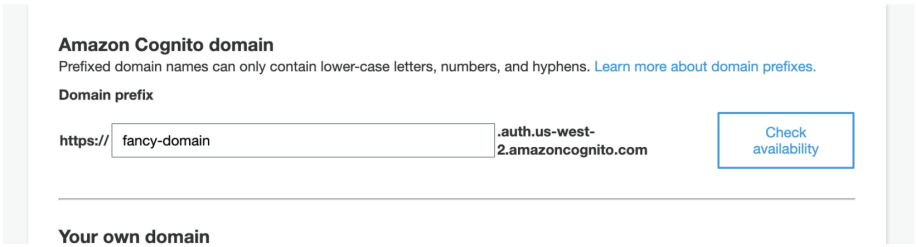
   You should see something like this:

Username	Enabled	Account status	Email verified	Phone number verified	Updated	Created
ricky	Enabled	FORCE_CHANGE_PASSWORD	true	true	Mar 26, 2020 6:36:42 PM	Mar 26, 2020 6:36:42 PM

8. At the left under **General settings** choose **Policies**. Choose **Only allow administrators to create users** and choose **Save changes**.
9. At the left click on **App integration**.



10. Choose **Domain name**. Type in `fancy-domain`



11. Choose **Check availability**. If it's not available increment it for *example: fancy2-domain (and make a note of what you used as you will need it later)*. Choose **Save Changes**.
12. At the left under **General settings** choose **App clients**. Choose **Add an app client**. For **App client name** use `FancyApp`. The **Refresh** will stay at `30`. Uncheck **Generate client secret**. Uncheck everything under **Auth Flows Configuration** except for **ALLOW_REFRESH_TOKEN_AUTH** which will already be selected. Leave **Enabled (Recommended)** checked. Choose **Create app client**.
13. Under **App integration** choose **App client settings**. Choose **Select all**. For **Callback URL(s)** paste in:

    ```
    http://localhost:8000/callback
    ```

For **Sign out URL(s)** paste in:

```
http://localhost:8000/sign-out
```

14. Under **OAuth 2.0** and **Allowed OAuth Flows** check **Implicit grant** only. Under **Allowed OAuth Scopes** check **openid** and **profile** only. Choose **Save changes**.
15. Choose **Launch Hosted UI**. Sign in with the username and password:

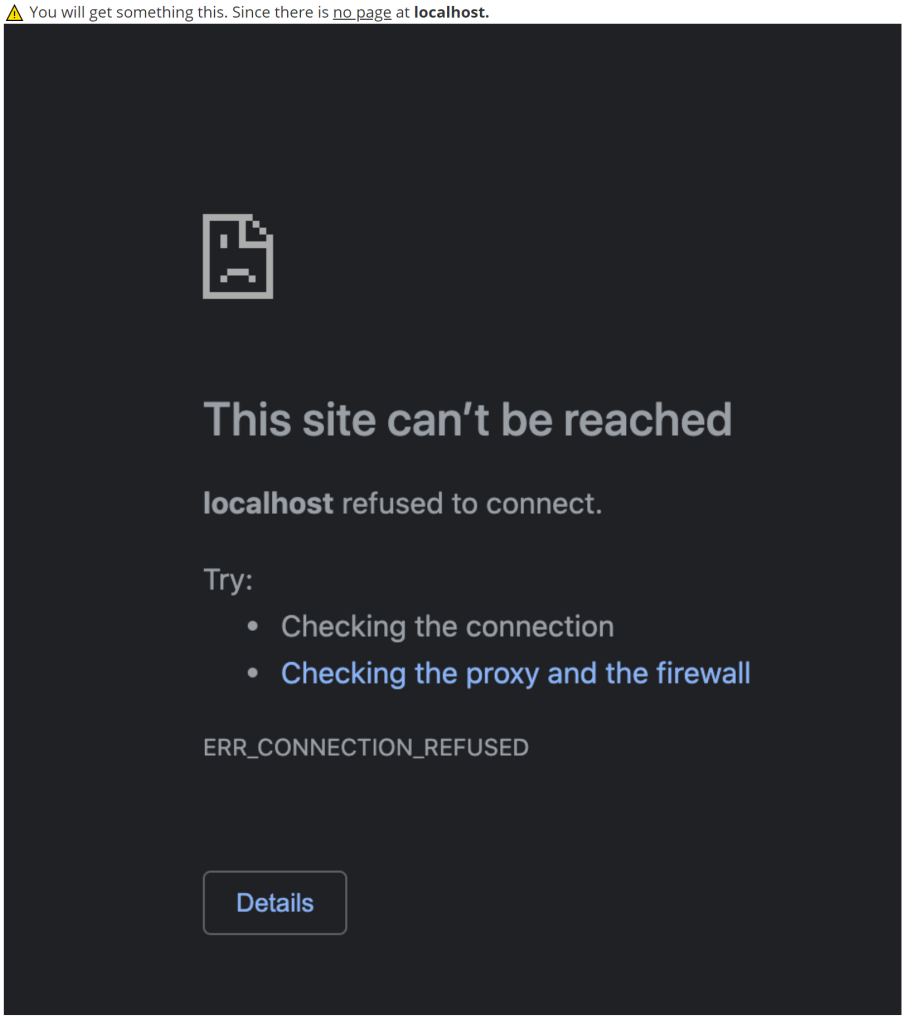
```
Username: ricky
Password: !FooBar55
```

