Assignment 10 – API Gateway and Cognito

PART I – Lambda and DynamoDB

In the last days, we prepared a CRUD app in Lambda backed with DynamoDB. Today, we will make it an API. So, it is publicly accessible. But only if the valid token is present.

PART II - API gateway and Cognito

- Create a CRUD API for the sample course app in API Gateway.

/course POST

/course/{courseName} GET – filter courses by course name. Query on the index. Get the course name as a path parameter.

/course GET – List all courses. Implement filter on non-key attributes. Get the month and year values as query strings.

/course/item GET – it returns one item by the composite key. Get the course code and teacher name as query strings.

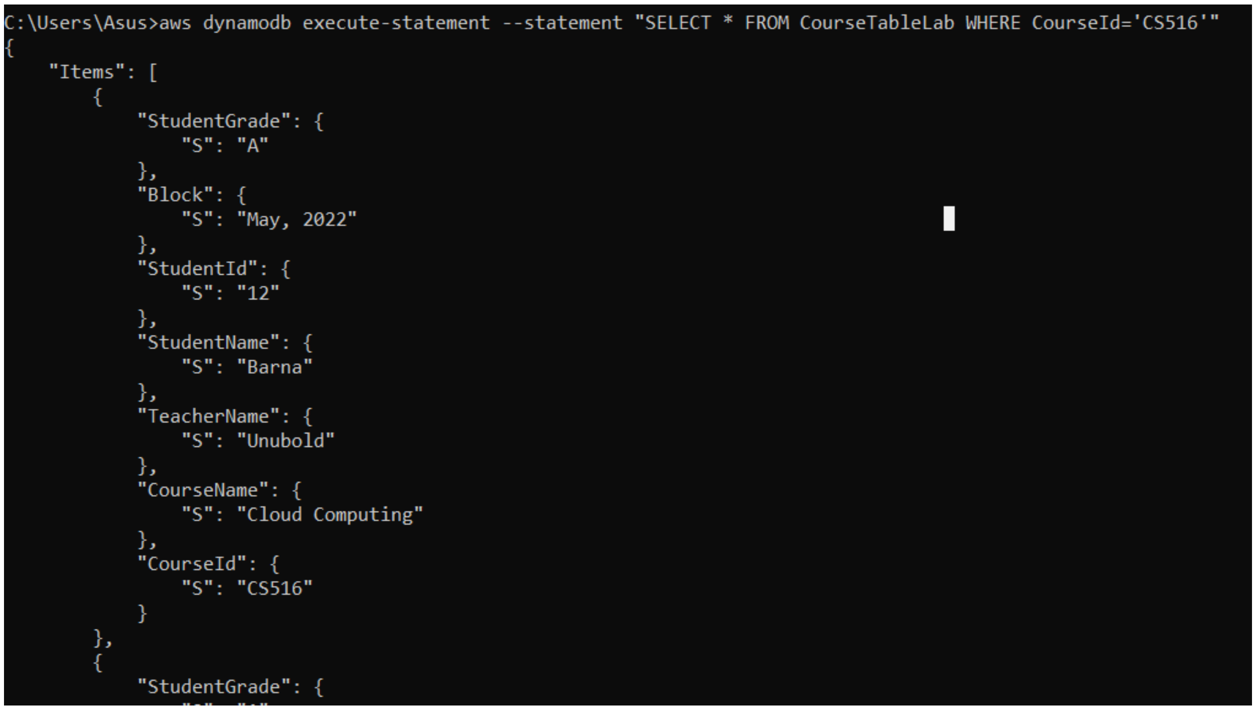
/course PATCH – That updates a course record.

- Create a **Cognito User pool** for the app. For hosted UI setup, refer: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-app-integration.html
    - Main conifugration. Go to HostedUI section -> Hit Edit -> OAuth 2.0 grant types -> Select IMPLICIT GRANT
- Secure the API using tokens from the Cognito User pool.
- Practice **direct DynamoDB integration** with **API key**

```
curl --location 'https://r1qos0kpx6.execute-api.us-east-1.amazonaws.com/dev/course-
db' \
--header 'x-api-key: QtCyvzc09e8WU4LJ56wSA6ZK9h6EhDCP3cSPMRo3' \
--header 'Content-Type: application/json' \
--data '{
    "TableName": "TableName",
    "Item": {
        "id": {
            "S": "Directly 2"
        },
        "courseName": {
            "S": "Cloud Computing"
        },
    }
}'
```

Extra

- Instead of Lambda, use StepFunctions to store data in DynamoDB.
- Practice the execute statement with SQL.

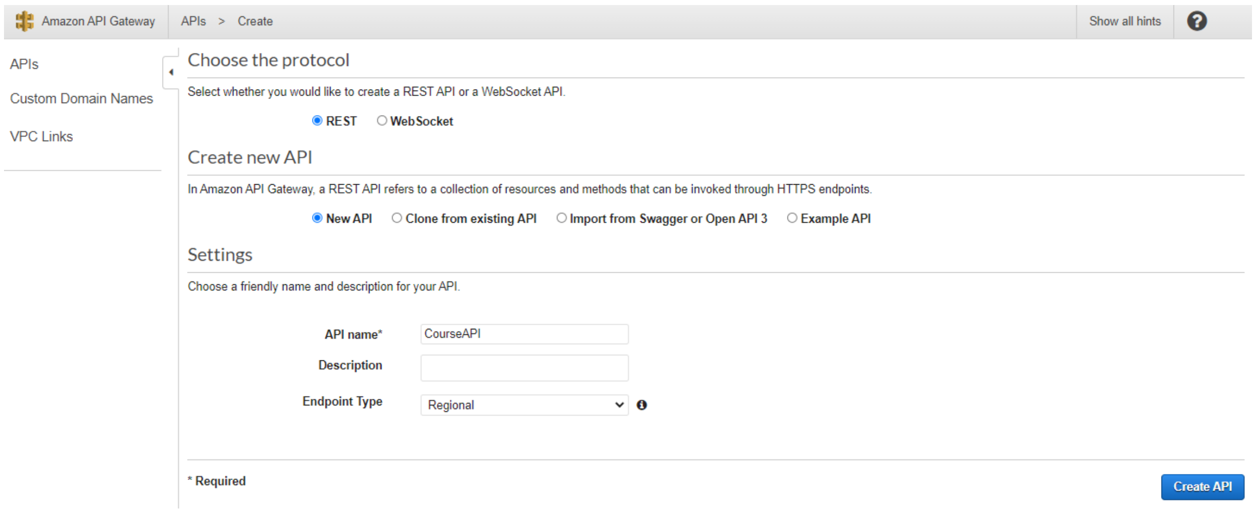


```
C:\Users\Asus>aws dynamodb execute-statement --statement "SELECT * FROM CourseTableLab WHERE CourseId='CS516'"
{
    "Items": [
        {
            "StudentGrade": {
                "S": "A"
            },
            "Block": {
                "S": "May, 2022"
            },
            "StudentId": {
                "S": "12"
            },
            "StudentName": {
                "S": "Barna"
            },
            "TeacherName": {
                "S": "Unubold"
            },
            "CourseName": {
                "S": "Cloud Computing"
            },
            "CourseId": {
                "S": "CS516"
            }
        },
        {
            "StudentGrade": {
```

Instructions

Creating an API

1. Create a “**CourseAPI**” API on API Gateway in front of the “**CourseLambda**”
   a. Search on the top bar and go to the **API Gateway** on AWS Console.
   b. **REST API** (Not REST API private!!)-> click on the orange **Build** button.
   c. On the popup, press **OK.**
   d. In **Create new API**, select **New API** radio button.
   e. In **Settings, API name** is **CourseAPI**. Hit **Create API.**



Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

[● REST] ○ WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[● New API] ○ Clone from existing API ○ Import from Swagger or Open API 3 ○ Example API

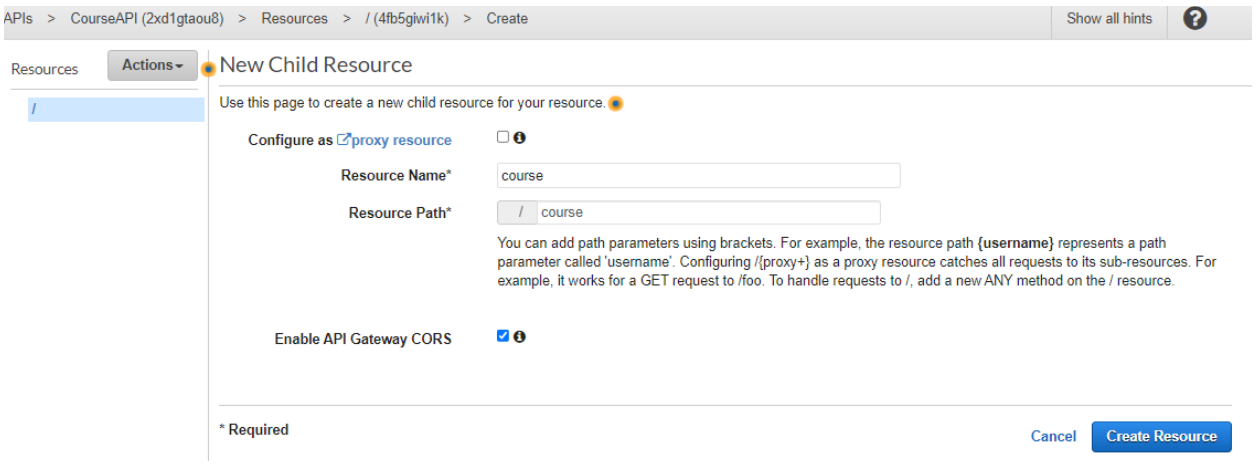
Settings

Choose a friendly name and description for your API.

