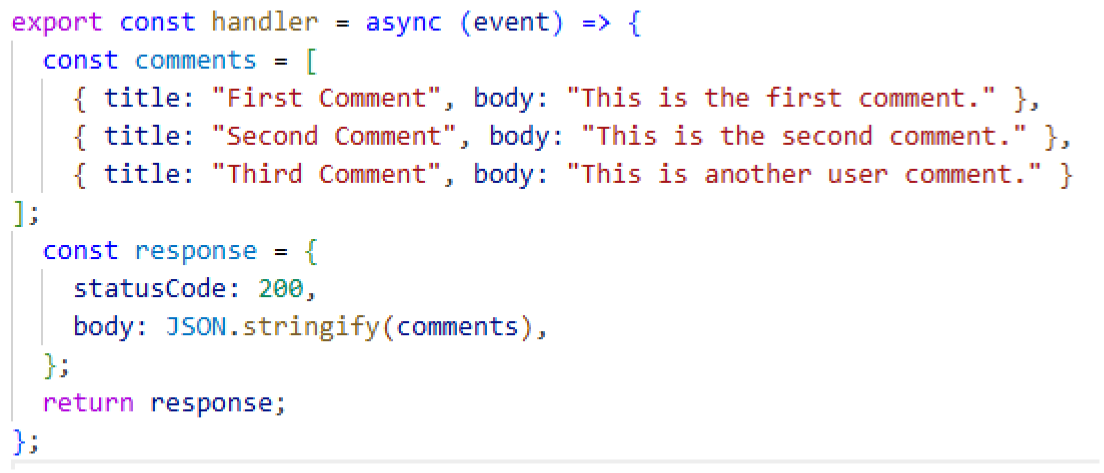
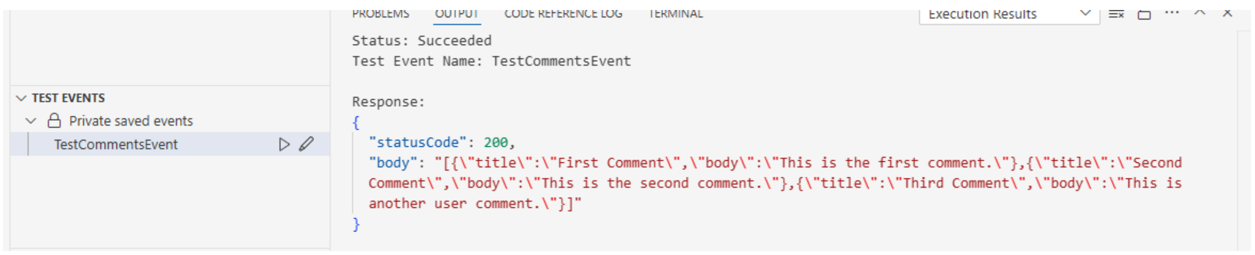
Task 1



```
export const handler = async (event) => {
  const comments = [
    { title: "First Comment", body: "This is the first comment." },
    { title: "Second Comment", body: "This is the second comment." },
    { title: "Third Comment", body: "This is another user comment." }
];
  const response = {
    statusCode: 200,
    body: JSON.stringify(comments),
  };
  return response;
};
```



```
Status: Succeeded
Test Event Name: TestCommentsEvent

Response:
{
  "statusCode": 200,
  "body": "[{\"title\":\"First Comment\",\"body\":\"This is the first comment.\"},{\"title\":\"Second
  Comment\",\"body\":\"This is the second comment.\"},{\"title\":\"Third Comment\",\"body\":\"This is
  another user comment.\"}]"
}
```