Choose **Sign in**. It will ask to **Change Password use the same password:**

```
Password: !FooBar55
```

Choose **Send**.

⚠ You will get something this. Since there is no page at **localhost.**



This is exactly **what's expected**.

16. Grab the URL in the address bar and paste it into a text editor. It should look something like the following:

```
http://localhost:8000/callback#id_token=eyJraWQiOiJaM3gOVUR5VGdSUjF6eFl5ZGhWYkZDNzN3dzFpeko2b2dRckNlQWdldzhRPSIsImFsZyI6IlJTMjU2InO.eyJhdF9oYXNoIjoiZlh5U2E5VlJjTXRDRWlCaO5iM1lKZyIsInN1YiI6IjE5OGRmMzhiLTdjZjYtNGM5MiO4NWQ1LTdiY2U1NTYxOWJkMSIsImVtYWlsX3Zlcmlmaa WVkIjpOcnVlLCJpc3MiOiJodHRwczpcL1wvY29nbmlOby1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cy5jb21cL3VzLXdlc3QtMl93cTZSQnRoZWEiLCJwaG9uZV9udW1iZXJfdmVyaWZpZWQiOnRydWUsImNvZ25pdG86dXNlcm5hbWUiOiJyaWNreSIsImF1dCI6IjJlY2tvbDFwOWZtbHFudmVvZ3MxMm45Y2JOIiwiZXZlbnRfaWQiOiI1NjFhMzhjZC0xMDk5LTQxMTMtOGVhNi1mNWE5MDE4MzRlYWEiLCJOb2tlbl91c2UiOiJpZCIsImF1dGhfdGltZSI6MTU5MzYyMjA1NywicGhvbmVfbnVtYmVyIjoiKzE0MTUzNTk3OTkzIiwiZXhwIjoxNTkzNjI1NjU3LCJpYXQiOjE1OTM2MjIwNTcsImVtYWlsIjoicmljaOBmdWxsc3RhYTSDbGlmZSJ9.HRKXJf56g0ka4g5VlbaSzffFt8_wOsaurPKRNR2XFk2jWJP62IReR3q4Nstna7MWJly1PqQXvu6HwCVNG1GNk415edBj7gHwOkmqtr7xgG-9EVuOB8DEgSKFaPbNRj9xdw-znyx_q_ZAZHqb6UlGvy5jaUiRq7MLy1QOrg1Q8PyI_Ewt4-5LApncaqY5-6Z445ZI83ouKKugetg7bfyYlCTrYgOe64Q_EHNLaEh6u-IgKNDVZqdQBRRzzmDhbJuEaXRsiMGfAevW7jAn258yNCLDDWLU-NeJsgKuy4eyxMxOgEHfYiSpXCgRKcjT_k2verp7WVAMlBwZu7qZLnRBJQ&access_token=eyJraWQiOiJYbkE5OHhiSmoxRFU2RElSnZHbE9BOTNtUlJqSmdwYVFuTGVxVVNaMERzPSIsImFsZyI6IlJTMjU2InO.eyJzdWIiOiIxOThkZjM4YiO3Y2Y2LTRjOTItODVkNS03YmNlNTU2MTliZDEiLCJldmVudF9pZCI6IjU2MWEzOGNkLTEwOTktNDExMyO4ZWE2LWY1YTkwMTgzNGVhYSIsInRva2VuX3VzZSI6ImFjY2VzcyIsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUiLCJhdXRoX3RpbWUiOjE1OTM2MjIwNTcsImlzcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkc51cy13ZXNOLTIuYW1hem9uYXdzLmNvbVwvdXMtd2VzdC0yX3dxNlJCdGhlYSIsImV4cCI6MTU5MzYyNTY1NywiaWF0IjoxNTkzNjIyMDU3LCJ2ZXJzaW9uIjoyLCJqdGkiOiJjOWM2Y2UyZS1iYzk4LTRkMDgtYmI1MC0yYzI1YzNmZGY2MWUiLCJjbGllbnRfaWQiOiIyZWNrb2xxcDlmbWxxbnZlb2dzMTJuOWNidCIsInVzZXJuYW1lIjoicmlja3kifQ.OluShkyN6Ow2XQHmxH5KzdZrxOLjt4NNvOyPf6oL8hO1MOo1OPBzjar8MOa_OWZq4GR2CHearZ8flFhN-LHiMP65Eu9-gIOy6fZTctjjxvL2JA7ZS8xq_cQHWVQwfl1DyUKPM1_lNawaOkl6ymYqhdR6R1bUAdm7OKoxWPOKAiXsNvWMZWqawkFMcQ955nSct3QGAoHVbrsQKvhGWIU_c-yA4MpQRptzhXqFOvtuCJLjlIZOOn9FDAilF2ssJUD4wyrITO8_9A-jfKhlNALKQxf13j7193AVN8il2NICg7Ckt5ZFw_iq2CIPdfQQcNAOTklBbiS9Ww-wrBsODIg1Gg&expires_in=3600&token_type=Bearer
```

