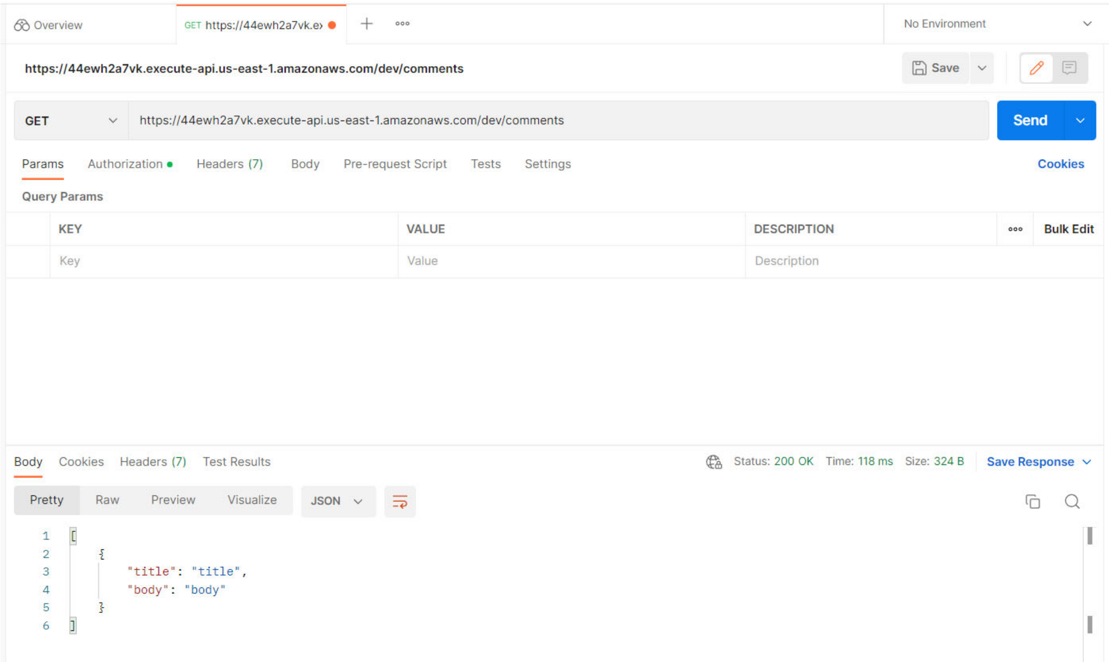
Task 1: For this task, you should fill in the function template code. Create a constant that is an array of Javascript objects and call it `comments`. The constant `comments` will store comments submitted by users. Each element of `comments` must have a title and a body e.g `[{title: “a”, body: “b”}, {...}]`. The returned value of the function should be a Javascript object. One of the fields in that object should be called `body`. Create the object having the body field and assign the `comments` constant to the `body` field. Report the completed function code.

```
const comments = [{title: "title", body: "body"}];
exports.handler = async (event) => {
    var response = {
        "body": JSON.stringify(comments)
    };
    return response;
};
```



https://44ewh2a7vk.execute-api.us-east-1.amazonaws.com/dev/comments

GET https://44ewh2a7vk.execute-api.us-east-1.amazonaws.com/dev/comments

Status: 200 OK Time: 118 ms Size: 324 B

```
[
    {
        "title": "title",
        "body": "body"
    }
]
```

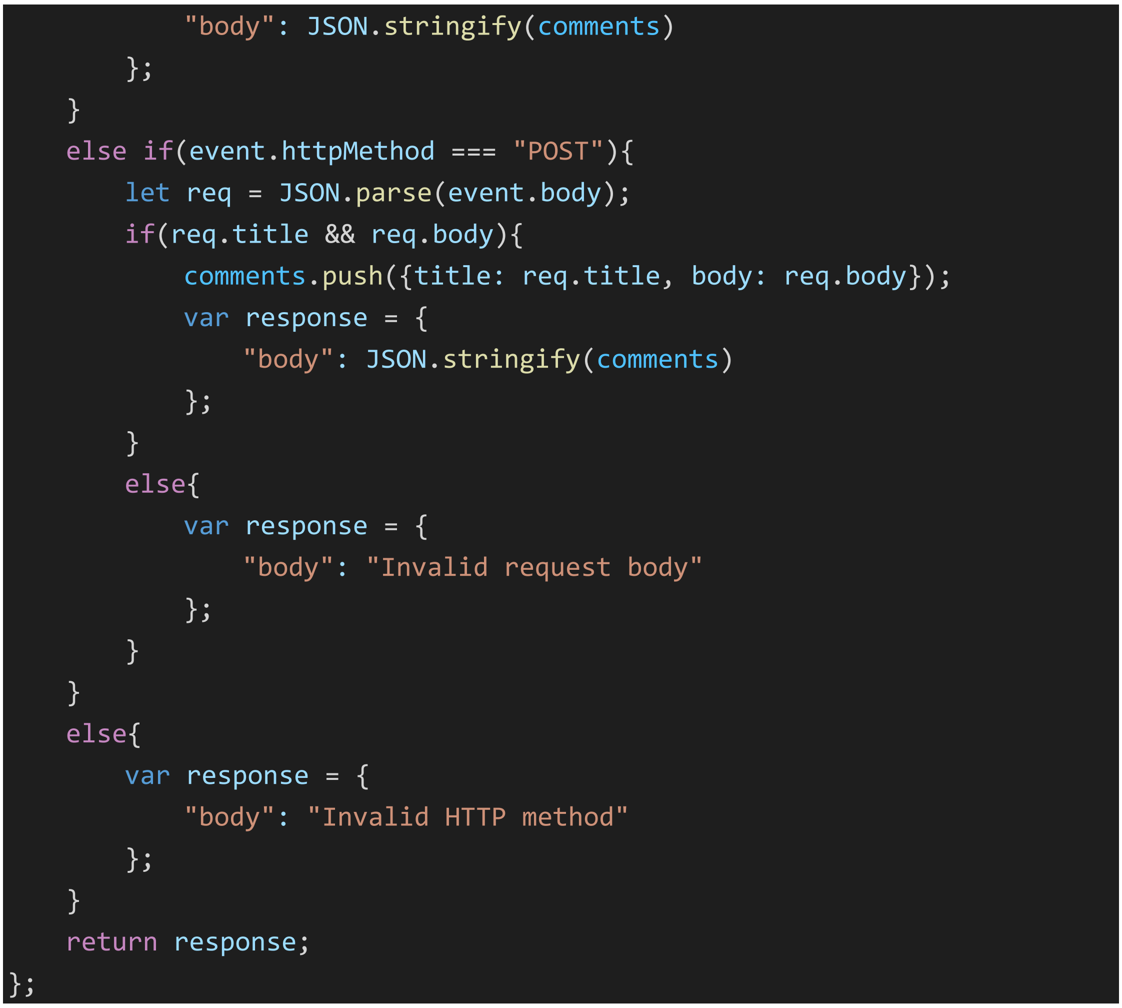
