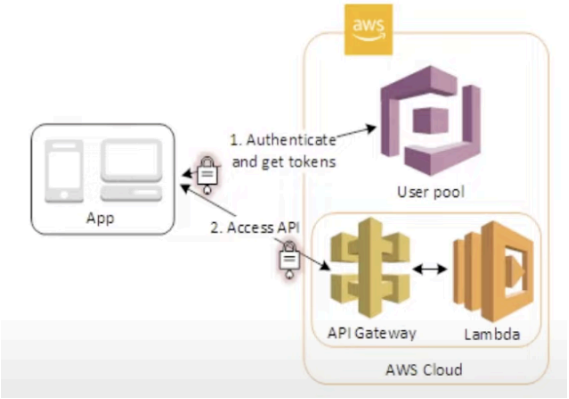
Task 1: Understand how passwords are held within cognito

- Something like this is abstracted. Cognito uses Oauth2.0 or other authentication policies in order to authenticate users (it is up to Team25).

Task 2: Understand how passwords are applied to other sources through cognito

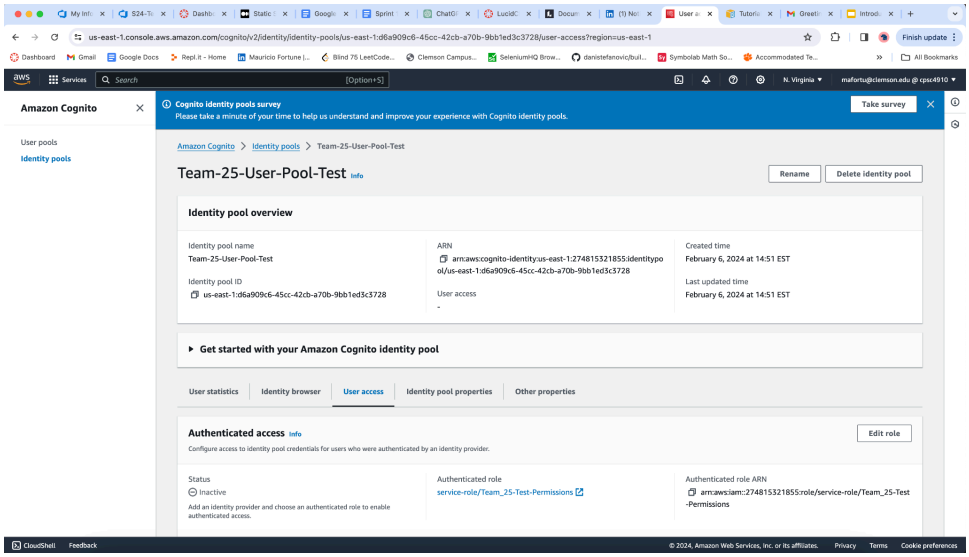
- Cognito offers authentication and authorization
- Will be useful to implement a login feature where only authorized users get to accessed our APIs
- Handles sign-on, registration, password resets, multi-factor authentication, can also leverage SSO with this
- Set up a user pool in order to do this and it appears that amazon also Identity roles are more for granting users access to AWS policies and IAM roles, that will not necessary for us so we will be using **User Pools**



https://youtu.be/oFSU6rhFETk?si=dcd9gxhfou8oT6ix

Task 3: Create a cognito instance in the dev sandbox

- Wasnt too bad but there are some features of it that we need to stand up with intent.



Task 4: Understand how lambda can be leveraged with React

- Lambda can be used in order to perform event-driven programming with react. Both instances need to be stood up in order to do this but they can both work in unison. Example I referenced to use a photo sharing app and info gets uploaded to s3:
    - Lambda code:

```
        # lambda_function.py
-
- import json
-
- def lambda_handler(event, context):
-     # Extract the details of the uploaded photo from the event
-     bucket_name = event['Records'][0]['s3']['bucket']['name']
-     object_key = event['Records'][0]['s3']['object']['key']
-
-     # Process the uploaded photo (for simplicity, just logging the details)
-     print(f'New photo uploaded: {object_key} in bucket {bucket_name}')
-
-     return {
-         'statusCode': 200,
-         'body': json.dumps('Photo processed successfully')
-     }
```

React integration

```
- // App.js
-
- import React, { useState } from 'react';
- import { Storage } from 'aws-amplify';
-
- function App() {
-   const [photoURL, setPhotoURL] = useState(null);
-
-   const handleFileUpload = async (event) => {
-     const file = event.target.files[0];
-
-     try {
-       // Upload the photo to the S3 bucket
-       await Storage.put(file.name, file);
-
-       // Display the uploaded photo in the UI
-       setPhotoURL(URL.createObjectURL(file));
-     } catch (error) {
-       console.error('Error uploading photo:', error);
-     }
-   };
-
```

```
-       return (
-         <div>
-           <h1>Photo Sharing App</h1>
-           <input type="file" onChange={handleFileUpload} />
-           {photoURL && <img src={photoURL} alt="Uploaded" />}
-         </div>
-       );
-     }
-
-     export default App;
```

Task 5: Stand up a lambda resource in AWS sandbox

- Pretty easy to do and quick, it was very straightforward. For lambda it will be more about figuring out the correct code.
-