Lab: Amazon Cognito

Lab overview and objectives

In this lab, you will use **Amazon Cognito** to create user pool and an application client.

After completing this lab, you should be able to:

- Create a user pool and application client from the **AWS Management Console**.
- Create a new **App Client.**
- Create a new user from Cognito App client login page.
- Connect an HTML page to the Cognito App client login page and use it to log in to your HTML page.
- Create a new user using Python SDK.

Scenario

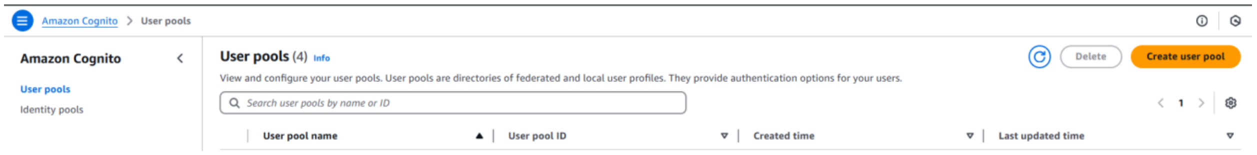
You will create a **User Pool** and **App Client** and connect an HTML page with the Cognito login page.

Accessing the AWS Management Console

For this lab, we will use the **AWS Academy learner lab** and the **AWS Management Console**.

Task 1: Create a User Pool

1. In the AWS Console search box to the right of **Services**, search for and choose **Cognito** to open the **Cognito** console.
2. Press **Create user pool**.



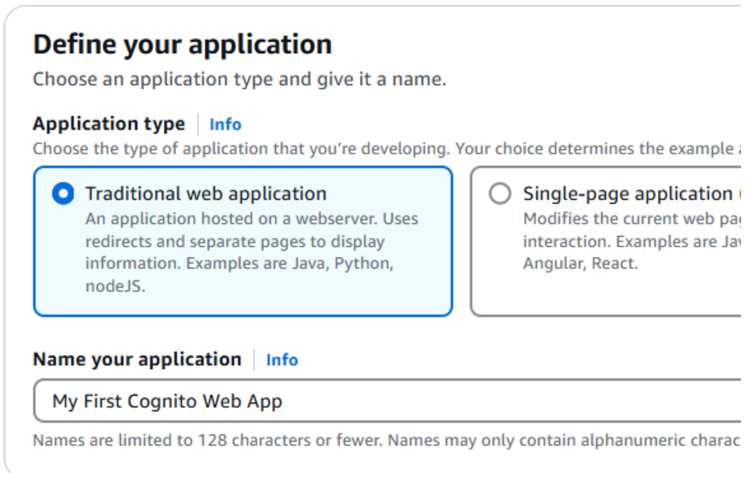
User pools (4) Info

View and configure your user pools. User pools are directories of federated and local user profiles. They provide authentication options for your users.

Search user pools by name or ID

User pool name	User pool ID	Created time	Last updated time

3. In the **Set up your application** screen, configure these settings:
   - In **Define your application** frame, set the **Name your application** name, as: My First Cognito Web App.



Define your application

Choose an application type and give it a name.

Application type | Info

Choose the type of application that you're developing. Your choice determines the example

[✓ Traditional web application] An application hosted on a webserver. Uses redirects and separate pages to display information. Examples are Java, Python, nodeJS.

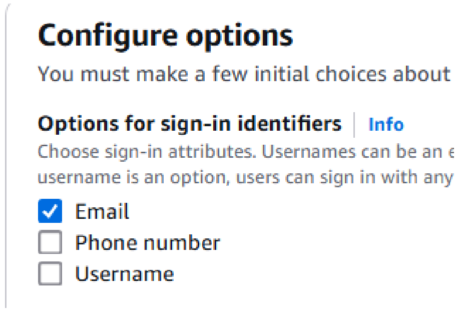
Single-page application Modifies the current web pa interaction. Examples are Ja Angular, React.

Name your application | Info

My First Cognito Web App

Names are limited to 128 characters or fewer. Names may only contain alphanumeric charac

- In **Configure options** frame, **Options for sign-in identifiers** check boxes, Select the check box **Email.**



Configure options

You must make a few initial choices about

Options for sign-in identifiers | Info

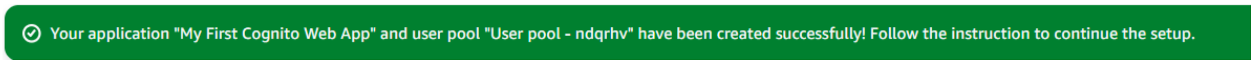
Choose sign-in attributes. Usernames can be an e

username is an option, users can sign in with any

- [✓] Email
- [ ] Phone number
- [ ] Username

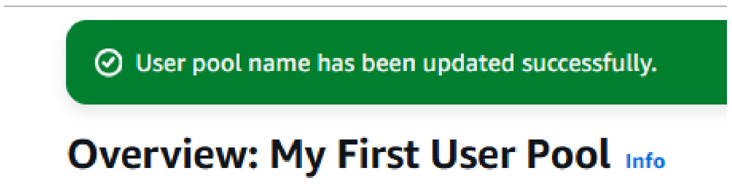
4. Press the **Create** button**.**

   Once done, an Upload succeeded message will appear.



Your application "My First Cognito Web App" and user pool "User pool - ndqrhv" have been created successfully! Follow the instruction to continue the setup.

5. Press the **Overview** button**.**
6. Press the **Rename** button and set the **User pool name** to My First User Pool, and press **Save changes**. A success message will appear, and you will see your user pool name has changed.

