Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Implement application authentication using Amazon cognito.**	
Experiment No: 13	**Date:**	**Enrolment No: 92200133030**

Aim :- Implement application authentication using Amazon cognito.

Lab overview and objectives

In this lab, you will work as Sofía to integrate Amazon Cognito into the café website. Frank wants to be able to log in and request a coffee bean inventory report directly from the café website. Amazon Cognito provides an authentication service, which Sofía wants to use for this website enhancement. This usability enhancement will use the state machine that you built in the previous lab using AWS Step Functions.

After completing this lab, you should be able to:

- Create an Amazon Cognito user pool
- Create an Amazon Cognito user pool user
- Configure an app client to use Amazon Cognito as an authentication service
- Integrate an Amazon Cognito app client into a website
- Update REST API endpoints that are built with Amazon API Gateway to call Amazon Cognito for authentication purposes
- Configure an API Gateway authorizer

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

Frank wants to be able to log in to the café website and request an inventory report to be sent to his email address (on his phone). He wants to be able to see the latest coffee bean inventory data quickly. In this lab, you will play the role of Sofía to implement this technical feature request.

In the previous lab, Sofía created a Step Functions state machine that, once invoked, can generate the report that Frank wants. However, the state machine can currently only be invoked by using the test feature in the Step Functions console or by running a curl command to invoke the *create_report* REST API endpoint.

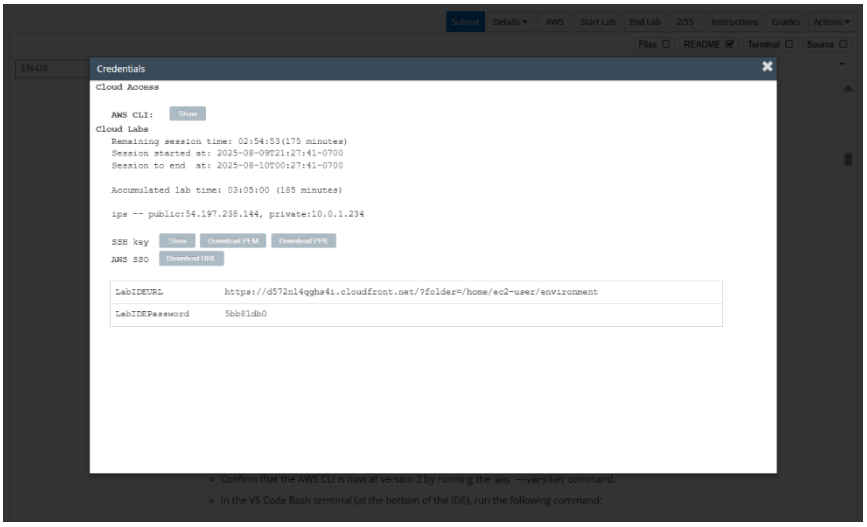
In this lab, Sofía will use Amazon Cognito to integrate an authentication mechanism into the website. Frank will be able to log in to the website to confirm his identity before he requests the report. Then, she will connect the REST API endpoint to the café website so that he can make his report request directly from the site. The API request will include the ID token that is retrieved as part of the authentication process, and Amazon Cognito will be used to validate the token. Then, the Step Functions state machine will be invoked, which will then invoke the AWS Lambda functions that generate the report.

Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Implement application authentication using Amazon cognito.**	
Experiment No: 13	**Date:**	**Enrolment No: 92200133030**

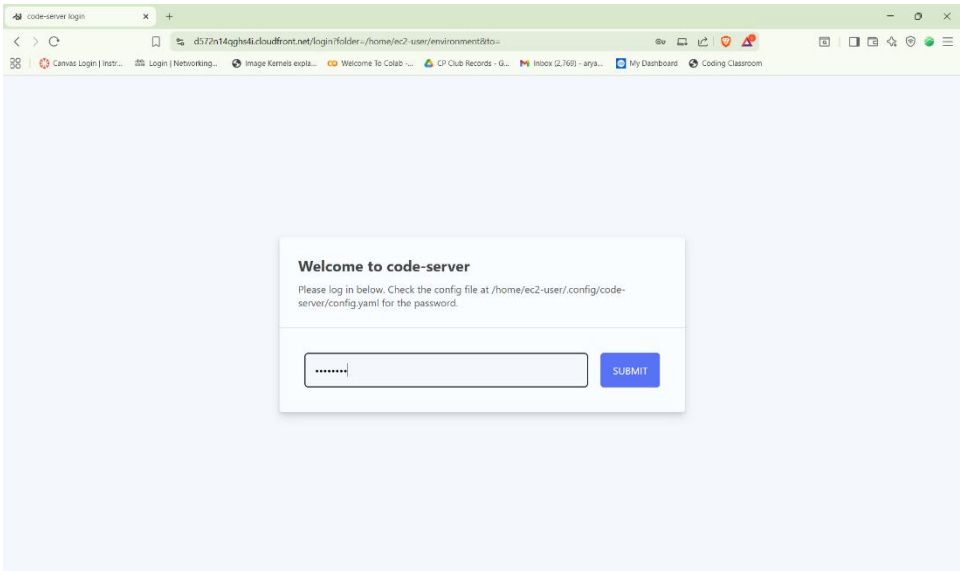
Task 1: Preparing the lab

Connect to the VS Code IDE.

1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
   a. **LabIDEURL**
   b. **LabIDEPassword**

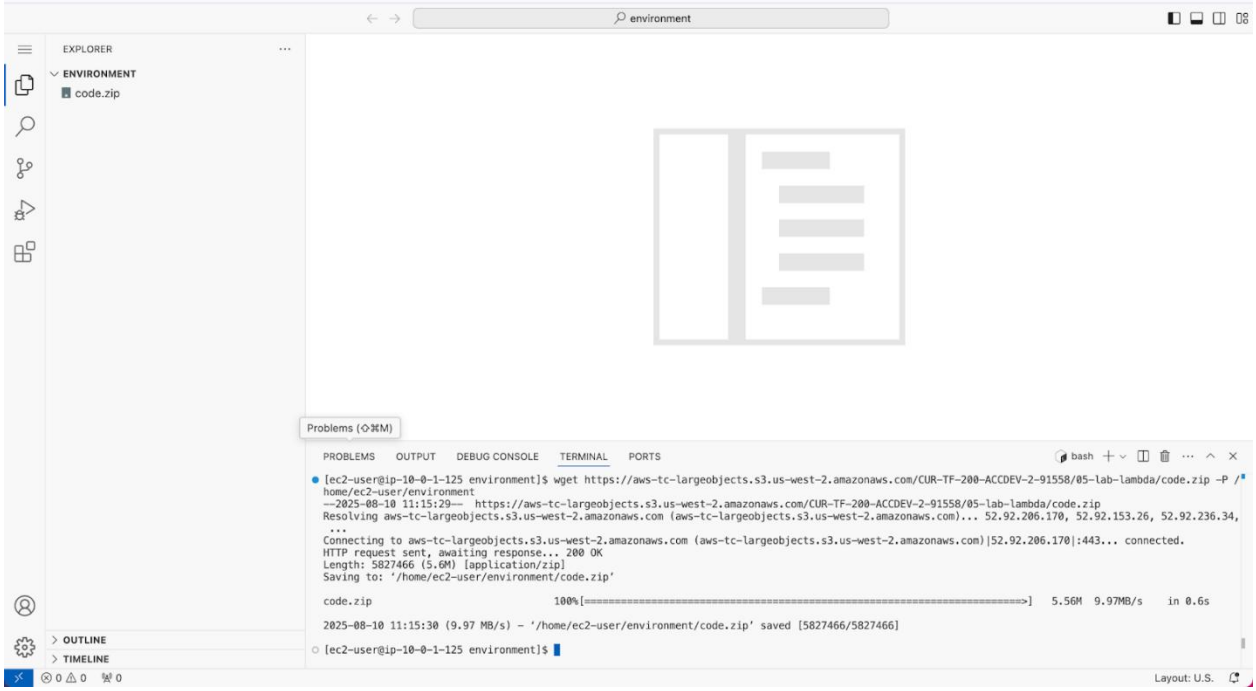


3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.