API name*	CourseAPI
Description	
Endpoint Type	Regional

* Required

   f. Click on **Actions** dropdown and hit **Create Resource.**
   g. Resource Name is **course.** check **Enable API Gateway CORS.** Hit **Create Resource**.


APIs > CourseAPI (2xd1gtaou8) > Resources > / (4fb5giwi1k) > Create
Show all hints
Resources
Actions
/

New Child Resource

Use this page to create a new child resource for your resource.

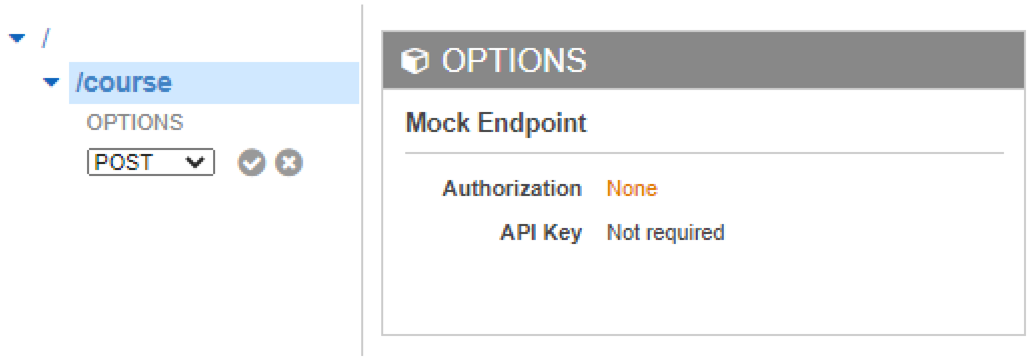
Configure as proxy resource	☐
Resource Name*	course
Resource Path*	/ course

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS ✓

* Required

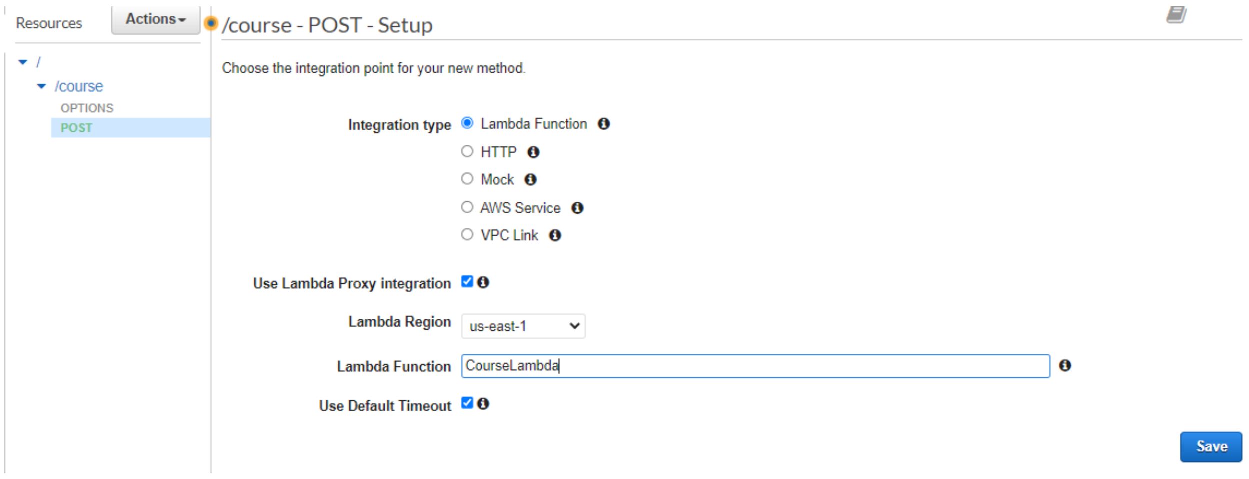
   h. Click on **Actions** dropdown and hit **Create Method.** Select **POST** in the small dropdown under the resource. Click on the small OK icon.



Authorization None

API Key Not required

i. Check **Use Lambda Proxy integration**
j. Type the lambda name **CourseLambda** as Lambda Function. Click **Save.**
k. There will be a popup. Read that and hit **OK**.



Choose the integration point for your new method.

Integration type: (● Lambda Function) ○ HTTP ○ Mock ○ AWS Service ○ VPC Link

Use Lambda Proxy integration ✓

Lambda Region us-east-1

Lambda Function CourseLambda

Use Default Timeout ✓

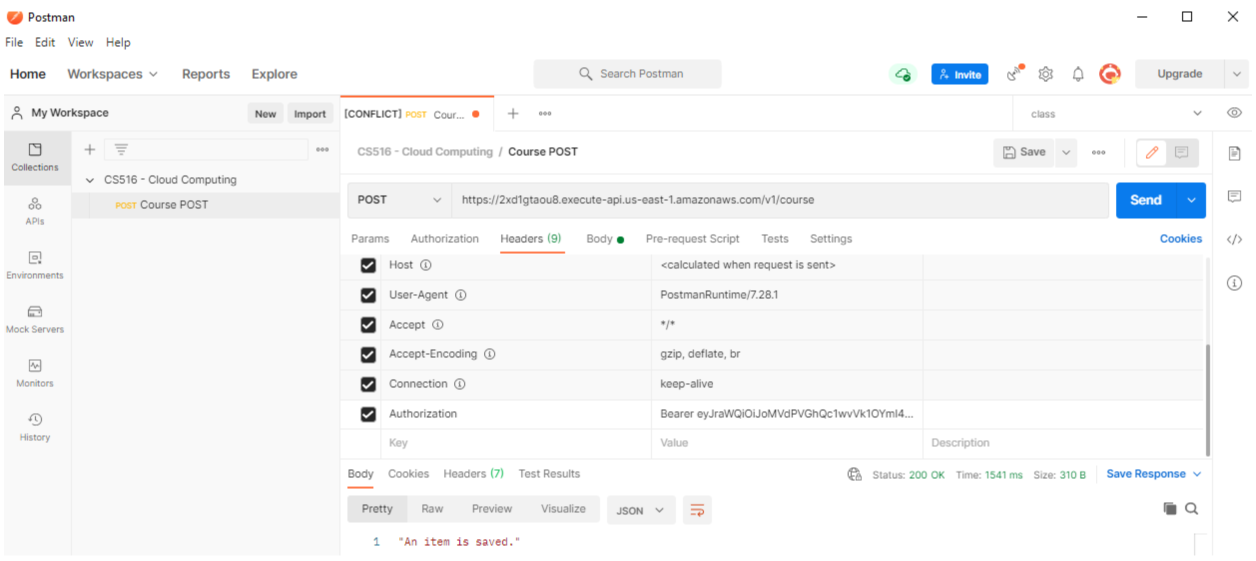
l. Click on the **Actions** dropdown and hit **Deploy API**
m. On the popup, the Deployment stage is **[New Stage]**. **Stage name** is v1. Hit **Deploy**.

2. Test your API with Postman.
   a. Click on **Stages** in left sidebar. Click on **v1**. Grab the **Invoke URL**.
   b. Create a new **POST** request in postman. Provide the URL. Append the **course** resource. It will look like this: https://2xd1gtaou8.execute-api.us-east-1.amazonaws.com/v1/**course**
   c. The body is below. Feel free to change the value. Body tab -> Select Raw -> Select JSON in the dropdown

```
{
    "courseCode": "CS100",
    "courseName": "My Course",
    "teacherName": "My Teacher",
    "month": 11,
    "year": 2022,
    "students": [
        "Student 1",
        "Student 2"
    ]
```

}

d. You should see the success response below.



POST https://2xd1gtaou8.execute-api.us-east-1.amazonaws.com/v1/course

Key	Value	Description
Host	<calculated when request is sent>	
User-Agent	PostmanRuntime/7.28.1	
Accept	*/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
Authorization	Bearer eyJraWQiOiJoMVdPVGhQc1wvVk1OYml4...	

Status: 200 OK Time: 1541 ms Size: 310 B

```
"An item is saved."
```

Creating a user pool

e. Go to Cognito -> click on User Pools -> Top right corner, click on Create a user pool.

f. In Cognito user pool sign-in options check Email. Click on Next.

g. In Multi-factor authentication section, select the No MFA. Leave the others with the default selections. Click on Next.

h. In Step 3 - Configure sign-up experience, you don't need to change anything. Click on Next.

i. In Email section, select the Send email with Cognito as an Email provider. Click on Next.

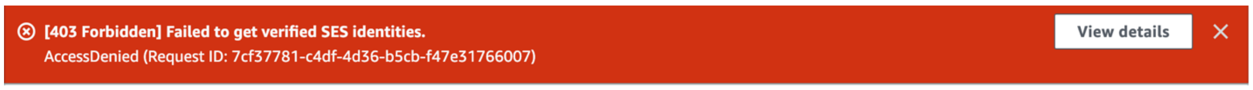
j. In User pool name section, CourseUserPool as Pool name. Then scroll down, In the Initial app client tab, CourseApiClient as App client name.

   i. In the Client secret tab, make sure that the Don't generate a client secret is selected, NOT the Generate a client secret.