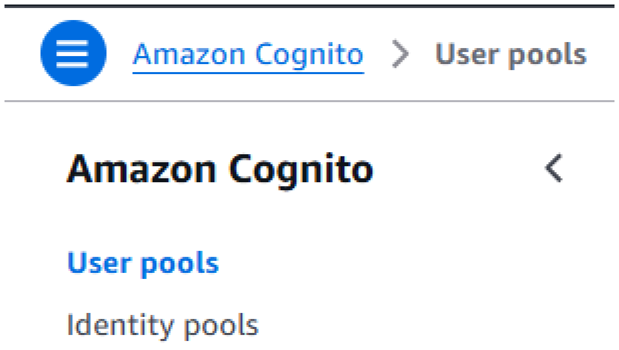


User pool name has been updated successfully.

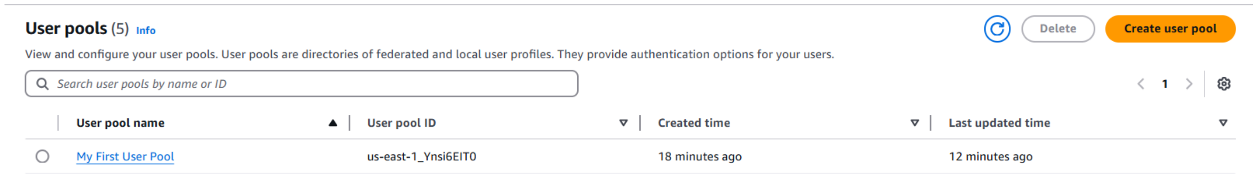
Overview: My First User Pool Info

Task 2: Create a user from Cognito application login page

1. In the AWS Console search box to the right of **Services**, search for and choose **Cognito** to open the **Cognito** console.
2. At the left menu, select the **User Pools** menu.



3. A list of all your User Pools will appear. Select the user pool you created **My First User Pool.**

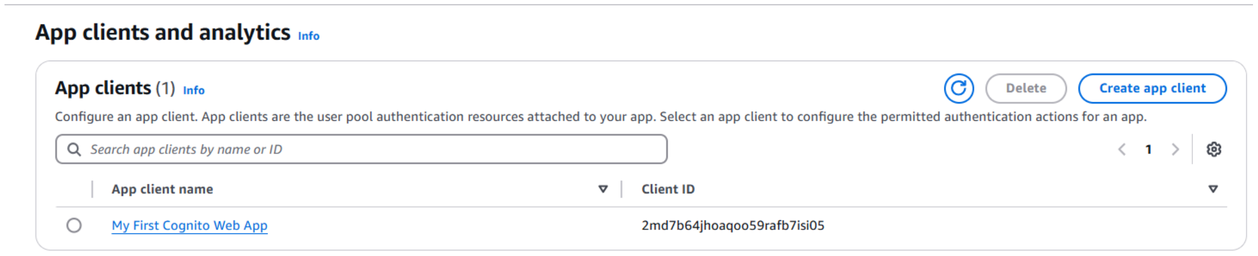

User pools (5) Info

View and configure your user pools. User pools are directories of federated and local user profiles. They provide authentication options for your users.

User pool name	User pool ID	Created time	Last updated time
My First User Pool	us-east-1_Ynsi6EIT0	18 minutes ago	12 minutes ago

The **Overview** page will appear.

4. At the left menu, under the **Applications** menu, select the **App Clients** sub menu. You will see your application:

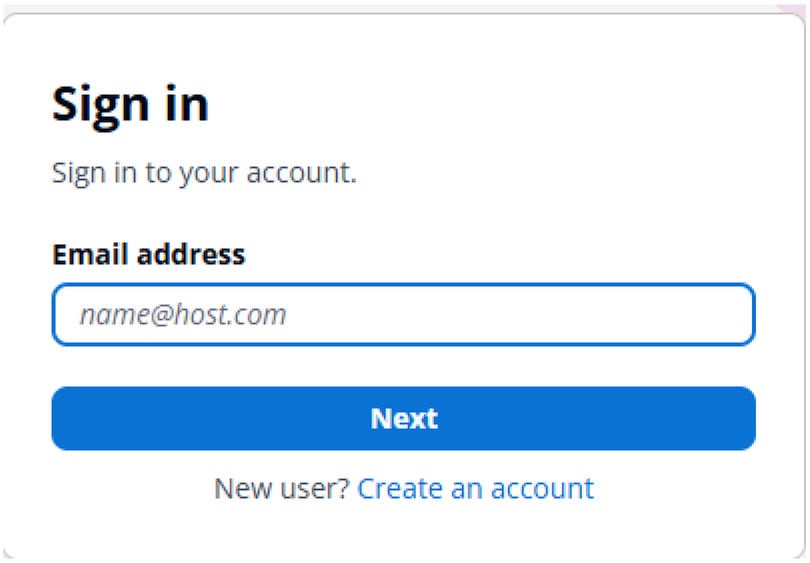

App clients and analytics Info

App clients (1) Info

Configure an app client. App clients are the user pool authentication resources attached to your app. Select an app client to configure the permitted authentication actions for an app.

App client name	Client ID
My First Cognito Web App	2md7b64jhoaqoo59rafb7isi05

5. Select the application, the **App Client** screen will open.
6. At the right, select the **View login page** link. The **Sign in** page will open.



Sign in

Sign in to your account.