Task 2

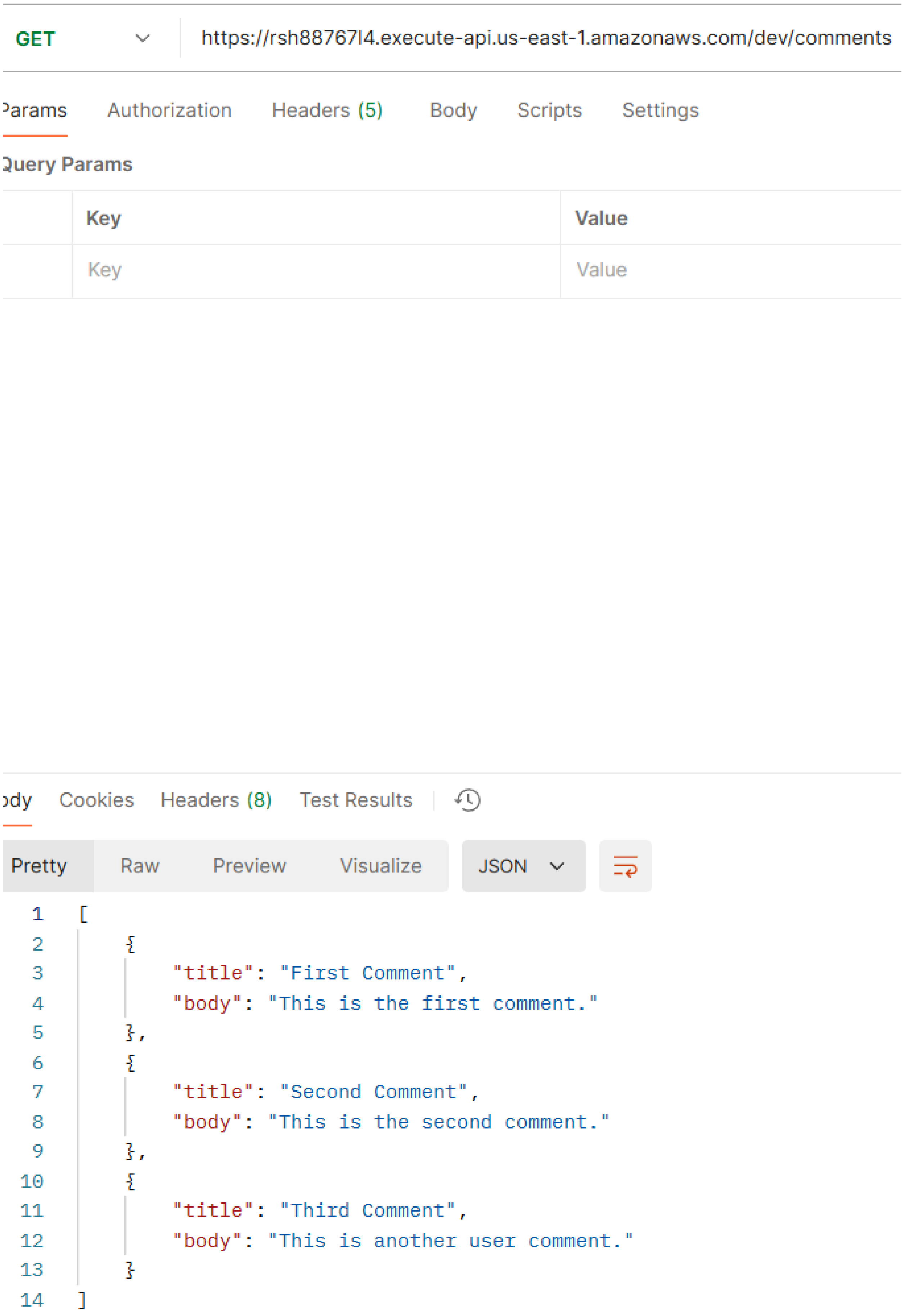
AWS Lambda is a serverless service, which means you don’t need to set up or manage any servers. It automatically runs your code when needed, and you only pay for the time it takes to run. This makes it great for tasks that happen occasionally or have unpredictable demand. However, Lambda has limits,such as only running tasks for up to 15 minutes. You also don’t have much control over how it runs behind the scenes.

Amazon EC2, on the other hand, gives you complete control over a virtual server. You can decide everything about how it runs, which is helpful for applications that need specific settings or need to run all the time. But with EC2, you are responsible for managing the server, including updates and maintenance, which takes more work. It can also be more expensive since you pay for the server the whole time it’s running, even if you’re not using it a lot.