17. Extract the **id_token** bit (ie. do not extract the access token or token type). It will end up looking something a bit like the following: Make a note of yours, as you will need it soon.

```
eyJraWQiOiJUemFDRjF2ZlNrNVdtekk0THJQc0IxQWFcL3NOUkdsYzA5MGlQOFByc1pjQTOiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNo
IjoiQnFUSmNvMk9zNDFjellLR1p6REd3ZyIsInN1YiI6ImY4OTc3NmMxLThmYWItNDFhYS05MjE1LWU2ZDhmMzk3NjQ3NSIsImVtYWlsX
3ZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczpcL1wvY29nbmlOby1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cyrf1cL3VzLXdlc3QtMl8w
R2JKeWpvdFciLCJwaG9uZV9udW1iZXJfdmVyaWZpZWQiOnRydWUsImNvZ25pdG86dXNlcm5hbWUiOiJyaWNreSIsImF1ZCI6IjVidWNvY
mR2Y2Fmc2FyOWZmcnBtcXNiaDU3IiwiZXZlbsdaWQiOiI0MWQ2NjhjYiO1NzEwLTQzMGMtYTE4NSOzMDY2NDAyZDhkMWMiLCJOb2tlbl9
1c2UiOiJpZCIsImF1dGhfdGltZSI6MTU4NTIOODI3MiwicGhvbmVfbnVtYmVyIjoiKzE0MTUzNTk3OTkzIiwiZXhwIjoxNTg1MjUxODcy
LCJpYXQiOjE1ODUyNDgyNzIsImVtYWlsIjoicmljaOBmdWxsc3RhY2subGlmZSJ9.VXqmvWkMPb8P8SpMeHw6j97ylhTBnBIAUQbfouCC
b8nwpUtrfPuPmmFNEwakQEVyuorvFP-pjJzFRjqo-OaLBUJl6Oupx3XQ75m-
dQuTXjqf2MNG9eEMEwRHpOwq85gyrhfBQZO1Y8yCX_Q6Xs7gF5Z1cTdKdW6fD1eYF7kLPHfHwAS64JDdtl5ck9etzwRQUvDjYEjkj93hE
R9ypERdYSLAkh2KZp7JfyJCwQmmmTRTByYuxxngew9knNfUIql_IO4ePtdT6b6zhKb9mHfZB4hvLAwAHDZkZJolOk5jEG5PXlh7OnH2bl
HN4mXjfk5EPtoQ5bXlvPjreGKSRpLtBw
```