Task 2: Discuss the differences, pros and cons between creating a serverless API using Lambda vs creating a server on Amazon’s EC2 you learned in the previous assignments.

	AWS Lambda	**Amazon EC2**
Management	User only provides code to the service. AWS handles all aspects of the infrastructure.	User manages all aspects of provisioning, deployment and operation.
Price	Pay-per-execution and compute time used for those executions.	Monthly/hourly cost varies depending on VM type and size.

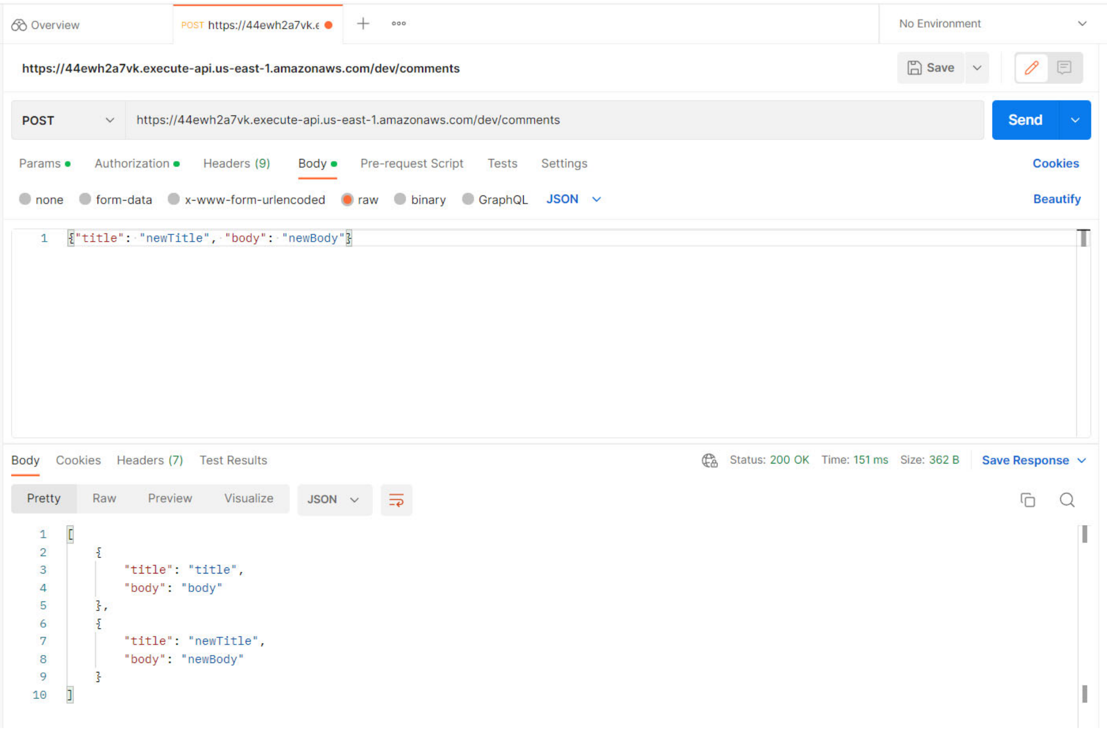
		User is billed monthly.
Performance	Always available but only runs when executed. Up to 100 ms delay before code is loaded and executed. Runtime limited to 15 minutes and cannot use more than 3008 MB.	Instance runs until it is deliberately stopped. The application in the instance is immediately available and running.
Security	Users can employ IAM role permissions for Lambda. The underlying infrastructure is patched and secured by AWS.	User is responsible for instance and application security. Additional AWS services may be needed to implement proper EC2 security.
