**Manual Setup of Student Health & Productivity Tracker on AWS Console**

Here's a step-by-step guide to manually create the architecture through the AWS Management Console:

# **Set up Cognito User Pool**

1. Navigate to **Amazon Cognito** service

2. Click **Manage User Pools** > **Create a user pool**

3. Enter pool name: **neo-cloud-student-pool**

4. **Under** "How do you want your end users to sign in?", **select** "Email address or phone number"

5. Click "**Next Step**" through defaults until "**Review**" page

6. Click "**Create pool**"

## **Configure App Client:**

1. Go to "**App clients**" tab

2. Click "**Add an app client**"

3. Name: `**student-tracker-client**`

4. Uncheck "**Generate client secret**"

5. Click "**Create app client**"

# **Create DynamoDB Table**

1. Navigate to **DynamoDB** service

2. Click "**Create table**"

3. Table name: `**StudentProductivityLogs**`

4. Partition key: `**userId**` (String)

5. Sort key: `**logDate**` (String)

6. Settings: Select "**Customize settings**"

- Read/write capacity mode: On-demand

7. Click "**Create table**"

## **Add Global Secondary Index**:

1. Go to "Indexes" tab

2. Click "Create index"

3. Partition key: `userId` (String)

4. Sort key: `entryType` (String)

5. Index name: `EntryTypeIndex`

6. Click "Create index"

# **Create Lambda Function**

1. Navigate to **Lambda** service

2. Click "**Create function**"

3. Function name: `**StudentTrackerLambda**`

4. Runtime: **Python 3.9**

5. Click "**Create function**"

## **Add Code:**

1. In the "**Code**" tab, replace the default code with the [lambda.py](#) content from earlier

2. Click "**Deploy**"

## **Configure Permissions:**

1. Go to "**Configuration**" > "**Permissions**"

2. Click on the **execution role name**

3. In IAM, click "**Add permissions**" > "**Attach policies**"

4. Add:

**- `AmazonDynamoDBFullAccess`**

**- `AWSLambdaBasicExecutionRole**`

# **Set up API Gateway**

1. Navigate to **API Gateway** service

2. Click "**Create API**" > "**REST API**" > "**Build**"

3. API name: `**StudentTrackerAPI`**

4. Click "**Create API**"

## 

## **Create Resources/Methods:**

1. Click "**Actions**" > "**Create Resource**"

- Resource Name: **`logs**`

- Click "**Create Resource**"

2. With `**/logs**` selected, click "**Actions**" > "**Create Method**"

- Select `**POST**` from dropdown

- Click checkmark

3. Configure:

- Integration type: **Lambda Function**

- Lambda region: (your region)

- Lambda function: **`StudentTrackerLambda`**

- Click "**Save**" > **"OK**"

## **Configure Authorizer:**

1. Click "**Authorizers**" in left menu

2. Click "**Create New Authorizer**"

- Name: **`CognitoAuthorizer`**

- Type: **Cognito**

- Cognito User Pool: `**neo-cloud-student-pool`**

- Token Source: **`Authorization`**

- Click **"Create"**

## **Enable CORS:**

1. Select **`/logs`** resource

2. Click **"Actions**" > "**Enable CORS**"

3. Keep defaults and click **"Enable CORS...**"

## **Deploy API:**

1. Click "**Actions**" > "**Deploy API**"

2. Deployment stage: **`[New Stage]`**

3. Stage name: **`prod`**

4. Click **"Deploy"**

# **Set up S3 for Frontend**

1. Navigate to **S3** service

2. Click **"Create bucket"**

- Bucket name: `**student-tracker-ui-[your-account-id]`**

- Uncheck **"Block all public access"**

- Acknowledge warning

- Click **"Create bucket"**

3. Go to **"Properties" tab**

- Scroll to **"Static website hosting"**

- Click **"Edit"**

- Enable static website hosting

- Index document: **`index.html`**

- Click **"Save changes"**

4. Go to **"Permissions"** tab

- Edit **"Bucket policy"** and add:

```json

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::student-tracker-ui-\*/\*"

}

]

}

```

# **Set up CloudFront Distribution**

1. Navigate to **CloudFront** service

2. Click **"Create distribution"**

3. Origin domain: Select your **S3 bucket**

4. Viewer protocol policy: **"Redirect HTTP to HTTPS"**

5. Default cache behavior:

**- Cache policy: "CachingOptimized"**

6. Default root object: **`index.html`**

7. Click **"Create distribution"**

# **Configure Frontend**

1. Create a **React app** with the structure shown earlier

2. Key configuration points:

**- Set API endpoint in `src/services/api.js`:**

```javascript

const API\_ENDPOINT = 'https://[your-api-id].execute-api.[region].amazonaws.com/prod';

```

**- Configure Amplify for Cognito in `src/services/auth.js`:**

```javascript

import { Amplify } from 'aws-amplify';

Amplify.configure({

Auth: {

region: '[your-region]',

userPoolId: '[user-pool-id]',

userPoolWebClientId: '[app-client-id]',

}

});

```

# **Deploy Frontend**

1. Build your React app: **`npm run build`**

2. Upload contents of **`build/`** folder to your S3 bucket

3. Wait for CloudFront distribution to deploy (status **"Deployed"**)

# **Verification Steps**

### **1. Register a test user in Cognito:**

- Go to Cognito > User Pools > Your pool > Users

- Click "Create user"

- Fill in details and click "Create user"

### **2. Test API:**

- Use Postman or curl to send a POST request to your API endpoint

- Include Cognito authentication token in headers

### **3. Check DynamoDB:**

- After making requests, check the DynamoDB table for new entries

# **Monitoring Setup**

### **1. CloudWatch Logs:**

- Lambda logs are automatically created

- API Gateway logs can be enabled in API Gateway settings

### **2. CloudWatch Alarms:**

- Create alarms for Lambda errors

- Monitor API 4XX/5XX errors