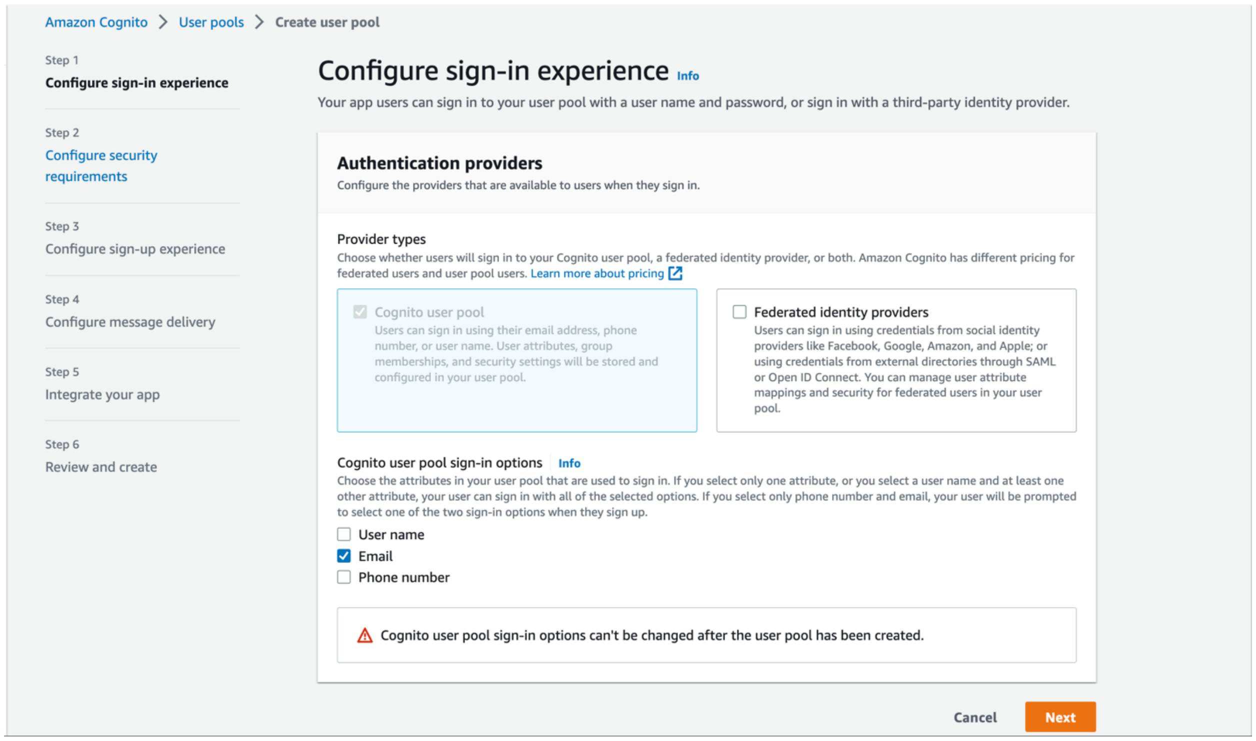
k. In Authentication flows section, Uncheck the ALLOW_CUSTOM_AUTH, and check the ALLOW_USER_PASSWORD_AUTH. Leave the others with the default selections. Click on Next.

l. In the last section Step 6 - Review and create, Click on Create user pool.

Note: You might be getting an error like the one below. You can ignore it.



[403 Forbidden] Failed to get verified SES identities.
AccessDenied (Request ID: 7cf37781-c4df-4d36-b5cb-f47e31766007)



Amazon Cognito > User pools > Create user pool

Step 1
Configure sign-in experience

Step 2
Configure security requirements

Step 3
Configure sign-up experience

Step 4
Configure message delivery

Step 5
Integrate your app

Step 6
Review and create

Configure sign-in experience Info

Your app users can sign in to your user pool with a user name and password, or sign in with a third-party identity provider.

Authentication providers

Configure the providers that are available to users when they sign in.

Provider types

Choose whether users will sign in to your Cognito user pool, a federated identity provider, or both. Amazon Cognito has different pricing for federated users and user pool users. Learn more about pricing

- [✓] Cognito user pool
  Users can sign in using their email address, phone number, or user name. User attributes, group memberships, and security settings will be stored and configured in your user pool.
- [ ] Federated identity providers
  Users can sign in using credentials from social identity providers like Facebook, Google, Amazon, and Apple; or using credentials from external directories through SAML or Open ID Connect. You can manage user attribute mappings and security for federated users in your user pool.

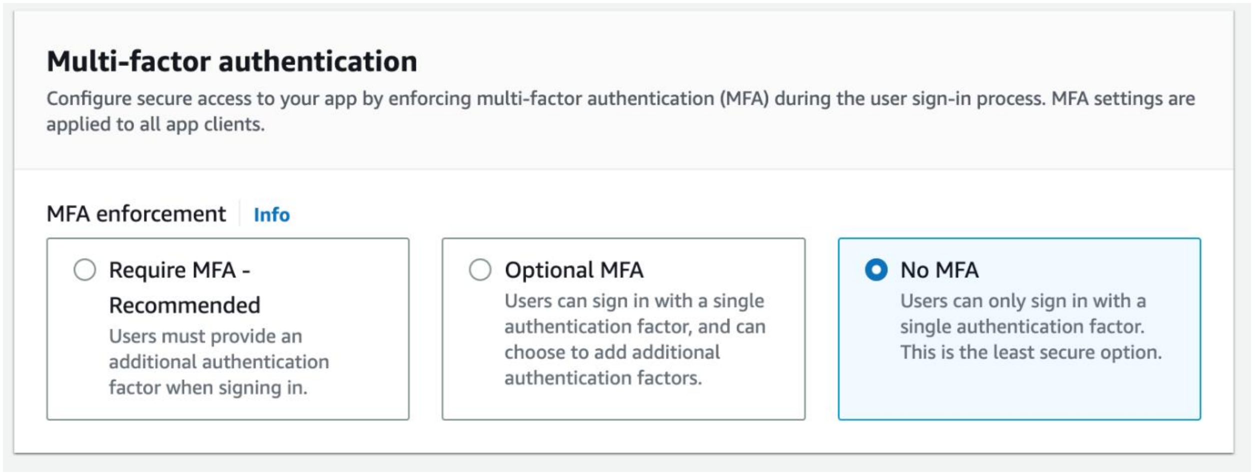
Cognito user pool sign-in options Info

Choose the attributes in your user pool that are used to sign in. If you select only one attribute, or you select a user name and at least one other attribute, your user can sign in with all of the selected options. If you select only phone number and email, your user will be prompted to select one of the two sign-in options when they sign up.

- [ ] User name
- [✓] Email
- [ ] Phone number

Cognito user pool sign-in options can't be changed after the user pool has been created.

Cancel Next

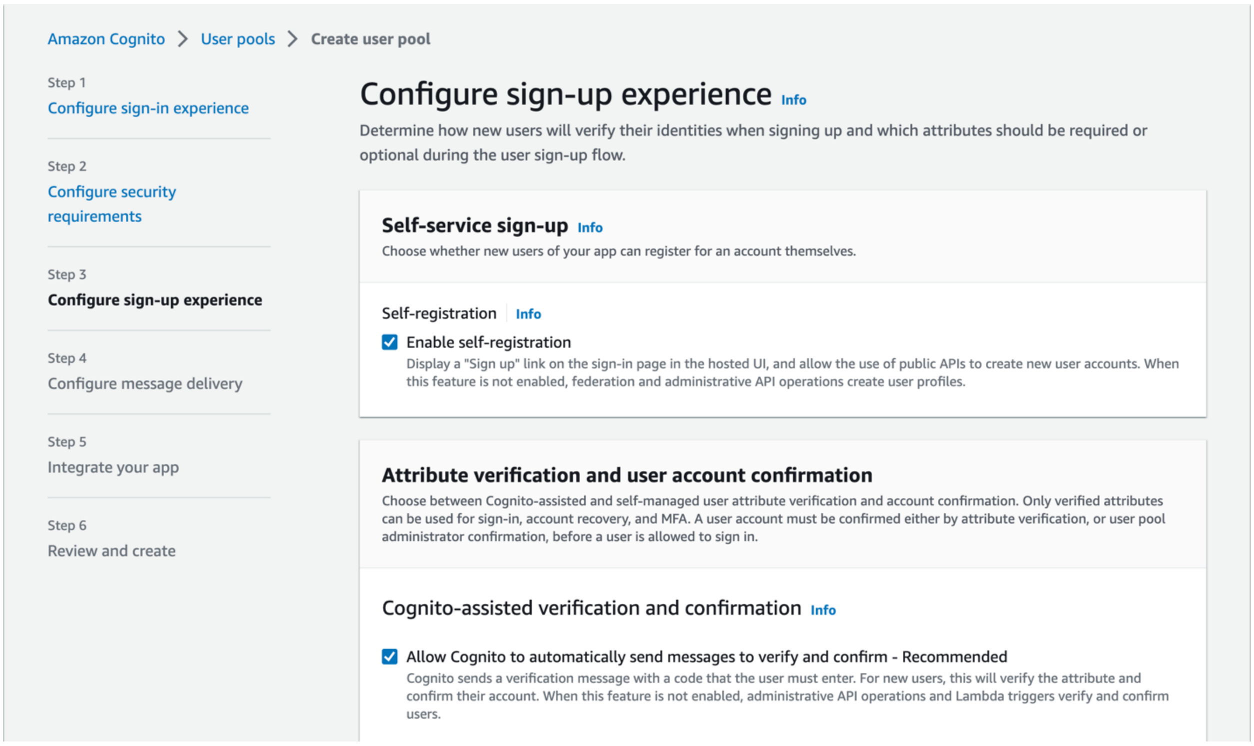


Multi-factor authentication

Configure secure access to your app by enforcing multi-factor authentication (MFA) during the user sign-in process. MFA settings are applied to all app clients.

MFA enforcement Info

- ○ Require MFA - Recommended
  Users must provide an additional authentication factor when signing in.
- ○ Optional MFA
  Users can sign in with a single authentication factor, and can choose to add additional authentication factors.
- [✓ No MFA]
  Users can only sign in with a single authentication factor. This is the least secure option.



Amazon Cognito > User pools > Create user pool

Step 1
Configure sign-in experience

Step 2
Configure security requirements

Step 3
Configure sign-up experience

Step 4
Configure message delivery

Step 5
Integrate your app

Step 6
Review and create

Configure sign-up experience Info

Determine how new users will verify their identities when signing up and which attributes should be required or optional during the user sign-up flow.