Dependencies	The only real dependency is the function code uploaded to the Lambda service.	Users must connect and configure any dependencies for the Ec2 instance.
Flexibility	Code size and execution time limits functions to specific, highly focused tasks.	Can run almost any application in a suitable instance.
Scalability	Scales dynamically in response to traffic and automatically adjusts the number of concurrently executing functions.	User is responsible for scaling but can use services such as EC2 Auto Scaling groups.

Task 3: In order for our Lambda to be able to work with GET and POST requests we need to modify it. Modify the Lambda function you created in order to be able to serve a GET and a POST method request. For a GET request, you should return the set of comments. For a POST request, read the body of the request, parse it using JSON.parse() and add it to the comments set. Please consider doing the bare minimum validation to the POST body on your Lambda so that you don't add null values or that it does not crash when provided with a null POST body. Report the code.

```
const comments = [{title: "title", body: "body"}];
exports.handler = async (event) => {
    if(event.httpMethod === "GET"){
        var response = {
```



```
            "body": JSON.stringify(comments)
        };
    }
    else if(event.httpMethod === "POST"){
        let req = JSON.parse(event.body);
        if(req.title && req.body){
            comments.push({title: req.title, body: req.body});
            var response = {
                "body": JSON.stringify(comments)
            };
        }
        else{
            var response = {
                "body": "Invalid request body"
            };
        }
    }
    else{
        var response = {
            "body": "Invalid HTTP method"
        };
    }
    return response;
};
```



https://44ewh2a7vk.execute-api.us-east-1.amazonaws.com/dev/comments

POST https://44ewh2a7vk.execute-api.us-east-1.amazonaws.com/dev/comments

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{"title": "newTitle", "body": "newBody"}
```

Status: 200 OK Time: 151 ms Size: 362 B

```
[
    {
        "title": "title",
        "body": "body"
    },
    {
        "title": "newTitle",
        "body": "newBody"
    }
]
```

Task 4: If you create a new comment and then do a GET, you should be able to get all previously inserted comments including your newly created comment. If you wait for a while (e.g. 30 minutes) and do a GET again you will realize that your newly created comment is not returned (only the default `comments` constant you created in your Lambda is returned). Explain the reason. What AWS component do you need to create in order for your comments to persist in time?

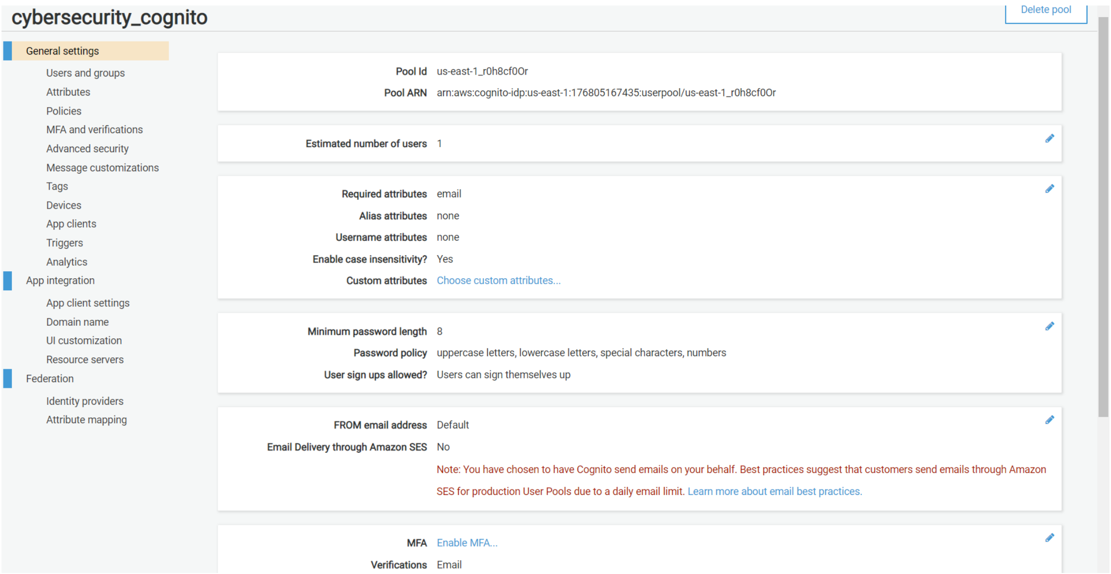
The reason:
AWS Lambda is an on-demand compute service that powers many serverless applications. Lambda functions are ephemeral, with execution environments only existing for a brief time when the function is invoked. Therefore, when we do a GET again 30 minutes after doing a POST, the array comments modified by the POST have been destroyed, and the lambda function will create a new constant array comments.
AWS component for persist data:
Amazon S3

Task 5: Explain the steps needed to integrate your newly created AWS Cognito to the POST method you developed in Task 3, so that your POST methods are authorized.

Make sure to test your configuration and verify that it works by doing a POST request on your API and checking that you are not authorized to access the API (Don't forget to redeploy the API).

Furthermore, report a short demonstration of the steps in order to invoke your Cognito Authorized endpoint of your API (POST method).