Task 3

For code plz see the js file

GET

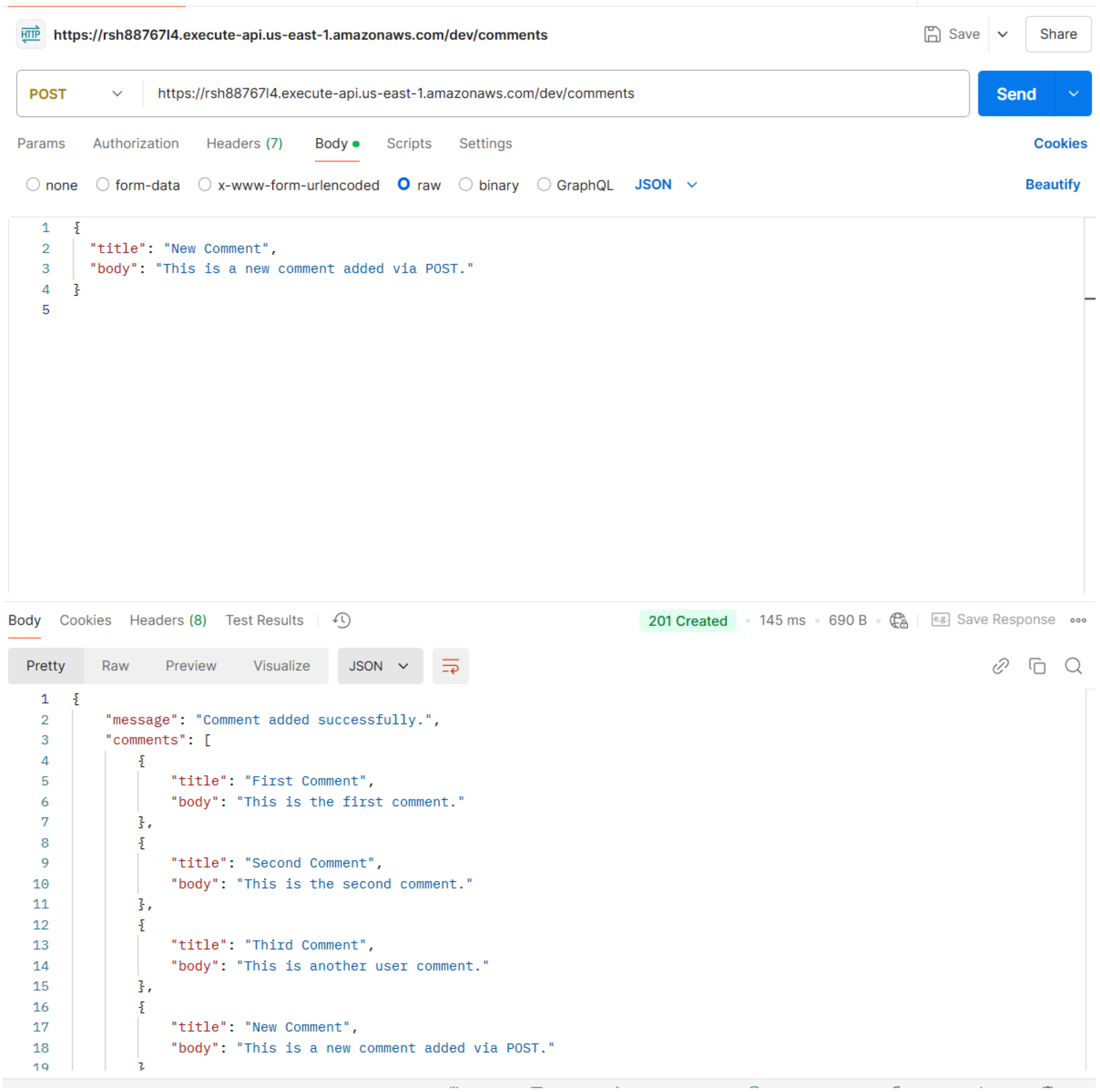


GET https://rsh88767l4.execute-api.us-east-1.amazonaws.com/dev/comments

Key	Value
Key	Value

```
[
    {
        "title": "First Comment",
        "body": "This is the first comment."
    },
    {
        "title": "Second Comment",
        "body": "This is the second comment."
    },
    {
        "title": "Third Comment",
        "body": "This is another user comment."
    }
]
```

POST


https://rsh887674.execute-api.us-east-1.amazonaws.com/dev/comments

POST https://rsh887674.execute-api.us-east-1.amazonaws.com/dev/comments

```
{
  "title": "New Comment",
  "body": "This is a new comment added via POST."
}
```

201 Created • 145 ms • 690 B

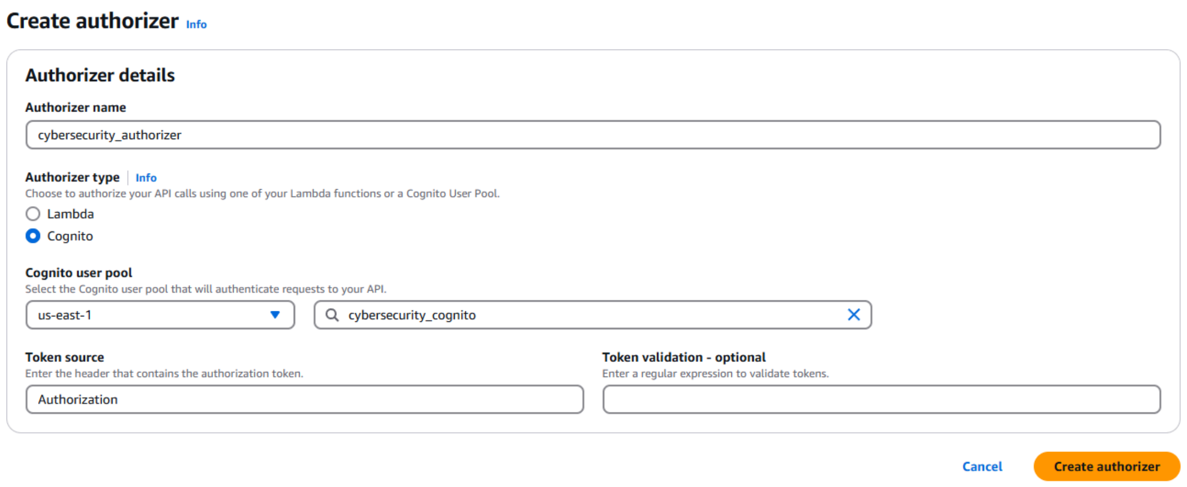
```
{
    "message": "Comment added successfully.",
    "comments": [
        {
            "title": "First Comment",
            "body": "This is the first comment."
        },
        {
            "title": "Second Comment",
            "body": "This is the second comment."
        },
        {
            "title": "Third Comment",
            "body": "This is another user comment."
        },
        {
            "title": "New Comment",
            "body": "This is a new comment added via POST."
        }
```

Task 4

The reason the newly created comment disappears after some time is because AWS Lambda is stateless. This means that any data stored in the Lambda function is not saved after the function completes. Every time the Lambda runs, it starts fresh, which is why only the original hardcoded comments are returned.

To keep comments persistent, we need a database to store them such as Amazon DynamoDB. DynamoDB can store the comments permanently, allowing you to read and write comments between Lambda executions. This way, your data won't be lost, and all your added comments will persist.

Task 5
First we need to set the Authorizer in API Gateway to the user pool we just created