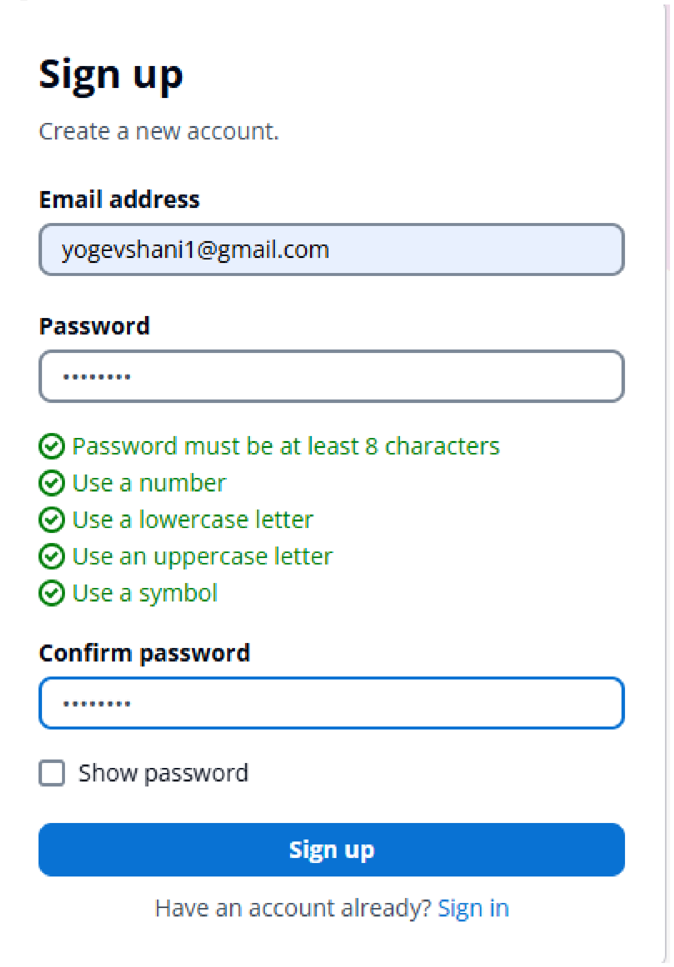
Email address

name@host.com

Next

New user? Create an account

7. As the user pool is empty, select the **Create an account** link, fill up your details, and sign up.



Sign up

Create a new account.

Email address

yogevshani1@gmail.com

Password

••••••••

✓ Password must be at least 8 characters
✓ Use a number
✓ Use a lowercase letter
✓ Use an uppercase letter
✓ Use a symbol

Confirm password

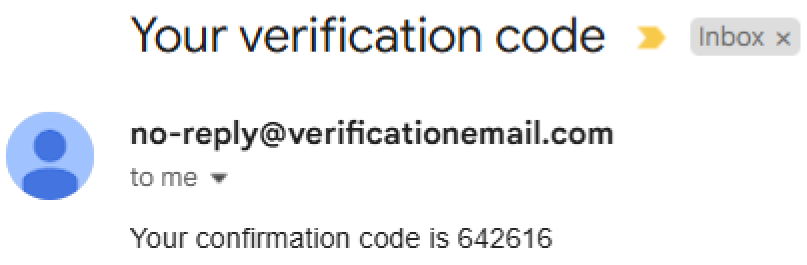
••••••••

☐ Show password

Sign up

Have an account already? Sign in

8. Go to your email inbox, look for You verification code email,

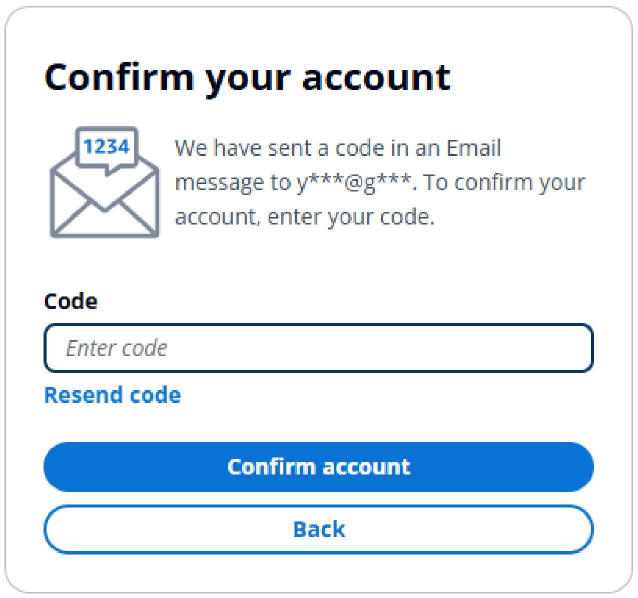

Your verification code Inbox ×

no-reply@verificationemail.com

to me

Your confirmation code is 642616

copy the code and paste it into the **Confirm your account** screen and press **Confirm account**.

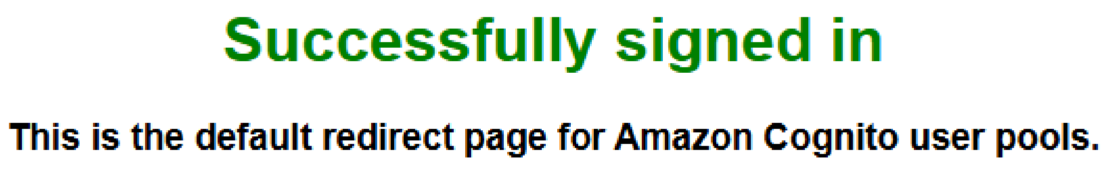

Confirm your account

We have sent a code in an Email message to y***@g***. To confirm your account, enter your code.

Code

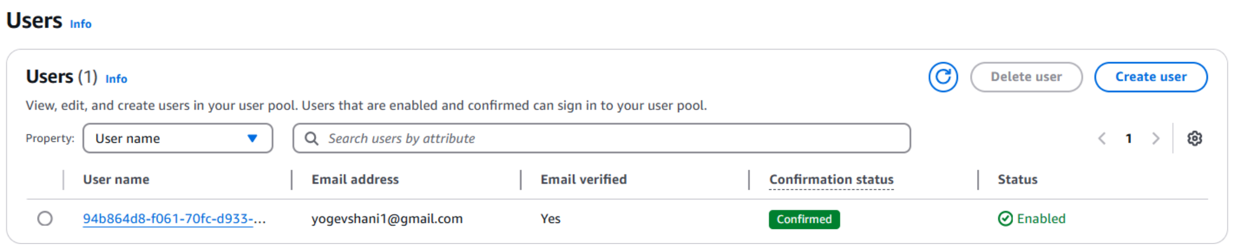
Enter code

9. Your account is now created, and you should see a default successful login page.


Successfully signed in

This is the default redirect page for Amazon Cognito user pools.