I used Way1:
Create a user pool:



Pool Id us-east-1_r0h8cf0Or

Pool ARN arn:aws:cognito-idp:us-east-1:176805167435:userpool/us-east-1_r0h8cf0Or

Estimated number of users 1

Required attributes email

Alias attributes none

Username attributes none

Enable case insensitivity? Yes

Custom attributes Choose custom attributes...

Minimum password length 8

Password policy uppercase letters, lowercase letters, special characters, numbers

User sign ups allowed? Users can sign themselves up

FROM email address Default

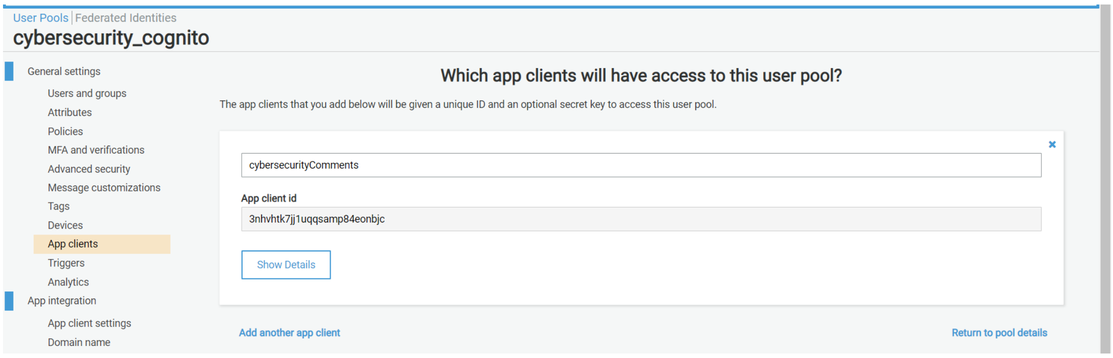
Email Delivery through Amazon SES No

Note: You have chosen to have Cognito send emails on your behalf. Best practices suggest that customers send emails through Amazon SES for production User Pools due to a daily email limit. Learn more about email best practices.

MFA Enable MFA...

Verifications Email

Add an app client:



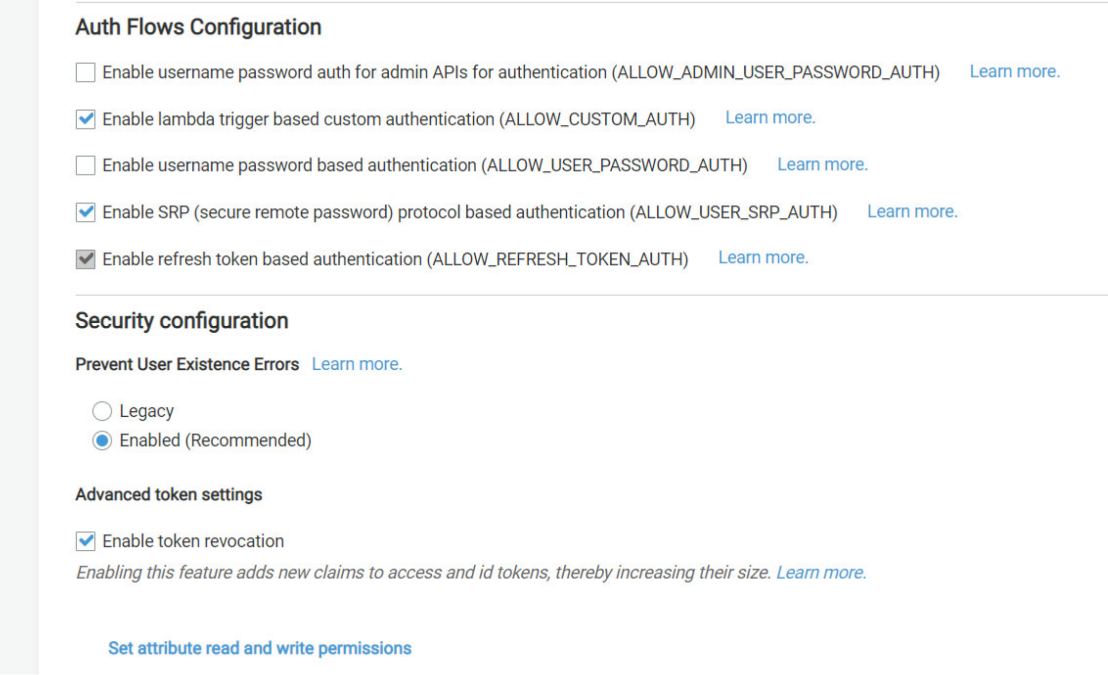
Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

cybersecurityComments

App client id

3nhvhtk7jj1uqqsamp84eonbjc



Auth Flows Configuration

□ Enable username password auth for admin APIs for authentication (ALLOW_ADMIN_USER_PASSWORD_AUTH) Learn more.

☑ Enable lambda trigger based custom authentication (ALLOW_CUSTOM_AUTH) Learn more.

□ Enable username password based authentication (ALLOW_USER_PASSWORD_AUTH) Learn more.

☑ Enable SRP (secure remote password) protocol based authentication (ALLOW_USER_SRP_AUTH) Learn more.

☑ Enable refresh token based authentication (ALLOW_REFRESH_TOKEN_AUTH) Learn more.