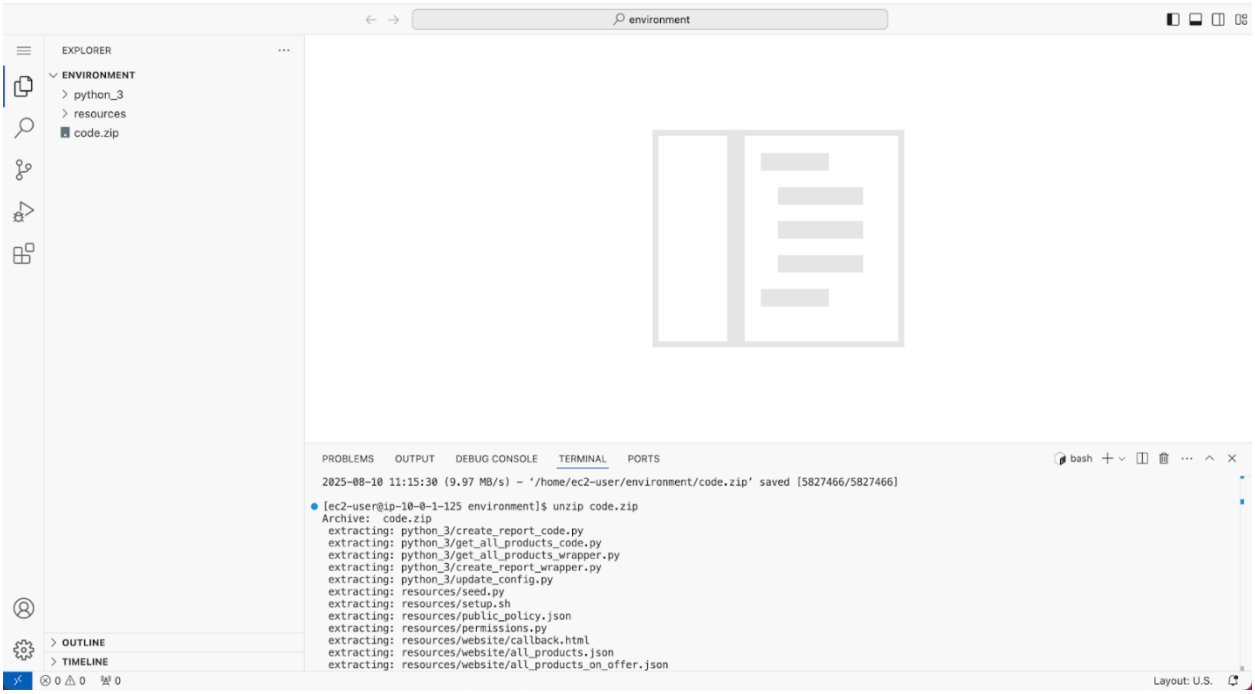
5. Download and extract the files that you need for this lab.
   - In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/12-lab-cognito/code.zip -P /home/ec2-user/environment
```



6. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.

- Extract the file by running the following command:
  `unzip code.zip`



7. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.
   - To set permissions on the script and then run it, run the following commands in the Bash terminal:

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

The script will prompt you for the **IP address** by which your computer is known to the internet. Use www.whatismyip.com to discover this address and then paste the IPv4 address into the command prompt and finish running the script.

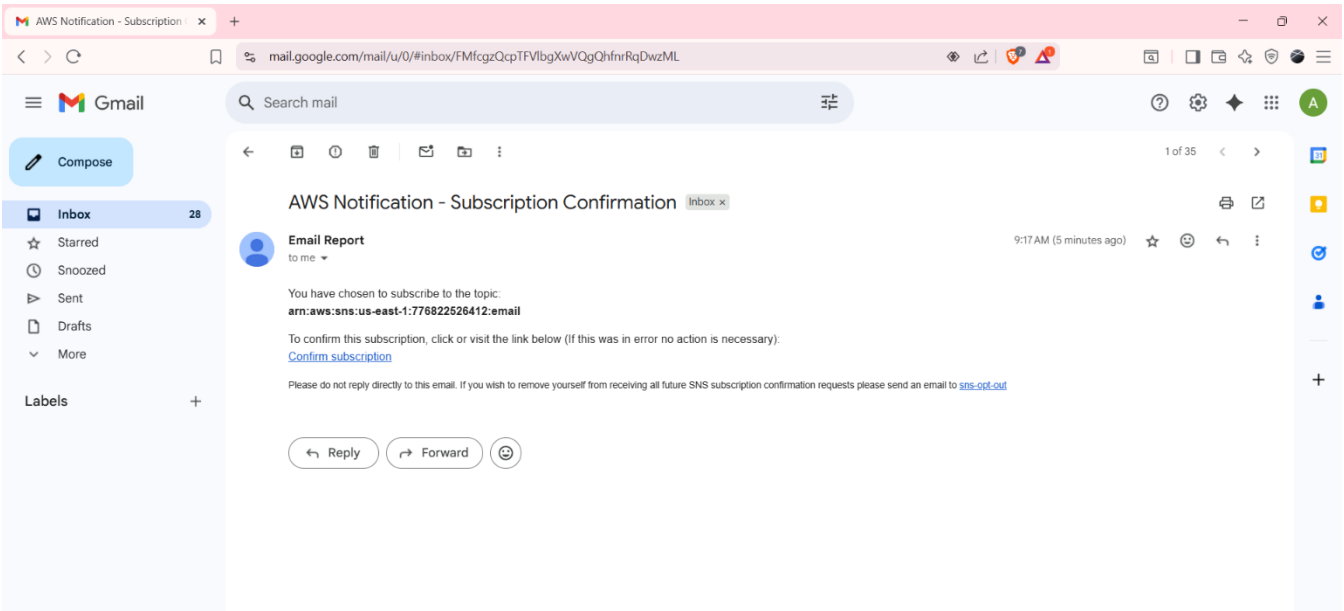
```
[ec2-user@ip-10-0-1-150 environment]$ chmod +x resources/setup.sh && resources/setup.sh
Please enter a valid IP address:
152.58.63.192
IP address:152.58.63.192
upload: resources/website/all_products_on_offer.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/all_products_on_offer.json
upload: resources/website/callback.html to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/callback.html
upload: resources/website/all_products.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/all_products.json
upload: resources/website/beans.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/beans.json
upload: resources/website/images/beans/excelsa.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/excelsa.png
upload: resources/website/config.js to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/config.js
upload: resources/website/images/items/blueberry_bagel.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry_bagel.pn
g
upload: resources/website/images/items/apple_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie.jpeg
upload: resources/website/images/items/blueberry_jelly_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberr
y_jelly_doughnut.jpeg
upload: resources/website/images/beans/robusta.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/robusta.png
upload: resources/website/images/items/boston_cream_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/boston_crea
m_doughnut.jpeg
upload: resources/website/images/expanded.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/expanded.png
upload: resources/website/images/beans/liberica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/liberica.png
upload: resources/website/images/items/apple_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie_slice.pn
g
upload: resources/website/images/items/apple_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie.png
upload: resources/website/images/beans/arabica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/arabica.png
upload: resources/website/images/items/boston_cream_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/boston_crear
_doughnut.png
upload: resources/website/favicon.ico to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/favicon.ico
upload: resources/website/images/items/apple_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie_slice.j
peg
upload: resources/website/images/items/cherry_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie.png
upload: resources/website/images/items/blueberry_bagel.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry_bagel.j
peg
upload: resources/website/images/items/blueberry_jelly_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry
_jelly_doughnut.png
upload: resources/website/images/items/cherry_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie_slice.
png
upload: resources/website/images/items/cherry_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie.jpeg
upload: resources/website/images/items/cherry_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie_slice
.jpeg
upload: resources/website/images/items/chocolate_chip_cupcake.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/chocolate_
```

Layout: US

8. Verify the AWS CLI version and also verify that the SDK for Python is installed.
   - Confirm that the AWS CLI is now at version 2 by running the **`aws --version`** command.
   - In the VS Code Bash terminal (at the bottom of the IDE), run the following command: **`pip3 show boto3`**