10. At the left menu, under the **User Management** menu, select the **Users** sub menu. You will see the user you have just created:


Users Info

Users (1) Info

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

Property: User name

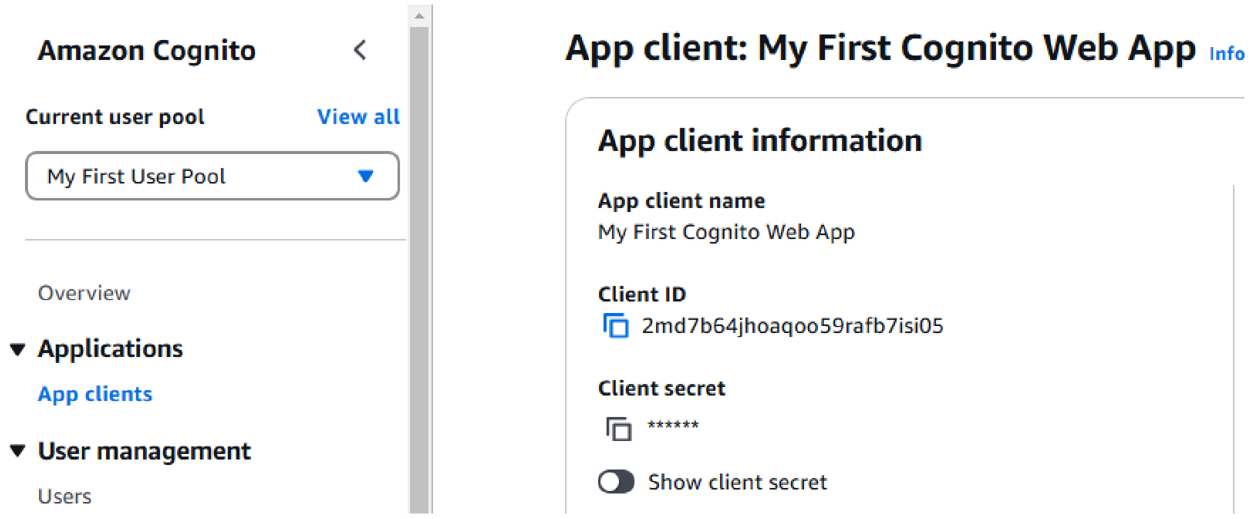
Search users by attribute

	User name	Email address	Email verified	Confirmation status	Status
○	94b864d8-f061-70fc-d933-...	yogevshani1@gmail.com	Yes	Confirmed	✓ Enabled

Task 3: Connect HTML page to Cognito application login page

You will connect the Cognito login page to a basic HTML page and view **Cognito** tokens based on **Cognito authorization code**.

1. We will now get the Cognito App **Client ID** & **Client Secret**. Go back to your application overview page and copy the **Client ID** & **Client Secret** values.



App client: My First Cognito Web App

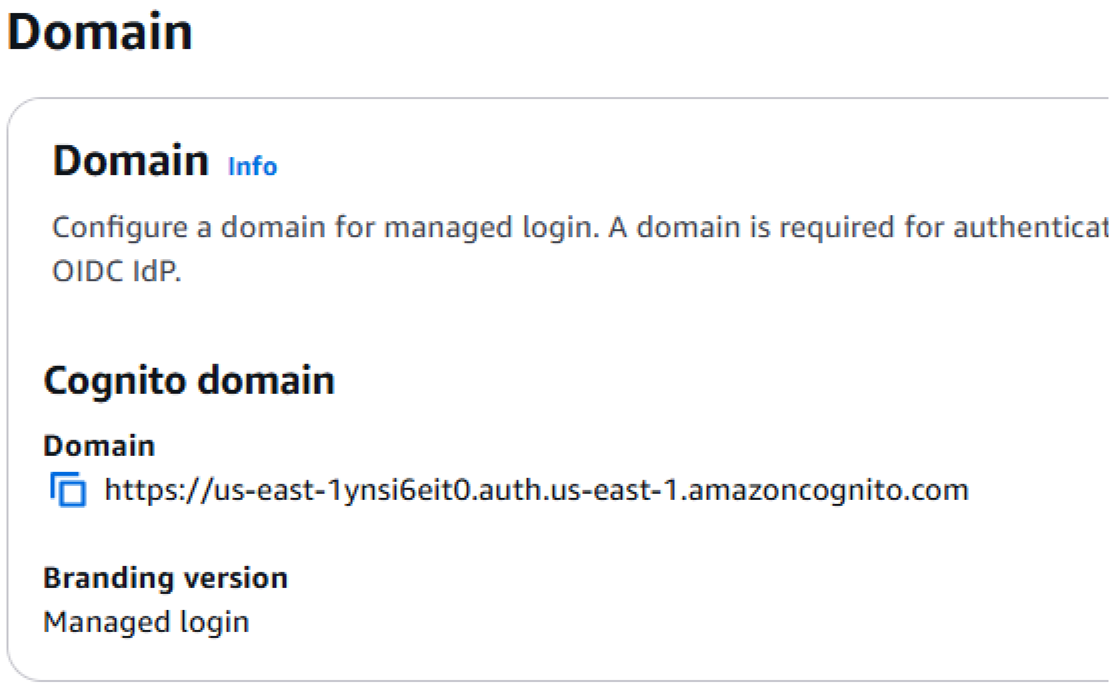
App client information

App client name
My First Cognito Web App

Client ID
2md7b64jhoaqoo59rafb7isi05

Client secret
******

2. We will now get the Cognito User Pool **Domain**. At the left menu, under the **Branding** menu, select the **Domain** sub menu. Once **Domain** Overview is open, in the **Domain** frame, copy the **Domain** value:



Domain

Domain

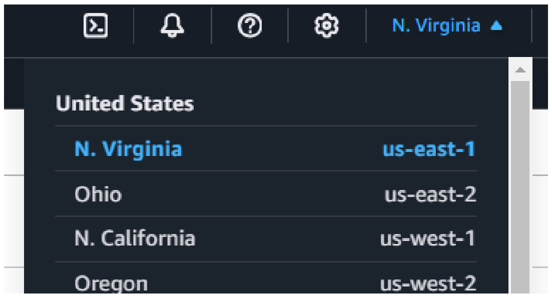
Configure a domain for managed login. A domain is required for authenticat
OIDC IdP.

Cognito domain

Domain
https://us-east-1ynsi6eit0.auth.us-east-1.amazoncognito.com

Branding version
Managed login

3. . Additionally, note down your **region**, available at the top drop-down menu.

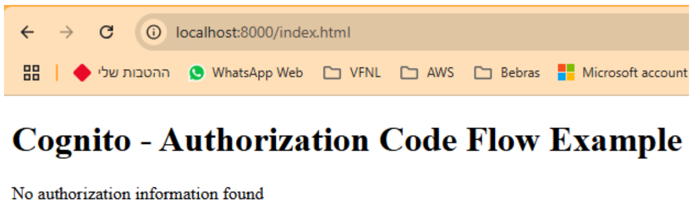


4. Create a basic HTML page that will be launched after Cognito successful login and query Cognito tokens with the token send back to the HTML.

   Alternatively, use the following code. Make sure to replace the next tags: <Your-App-Client-ID >, <-App-Client-Secret >; your redirect URI; <Your-Cognito-Domain> & <region>.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Hello World with Cognito</title>
  <script>
    async function exchangeCodeForTokens(code) {
      const clientId = '<Your-App-Client-ID>';  // Replace with your Cognito App Client ID
      const clientSecret = '<Your-App-Client-Secret>'; // Replace with your Cognito App Client Secret (if
applicable)
      const redirectUri = 'http://localhost:8000/index.html'; // Replace with your redirect URI
      const tokenUrl = 'https://<Your-Cognito-Domain>.auth.<region>.amazoncognito.com/oauth2/token'; //
Replace with your Cognito token endpoint

      const body = new URLSearchParams({
        grant_type: 'authorization_code',
        client_id: clientId,
        redirect_uri: redirectUri,
        code: code
      });

      try {
        const response = await fetch(tokenUrl, {
          method: 'POST',
          headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
            Authorization: 'Basic ' + btoa(`${clientId}:${clientSecret}`)
          },
            body
        });

        if (response.ok) {
            const data = await response.json();
            return data;
        } else {
            throw new Error('Failed to exchange code for tokens');
        }
    } catch (error) {
        console.error('Error exchanging code:', error);
        document.getElementById('userInfo').innerText = 'Error retrieving user information';
    }
}

function parseJwt(token) {
    const base64Url = token.split('.')[1];
    const base64 = base64Url.replace(/-/g, '+').replace(/_/g, '/');
    const jsonPayload = decodeURIComponent(atob(base64).split('').map(function(c) {
        return '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2);
    }).join(''));

    return JSON.parse(jsonPayload);
}

async function displayUserInfo() {
    const searchParams = new URLSearchParams(window.location.search);
    const code = searchParams.get('code');

    if (code) {
        // Handle Authorization Code Flow
        const tokens = await exchangeCodeForTokens(code);

        if (tokens && tokens.id_token) {
            const userDetails = parseJwt(tokens.id_token);
            const userInfoDiv = document.getElementById('userInfo');

            // Display all user details
            userInfoDiv.innerHTML = '<h3>User Details:</h3><ul>';
            for (const [key, value] of Object.entries(userDetails)) {
                userInfoDiv.innerHTML += `<li><strong>${key}:</strong> ${value}</li>`;
            }
            userInfoDiv.innerHTML += '</ul>';
        } else {
            document.getElementById('userInfo').innerText = 'Unable to retrieve user details';
        }
    } else {
        document.getElementById('userInfo').innerText = 'No authorization information found';
    }
    }