It is easy to accidentally have the access token or some extra characters still in there, Check again that you have done this step correctly, by comparing the two snippets above.

You would normally provide a LIVE callback URL in the previous setup steps. Which would take the full URL and extract the ID token from it and then use that to call the API. Although we do actually have that in your website already (callback.html). It is very useful to do this stage *via localhost* and manually extract the token and test everything out *first* before putting it into production to test via a live website.

Awesome. You have your Cognito Set up. Now you need to tell your POST API `create_report` in API Gateway to only allow access to managers who have pre-registered accounts with Cognito.

For now we use our dummy account `ricky`. The managers will set up their own Cognito accounts later.

Stage 2 - How to set up, and test, the API GW cognito authentication integration.

OK time to tell our POST API to authorize with Cognito.

1. Back at the **Cognito** tab. Choose **Services** and **API Gateway**. Choose the `Fancy-Api`. Then choose **Authorizers** at the left. Choose **Create New Authorizer** then name it `Fancy-Auth`.
2. Choose **Cognito** for **Type** and select the `FancyPool`. For **Token Source** paste in `Authorization`. Leave **Token Validation** blank.
3. Choose **Create**. Choose **Test**. Choose **Test** again. The expected response should look *similar* to the following:

```
Response Code:401
Latency 1
Unauthorized request: 6b15860c-c8ee-4758-a6d3-da076b9058f9
```

Below **Authorization Token** where it says **Authorization (header)** paste in the **id_token** which again should look *similar* to the following:

```
eyJraWQiOiJQUFZvVk5HdkVzanJhVmh3cXpJdHJsUlZcL21KTHh6XC8zOG12a2lLNzBLYlE9IiwiYWxnIjoiUlMyNTYifQ.eyJhdF9oYX
NoIjoiNDI1OExhRUlVMnRRNWxQdTNSV0dGQSIsInN1YiI6ImZhNjc4ZDFiLTg5NWQtNDYxYi1iNmFlLTk1Y2E1MTkzN2U3ZiIsImVtYWl
sX3ZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczpcL1wvY29nbmlOby1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cy5jb21cL3VzLXdlc3Qt
Ml8zRkpHNHhoeFEiLCJwaG9uZV9udW1iZXJfdmVyaWZpZWQiOnRydWUsImNvZ25pdG86dXNlcm5hbWUiOiJyaWNreSIsImF1ZCI6InU1Z
WJpYWg2OWUyajBpa2wzMmRzZjM2bTYiLCJOb2tlbl91c2UiOiJpZCIsImF1dGhfdGltZSI6MTU4Mzc3OTgzMSwicGhvbmVfbnVtYmVyIj
oiKzQ2OTIzMDcwNjIiLCJleHAiOjE1ODM3ODM0MzEsImlhdCI6MTU4Mzc3OTgzMSwiZW1haWwiOiJldmFucmluQGdtYWlsLmNvbSJ9.Qq
CgmB8BlpqramoaFoiVBAevxzwJlCf8OSeFe8j4NQF6ge274R7_XTcXNTnjizMiI35Yyar3Dqa3qQXEV35DTtO16FVaY-
UUop63A1G81asddlmejZ2L_FbZUGLA975xpJjJ96dvtDANFPOIa62nYbRNUzbWVLipdkE3M_-
lVXDxckNCz8omU6Eo1csE_QSZoR8B4AM3a7hEp94vEeU3OzG4sGflReTE6lD2wzMTPyYKKAOF5TPDOfPwQWt-yv-
sm155XWf8oOZUNFl9v3liAqVaT39PncBNCKwx3DOwOOuFuNgxlTCzDpTD9l7zHdWnRB3AJ2uC-YJ9mkRlukU8IA
```