```
[ec2-user@ip-10-0-1-234 environment]$ aws --version
aws-cli/2.28.6 Python/3.13.4 Linux/6.1.147-172.266.amzn2023.x86_64 exe/x86_64.amzn.2023
[ec2-user@ip-10-0-1-234 environment]$ pip3 show boto3
Name: boto3
Version: 1.40.6
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: /usr/local/lib/python3.11/site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
[ec2-user@ip-10-0-1-234 environment]$
```

9. Confirm the Amazon Simple Notification Service (Amazon SNS) email subscription.
    a. Check your email for a message from AWS Notifications.
    b. In the email body, choose the **Confirm subscription** link.
    c. A webpage opens and displays a message that the subscription was successfully confirmed. You can close this page.



10. Verify that you can access the café website through Amazon CloudFront, and test the current login button behavior.

- Navigate to the CloudFront console.

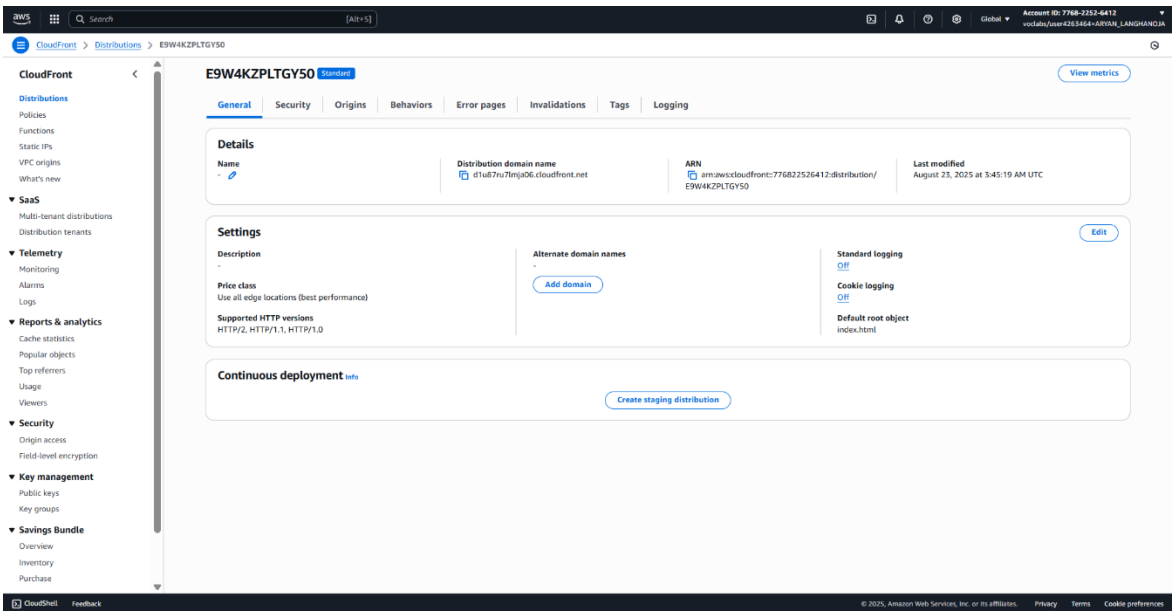
- Choose the hyperlink for the distribution that was created for the lab. You can get the ID from the script output you ran earlier. Look for the message *This is the Cloudfront Distribution created for the lab:* in the output.
- Copy the **Distribution domain name** value, and then load that URL in a new browser tab.

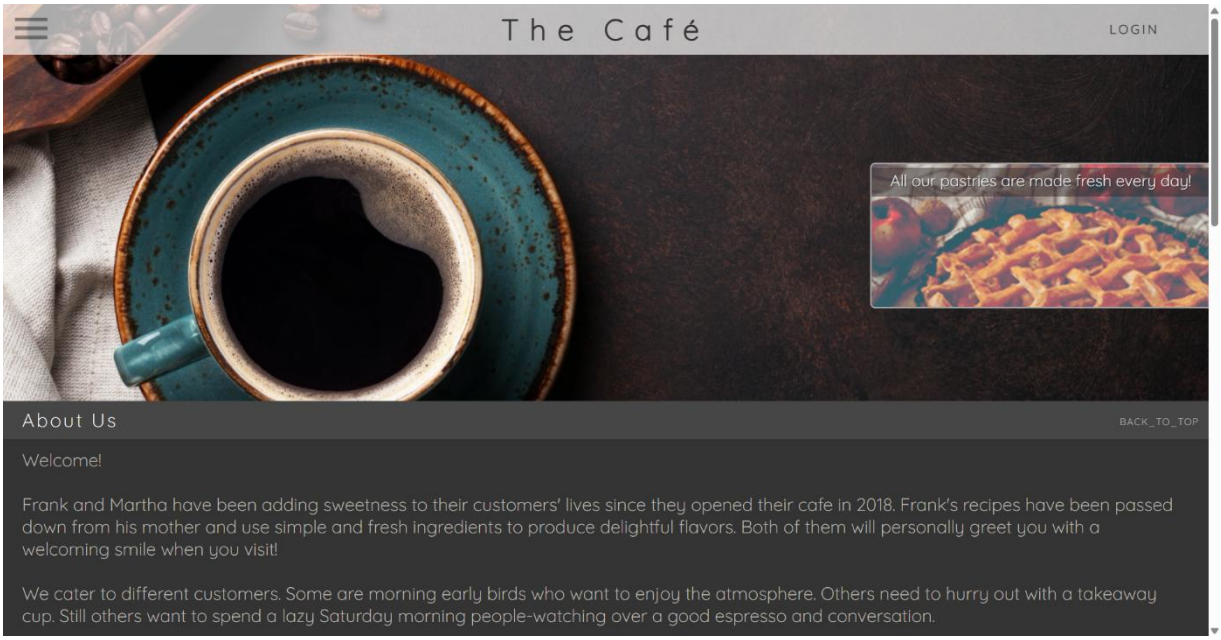
The café website displays. If it doesn't, see the following troubleshooting tip.

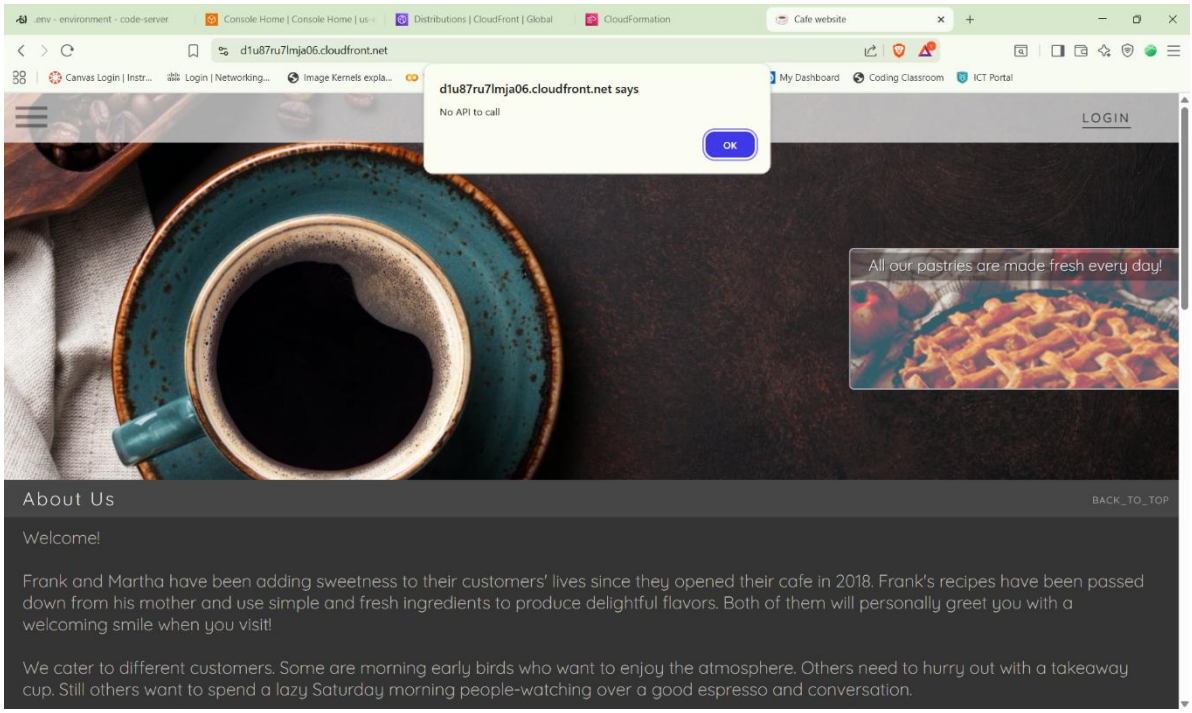
- Choose the **LOGIN** link in the upper-right corner.

Notice that the link does not currently work. Instead, it returns the message *No API to call*. This is expected. You will configure the login functionality during this lab.

- Keep the café website open in this browser tab. You will return to it later in this lab.



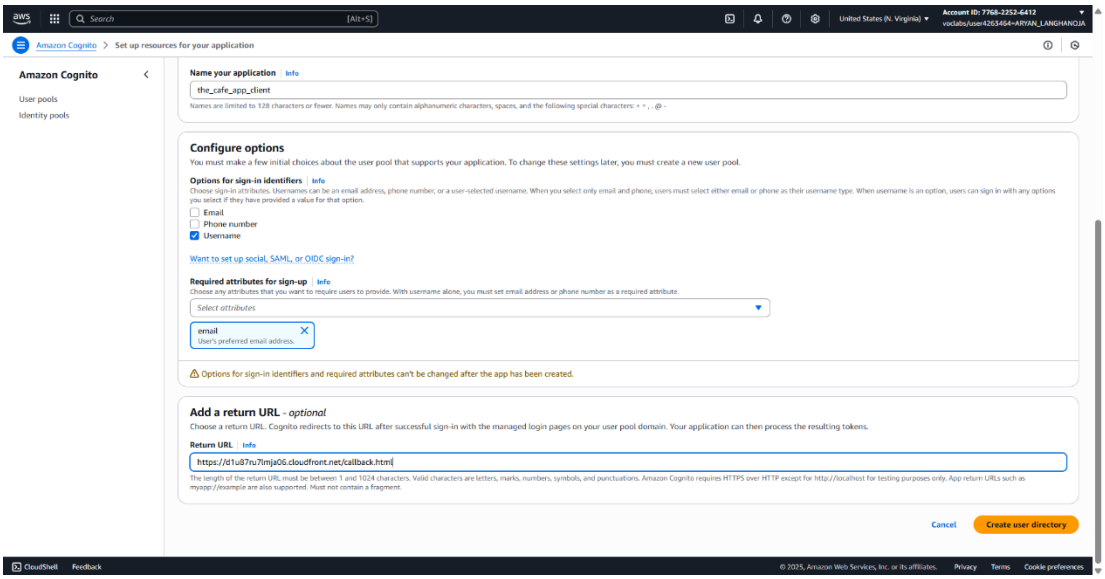




Task 2: Configuring an Amazon Cognito user pool and app client

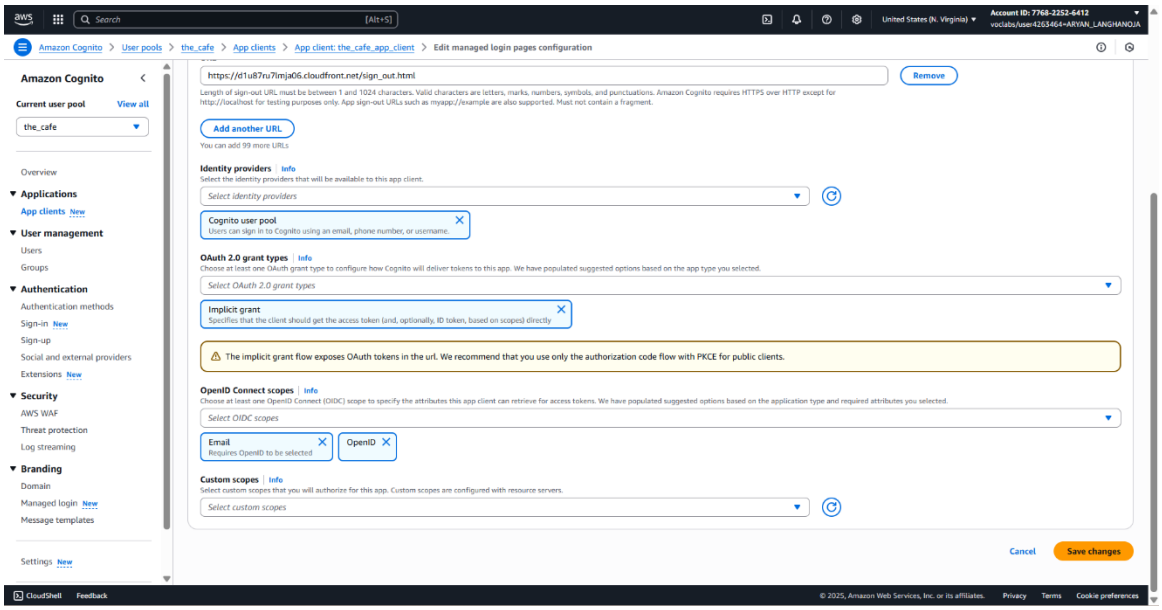
11. Begin to create an Amazon Cognito user pool.

- Navigate to the Amazon Cognito console.
- Choose **Create user pool**.
- If this is not visible, choose menu and choose **User pools** and then choose **Create user pool**.
- On the **Set up resources for your application** page, configure the following options:
- For **Application type**, ensure that **Traditional web application** is already selected.
- For **Name your application**: the_cafe_app_client
- Under the **Configure options**, choose **Username**
- For **Required attributes for sign-up**, choose **email** from the dropdown.
- For **Add a return URL - optional**, enter https://<cloudfront-domain>/callback.html, and replace with the CloudFront distribution domain from your text editor.
- **Note:** The updated URL should be similar to the following: **https://d123456acbdef.cloudfront.net/callback.html**
- Choose **Create user directory** button.

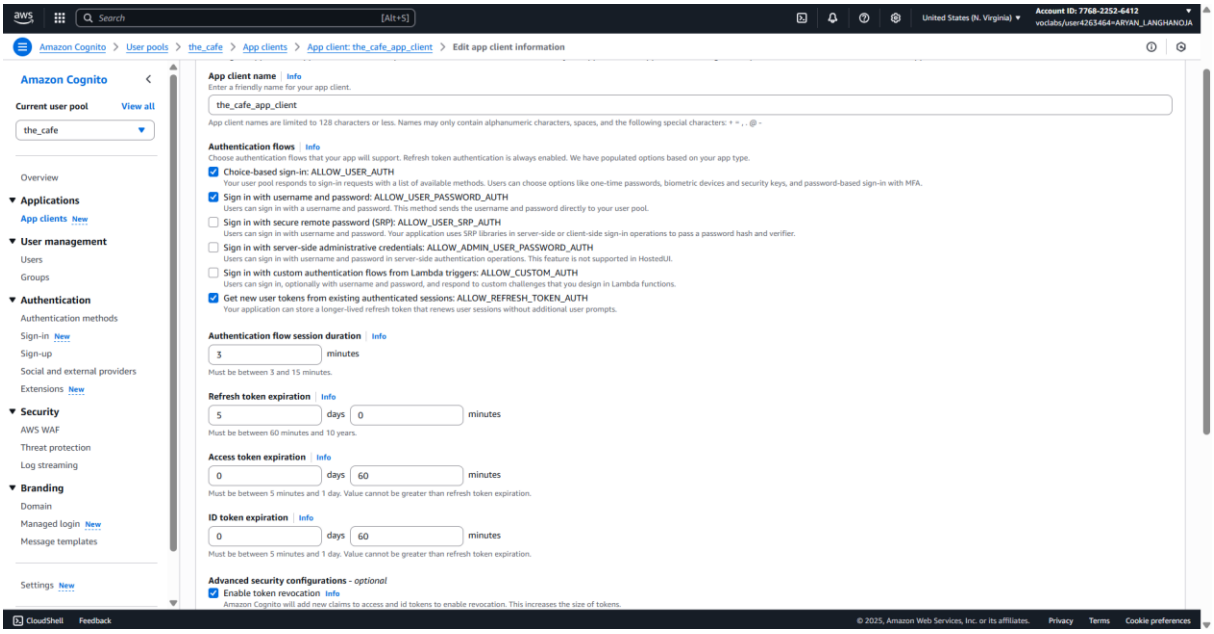