Create authorizer Info

Authorizer details

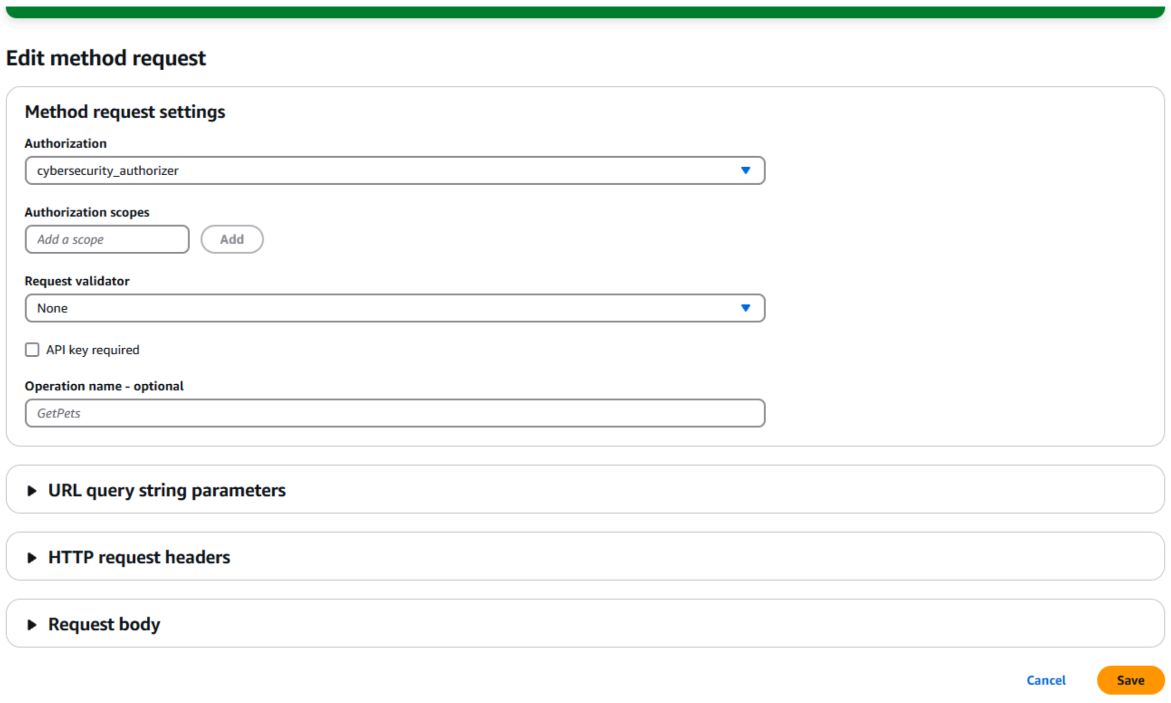
Authorizer name: cybersecurity_authorizer

Authorizer type | Info
Choose to authorize your API calls using one of your Lambda functions or a Cognito User Pool.
○ Lambda
● Cognito

Cognito user pool
Select the Cognito user pool that will authenticate requests to your API.
us-east-1 | cybersecurity_cognito

Token source
Enter the header that contains the authorization token.
Authorization

Token validation - optional
Enter a regular expression to validate tokens.



Edit method request

Method request settings

Authorization: cybersecurity_authorizer

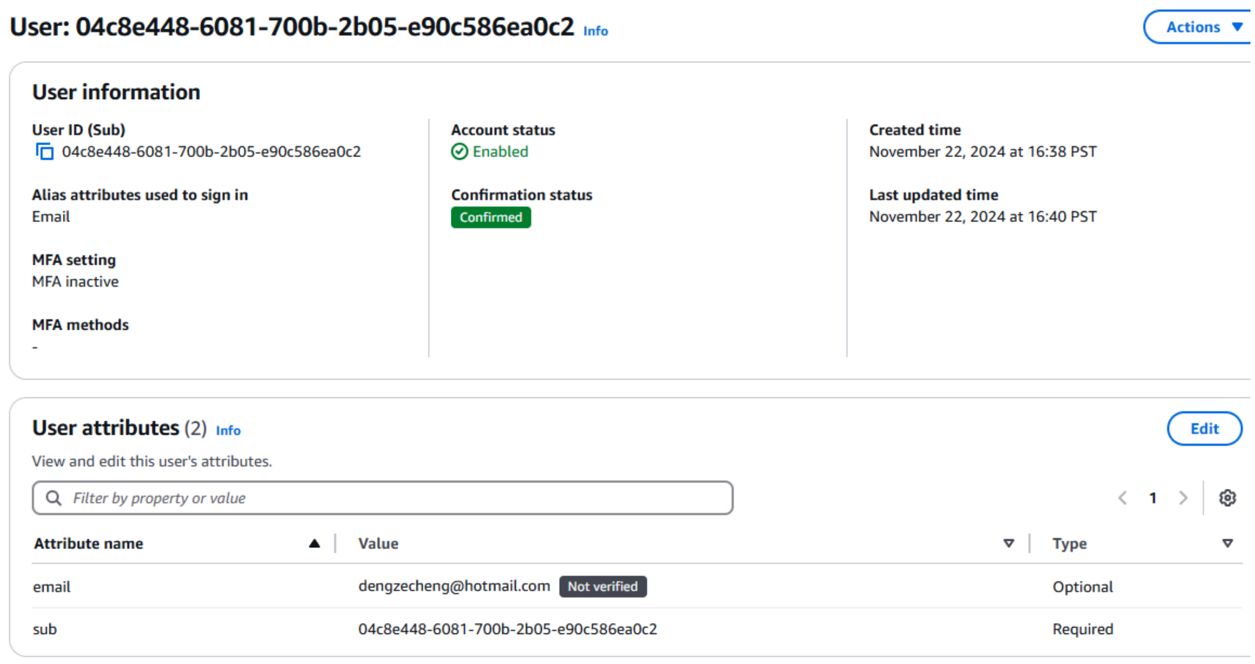
Authorization scopes: Add a scope

Request validator: None

☐ API key required

Operation name - optional: GetPets

On the Cognito side, we need to create a new user.