Press **Test** again. The output should look similar to the following:

```
{
    "at_hash": "nBZuhu12hP7MXXSq3hk4kg",
    "aud": "6igujr7a5emupfgbc4el3s9rs5",
    "auth_time": "1593448363",
    "cognito:username": "ricky",
    "email": "xxxxxxxxxxx",
    "email_verified": "true",
    "event_id": "7cb09389-805d-42fe-a512-5be98d136eb1",
    "exp": "Mon Jun 29 17:32:43 UTC 2020",
    "iat": "Mon Jun 29 16:32:43 UTC 2020",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_PajrAtHMK",
    "phone_number": "+1xxxxxxxxxx",
    "phone_number_verified": "true",
    "sub": "646e50c7-3928-48e8-b7ad-813c174359db",
    "token_use": "id"
}
```

This is great, as now your API can be told to only allow access if a valid token is passed. Later you will be able to extract this information such as the phone number (to be able to send out reports). #winning

4. Choose **Close** and let's wire up this new authenticator with the POST API.
5. Choose **Resources** and **POST** under **/create_report**. Choose **Method Request** and **Authorization**. Choose the **pencil** icon at the right. Choose the **Fancy-Auth** user pool.

   You may need to refresh the page if its not showing up.

   Choose the **checkmark**. Leave everything else **"as-is"**, and choose **Actions** and **Deploy API**. For **Deployment stage** choose **test**. Choose **Deploy**. Ignore any warnings.

   Make a note of the API Gateway Endpoint. You will need that later.
6. Now back in your Cloud9 `CMD_LINE` test it via CURL against the **Invoke URL**. (*This can be found under **Stages** and **test***). First

test it without the token in the Cloud9 terminal:

```
curl --location -vk --request POST '<FMI>'
```

Example: Don't forget the `create_report` path bit.

```
curl --location -vk --request POST 'https://9gt9cz2kp0.execute-api.us-west-
2.amazonaws.com/test/create_report'
```

It should give you something like this. Telling you that you are not authorized to view it.

Perfect!

```
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
< HTTP/2 401
< date: Mon, 29 Jun 2020 16:57:18 GMT
< content-type: application/json
< content-length: 26
< x-amzn-requestid: 616bd81d-db20-4e3c-96b5-c63e12462b6e
< x-amzn-errortype: UnauthorizedException
```

7. Now add in the Authorization token in the header a a Bearer Token, using **your ID token** like so:

```
curl --location -vk --request POST --url 'https://kd6pcugh57.execute-api.us-west-
2.amazonaws.com/test/create_report' --header 'Authorization: Bearer <FMI>'
```

Example. Your token will be different:

```
curl --location -vk --request POST --url 'https://q8hu3zlwk4.execute-api.us-west-
2.amazonaws.com/test/create_report' --header "Authorization: Bearer
eyJraWQiOiJQUFZvVk5HdkVzanJhVmh3cXpJdHJsUlZcL21KTHh6XC8zOG12a2lLNzBLYlE9IiwiYWxnIjoiUlMyNTYifQ.eyJhdF9oYX
NoIjoiNDI1OExhRUlVMnRRNWxQdTNSV0dGQSIsInN1YiI6ImZhNjc4ZDFiLTg5NWQtNDYxYi1iNmFlLTk1Y2E1MTkzN2U3ZiIsImVtYWl
sX3ZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cy5jb21cL3VzLXdlc3Qt
Ml8zRkpHNHhoeFEiLCJwaG9uZV9udW1iZXJfdmVyaWZpZWQiOnRydWUsImNvZ25pdG86dXNlcm5hbWUiOiJyaWNreSIsImF1ZCI6InU1Z
WJpYWg2OWUyajBpa2wzMmRzZjM2bTYiLCJ0b2tlbl91c2UiOiJpZCIsImF1dGhfdGltZSI6MTU4Mzc3OTgzMSwicGhvbmVfbnVtYmVyIj
oiKzQ2OTIzMDcwNjIiLCJleHAiOjE1ODM3ODM0MzEsImlhdCI6MTU4Mzc3OTgzMSwiZW1haWwiOiJldmFucmluQGdtYWlsLmNsddSJ9.Q
qCgmB8BlpqramoaFoiVBAevxzwJlCf8OSeFe8j4NQF6ge274R7_XTcXNTnjizMiI35Yyar3Dqa3qQXEV35DTtO16FVaY-
UUop63A1G81a6Y3lmejZ2L_FbZUGLA975xpJjJ96dvtDANFPOIa62nYbRNUzbWVLipdkE3M_-
lVXDxckNCz8omU6Eo1csE_QSZoR8B4AM3a7hEp94vEeU30zG4sGflReTE6lD2wzMTPyYKKAOF5TPDOfPwQWt-yv-
sm155XWf8oOZUNFl9v3liAqVaT39PncBNCKwx3DOwOOuFuNgxlTCzDpTD9l7zHdWnRB3AJ2uC-YJ9mkRlukU8IA"
```