- On the next page displayed, under **Check out your sign-in page**, choose **View login page**.
- Notice the **Sign-in** page which you will configure later, close the browser tab.
- Go back to the **User pools** and choose the pool you created.
- User pool Name should be similar to *User pool - zzzzzz*
- On the **Overview:** page, you notice the pool details and other configuraiton information.
- Choose **Rename** button and under **User pool name** enter the_cafe, choose **Save changes**.
- Copy the value for **User pool ID** to the editor.
- From the bottom **Recommendations** pane, choose **the_cafe_app_client** link.
- Go to **Login pages**:
- On the **Managed login pages configuration** choose **edit**.
- Under **Allowed sign-out URLs - optional**:
- Choose **Add sign-out URL**
- in the **URL** field, enter https://<cloudfront-domain>/sign_out.html, and replace the placeholder with the CloudFront distribution domain.
- For **OAuth 2.0 grant types**, **Clear** the **Authorization code grant**.
- From the dropdown list, choose **Implicit grant**.
- For **OpenID Connect scopes**, ensure that **Email** and **OpenID** are chosen, and clear **Phone**.
- Choose **Save changes** button.

Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Implement application authentication using Amazon cognito.**	
Experiment No: 13	**Date:**	**Enrolment No: 92200133030**



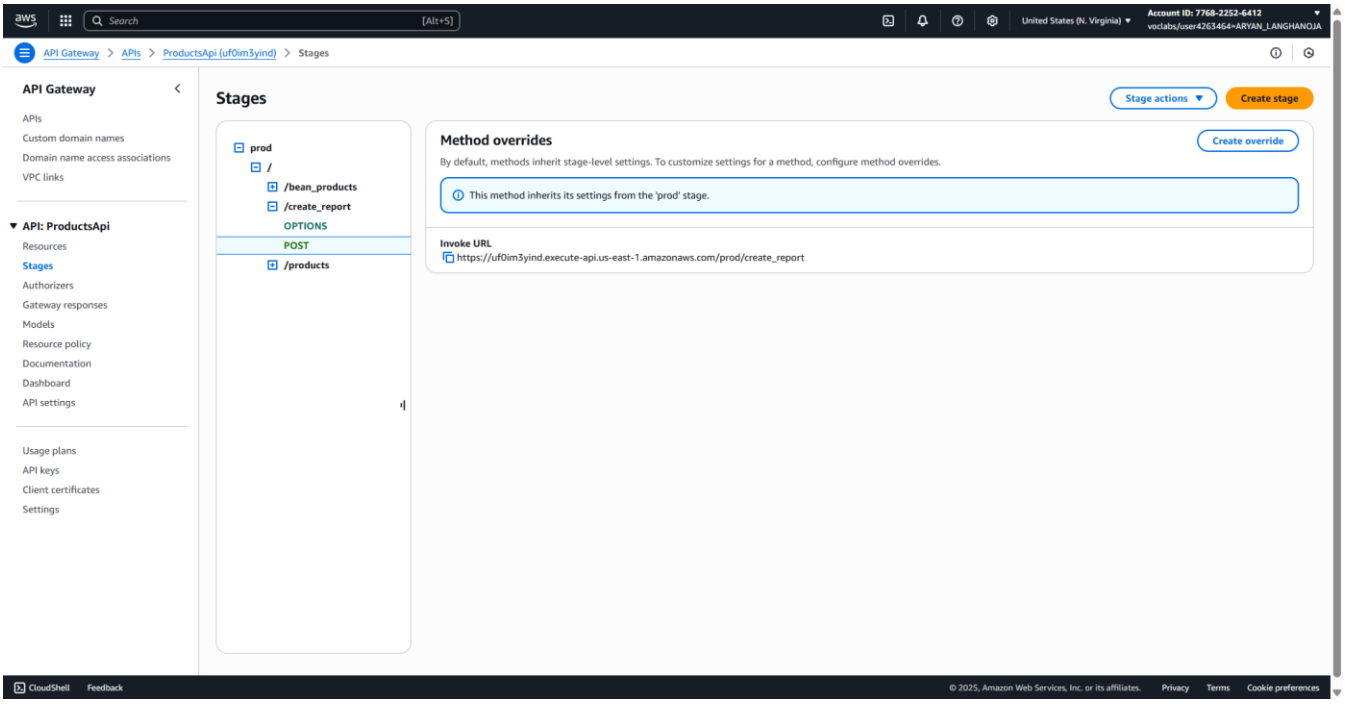
- On the **App client: the_cafe_app_client** page, choose **Edit**.
- For **Authentication flows**, from the dropdown list
- Choose **ALLOW_USER_PASSWORD_AUTH**,
- Clear **ALLOW_USER_SRP_AUTH**.
- Choose **Save changes**.
- Next you create and add users to the user pool.

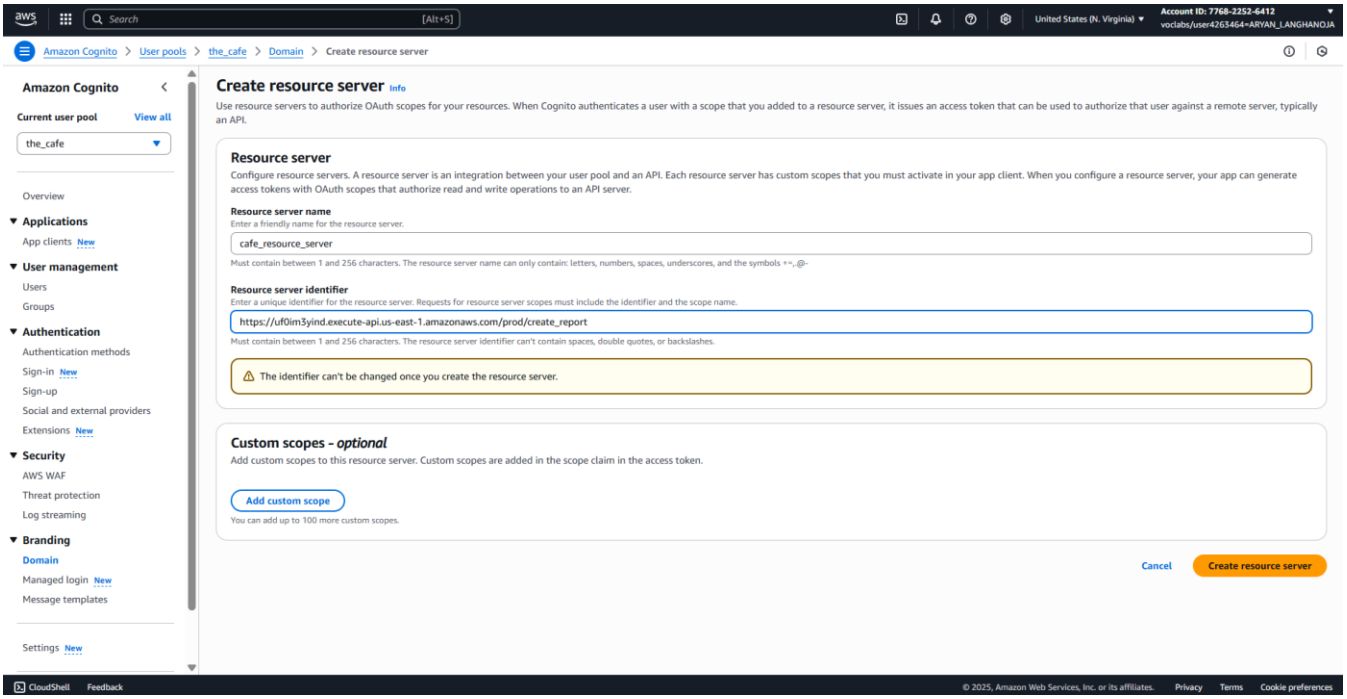


Task 3: Configuring the app client

12. Configure a resource server for the user pool to call the *create_report* REST API endpoint.

- In the **the_cafe** user pool page, from the navigation on the left, choose the **Domain**.
- In the Resource servers area, choose **Create resource server** and configure:
  - **Resource server name:** Enter `cafe_resource_server`
  - **Resource server identifier:** Use the following steps to retrieve the invoke URL and paste it here.
- To find the invoke URL:
  - In a new browser tab, navigate to the API Gateway console.
  - Choose the link for **ProductsApi**.
  - In the left navigation pane, choose **Stages**.
  - In the **Stages** tree, expand the **prod** stage.
  - Under **/create_report**, choose **POST**.
  - Copy the **Invoke URL**, which looks similar to the following: *https://some-unique-string.execute-api.us-east-1.amazonaws.com/prod/create_report*





13. Confirm that the hosted UI is now available.

- Choose overview, and choose the **the_cafe_app_client** link
- Choose **View login page**.
- A webpage opens and displays a login screen.
- Copy the full URL from the address bar in **your browser** to your clipboard.
- The URL looks similar to the following,