Self-service sign-up Info

Choose whether new users of your app can register for an account themselves.

Self-registration Info

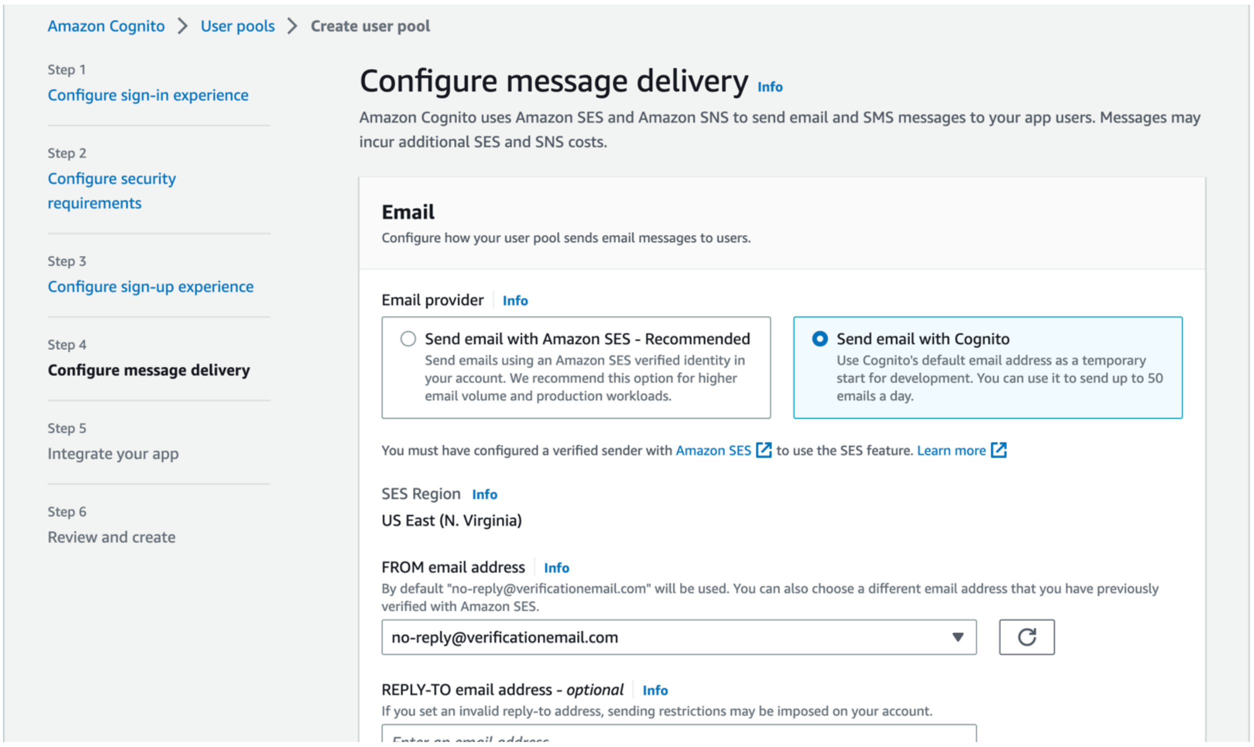
[✓] Enable self-registration
Display a "Sign up" link on the sign-in page in the hosted UI, and allow the use of public APIs to create new user accounts. When this feature is not enabled, federation and administrative API operations create user profiles.

Attribute verification and user account confirmation

Choose between Cognito-assisted and self-managed user attribute verification and account confirmation. Only verified attributes can be used for sign-in, account recovery, and MFA. A user account must be confirmed either by attribute verification, or user pool administrator confirmation, before a user is allowed to sign in.

Cognito-assisted verification and confirmation Info

[✓] Allow Cognito to automatically send messages to verify and confirm - Recommended
Cognito sends a verification message with a code that the user must enter. For new users, this will verify the attribute and confirm their account. When this feature is not enabled, administrative API operations and Lambda triggers verify and confirm users.



Amazon Cognito > User pools > Create user pool

Step 1
Configure sign-in experience

Step 2
Configure security requirements

Step 3
Configure sign-up experience

Step 4
Configure message delivery

Step 5
Integrate your app

Step 6
Review and create

Configure message delivery Info

Amazon Cognito uses Amazon SES and Amazon SNS to send email and SMS messages to your app users. Messages may incur additional SES and SNS costs.

Email

Configure how your user pool sends email messages to users.

Email provider Info

○ Send email with Amazon SES - Recommended
Send emails using an Amazon SES verified identity in your account. We recommend this option for higher email volume and production workloads.

[✓ Send email with Cognito]
Use Cognito's default email address as a temporary start for development. You can use it to send up to 50 emails a day.

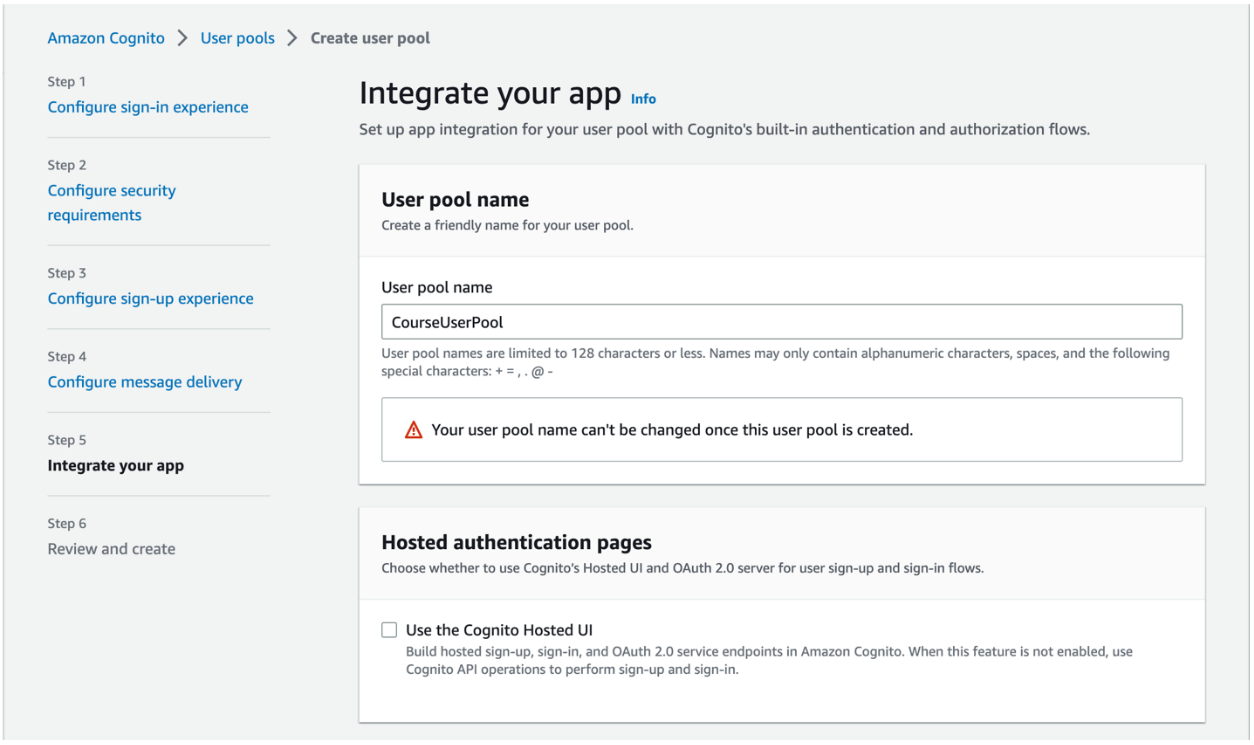
You must have configured a verified sender with Amazon SES to use the SES feature. Learn more

SES Region Info
US East (N. Virginia)

FROM email address Info
By default "no-reply@verificationemail.com" will be used. You can also choose a different email address that you have previously verified with Amazon SES.

no-reply@verificationemail.com

REPLY-TO email address - *optional* Info
If you set an invalid reply-to address, sending restrictions may be imposed on your account.


Amazon Cognito > User pools > Create user pool

Step 1
Configure sign-in experience

Step 2
Configure security requirements

Step 3
Configure sign-up experience

Step 4
Configure message delivery

Step 5
Integrate your app

Step 6
Review and create

Integrate your app Info

Set up app integration for your user pool with Cognito's built-in authentication and authorization flows.

User pool name

Create a friendly name for your user pool.