User: 04c8e448-6081-700b-2b05-e90c586ea0c2 Info

Actions

User information

User ID (Sub)
04c8e448-6081-700b-2b05-e90c586ea0c2

Alias attributes used to sign in
Email

MFA setting
MFA inactive

MFA methods
-

Account status
Enabled

Confirmation status
Confirmed

Created time
November 22, 2024 at 16:38 PST

Last updated time
November 22, 2024 at 16:40 PST

User attributes (2) Info

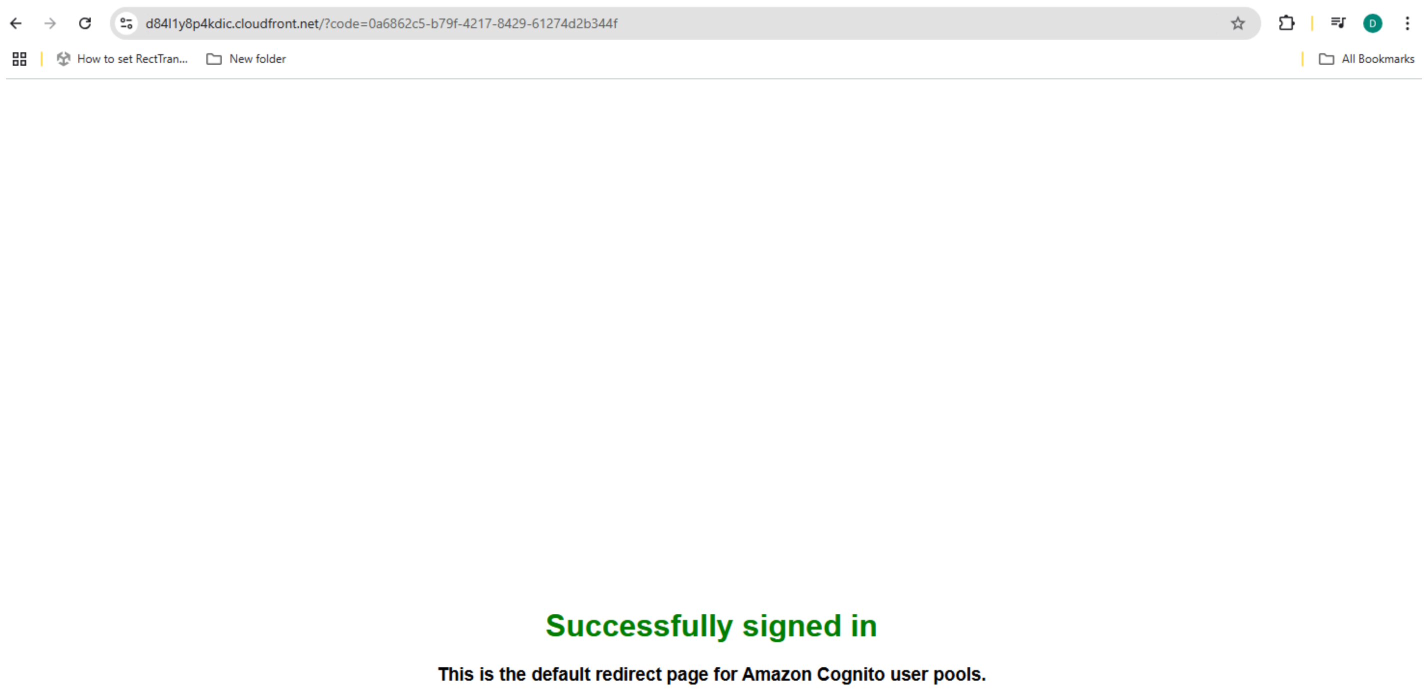
Edit

View and edit this user's attributes.

Filter by property or value

Attribute name	Value	Type
email	dengzecheng@hotmail.com Not verified	Optional
sub	04c8e448-6081-700b-2b05-e90c586ea0c2	Required

We login-in the Hosted-UI with that user, and we can see there is a Authorization Code in the URL