```
https://us-east-1dpnepwe7y.auth.us-east-1.amazoncognito.com/login?client_id=1t54jjqpknkrgefm76hnsmomhl&response_type=token&scope=email+openid&redirect_uri=https%3A%2F%2Fd1u87ru7lmja06.cloudfront.net%2Fcallback.html
```

Task 4: Integrating the Amazon Cognito hosted UI into the website

14. Configure the website to invoke the new app.

- Return to the VS Code IDE.
- Open the **resources/website/config.js** file in the text editor.
    - **Analysis**: This is a copy of the configuration file in Amazon S3 where the website .js and .html files are hosted. You will use the config.js file to integrate authentication into the café website. As you can see, `COGNITO_LOGIN_BASE_URL_STR` does not have a value yet. In this task, you will update this local copy of the configuration file and then upload it to Amazon S3 to deploy the change to the application behavior.
- Replace the `null` value for COGNITO_LOGIN_BASE_URL_STR with the hosted login URL that you just copied. Surround the URL in double quotation marks.

```
.env    JS config.js
resources > website > JS config.js > ...
1  window.COFFEE_CONFIG = {
2      API_GW_BASE_URL_STR: "https://uf0im3yind.execute-api.us-east-1.amazonaws.com/prod",
3      COGNITO_LOGIN_BASE_URL_STR: "https://us-east-1dpnepwe7y.auth.us-east-1.amazoncognito.com/login?client_id=1t54jjqpknkrgefm76hnsm
4  };
```

- Close the file using X, the changes to the file are saved automatically.
- In the VS Code IDE Bash terminal, to update the copy of the config.js file that is hosted on Amazon S3, run the following two commands.

```
bucket=$(aws s3api list-buckets --query "Buckets[].Name" | grep
s3bucket | tr -d ',' | sed -e 's/"//g' | xargs)
aws s3 cp ./resources/website/config.js s3://$bucket/ --cache-control
"max-age=0"
```

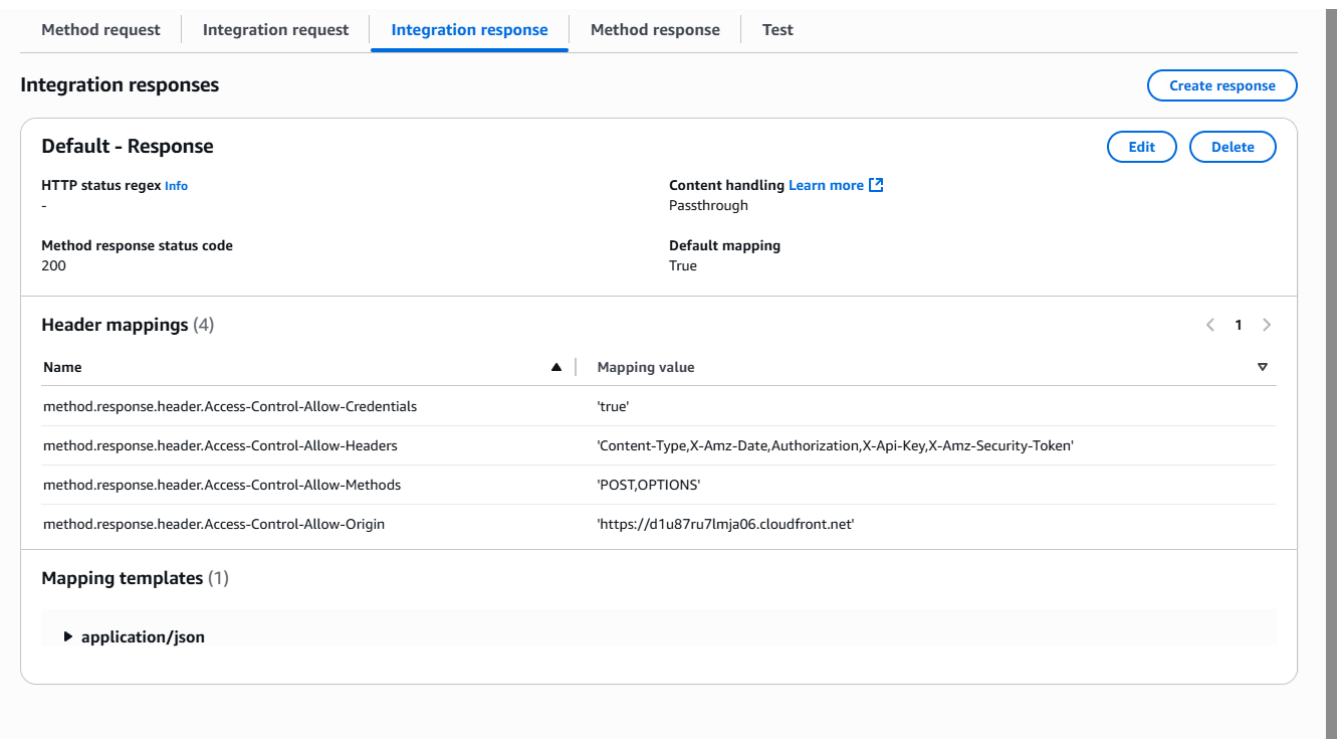
```
[ec2-user@ip-10-0-1-213 environment]$ bucket=$(aws s3api list-buckets --query "Buckets[].Name" | grep s3bucket | tr -d ',' | sed -e 's/"//g' | xargs)
[ec2-user@ip-10-0-1-213 environment]$ aws s3 cp ./resources/website/config.js s3://$bucket/ --cache-control "max-age=0"
upload: resources/website/config.js to s3://c168617a4340264l11304714t1w776822526412-s3bucket-pxwxgazbee3l/config.js
[ec2-user@ip-10-0-1-213 environment]$
```

15. Confirm the café website's updated behavior.

- Return to the browser tab where you have the café website open.
    - **Note**: If you no longer have the café website tab open, browse to the CloudFront console, choose the link for the distribution, and copy the **Distribution domain name** value. Load that URL in a new browser tab.
- Refresh the browser tab to load the change to the website configuration into the browser session.
- Choose the **LOGIN** link in the upper-right corner.

Task 5: Observing the REST API endpoint details and testing

16. Observe the CORS settings for the *create_report* API endpoint.
    - Navigate to the API Gateway console.
    - Choose the link for **ProductsApi**.
    - Choose **Resources**.
    - In the **Resources** tree, under **/create_report**, choose **OPTIONS**.
    - Choose **Integration Response**.
    - Expand the existing row that has a **200** method response status.
    - Expand the **Header Mappings** section.
    - Notice that the **Access-Control-Allow-Origin** setting already points to your CloudFront distribution and looks similar to the following (the unique-string value in your setting will be a unique string):



Test sending a report from the API Gateway console.

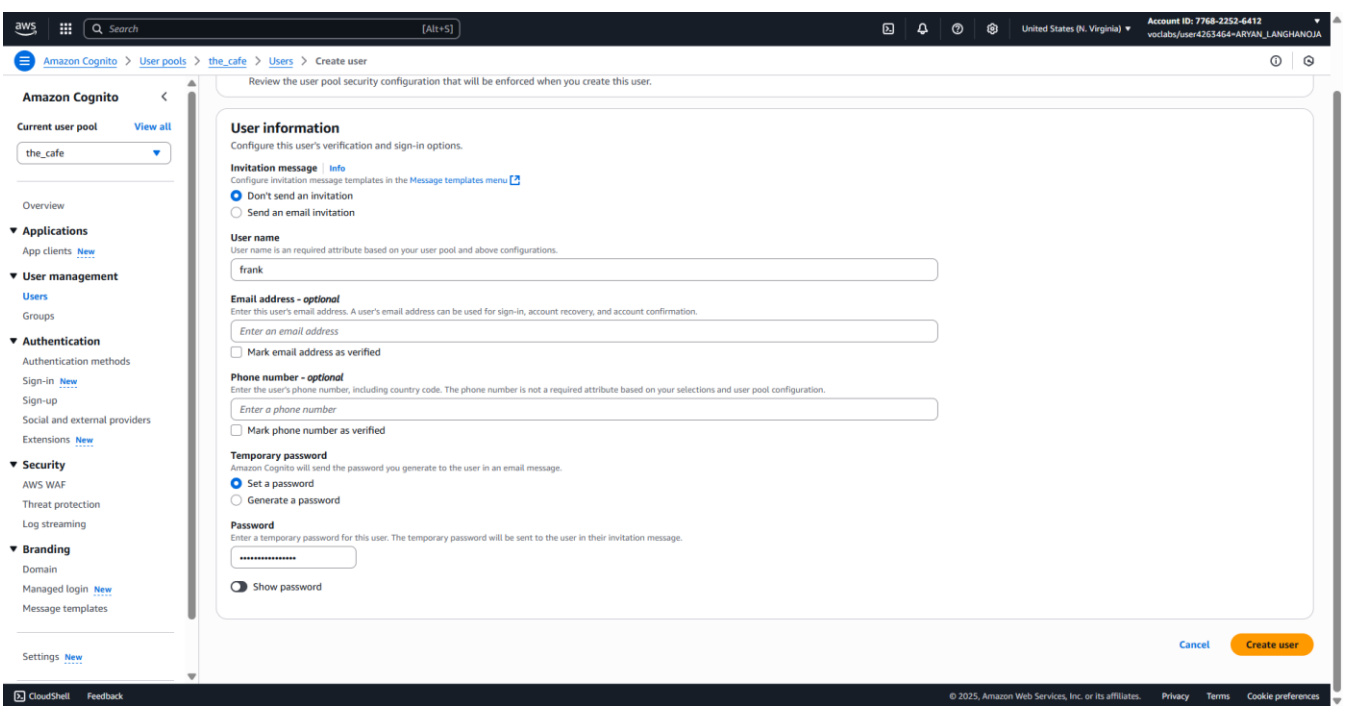
17. In the **Resources** tree, under **/create_report**, choose **POST**.

- From the lower pane, choose **TEST** , and then scroll down and choose the **Test** button.
    - Observe the *Response Body* and *Response Headers* in the test panel. JSON code is displayed in both.
    - Also observe the *Logs* panel details. A final line states `Method completed with status: 200`.
- Check your email. You receive an email with a presigned URL for the inventory report. Open the presigned URL in a new browser tab or window.

- **Note**: It might take a minute or two for the email to arrive. By choosing **Test**, you invoked the Step Functions state machine, which in turn invokes a few Lambda functions to generate the report.
- **Troubleshooting tip if you don't receive the email**: If you do not receive the email after waiting at least 2 minutes, follow these steps:
- In a new browser tab, navigate to the **WAF & Shield** console.
- In the navigation pane, choose **IP sets**.
- Choose the link for **office_regional** to view the details.
- Verify that the IP address entry contains your IP address as returned by https://whatismyipaddress.com with /32 appended.
- If your IP address is not listed, choose **Add IP address**, and add it (append /32 to the address).

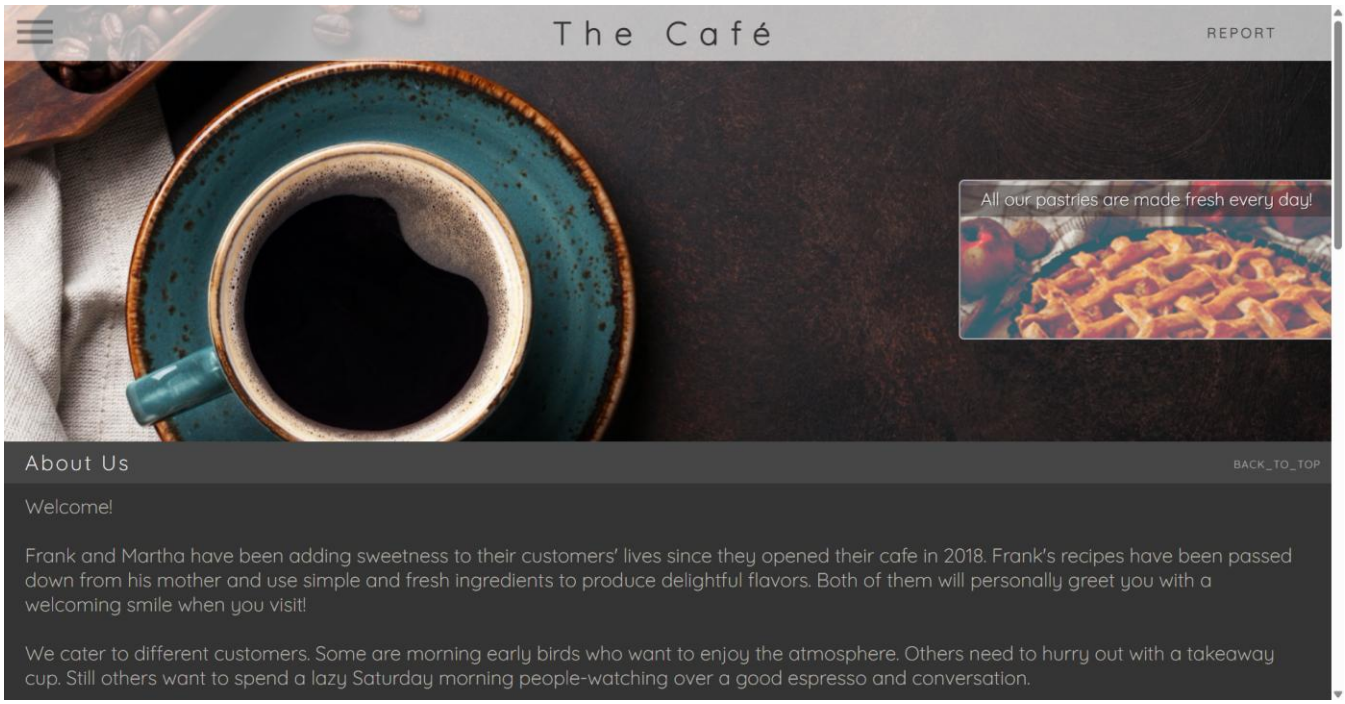
Task 6: Creating a user for the Amazon Cognito user pool

18. Create a user for the Amazon Cognito user pool.
    - In the Amazon Cognito console choose the link for the **the_cafe** user pool.
    - From the navigation pane, under **User management** choose **Users**.
    - In the Users panel, choose **Create user** and configure:
    - Invitation message: **Don't send an invitation**
    - Username: frank
    - Temporary password: Set a password
    - **Password**: Enter a password that you will remember; for example: !CoffeeIsGreat34
    - Choose **Create user**.
    - A new row appears on the **Users** tab with the details of the new user.



19. Test logging in as Frank.

- Return to the café website.
  - If you closed it, you can load the website by using the CloudFront distribution domain name.
- Choose **LOGIN** and enter the credentials that you just created:
  - **Username:** Enter `frank`
  - **Password:** Enter the password that you created a moment ago.
  - Choose **Sign in**.
  - If successful, you are prompted to enter a new password.
    - For example, you could enter: `!CoffeeIsGreat35`
    - Enter the new password in both fields, and then choose **Change password**.
  - If successful, you see a link named **REPORT** where the LOGIN button previously appeared.
    - **Analysis**: The website redirected the *frank* user to the callback page, and then it immediately returned the user to the café website.
  - Choose the **REPORT** hyperlink.
    - You see the message *Report is being generated, please check your email*.
  - Check your email, and verify that you can access the report by using the presigned URL that is included in the email.



20. Test the ability to bypass the website and instead invoke *create_report* by using a curl command.

- In the VS Code IDE Bash terminal, run the following command. Replace the URL with the *create_report* invoke URL, which you copied from the API Gateway console earlier.