Security configuration

Prevent User Existence Errors Learn more.

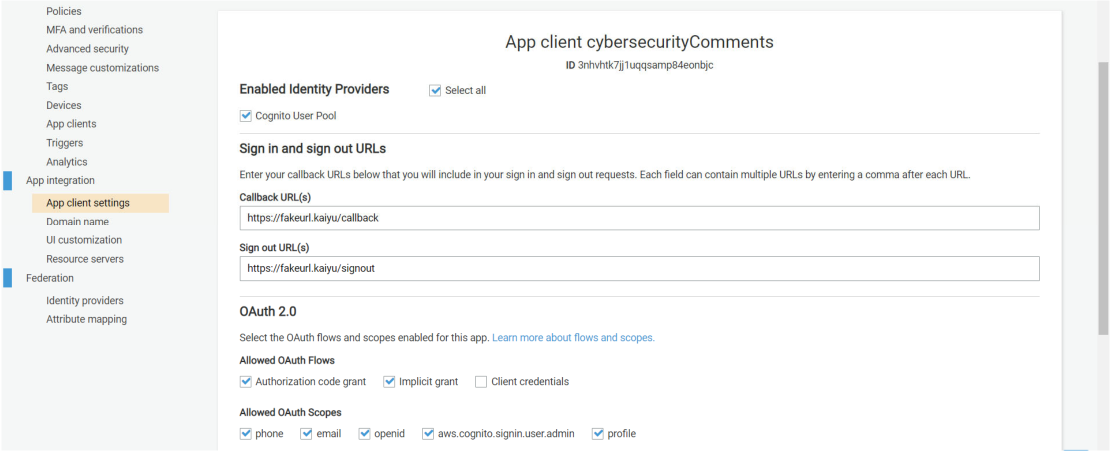
○ Legacy
● Enabled (Recommended)

Advanced token settings

☑ Enable token revocation

Enabling this feature adds new claims to access and id tokens, thereby increasing their size. Learn more.

Configure the app



App client cybersecurityComments

ID 3nhvhtk7jj1uqqsamp84eonbjc

Enabled Identity Providers ☑ Select all

☑ Cognito User Pool

Sign in and sign out URLs

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

Callback URL(s)

https://fakeurl.kaiyu/callback

Sign out URL(s)

https://fakeurl.kaiyu/signout

OAuth 2.0

Select the OAuth flows and scopes enabled for this app. Learn more about flows and scopes.

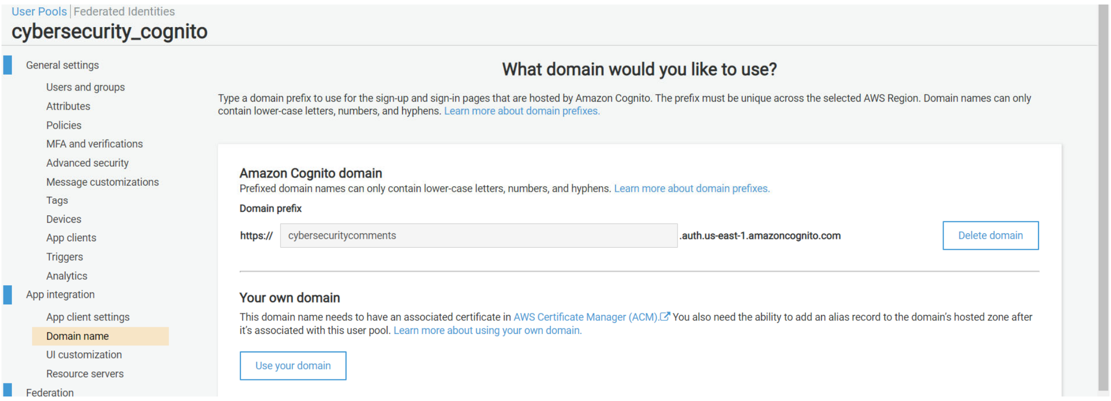
Allowed OAuth Flows

☑ Authorization code grant ☑ Implicit grant □ Client credentials

Allowed OAuth Scopes

☑ phone ☑ email ☑ openid ☑ aws.cognito.signin.user.admin ☑ profile

Configure a domain



cybersecurity_cognito

What domain would you like to use?

Type a domain prefix to use for the sign-up and sign-in pages that are hosted by Amazon Cognito. The prefix must be unique across the selected AWS Region. Domain names can only contain lower-case letters, numbers, and hyphens. Learn more about domain prefixes.

Amazon Cognito domain

Prefixed domain names can only contain lower-case letters, numbers, and hyphens. Learn more about domain prefixes.