User pool name: CourseUserPool

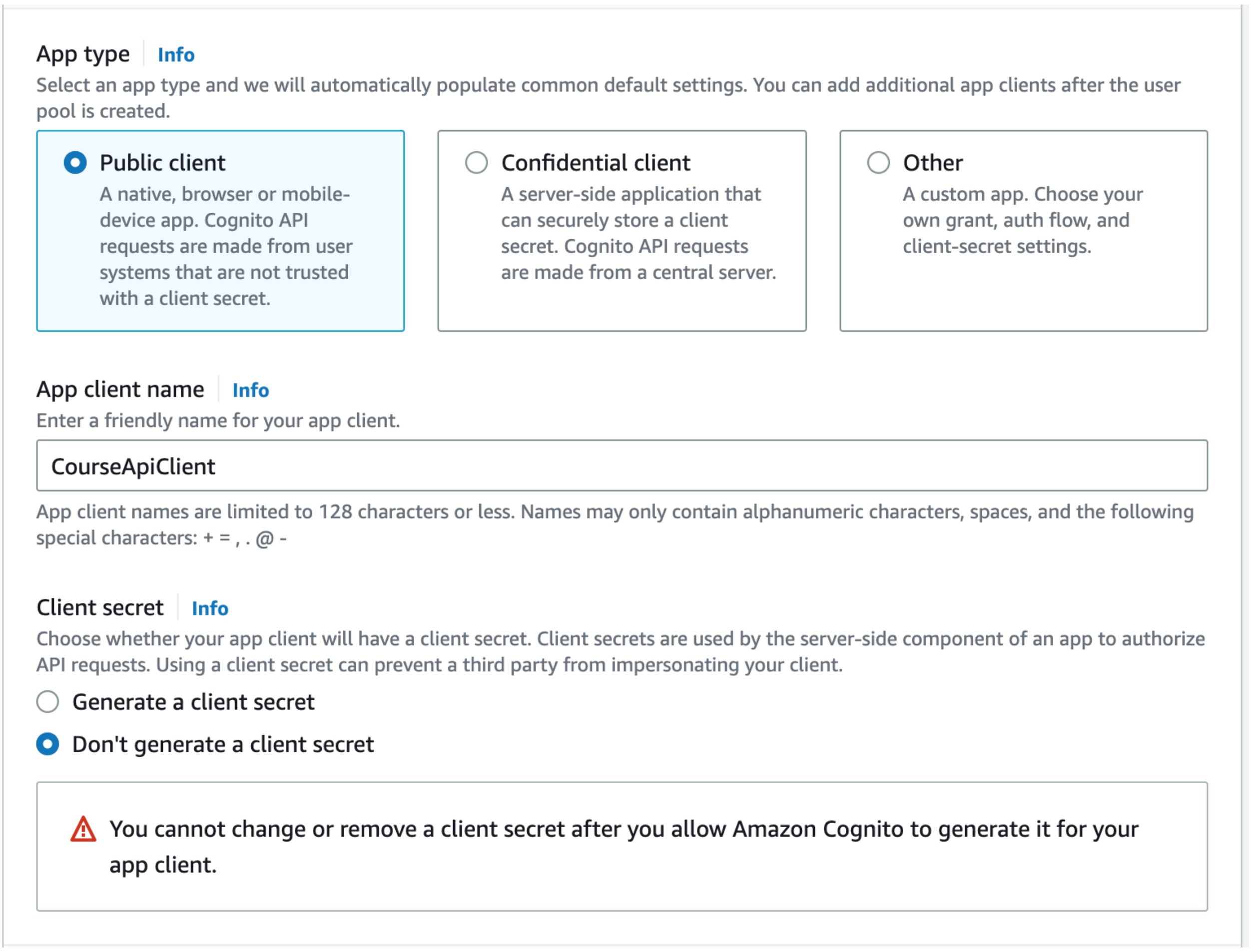
User pool names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = , . @ -

⚠ Your user pool name can't be changed once this user pool is created.

Hosted authentication pages

Choose whether to use Cognito's Hosted UI and OAuth 2.0 server for user sign-up and sign-in flows.

☐ Use the Cognito Hosted UI
Build hosted sign-up, sign-in, and OAuth 2.0 service endpoints in Amazon Cognito. When this feature is not enabled, use Cognito API operations to perform sign-up and sign-in.


App type | Info

Select an app type and we will automatically populate common default settings. You can add additional app clients after the user pool is created.

[✓ Public client] A native, browser or mobile-device app. Cognito API requests are made from user systems that are not trusted with a client secret.

Confidential client: A server-side application that can securely store a client secret. Cognito API requests are made from a central server.

Other: A custom app. Choose your own grant, auth flow, and client-secret settings.

App client name | Info

Enter a friendly name for your app client.

CourseApiClient

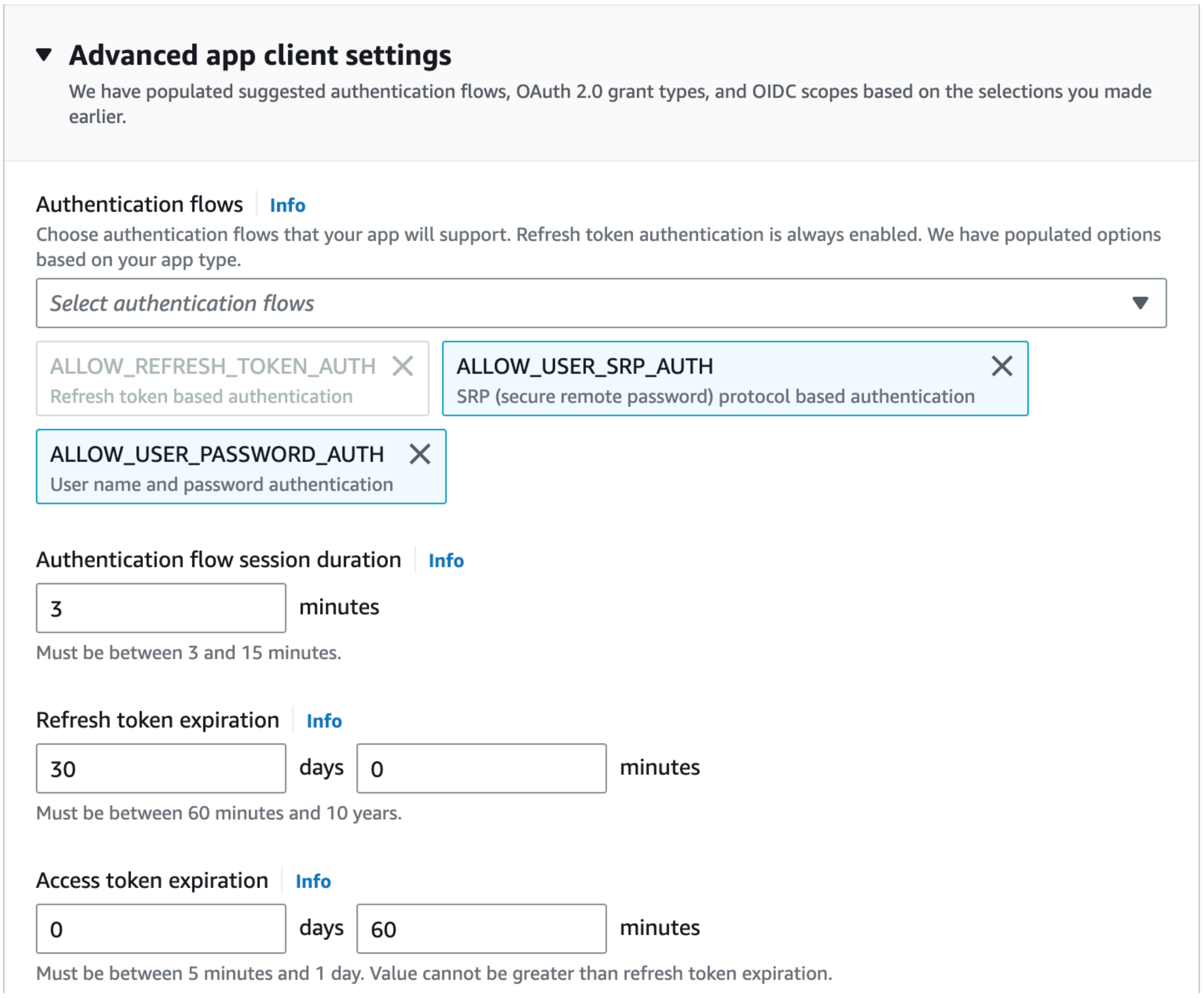
App client names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = , . @ -

Client secret | Info

Choose whether your app client will have a client secret. Client secrets are used by the server-side component of an app to authorize API requests. Using a client secret can prevent a third party from impersonating your client.

Generate a client secret [✓ Don't generate a client secret]

⚠ You cannot change or remove a client secret after you allow Amazon Cognito to generate it for your app client.


Advanced app client settings

We have populated suggested authentication flows, OAuth 2.0 grant types, and OIDC scopes based on the selections you made earlier.

Authentication flows | Info

Choose authentication flows that your app will support. Refresh token authentication is always enabled. We have populated options based on your app type.

Select authentication flows

ALLOW_REFRESH_TOKEN_AUTH
Refresh token based authentication

ALLOW_USER_SRP_AUTH
SRP (secure remote password) protocol based authentication

ALLOW_USER_PASSWORD_AUTH
User name and password authentication

Authentication flow session duration | Info

3 minutes

Must be between 3 and 15 minutes.

Refresh token expiration | Info

30 days 0 minutes

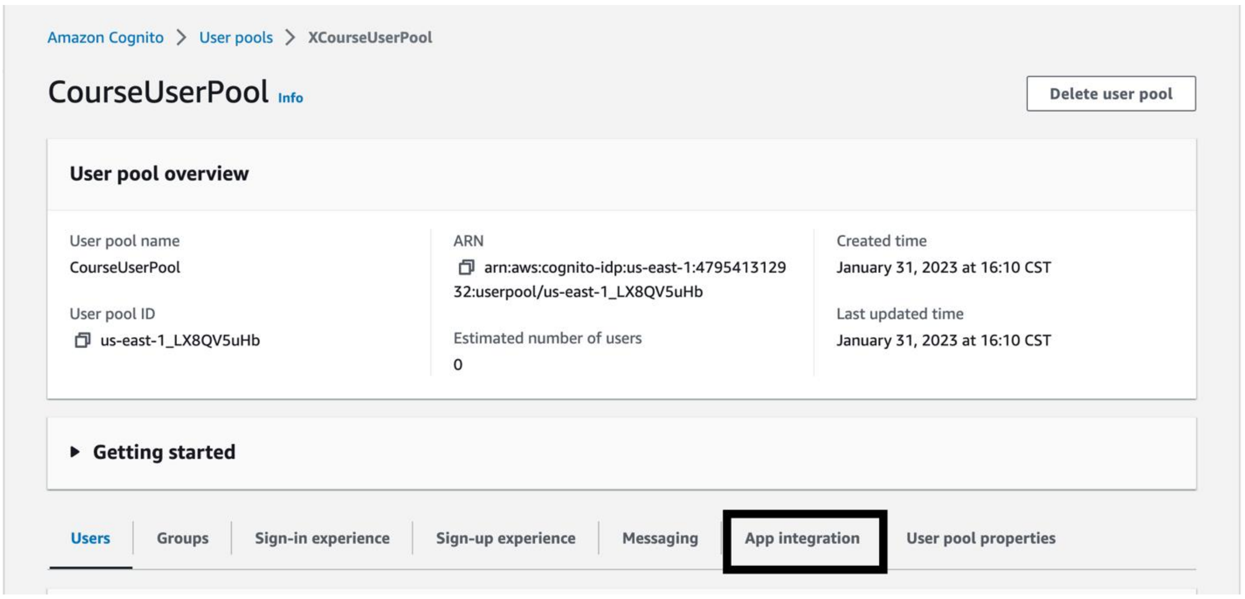
Must be between 60 minutes and 10 years.