```
curl -X POST https://unique-string.execute-api.us-east-1.amazonaws.com/prod/create_report
```

- You see the following response:

```
{"message":"Forbidden"}
```

```
[ec2-user@ip-10-0-1-213 environment]$ curl -X POST https://uf0im3yind.execute-api.us-east-1.amazonaws.com/prod/create_report
{"message":"Forbidden"}[ec2-user@ip-10-0-1-213 environment]$
```

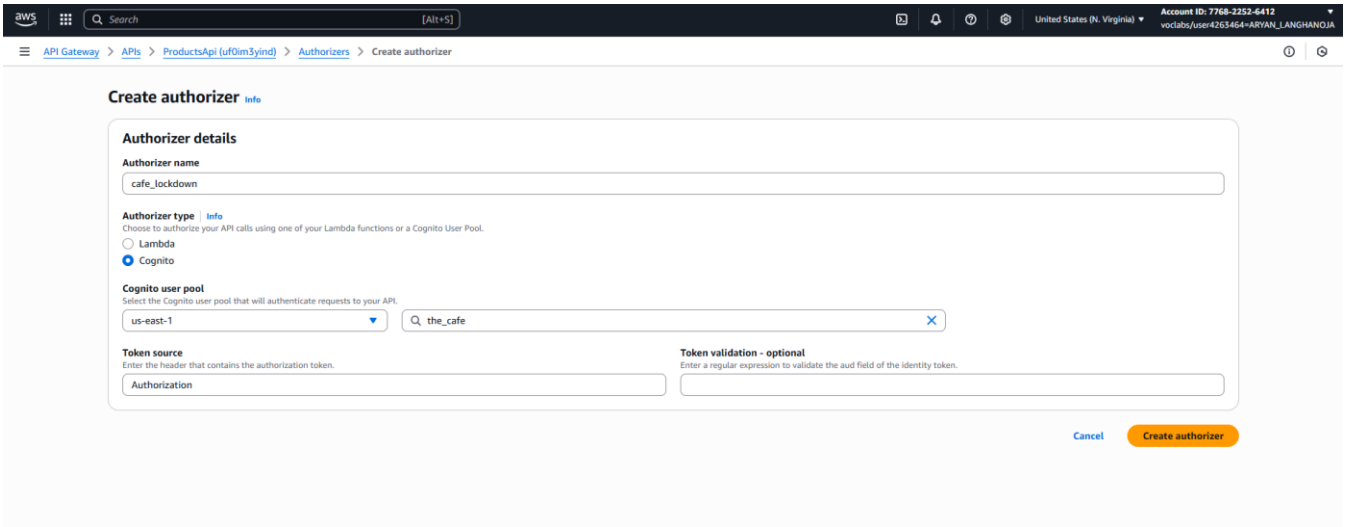
- Now try to run the same command from your local machine. For example, if you are using a Windows machine, run the command in a Command Prompt window. If you are using macOS, run the command in a Terminal window.
- You see a response similar to the following:

```
C:\Users\aryan>curl -X POST https://uf0im3yind.execute-api.us-east-1.amazonaws.com/prod/create_report
{"executionArn":"arn:aws:states:us-east-1:776822526412:express:MyStateMachine:16e943c6-d83a-454f-acb8-cd736a817bd4:14865fe5-51f7-4415-8a95-cb609942bc16","startDate":1.755924665567E9}
C:\Users\aryan>
```

Task 7: Configuring an API Gateway authorizer

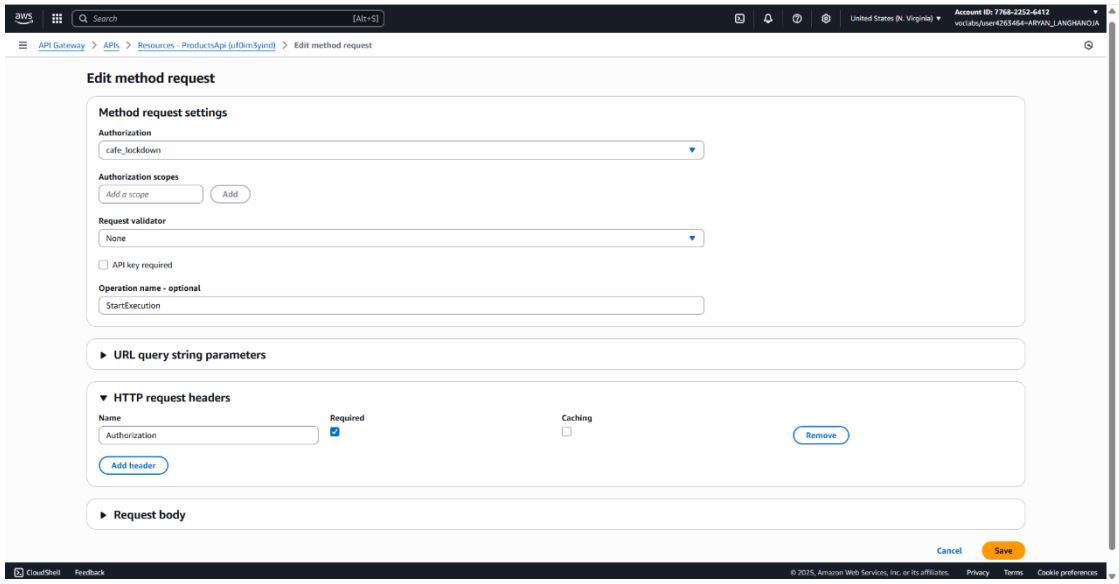
21. Add an authorizer to the *ProductsApi* REST API.

- Return to the API Gateway console.
- Choose the link for **ProductsApi**.
- In the navigation pane, choose **Authorizers**.
- Choose **Create authorizer** and configure:
- **Name:** Enter cafe_lockdown
- **Type:** Choose **Cognito**.
- **Cognito User Pool:** Choose the dialog box to the right of **us-east-1**, and then select **the_cafe**.
- **Token Source:** Enter Authorization
- Keep **Token Validation** blank.
- Choose **Create authorizer**.
- The page refreshes.



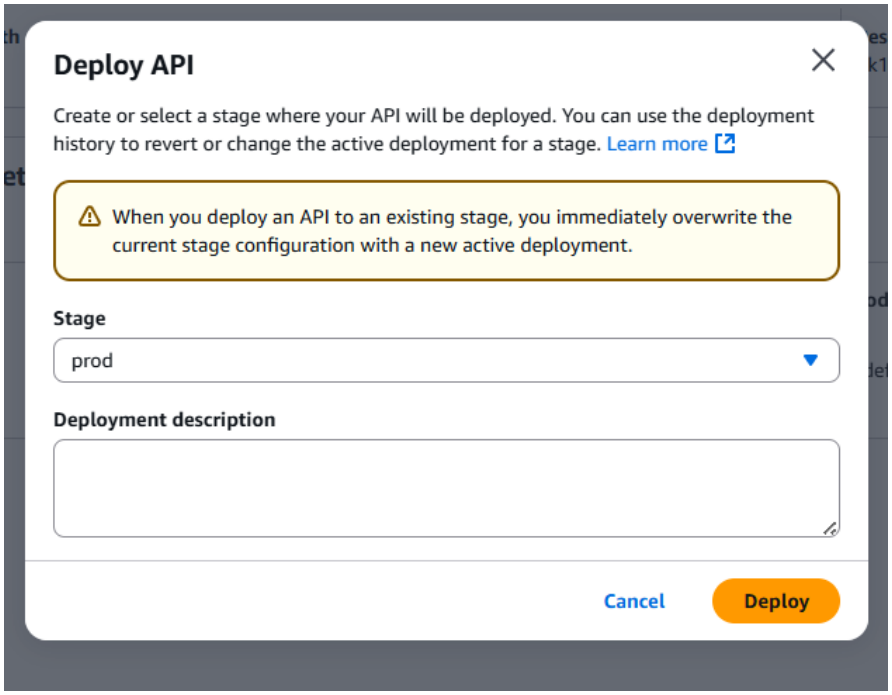
22. Configure the *create_report* method request to call Amazon Cognito to verify authorization.

- Refresh the API Gateway console page.
- In the navigation pane, choose **Resources**.
- In the **Resources** tree, under **/create_report**, choose **POST**.
- Choose **Method Request**.
  - In the **Settings** area, notice that **Authorization** is currently set to NONE.
- Edit the authorization setting:
  - Choose the pencil icon next to **Authorization**.
  - In the dropdown menu, under **Cognito user pool authorizers**, choose **cafe_lockdown**.
  - To save the setting change, choose the check mark icon.
- Expand the **HTTP Request Headers** section.
- Choose **Add header** and configure:
  - For **Name**, enter `Authorization`
  - To save the setting change, choose the check mark icon.
  - Select **Required**.



23. Deploy the REST API updates.

- In the **Resources** tree, choose /, which is the root of the API.
- From the **Actions** menu, choose **Deploy API** and configure:
  - **Deployment stage:** Choose **prod**.
  - Choose **Deploy**.
- Choose **Save Changes**.



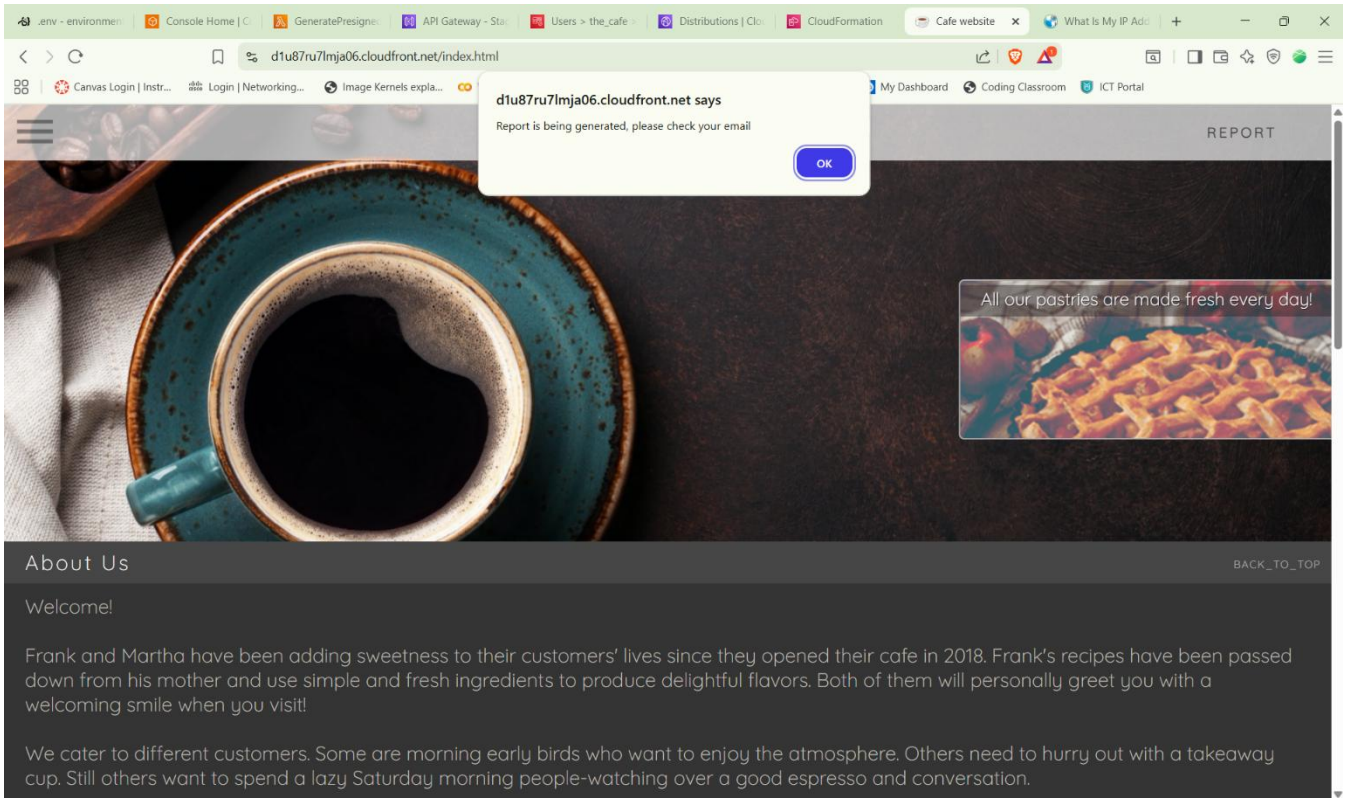
24. Test sending a report by using the `curl` command again.

- Return to the Command Prompt or Terminal on your local machine.
- Press the Up arrow key to load the same `curl` command that you ran previously. Then, press Enter on your keyboard to run it.