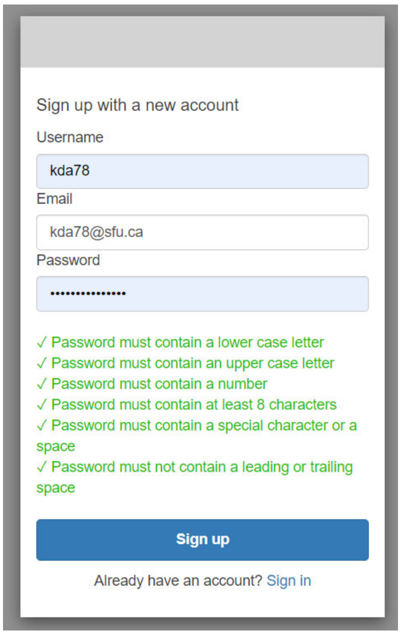
Domain prefix

https:// cybersecuritycomments .auth.us-east-1.amazoncognito.com

Your own domain

This domain name needs to have an associated certificate in AWS Certificate Manager (ACM). You also need the ability to add an alias record to the domain's hosted zone after it's associated with this user pool. Learn more about using your own domain.

To view the sign-in page, sign up and sign in:



Sign up with a new account

Username

kda78

Email

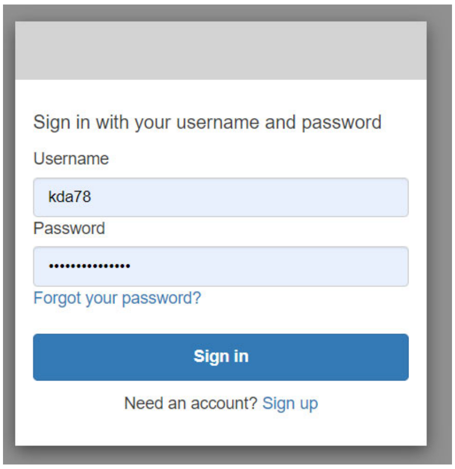
kda78@sfu.ca

Password

•••••••••••••••

✓ Password must contain a lower case letter
✓ Password must contain an upper case letter
✓ Password must contain a number
✓ Password must contain at least 8 characters
✓ Password must contain a special character or a space
✓ Password must not contain a leading or trailing space

Already have an account? Sign in



Sign in with your username and password

Username

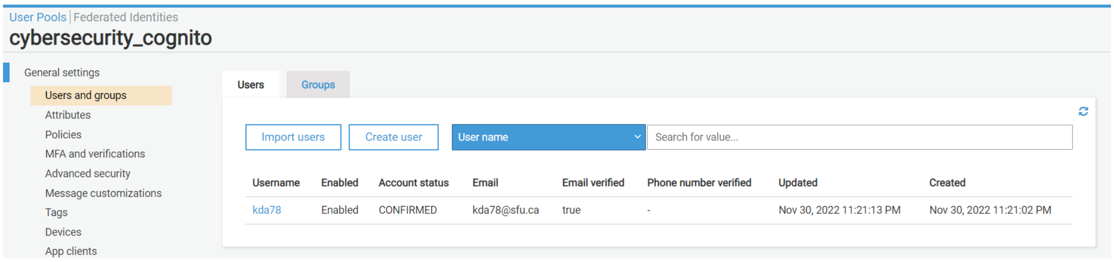
kda78

Password

•••••••••••••••

Forgot your password?

Need an account? Sign up



User Pools | Federated Identities

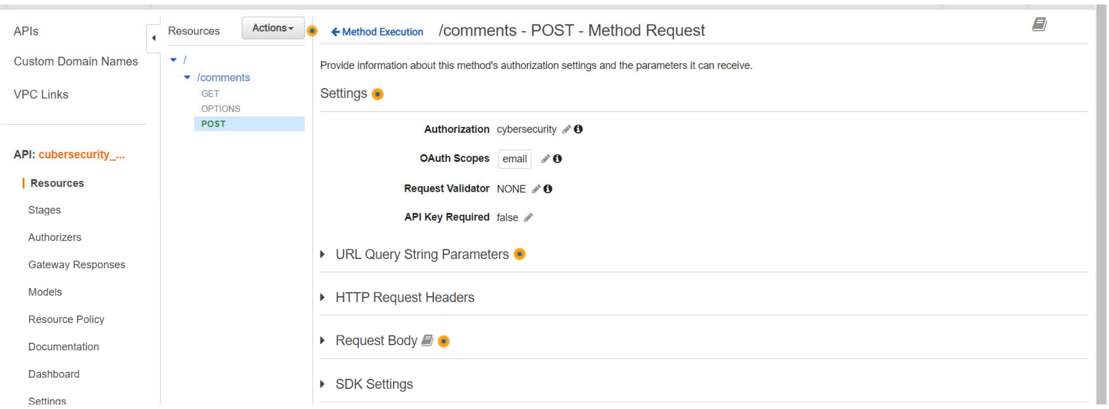
cybersecurity_cognito

Username	Enabled	Account status	Email	Email verified	Phone number verified	Updated	Created
kda78	Enabled	CONFIRMED	kda78@sfu.ca	true	-	Nov 30, 2022 11:21:13 PM	Nov 30, 2022 11:21:02 PM

Change response_type=token to get id_token from Amazon Cognito:

https://fakeurl.kaiyu/callback#id_token=eyJraWQiOiJVRzdhc3QrMlpyb0NVRndOV2w5MUozc0xSZWhzUmRlemVQMklcL3IzNVJuWT0iLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiSW1sYndZT0VFTG53aU82SlRZSVJjZyIsInN1YiI6ImJkYTFiOTAzLWY3M2MtNGQ0Ny1hMTg4LTVjMTI3ODc5Yzk0ZSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAudXMtZWFzdC0xLmFtYXpvbmF3cy5jb21cL3VzLWVhc3QtMV9yMGg4Y2YwT3IiLCJjb2duaXRvOnVzZXJuYW1lIjoia2RhNzgiLCJhdWQiOiIzbmh2aHRrN2pqMXVxcXNhbXA4NGVvbmJqYyIsInRva2VuX3VzZSI6ImlkIiwiYXV0aF90aW1lIjoxNjY5ODY2NjQzLCJleHAiOjE2Njk4NzAyNDMsImlhdCI6MTY2OTg2NjY0MywianRpIjoiZTVjOTdlNjktOTA0OC00MjVhLTg4NDUtNGJlZmM3NmFiMTgxIiwiZW1haWwiOiJrZGE3OEBzZnUuY2EifQ.pHVXYEO0MT--aehJ6wwPLdUsW9j6C1YHe2jBuX5d-LbOET8K_49xlJOsB-l_nYZq5geezC8HGDKJM50b1E4UXeW6GN1ALXe9Q74kzXFulizMjSJfZVgJscsDup57X6Bs6Z3kawW17p6Ur9MtfOVpOpQDe6V49At8a_gHY_mgVaPVrQfGLcdkLTwc6Xwz3hOhWgjwsOQKqPQ5ernd7BDTfG35jtTfXSlLOhDIjCtCxRAlh2-MKS4iuCO6pJblFLaN0xCBgNYkFexPAPEvyyfWJeAs4BrxYDAGFmDJm4VSR3O0flAuR9XUYMVf30ARfVZL6KjkWzdpnztxOFW2SrjBgw&access_token=eyJraWQiOiJROGZEd2hFbXJuenJMYld1akZ1SkgrZ24zYWFBK3c2MjJhUlFoK3E0VUhRPSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiJiZGExYjkwMy1mNzNjLTRkNDctYTE4OC01YzEyNzg3OWM5NGUiLCJ0b2tlbl91c2UiOiJhY2Nlc3MiLCJzY29wZSI6ImF3cy5jb2duaXRvLnNpZ25pbi51c2VyLmFkbWluIHBob25lIG9wZW5pZCBwcm9maWxlIGVtYWlsIiwiYXV0aF90aW1lIjoxNjY5ODY2NjQzLCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAudXMtZWFzdC0xLmFtYXpvbmF3cy5jb21cL3VzLWVhc3QtMV9yMGg4Y2YwT3IiLCJleHAiOjE2Njk4NzAyNDMsImlhdCI6MTY2OTg2NjY0MywidmVyc2lvbiI6MiwianRpIjoiZjEzOWM5MmUtNzVmZC00M2YyLTkyMjEtMWZmODFmMDY5ZWE1IiwiY2xpZW50X2lkIjoiM25odmh0azdqajF1cXFzYW1wODRlb25iamMiLCJ1c2VybmFtZSI6ImtkYTc4In0.jDN1j7Cgeq-FP6PujEY5uoAg4kTezBZ6-Aa1rA_hqnYU0JVnDuy6ZyZQXZeef7Y10ye8RfXHfinjFZC1aR6NspfftsgdVzAdhr5VAIQVt58eMtR9uwTWs_Wf2f1OLf2JfxK1ThMSBr7eXVami39G-DITdpuBBhAbptVH04Dg7xyKHxT_OIO-Qqm7heLUc63daChrjxz_uQCb72Q1YkbJZvMVIFk99ocpd9JHDaEQprFV0WE5KucBLI5h8b-x5n2XVnSXi51wZ4-t561wo-uM4hk2jkdB7pGuYw422zC1EBHldoMPQFOBHNrzIG-GugMI9syWdhcIKvf7-VDXb7LQjg&expires_in=3600&token_type=Bearer

Config API gateway:



/comments - POST - Method Request

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

Settings

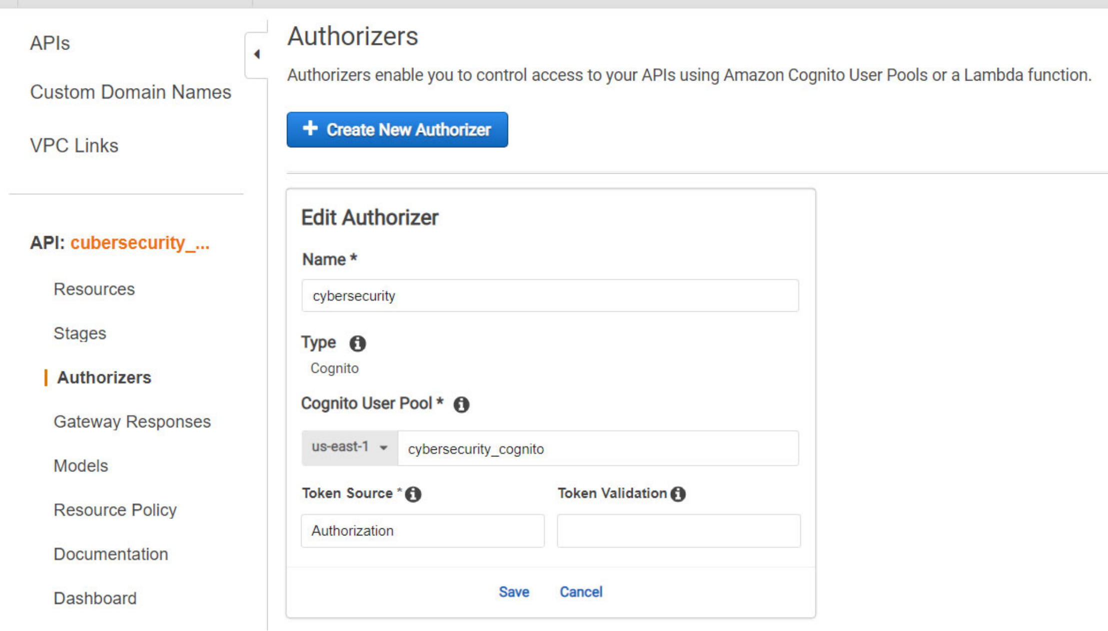
Authorization cybersecurity

OAuth Scopes email

Request Validator NONE

API Key Required false

- URL Query String Parameters
- HTTP Request Headers
- Request Body
- SDK Settings



Authorizers

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.