To get:

```
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
< HTTP/2 200
< date: Mon, 29 Jun 2020 16:59:45 GMT
< content-type: application/json
< content-length: 116
< x-amzn-requestid: 5365a0b5-ff98-48c2-9c30-6fa62d2d1e6e
< access-control-allow-origin: *
< access-control-allow-headers: Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token
< x-amz-apigw-id: O5mgQF7rPHcFUHg=
< access-control-allow-methods: POST,OPTIONS
<
{
    "message_str": "report requested, check your phone shortly"
}
* Connection #0 to host 9gt9cz2kp0.execute-api.us-west-2.amazonaws.com left intact
```

Excellent! You have access to your POST API, BUT only if you have a token.

This is just what we want.

*Just an FYI. If you tried to test this in the API GW console. i.e clicking **test** on create_report it will always allow access. The AWS console does NOT enforce this authentication check. Hence I had you do that curl stuff ;).*

Stage 3 - Link your API and Cognito to your website.

We are on the final task of the day where we prove that this can work on the kiosk (website).

Step 1 - Tell Cognito to use your website callback link instead of localhost

Step 2 - Tell the website that you have a new Cognito HOSTED URL.

Step 3 - Visit the website, and try to access a report (and fail). Then follow the login link to the new hosted UI. Finally proceed to login and attempt to get a new report. Hopefully receive a message saying that you are being sent a report. *Which of course you are not, as we haven't built that bit yet.*

Step 1 Update Cognito

1. First grab the Callback URL:

```
aws s3api list-buckets --query "Buckets[].Name" | grep s3bucket | tr -d ',' | sed -e 's/"//g' | xargs

#Output
your-bucket
```

Example:

```
aws s3api list-buckets --query "Buckets[].Name" | grep s3bucket | tr -d ',' | sed -e 's/"//g' | xargs
```

```
#Output
c11284a125436u294892t1w852315532251-s3bucket-pteedic3sfy5
```