```
C:\Users\aryan>curl -X POST https://uf0im3yind.execute-api.us-east-1.amazonaws.com/prod/create_report
{"message":"Unauthorized"}
C:\Users\aryan>
```

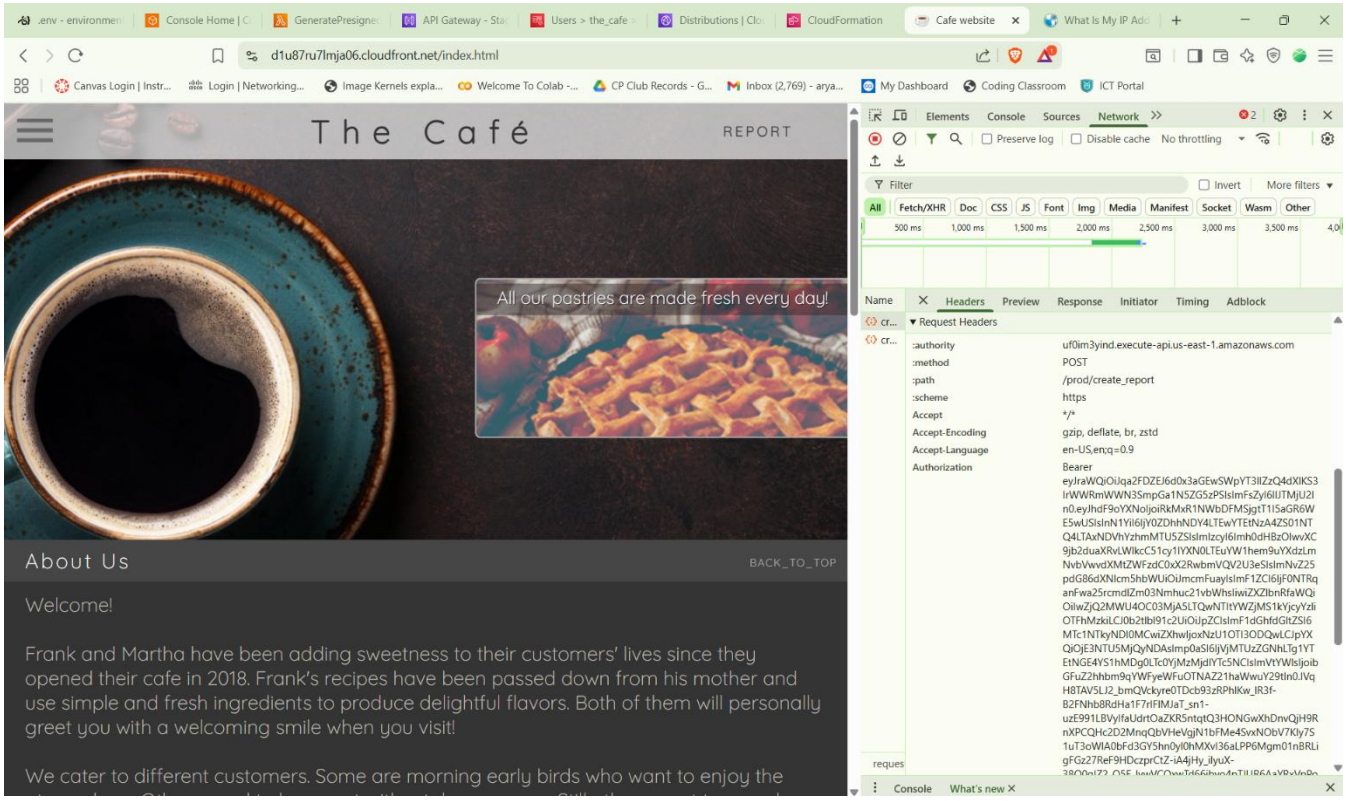
Task 8: Testing the request process from the website

25. Test the ability to create the report from the website.
    - Return to the café website.
    - If you closed it, you can load the website by using the CloudFront distribution domain name.
    - If you are not already logged in, choose **LOGIN** and enter the credentials that you created previously:
    - **Username:** Enter frank
    - **Password:** Enter the password; for example: !CoffeeIsGreat35
    - Choose **Sign in**.
    - On the website, choose the **REPORT** hyperlink.
    - You see the message *Report is being generated, please check your email*.
    - Check your email, and verify that you can access the report by using the presigned URL.



26. *Optional* step: Find the authorization token in the browser session and test it with the curl command.
    - On the café website page, open the context menu (right-click) and choose **Inspect** (if using Chrome) or **Inspect Element** (if using Firefox).
    - In the developer tools area that appears, choose the **Network** tab.
    - On the café website, choose the **REPORT** hyperlink again.
    - A list of all the files and accessed locations displays in the developer tools area.
    - Choose (double-click) the **create_report** entry for the POST action that appears.
    - **Note:** Depending on your browser, you might need to choose **All** to see the entry. In Chrome, the entry type for *create_report* is xhr. In Firefox, choose the *create_report* entry that shows POST in the **Method** column.
    - Observe the *Request Headers* details. Notice the field that shows authorization and contains the word Bearer followed by a long credentials value.
    - Copy the entire string value, including Authorization: Bearer.

Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Implement application authentication using Amazon cognito.**	
Experiment No: 13	**Date:**	**Enrolment No: 92200133030**



- Return to the Command Prompt window (or Terminal window) on your local machine.
- Press the Up arrow key to load the last curl command that you ran. Do not run the command yet.
- At the end of the command, add -H. Then, paste in the credentials value that you just copied from your browser. Surround the credentials value in double quotation marks.
- The resulting command looks similar to the following (where unique-string is part of your API Gateway invoke URL and VeryLongStringValue is the authorization token value that you copied):
- curl -X POST https://unique-string.execute-api.us-east-1.amazonaws.com/prod/create_report -H "Authorization: Bearer VeryLongStringValue"
- Run the command.

Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Implement application authentication using Amazon cognito.**	
Experiment No: 13	**Date:**	**Enrolment No: 92200133030**

```
C:\Users\aryan>curl -X POST https://uf0im3yind.execute-api.us-east-1.amazonaws.com/prod/create_report -H "Authorization:
Bearer eyJraWQiOiJqa2FDZEJ6d0x3aGEwSWpYT3lIZzQ4dXlKS3IrWWRmWWN3SmpGa1N5ZG5zPSIsImFsZyI6IlJTMjU2In0.eyJhdF9oYXNoIjoiRkMxR
1NWbDFMSjgtT1I5aGR6WE5wUSIsInN1YiI6IjY0ZDhhNDY4LTEwYTEtNzA4ZS01NTQ4LTAxNDVhYzhmMTU5ZSIsImlzcyI6Imh0dHBzOlwvXC9jb2duaXRvL
WlkcC51cy1lYXN0LTEuYW1hem9uYXdzLmNvbVwvdXMtZWFzdC0xX2RwbmVQV2U3eSIsImNvZ25pdG86dXNlcm5hbWUiOiJmcmFuayIsImF1ZCI6IjF0NTRqa
nFwa25rcmdlZm03Nmhuc21vbWhsIiwiZXZlbnRfaWQiOiIwZjQ2MWU4OC03MjA5LTQwNTItYWZjMS1kYjcyYzliOTFhMzkiLCJ0b2tlbl91c2UiOiJpZCIsI
mF1dGhfdGltZSI6MTc1NTkyNDI0MCwiZXhwIjoxNzU1OTI3ODQwLCJpYXQiOjE3NTU5MjQyNDAsImp0aSI6IjVjMTUzZGNhLTg1YTEtNGE4YS1hMDg0LTc0Y
jMzMjdlYTc5NCIsImVtYWlsIjoibGFuZ2hhbm9qYWFyeWFuOTNAZ21haWwuY29tIn0.IVqH8TAV5LJ2_bmQVckyre0TDcb93zRPhlKw_lR3f-B2FNhb8RdHa
1F7rIFlMJaT_sn1-uzE991LBVyIfaUdrtOaZKR5ntqtQ3HONGwXhDnvQjH9RnXPCQHc2D2MnqQbVHeVgjN1bFMe4SvxNObV7Kly7S1uT3oWlA0bFd3GY5hn0
yl0hMXvI36aLPP6Mgm01nBRLigFGz27ReF9HDczprCtZ-iA4jHy_ilyuX-38Q0qIZ2_O5E_lvwVCQxwTd66jbyo4pTJUR6AaYRxVpPoU8--86jks7ViTCQd6
nq6JftAjJ2ZzeA3peEk8VcWD9JzGjPKir0UI7n1pBn8SOAjBw"
{"executionArn":"arn:aws:states:us-east-1:776822526412:express:MyStateMachine:57b0c279-3b71-4696-9832-623f17101903:a4c1e
7eb-285f-4181-ba64-61641684fad7","startDate":1.755925691938E9}
C:\Users\aryan>
```

Conclusion:-

- Connected to the VS Code IDE using provided credentials.
- Downloaded and extracted the lab resources:
  `wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/12-lab-cognito/code.zip -P /home/ec2-user/environment`
  `unzip code.zip`
- Upgraded AWS CLI and verified Python SDK:
  `chmod +x ./resources/setup.sh && ./resources/setup.sh`
  `aws --version`
  `pip3 show boto3`
- Confirmed Amazon SNS email subscription for report delivery.
- Created a user pool named `the_cafe` with:
  - App client: `the_cafe_app_client`
  - Username as the sign-in attribute.
  - Email as a required attribute.
  - Callback URL: `https://<cloudfront-domain>/callback.html`
- Configured OAuth 2.0 with Implicit Grant and OpenID Connect Scopes (Email, OpenID).
- Set Allowed Sign-Out URL: `https://<cloudfront-domain>/sign_out.html`
- Enabled ALLOW_USER_PASSWORD_AUTH authentication flow.
- Created a Resource Server for the REST API:
  - Name: `cafe resource server`
  - Identifier: API Gateway Invoke URL for `/create_report`
- Updated `config.js` with the Cognito Hosted UI URL:
  `COGNITO_LOGIN_BASE_URL_STR = "https://<cognito-domain>/login?..."`
- Uploaded the updated config to S3:

```
bucket=$(aws s3api list-buckets --query "Buckets[].Name" | grep
s3bucket | tr -d ',' | sed -e 's/"//g' | xargs)
aws s3 cp ./resources/website/config.js s3://$bucket/ --cache-control
"max-age=0"
```

- Created a user frank in the Cognito user pool with a temporary password.
- Tested login via the café website:
  - User prompted to change password on first login.
  - Successful login replaced LOGIN with REPORT button.
- Created a Cognito Authorizer named `cafe_lockdown` for the `ProductsApi.`
- Configured the `/create_report` POST method to use the authorizer.
- Added required Authorization header.
- Deployed the API to the prod stage.
- Before authorizer: curl -X POST <invoke-url> returned {"message":"Forbidden"}.
- After authorizer: Same command returned {"message":"Unauthorized"}.
- With valid token (from browser session):

  `curl -X POST <invoke-url> -H "Authorization: Bearer <token>"`
  Returned successful execution ARN and start date.
- Website integration: Clicking REPORT generated an email with a presigned URL to the inventory report.
- Verified CORS settings in API Gateway for /create_report:
  - Access-Control-Allow-Origin pointed to the CloudFront domain.
  - Headers and methods properly configured for cross-origin requests.

Result :-

Total score	25/25
[Task 1] Confirmed SNS subscription	5/5
[Task 2] Created the_cafe Cognito user pool	5/5
[Task 3] Created Resource Server for Cognito	5/5
[Task 6] Created frank user for the Cognito user pool	5/5
[Task 7] Created API Gateway authorizer	5/5