Edit Authorizer

Name *: cybersecurity

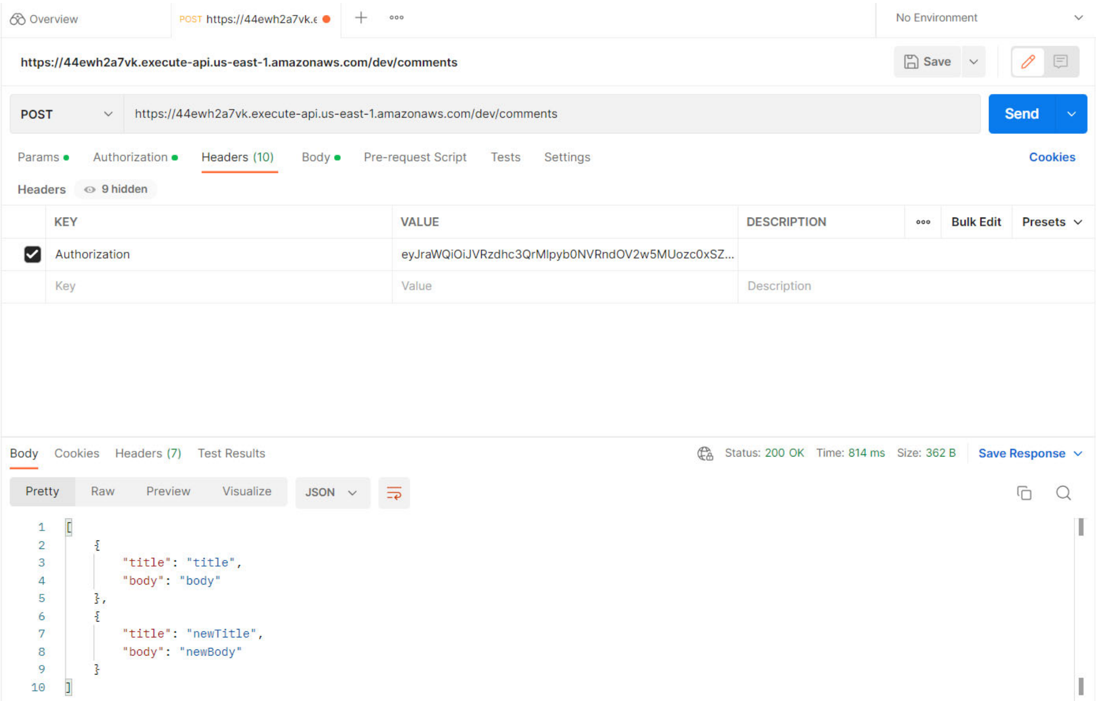
Type: Cognito

Cognito User Pool *: us-east-1 cybersecurity_cognito

Token Source *: Authorization

Token Validation:

Do a POST test:



https://44ewh2a7vk.execute-api.us-east-1.amazonaws.com/dev/comments

POST https://44ewh2a7vk.execute-api.us-east-1.amazonaws.com/dev/comments

	KEY	VALUE	DESCRIPTION
☑	Authorization	eyJraWQiOiJVRzdhc3QrMlpyb0NVRndOV2w5MUozc0xSZ...	
	Key	Value	Description

Status: 200 OK Time: 814 ms Size: 362 B

```
[
    {
        "title": "title",
        "body": "body"
    },
    {
        "title": "newTitle",
        "body": "newBody"
    }
]
```

Task 6: A user or a potential attacker can manipulate that token to increase the expiration time. However, this wouldn’t work as the team that created JWT has taken measures against such attacks. Now, suppose that as an attack you could change the expiration time or the authenticated user type of the JWT token from your browser. Explain why this attack would not work.

Because JWT uses digital signature to verify the authenticity of the token.
The server that issues the token typically generates the signature by hashing the header and payload. In some cases, they also encrypt the resulting hash. Either way, this process involves a secret signing key. This mechanism provides a way for servers to verify that none of the data within the token has been tampered with since it was issued:
As the signature is directly derived from the rest of the token, changing a single byte of the header or payload results in a mismatched signature.
Without knowing the server's secret signing key, it shouldn't be possible to generate the correct signature for a given header or payload.