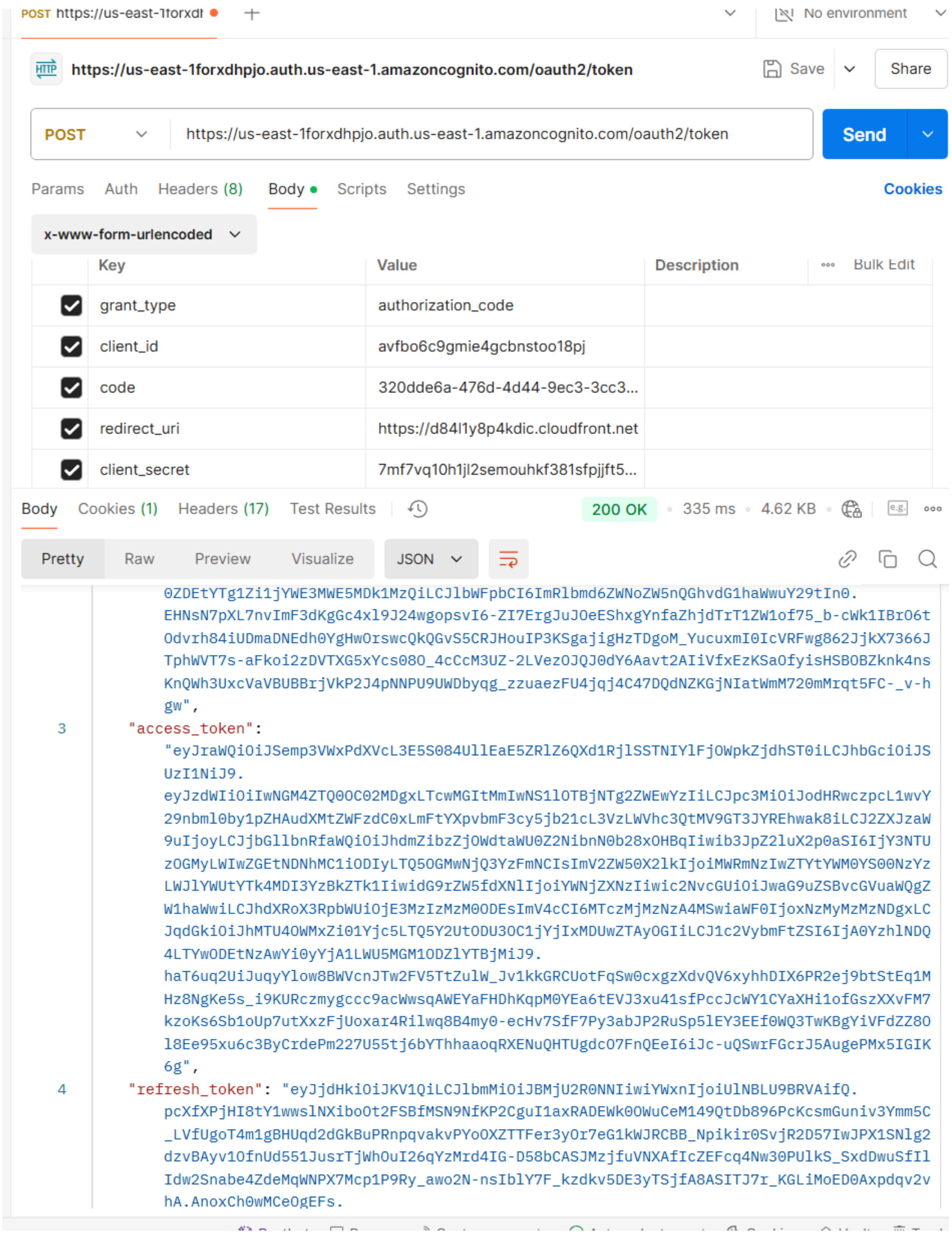

d84l1y8p4kdic.cloudfront.net/?code=0a6862c5-b79f-4217-8429-61274d2b344f

How to set RectTran... New folder All Bookmarks

Successfully signed in

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

With that code, we can send the Authorization Code to exchange JWT tokens.
All the information in header can be found in Cognito



https://us-east-1forxdhpjo.auth.us-east-1.amazoncognito.com/oauth2/token

POST https://us-east-1forxdhpjo.auth.us-east-1.amazoncognito.com/oauth2/token

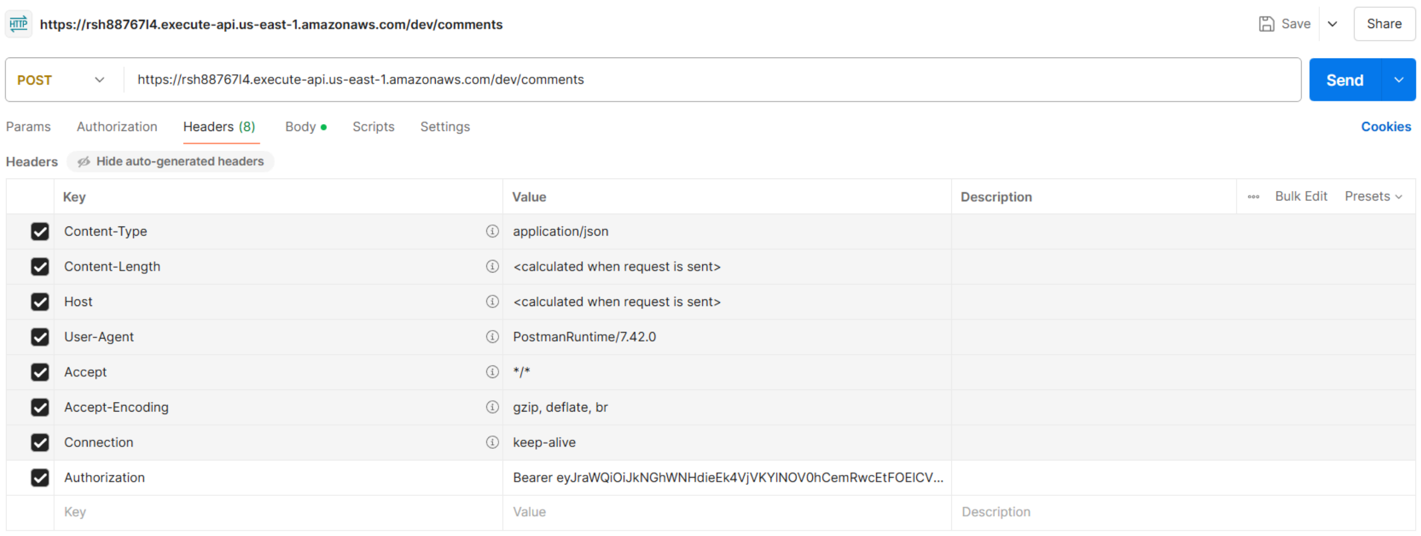
x-www-form-urlencoded

	Key	Value	Description
✓	grant_type	authorization_code	
✓	client_id	avfbo6c9gmie4gcbnstoo18pj	
✓	code	320dde6a-476d-4d44-9ec3-3cc3...	
✓	redirect_uri	https://d84l1y8p4kdic.cloudfront.net	
✓	client_secret	7mf7vq10h1jl2semouhkf381sfpjjft5...	