Access token expiration | Info

0 days 60 minutes

Must be between 5 minutes and 1 day. Value cannot be greater than refresh token expiration.

3. Click on the **User pools** that you created. Then click on the **App integration** tab. Scroll down to the bottom. Then you will see your **Client ID**. Grab the **App client id** and store it somewhere. You will need it in the next steps.



Amazon Cognito > User pools > XCourseUserPool

CourseUserPool Info

Delete user pool

User pool overview

User pool name
CourseUserPool

User pool ID
us-east-1_LX8QV5uHb

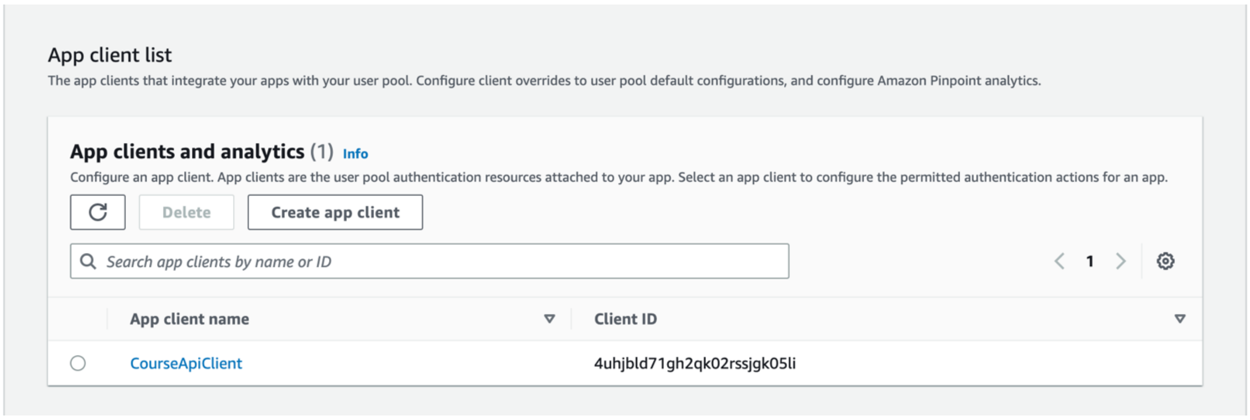
ARN
arn:aws:cognito-idp:us-east-1:479541312932:userpool/us-east-1_LX8QV5uHb

Estimated number of users
0

Created time
January 31, 2023 at 16:10 CST

Last updated time
January 31, 2023 at 16:10 CST

▶ Getting started



App client list

The app clients that integrate your apps with your user pool. Configure client overrides to user pool default configurations, and configure Amazon Pinpoint analytics.

App clients and analytics (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
CourseApiClient	4uhjbld71gh2qk02rssjgk05li

4. Create a user in your user pool with hosted UI or AWS CLI.

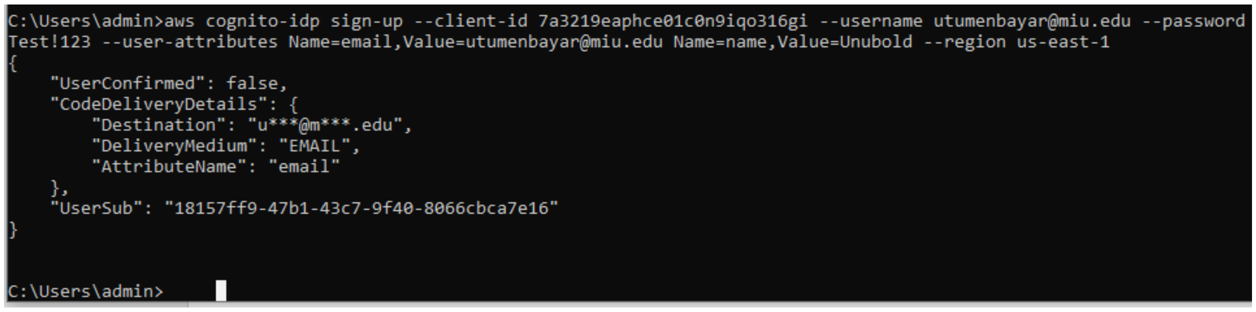
Hosted UI:

https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-app-integration.html

https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=http://localhost:3000

OR

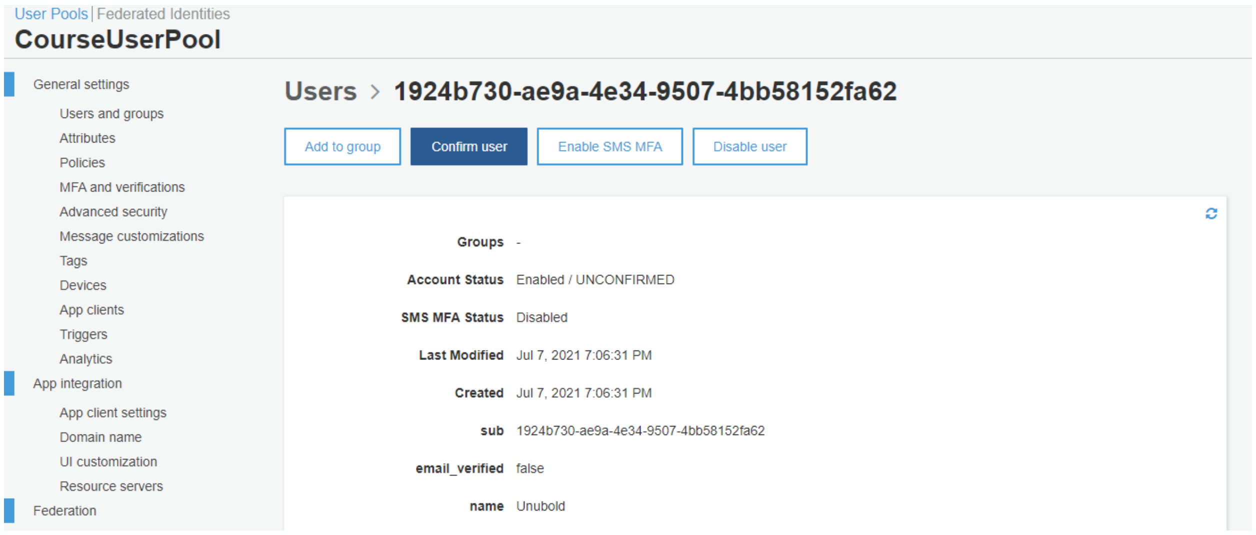
```
CLI: aws cognito-idp sign-up --client-id <<app_client_id>> --username <<your_email>>
--password Test123 --user-attributes Name=email,Value=<<your_email>>
Name=name,Value=<<your_first_name>> --region us-east-1
```



```
C:\Users\admin>aws cognito-idp sign-up --client-id 7a3219eaphce01c0n9iqo316gi --username utumenbayar@miu.edu --password
Test!123 --user-attributes Name=email,Value=utumenbayar@miu.edu Name=name,Value=Unubold --region us-east-1
{
    "UserConfirmed": false,
    "CodeDeliveryDetails": {
        "Destination": "u***@m***.edu",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    },
    "UserSub": "18157ff9-47b1-43c7-9f40-8066cbca7e16"
}

C:\Users\admin>
```

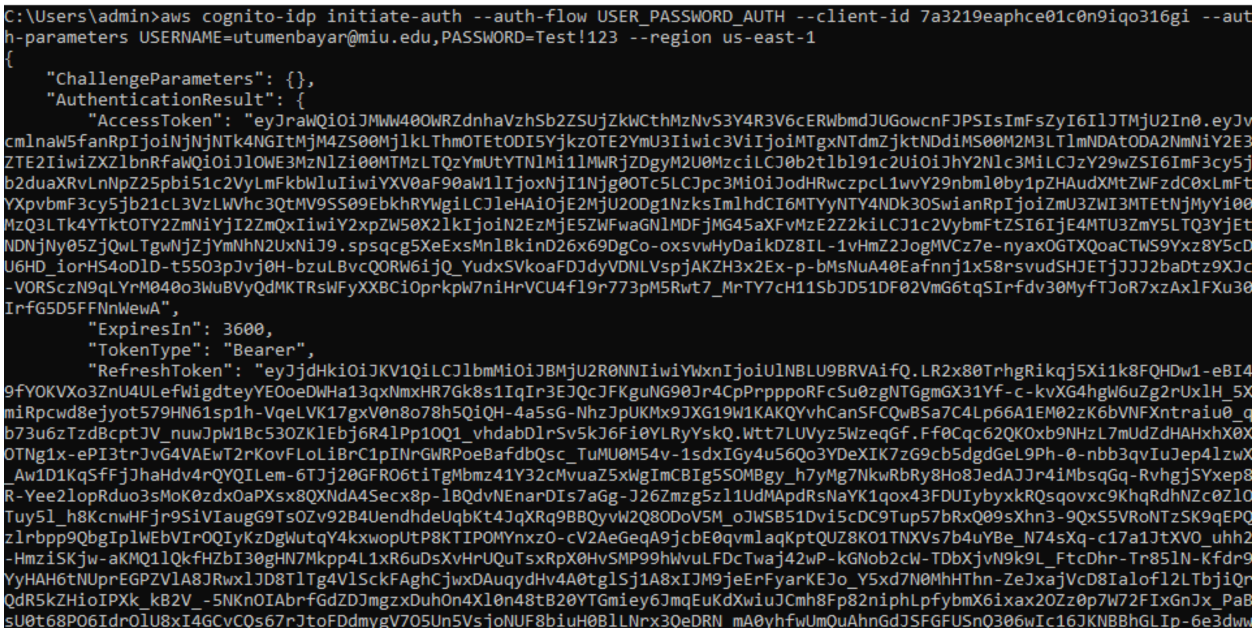
a. Go to your user pool and click on **Users and groups** in the left sidebar. Hit refresh icon on top right corner. That will pull the newly-created user. Click on the username which is UUID hyperlink. Click on **Confirm user** button.