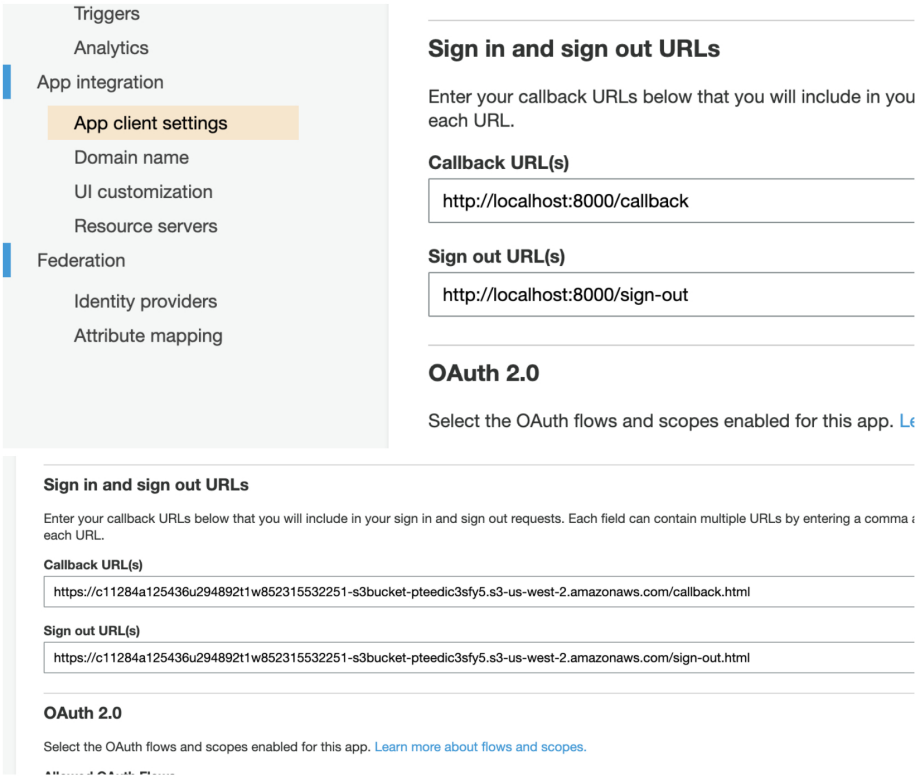
We will need to put that together with the Region to get the Callback URL:

```
https://<FMI>.s3-us-west-2.amazonaws.com/callback.html
```

Example:

```
https://c11284a125436u294892t1w852315532251-s3bucket-pteedic3sfy5.s3-us-west-
2.amazonaws.com/callback.html
```

2. Switch back to the **API Gateway** tab. Choose **Services** and choose **Cognito**. Choose **Manage User Pools**. Choose **FancyPool**.
3. Under **App integration** and **App client settings**. Replace the **callback** and **sign-out** URLs with the one we just created from above:



4. Choose **Save changes**.

Step 2 - Update The Website

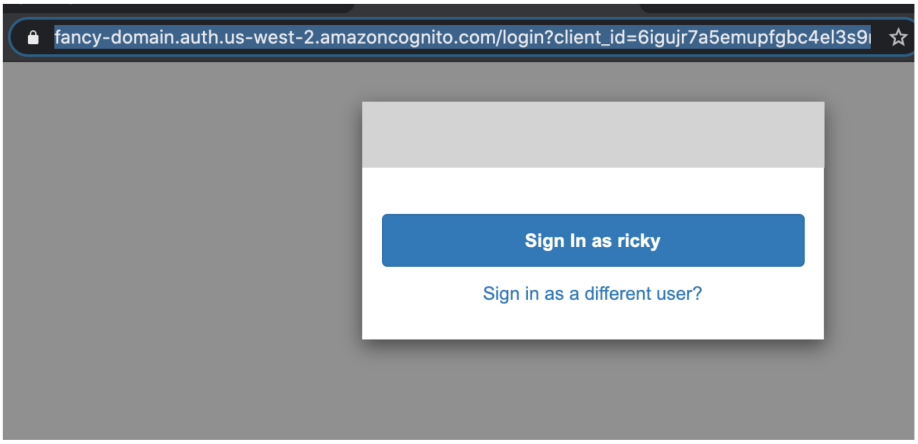
You will need to edit the website's config file.

First grab the hosted URL from Cognito.

1. Choose **Launch Hosted UI**.
2. This will give us the **Hosted UI** address in the browser address bar:

   Example:

```
https://fancy-domain.auth.us-west-2.amazoncognito.com/login?
client_id=6igujr7a5emupfgbc4el3s9rs5&response_type=token&scope=openid+profile&redirect_uri=https://c11284
a125436u294892t1w852315532251-s3bucket-pteedic3sfy5.s3-us-west-2.amazonaws.com/callback.html
```



3. Copy that to your clipboard and switch back to the **Cloud9** tab. Open `resources/website/config.js`

Which should look like this:

```
var G_API_GW_URL_STR = null;
var G_COGNITO_HOSTED_URL_STR = null;
```

Replace them using your respective URLs. ***Remember*** the Invoke URL was used in the curl tests from earlier. (i.e Your API Gateway Endpoint)