200 OK • 335 ms • 4.62 KB

```
        0ZDEtYTg1Zi1jYWE3MWE5MDk1MzQiLCJlbWFpbCI6ImRlbmd6ZWNoZW5nQGhvdG1haWwuY29tIn0.
        EHNsN7pXL7nvImF3dKgGc4xl9J24wgopsvI6-ZI7ErgJuJ0eEShxgYnfaZhjdTrT1ZW1of75_b-cWk1IBr06t
        Odvrh84iUDmaDNEdh0YgHwOrswcQkQGvS5CRJHouIP3KSgajigHzTDgoM_YucuxmI0IcVRFwg862JjkX7366J
        TphWVT7s-aFkoi2zDVTXG5xYcs080_4cCcM3UZ-2LVezOJQJ0dY6Aavt2AIiVfxEzKSaOfyisHSBOBZknk4ns
        KnQWh3UxcVaVBUBBrjVkP2J4pNNPU9UWDbyqg_zzuaezFU4jqj4C47DQdNZKGjNIatWmM720mMrqt5FC-_v-h
        gw",
    "access_token":
        "eyJraWQiOiJSemp3VWxPdXVcL3E5S084UllEaE5ZRlZ6QXd1RjlSSTNIYlFjOWpkZjdhST0iLCJhbGciOiJS
        UzI1NiJ9.
        eyJzdWIiOiIwNGM4ZTQ0OC02MDgxLTcwMGItMmIwNS1lOTBjNTg2ZWEwYzIiLCJpc3MiOiJodHRwczpcL1wvY
        29nbml0by1pZHAudXMtZWFzdC0xLmFtYXpvbmF3cy5jb21cL3VzLWVhc3QtMV9GT3JYREhwak8iLCJ2ZXJzaW
        9uIjoyLCJjbGllbnRfaWQiOiJhdmZibzZjOWdtaWU0Z2NibnN0b28xOHBqIiwib3JpZ2luX2p0aSI6IjY3NTU
        zOGMyLWIwZGEtNDNhMC1iODIyLTQ5OGMwNjQ3YzFmNCIsImV2ZW50X2lkIjoiMWRmNzIwZTYtYWM0YS00NzYz
        LWJlYWUtYTk4MDI3YzBkZTk1IiwidG9rZW5fdXNlIjoiYWNjZXNzIiwic2NvcGUiOiJwaG9uZSBvcGVuaWQgZ
        W1haWwiLCJhdXRoX3RpbWUiOjE3MzIzMzM0ODEsImV4cCI6MTczMjMzNzA4MSwiaWF0IjoxNzMyMzMzNDgxLC
        JqdGkiOiJhMTU4OWMxZi01Yjc5LTQ5Y2UtODU3OC1jYjIxMDUwZTAyOGIiLCJ1c2VybmFtZSI6IjA0YzhlNDQ
        4LTYwODEtNzAwYi0yYjA1LWU5MGM1ODZlYTBjMiJ9.
        haT6uq2UiJuqyYlow8BWVcnJTw2FV5TtZulW_Jv1kkGRCUotFqSw0cxgzXdvQV6xyhhDIX6PR2ej9btStEq1M
        Hz8NgKe5s_i9KURczmygccc9acWwsqAWEYaFHDhKqpM0YEa6tEVJ3xu41sfPccJcWY1CYaXHi1ofGszXXvFM7
        kzoKs6Sb1oUp7utXxzFjUoxar4Rilwq8B4my0-ecHv7SfF7Py3abJP2RuSp5lEY3EEf0WQ3TwKBgYiVFdZZ8O
        l8Ee95xu6c3ByCrdePm227U55tj6bYThhaaoqRXENuQHTUgdcO7FnQEeI6iJc-uQSwrFGcrJ5AugePMx5IGIK
        6g",
    "refresh_token": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.
        pcXfXPjHI8tY1wwslNXiboOt2FSBfMSN9NfKP2CguI1axRADEWk0OWuCeM149QtDb896PcKcsmGuniv3Ymm5C
        _LVfUgoT4m1gBHUqd2dGkBuPRnpqvakvPYoOXZTTFer3yOr7eG1kWJRCBB_Npikir0SvjR2D57IwJPX1SNlg2
        dzvBAyv1OfnUd551JusrTjWhOuI26qYzMrd4IG-D58bCASJMzjfuVNXAfIcZEFcq4Nw30PUlkS_SxdDwuSfIl
        Idw2Snabe4ZdeMqWNPX7Mcp1P9Ry_awo2N-nsIblY7F_kzdkv5DE3yTSjfA8ASITJ7r_KGLiMoED0Axpdqv2v
        hA.AnoxCh0wMCeOgEFs.
```

After that we can use the Token we got and input it in the Header of the Post request for the Lambda function.