    window.onload = displayUserInfo;
  </script>
</head>
<body>
  <h1>Cognito - Authorization Code Flow Example</h1>
  <div id="userInfo">Loading user information...</div>
</body>
</html>
```

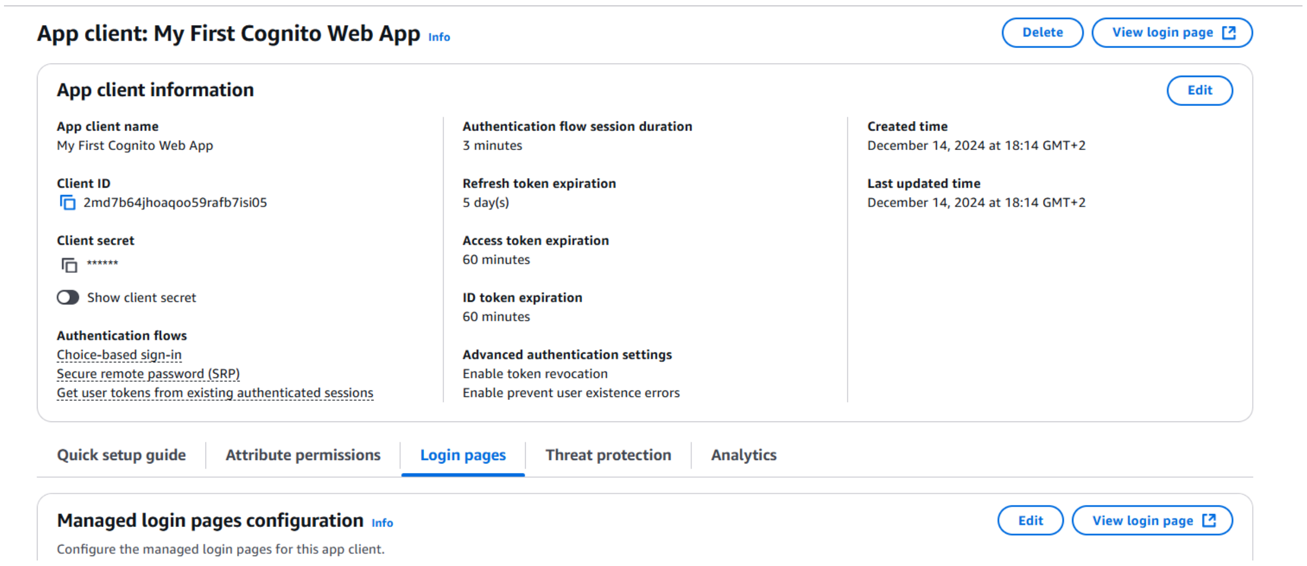
5. Save the html page as index.html and put it in any folder. Start a Local Web Server to access the HTML page from web browser.
    1. Open the **Command Prompt** or **Terminal**.
    2. Navigate to the folder containing your HTML file: *cd <path>*
    3. Start the web server. Example with Python 3.x: *python -m http.server 8000*
    4. Verify you can access your HTML file from a web browser (i.e. Chrome): *http://localhost:8000/index.html*



Cognito - Authorization Code Flow Example

No authorization information found

6. Go back to your Cognito App Client overview page, select the **Login Pages** tab, and select the **Edit** button.



App client: My First Cognito Web App Info

App client information

App client name
My First Cognito Web App

Client ID
2md7b64jhoaqoo59rafb7isi05

Client secret
******

Show client secret

Authentication flows
Choice-based sign-in
Secure remote password (SRP)
Get user tokens from existing authenticated sessions

Authentication flow session duration
3 minutes

Refresh token expiration
5 day(s)

Access token expiration
60 minutes

ID token expiration
60 minutes

Advanced authentication settings
Enable token revocation
Enable prevent user existence errors

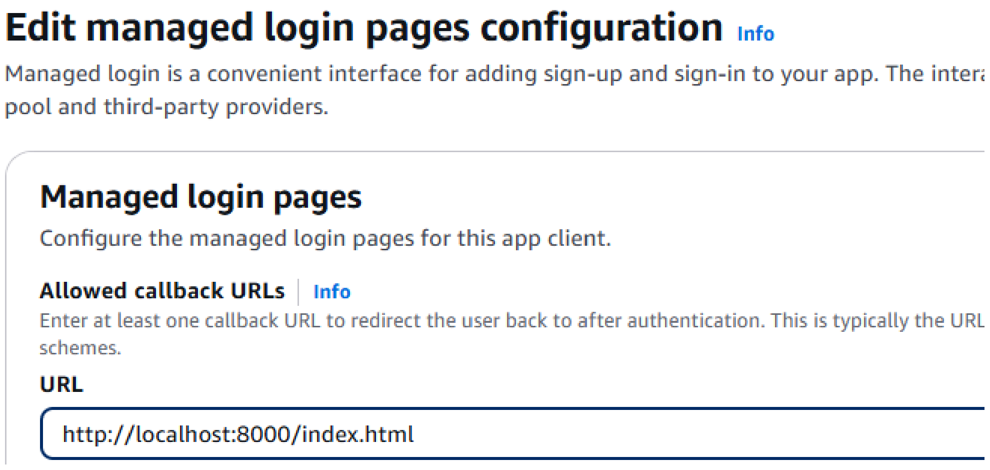
Created time
December 14, 2024 at 18:14 GMT+2

Last updated time
December 14, 2024 at 18:14 GMT+2

Managed login pages configuration Info

Configure the managed login pages for this app client.

7. In the Edit managed login pages configuration page, in the Managed login pages frame, set the URL to your html page, i.e. *http://localhost:8000/index.html* and press **Save changes**.



Edit managed login pages configuration Info

Managed login is a convenient interface for adding sign-up and sign-in to your app. The intera
pool and third-party providers.

Managed login pages

Configure the managed login pages for this app client.

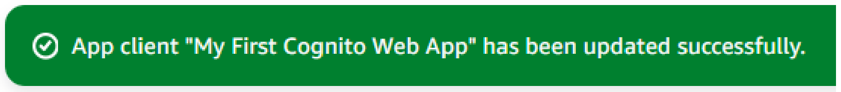
Allowed callback URLs | Info

Enter at least one callback URL to redirect the user back to after authentication. This is typically the URL
schemes.

URL

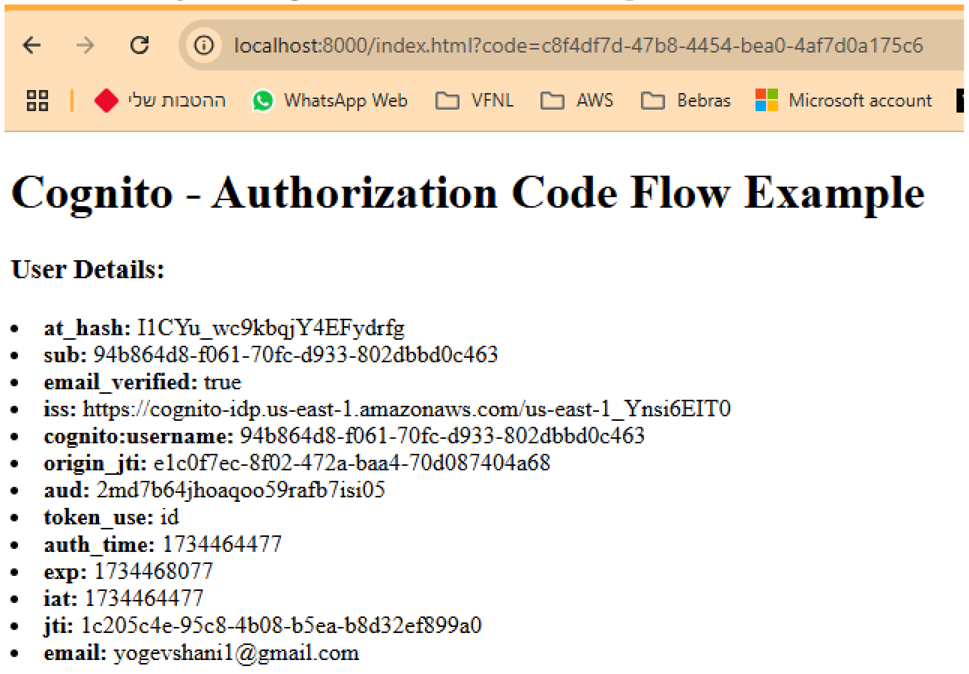
http://localhost:8000/index.html

8. A success message should appear:



App client "My First Cognito Web App" has been updated successfully.