Groups	-
Account Status	Enabled / UNCONFIRMED
SMS MFA Status	Disabled
Last Modified	Jul 7, 2021 7:06:31 PM
Created	Jul 7, 2021 7:06:31 PM
sub	1924b730-ae9a-4e34-9507-4bb58152fa62
email_verified	false
name	Unubold

b. Execute the command below that returns token associated with the user. That you need to provide after securing the API to store and retrieve data from the back-end or lambda. You may need to re-execute this command to get the new tokens in case it is expired. Based on how you configured the custom attributes, it could be slightly different.

```
aws cognito-idp initiate-auth --auth-flow USER_PASSWORD_AUTH --client-id
<<app_client_id>> --auth-parameters USERNAME=<<your_email>>,PASSWORD=Test123# --
region us-east-1
```

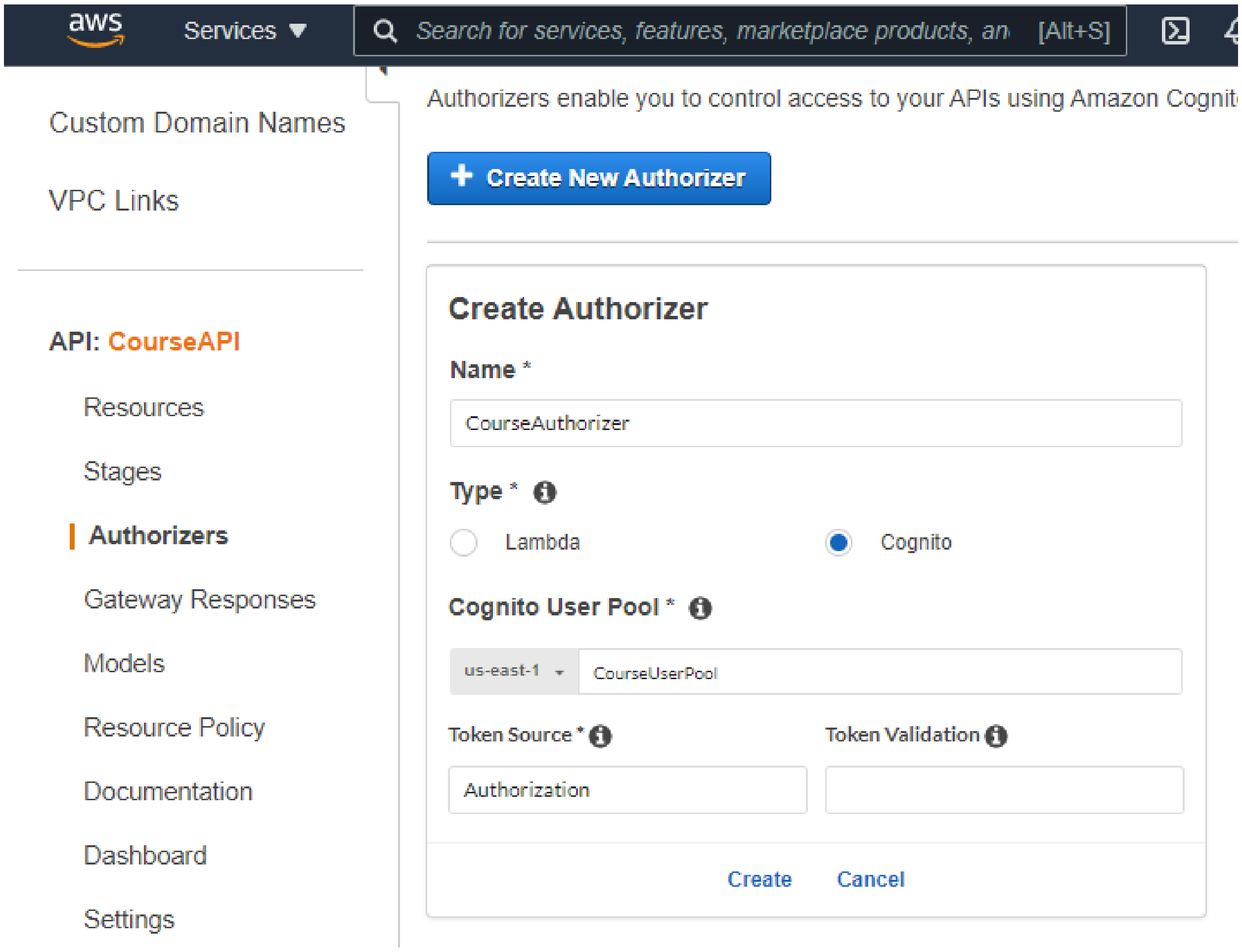


```
C:\Users\admin>aws cognito-idp initiate-auth --auth-flow USER_PASSWORD_AUTH --client-id 7a3219eaphce01c0n9iqo316gi --aut
h-parameters USERNAME=utumenbayar@miu.edu,PASSWORD=Test!123 --region us-east-1
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "eyJraWQiOiJMWW40OWRZdnhaVzhSb2ZSUjZkWCthMzNvS3Y4R3V6cERWbmdJUGowcnFJPSIsImFsZyI6IlJTMjU2In0.eyJv
cmlnaW5fanRpIjoiNjNjNTk4NGItMjM4ZS00MjlkLThmOTEtODI5YjkzOTE2YmU3Iiwic3ViIjoiMTgxNTdmZjktNDdiMS00M2M3LTlmNDAtODA2NmNiY2E3
ZTE2IiwiZXZlbnRfaWQiOiJlOWE3MzNlZi00MTMzLTQzYmUtYTNlMi1lMWRjZDgyM2U0MzciLCJ0b2tlbl91c2UiOiJhY2Nlc3MiLCJzY29wZSI6ImF3cy5j
b2duaXRvLnNpZ25pbi51c2VyLmFkbWluIiwiYXV0aF90aW1lIjoxNjI1Njg0OTc5LCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAudXMtZWFzdC0xLmFt
YXpvbmF3cy5jb21cL3VzLWVhc3QtMV9SS09EbkhRYWgiLCJleHAiOjE2MjU2ODg1NzksImlhdCI6MTYyNTY4NDk3OSwianRpIjoiZmU3ZWI3MTEtNjMyYi00
MzQ3LTk4YTktOTY2ZmNiYjI2ZmQxIiwiY2xpZW50X2lkIjoiN2EzMjE5ZWFwaGNlMDFjMG45aXFvMzE2Z2kiLCJ1c2VybmFtZSI6IjE4MTU3ZmY5LTQ3YjEt
NDNjNy05ZjQwLTgwNjZjYmNhN2UxNiJ9.spsqcg5XeExsMnlBkinD26x69DgCo-oxsvwHyDaikDZ8IL-1vHmZ2JogMVCz7e-nyaxOGTXQoaCTWS9Yxz8Y5cD
U6HD_iorHS4oDlD-t55O3pJvj0H-bzuLBvcQORW6ijQ_YudxSVkoaFDJdyVDNLVspjAKZH3x2Ex-p-bMsNuA40Eafnnj1x58rsvudSHJETjJJJ2baDtz9XJc
-VORSczN9qLYrM040o3WuBVyQdMKTRsWFyXXBCiOprkpW7niHrVCU4fl9r773pM5Rwt7_MrTY7cH11SbJD51DF02VmG6tqSIrfdv30MyfTJoR7xzAxlFXu30
IrfG5D5FFNnWewA",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.LR2x80TrhgRikqj5Xi1k8FQHDw1-eBI4
9fYOKVXo3ZnU4ULefWigdteyYEOoeDWHa13qxNmxHR7Gk8s1IqIr3EJQcJFKguNG90Jr4CpPrpppoRFcSu0zgNTGgmGX31Yf-c-kvXG4hgW6uZg2rUxlH_5X
miRpcwd8ejyot579HN61sp1h-VqeLVK17gxV0n8o78h5QiQH-4a5sG-NhzJpUKMx9JXG19W1KAKQYvhCanSFCQwBSa7C4Lp66A1EM02zK6bVNFXntraiu0_q
b73u6zTzdBcptJV_nuwJpW1Bc53OZKlEbj6R4lPp1OQ1_vhdabDlrSv5kJ6Fi0YLRyYskQ.Wtt7LUVyz5WzeqGf.Ff0Cqc62QKOxb9NHzL7mUdZdHAHxhX0X
OTNg1x-ePI3trJvG4VAEwT2rKovFLoLiBrC1pINrGWRPoeBafdbQsc_TuMU0M54v-1sdxIGy4u56Qo3YDeXIK7zG9cb5dgdGeL9Ph-0-nbb3qvIuJep4lzwX
_Aw1D1KqSfFjJhaHdv4rQYQILem-6TJj20GFRO6tiTgMbmz41Y32cMvuaZ5xWgImCBIg5SOMBgy_h7yMg7NkwRbRy8Ho8JedAJJr4iMbsqGq-RvhgjSYxep8
R-Yee2lopRduo3sMoK0zdxOaPXsx8QXNdA4Secx8p-lBQdvNEnarDIs7aGg-J26Zmzg5zl1UdMApdRsNaYK1qox43FDUIybyxkRQsqovxc9KhqRdhNZc0ZlO
Tuy5l_h8KcnwHFjr9SiVIaugG9TsOZv92B4UendhdeUqbKt4JqXRq9BBQyvW2Q8ODoV5M_oJWSB51Dvi5cDC9Tup57bRxQ09sXhn3-9QxS5VRoNTzSK9qEPQ
zlrbpp9QbgIplWEbVIrOQIyKzDgWutqY4kxwopUtP8KTIPOMYnxzO-cV2AeGeqA9jcbE0qvmlaqKptQUZ8KO1TNXVs7b4uYBe_N74sXq-c17a1JtXVO_uhh2
-HmziSKjw-aKMQ1lQkfHZbI30gHN7Mkpp4L1xR6uDsXvHrUQuTsxRpX0HvSMP99hWvuLFDcTwaj42wP-kGNob2cW-TDbXjvN9k9L_FtcDhr-Tr85lN-Kfdr9
YyHAH6tNUprEGPZVlA8JRwxlJD8TlTg4VlSckFAghCjwxDAuqydHv4A0tglSj1A8xIJM9jeErFyarKEJo_Y5xd7N0MhHThn-ZeJxajVcD8Ialofl2LTbjiQr
QdR5kZHioIPXk_kB2V_-5NKnOIAbrfGdZDJmgzxDuhOn4Xl0n48tB20YTGmiey6JmqEuKdXwiuJCmh8Fp82niphLpfybmX6ixax2OZz0p7W72FIxGnJx_PaB
sU0t68PO6IdrOlU8xI4GCvCOs67rJtoFDdmygV7O5Un5VsjoNUF8biuH0BlLNrx3QeDRN_mA0yhfwUmQuAhnGdJSFGFUSnQ306wIc16JKNBBhGLIp-6e3dww
```