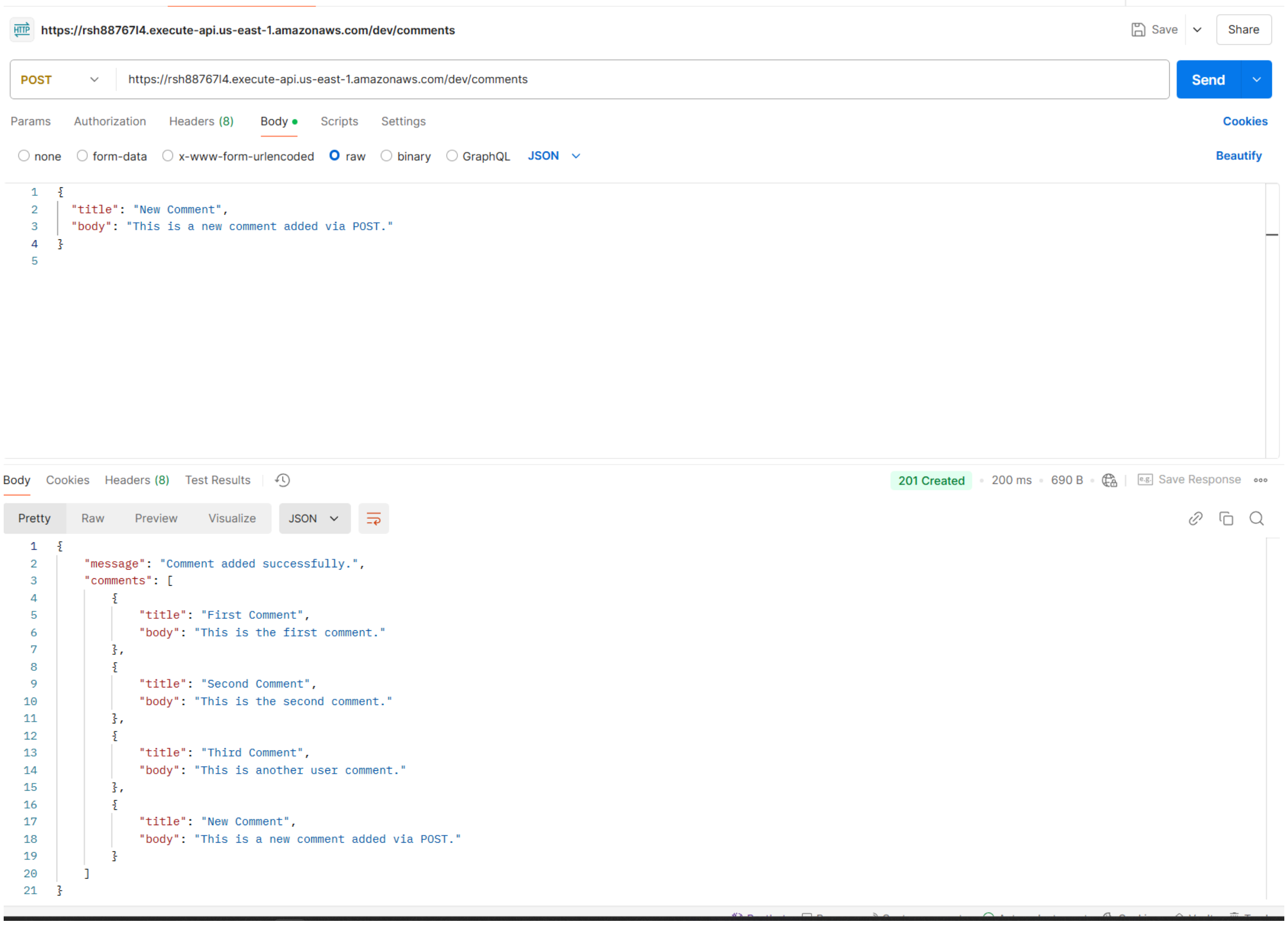
https://rsh88767l4.execute-api.us-east-1.amazonaws.com/dev/comments

POST https://rsh88767l4.execute-api.us-east-1.amazonaws.com/dev/comments

Params Authorization Headers (8) Body Scripts Settings

Headers — Hide auto-generated headers

	Key	Value	Description
✓	Content-Type	application/json	
✓	Content-Length	<calculated when request is sent>	
✓	Host	<calculated when request is sent>	
✓	User-Agent	PostmanRuntime/7.42.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Authorization	Bearer eyJraWQiOiJkNGhWNHdieEk4VjVKYlNOV0hCemRwcEtFOElCV...	
	Key	Value	Description



https://rsh88767l4.execute-api.us-east-1.amazonaws.com/dev/comments

POST https://rsh88767l4.execute-api.us-east-1.amazonaws.com/dev/comments

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
  "title": "New Comment",
  "body": "This is a new comment added via POST."
}
```

Body Cookies Headers (8) Test Results — 201 Created · 200 ms · 690 B

Pretty Raw Preview Visualize JSON

```
{
    "message": "Comment added successfully.",
    "comments": [
        {
            "title": "First Comment",
            "body": "This is the first comment."
        },
        {
            "title": "Second Comment",
            "body": "This is the second comment."
        },
        {
            "title": "Third Comment",
            "body": "This is another user comment."
        },
        {
            "title": "New Comment",
            "body": "This is a new comment added via POST."
        }
    ]
}
```

Task 6

This attack wouldn't work because JWT tokens are cryptographically signed.If an attacker were to change the expiration time or other contents of the JWT, the signature of the token would become invalid. Without the secret key, the attacker cannot generate a valid signature. Similarly, if asymmetric encryption is used, the attacker cannot recreate the valid signature without access to the private key. When a modified JWT with an altered payload (like an extended expiration time) is sent to the server, the server verifies the token's signature. Since the attacker does not have the secret or private key, the modified token will fail verification, and the server will reject it. This makes it extremely difficult for an attacker to tamper with the contents of the token in any meaningful way.

This is why such an attack, where a token is modified to change its expiration time or user type, would not succeed.