9. Press the **View login page**, and login, your HTML page should open with the tokens received using the **Cognito authorization code** passed to the HTML:



localhost:8000/index.html?code=c8f4df7d-47b8-4454-bea0-4af7d0a175c6

Cognito - Authorization Code Flow Example

User Details:

- **at_hash:** I1CYu_wc9kbqjY4EFydrfg
- **sub:** 94b864d8-f061-70fc-d933-802dbbd0c463
- **email_verified:** true
- **iss:** https://cognito-idp.us-east-1.amazonaws.com/us-east-1_Ynsi6EIT0
- **cognito:username:** 94b864d8-f061-70fc-d933-802dbbd0c463
- **origin_jti:** e1c0f7ec-8f02-472a-baa4-70d087404a68
- **aud:** 2md7b64jhoaqoo59rafb7isi05
- **token_use:** id
- **auth_time:** 1734464477
- **exp:** 1734468077
- **iat:** 1734464477
- **jti:** 1c205c4e-95c8-4b08-b5ea-b8d32ef899a0
- **email:** yogevshani1@gmail.com

Task 4: Create a user from Python Boto3 SDK

In this task, you will create a user using Boto3. Python knowledge is required.

1. Create a Python script that interacts with an AWS Cognito and create a new user.

   1.1 Import Required Libraries
   - Import \`boto3\` for AWS SDK.
   - Import \`hmac\`, \`hashlib\`, and \`base64\` for cryptographic operations.

   2.1 Set Up Configuration Variables
   - Define \`client_id\`, \`client_secret\`, \`username\`, \`password\`, and \`email\` with appropriate values.

   3.1 Define Function to Calculate \`SECRET_HASH\`:
   - Create a function \`calculate_secret_hash\` that takes \`client_id\`, \`client_secret\`, and \`username\` as inputs.
   - Concatenate \`username\` and \`client_id\` to create a message.
   - Create a HMAC-SHA256 hash of the message using the \`client_secret\`.
   - Base64-encode the hash and return it.

   4.1 Initialize Cognito Client:
   - Use \`boto3\` to initialize a Cognito client with the specified AWS region (\`us-east-1\`).

   5.1 Sign Up User:
   - Use a \`try\` block to attempt to sign up the user with the \`sign_up\` method of the Cognito client.
   - Pass \`ClientId\`, \`Username\`, \`Password\`, and \`UserAttributes\` (which includes the email).
   - Include the \`SECRET_HASH\` calculated earlier.
   - Print the response if the sign-up is successful.
   - Use an \`except\` block to catch any exceptions and print the error message.

2. Alternatively, copy the following code, and paste it in your IDE:

   Note: Replace **COGNITO_APP_CLIENT_ID** and **COGNITO_APP_CLIENT_SECRET**, with your actual AWS credentials, similarly, replace the region-name with the region-name you are using. Additionally, **replace USER_NAME** and **USER_EMAIL.**

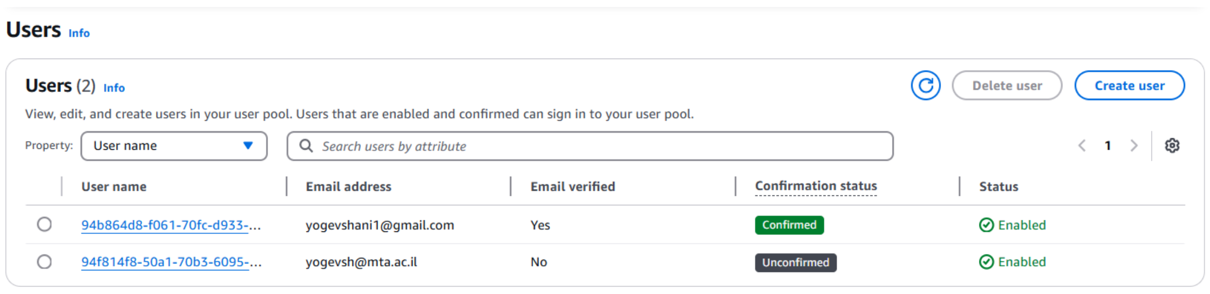
```
import boto3
import hmac
import hashlib
import base64