Example:

```
var G_API_GW_URL_STR = "https://9gt9cz2kp0.execute-api.us-west-2.amazonaws.com/test";
var G_COGNITO_HOSTED_URL_STR = "https://fancy-domain.auth.us-west-2.amazoncognito.com/login?
client_id=6igujr7a5emupfgbc4el3s9rs5&response_type=token&scope=openid+profile&redirect_uri=https://c11284
a125436u294892t1w852315532251-s3bucket-pteedic3sfy5.s3-us-west-2.amazonaws.com/callback.html";
```

4. Choose **File** and **Save**.
5. Now we will upload the updated config.js file using the provided script. Run this:

```
chmod +x ./resources/setup2.sh && ./resources/setup2.sh
```

To get:

```
upload: resources/website/config.js to s3://c11284a125436u294892t1w852315532251-s3bucket-
pteedic3sfy5/config.js
```

Step 3 - Check It Blocks You, And Then Works Once Logged In.

1. Now visit the website at: Using your URL:

```
https://<FMI>/index.html
```

Example:

```
https://c11284a125436u294892t1w852315532251-s3bucket-pteedic3sfy5.s3-us-west-2.amazonaws.com/index.html
```

2. The website should now work, in terms of getting ratings and reviews. However the request report feature should fail (at first)
3. Choose **REQUEST A REPORT**.

You should get this message:

```
Something Went Wrong
```

If you are curious you could look in the chrome dev tools at the network to see that you are getting this as a response:

```
content-length: 27
content-type: application/json
date: Thu, 02 Jul 2020 16:11:17 GMT
status: 403
x-amz-apigw-id: PDYN7HUEPHcFVlg=
x-amzn-errortype: AccessDeniedException
x-amzn-requestid: 26ed2bd5-4fec-44d9-9704-bc39ee119e45
```

4. Choose the **Admin Login**. It will redirect the page to the Cognito hosted login. Log in using `ricky` and `!FooBar55`

Once logged in, it will redirect you back to the site, where your bearer token is handled. Now try **REQUEST A REPORT**.

Because you are logged in, and because you have set up CORS correctly to allow authenticated "non-simple" POST requests. You should now see a different message:

```
"Report Requested, Check Your Phone Shortly"
```

🤦 *No need to check you phone, as we have not set up that bit yet.*

If you are curious you could look again in the chrome dev tools at the network to see that you are getting this as a response:

```
access-control-allow-headers: Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token
access-control-allow-methods: POST,OPTIONS
access-control-allow-origin: *
content-length: 100
content-type: application/json
date: Thu, 02 Jul 2020 16:13:00 GMT
status: 200
x-amz-apigw-id: PDYd5GTsvHcFeXA=
x-amzn-requestid: 0be15d60-7a00-43cf-be08-d65eba3e5668
```

🤔 You will notice that you have a `*` for origin, however this is on the already protected POST resource where the Authorization takes place. Remember that it is the OPTIONS resource where the Cross domain protection is happening. This OPTION requires the domain origin to match the website; hence preventing access to the POST resource from any non whitelisted domain. You could lock that POST down more by swapping out the `*` with the website origin for POST but because we are requiring on Authentication Token on the POST request the browser considers it "non simple" and thus CORS would just block it anyway.

🤦 Also note you can't use `*` for the Origin if you are using credentials, that's just another security feature of the browser.

So. It all works! 🎉

The kiosks allow you to login with a dummy account, and request a report.

This is designed to only work from inside the store. Due to your IP bucket policy, and the report can only be requested by logged in users.

Just as you are packing up for the day. Sandra comes by your desk, asking how things are going. She invites you to the team meal tonight.

You feel accomplished, and as you're on track, you decide to accept. You promise yourself you are not going to drink that much, because you have a lot of back end code to write tomorrow, and you will need all your brain cells

Lab Complete

Congratulations! You have completed the lab.

1. Click **End Lab** at the top of this page and then click **Yes** to confirm that you want to end the lab.
2. A panel will appear, indicating that "DELETE has been initiated... You may close this message box now."
3. Click the **X** in the top right corner to close the panel.

For feedback, suggestions, or corrections, please contact us at:https://support.aws.amazon.com/#/contacts/aws-training