Securing the API

c. Go to API Gateway. Go to your API. Click on **Authorizers** in the left sidebar.Click on **Create New Authorizer.**

d. Name as **CourserAuthorizer.** Type is **Cognito.** Select the user pool you created. **Token Source** is **Authorization**.



Authorizers enable you to control access to your APIs using Amazon Cognit

Create Authorizer

Name * CourseAuthorizer

Type * Lambda [✓ Cognito]

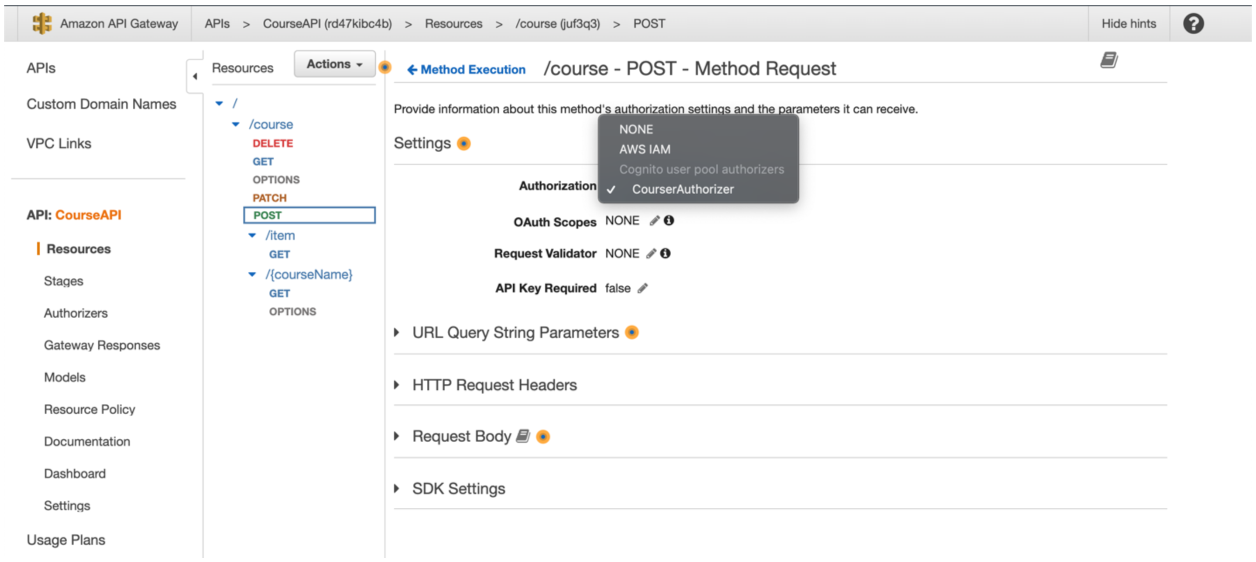
Cognito User Pool * us-east-1 CourseUserPool

Token Source * Authorization

Token Validation

e. Go to **Resources.** Select the **POST** method under course resource.

f. Refresh the whole page. Click on **Method Request. Authorization** is the authorizer you just created. Click on OK icon.



/course - POST - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization: CourserAuthorizer

OAuth Scopes NONE

Request Validator NONE

API Key Required false

URL Query String Parameters

HTTP Request Headers

Request Body

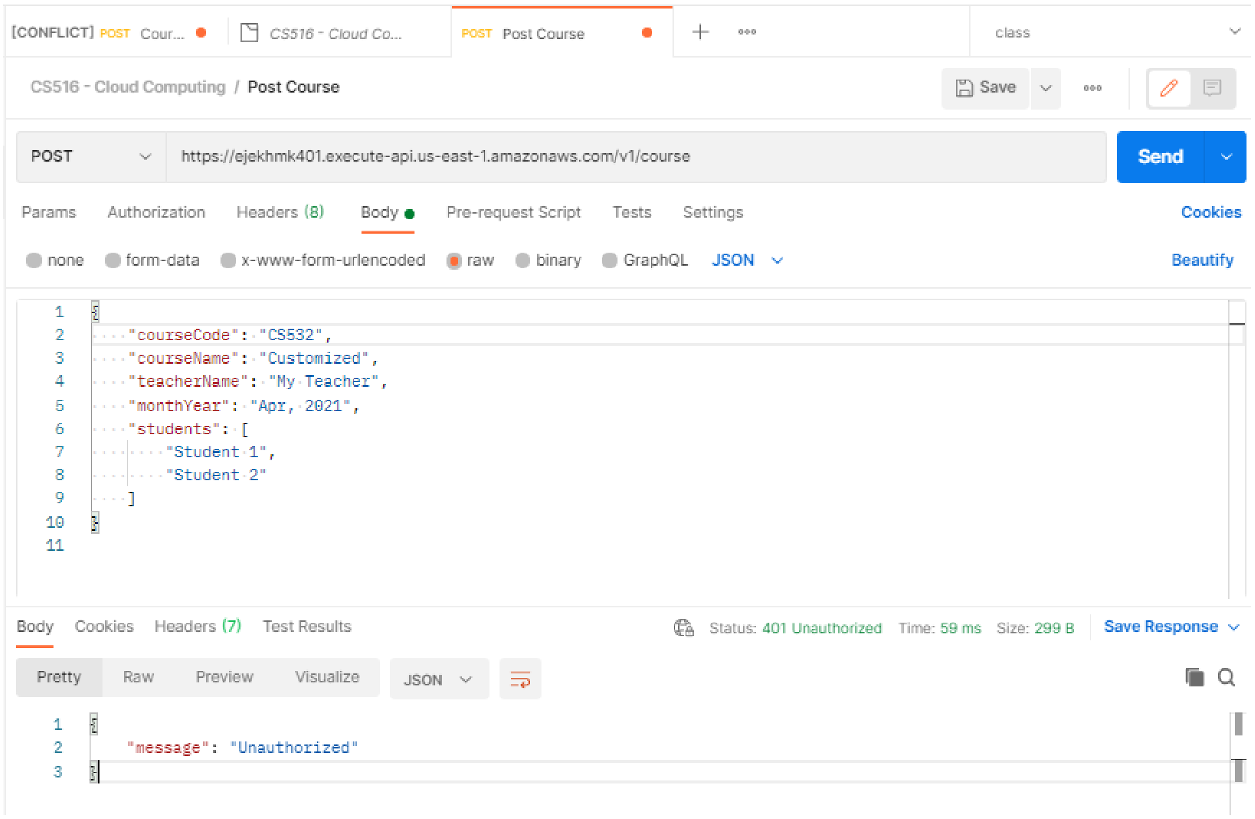
SDK Settings

g. Secure the GET endpoint as well by using the authorizer you created earlier. Do the step c and d on the GET.

h. Actions -> Deploy API -> Go with the existing stage.

5. Test.

a. As see you below. Your endpoint is secured. You must provide the tokens that we generated in previous steps in Authorization header.



POST https://ejekhmk401.execute-api.us-east-1.amazonaws.com/v1/course

```
{
    "courseCode": "CS532",
    "courseName": "Customized",
    "teacherName": "My Teacher",
    "monthYear": "Apr, 2021",
    "students": [
        "Student 1",
        "Student 2"
    ]
}
```

Status: 401 Unauthorized Time: 59 ms Size: 299 B

```
{
    "message": "Unauthorized"
}
```

b. Copy the **ID Token**. Provide it in the header as **Authorization.**