# Configuration
client_id = '<COGNITO_APP_CLIENT_ID>'   # Replace with your App Client ID
client_secret = '<COGNITO_APP_CLIENT_SECRET>'   # Replace with your App Client Secret
username = '<USER_NAME>'
password = 'StrongPassword123!'
email = '<USER_EMAIL>'

# Function to calculate SECRET_HASH
def calculate_secret_hash(client_id, client_secret, username):
    message = username + client_id
    dig = hmac.new(client_secret.encode('utf-8'), message.encode('utf-8'), hashlib.sha256).digest()
    return base64.b64encode(dig).decode('utf-8')

# Initialize Cognito client
client = boto3.client('cognito-idp', region_name='us-east-1')

# Sign up user
try:
    response = client.sign_up(
        ClientId=client_id,
        Username=username,
        Password=password,
        UserAttributes=[
            {'Name': 'email', 'Value': email}
        ],
        SecretHash=calculate_secret_hash(client_id, client_secret, username)
    )
    print("User signed up successfully:", response)
except Exception as e:
    print("Error during sign up:", e)
```

3. Install boto3 module.
   Verify if boto3 is installed, in your terminal run:
   pip show boto3
   if you’re getting a warning:
   WARNING: Package(s) not found: boto3
   You should install boto3 b running the next command in your terminal:
   pip install boto3
4. Run the code. You should get the following successful reply:

```
{'UserConfirmed': False, 'CodeDeliveryDetails': {'Destination': 'y***@m***', 'DeliveryMedium':
'EMAIL', 'AttributeName': 'email'}, 'UserSub': '94f814f8-50a1-70b3-6095-b818c88fc7a1', 'Session':
'AYABeIkToOKP72so58IwwoO4ozEAHQABAAdTZXJ2aWNlABBDb2duaXRvVXNlclBvb2xzAAEAB
2F3cy1rbXMAS2Fybjphd3M6a21zOnVzLWVhc3QtMTo3NDU2MjM0Njc1NTU6a2V5L2IxNTVhZmNh
LWJmMjktNGVlZC1hZmQ4LWE5ZTA5MzY1M2RiZQC4AQIBAHjHL4WD3WpekpFe85nxP9Nwg99u
3bPN6BTSaB-
uHZcTLAFEwyP7GotYDSqP0ppayLvpAAAAfjB8BgkqhkiG9w0BBwagbzBtAgEAMGgGCSqGSIb3DQ
EHATAeBglghkgBZQMEAS4wEQQMDR0VySo8iOe9sytWAgEQgDs-
UvEijQylZHERXD9B72Dpn2DyOkOKPHVUKFfIY_DILLr_lA4_XDFQK0u2OFDy_piUy-
jtOgvBYdV5jAIAAAAADAAAEAAAAAAAAAAAAAAAATjEu3LGDK3dMuQWBMHh2l_____w
AAAAEAAAAAAAAAAAAAAAAEAAAFcriKf1_9siy22hMD2YDe0BdQzvTGuoq9DXnzsDOq41syk7sj
fUNN0Be82WlA6rxLTtwmxXrLPKLbMRj2eNpSKgVj9NAb-rcCPh-
GA3NfjEeOf6uIgOyBbenfhWsuYwzpoG8gDT2h74Bg_NGby5f3MuCxDD78-cq3NlgGSAY-
fDudw3CMyU3WsZijVIG_V3Gzde6tvsu1PKvRYAFY5bB8p8Suuf_uT_AGLOuZy-dPb0-
VJGoVWDlI6VIUnjTIYowHv8vbR6lQfV8ElXJimluXwMVAhH6POAlLY7inHJdikD6W52x2Znl0MwU
S6NCqd9HH7seUBD_hOPGM5f1e_OWtnuzFkAHsQ08pp8mQsWRoP-
_ayMmqhYuOxzd3X_cilWnyUJSh2Cs4O3dz-uMqmg1umwOS3RXPmiJcd3b3XPcaLNWP3iTfY-
uOZqqEK_EUbQTL67enwXPCgHym9RIFMKNasaqtuMnV-lkRvECwpJg', 'ResponseMetadata':
{'RequestId': '23370591-dd75-4068-b829-6f1e04dfeb74', 'HTTPStatusCode': 200, 'HTTPHeaders':
{'date': 'Sat, 21 Dec 2024 15:11:06 GMT', 'content-type': 'application/x-amz-json-1.1', 'content-length':
'1186', 'connection': 'keep-alive', 'x-amzn-requestid': '23370591-dd75-4068-b829-6f1e04dfeb74'},
'RetryAttempts': 0}}
```

Task 5: Explore the new user created using AWS SDK (Boto3) from the AWS Console

1. In the AWS Console go back to Cognito to the User Pool and in the left menu, under the **User Management** menu, select the **Users** sub menu. You will see you’re the user you have just created:

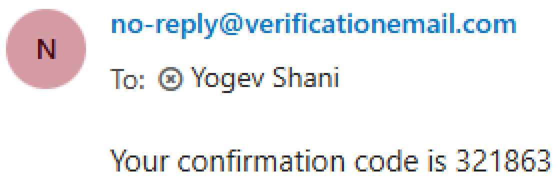

Users Info

Users (2) Info

View, edit, and create users in your user pool. Users that are enabled and confirmed can sign in to your user pool.

User name	Email address	Email verified	Confirmation status	Status
94b864d8-f061-70fc-d933-...	yogevshani1@gmail.com	Yes	Confirmed	✓ Enabled
94f814f8-50a1-70b3-6095-...	yogevsh@mta.ac.il	No	Unconfirmed	✓ Enabled

2. The user **Confirmation status** will be **Unconfirmed**. In your mailbox you should see a confirmation code email, that can be used to confirm the user from a python script.


no-reply@verificationemail.com

To: Yogev Shani

Your confirmation code is 321863

Activity complete

Congratulations! You have successfully completed the activity.