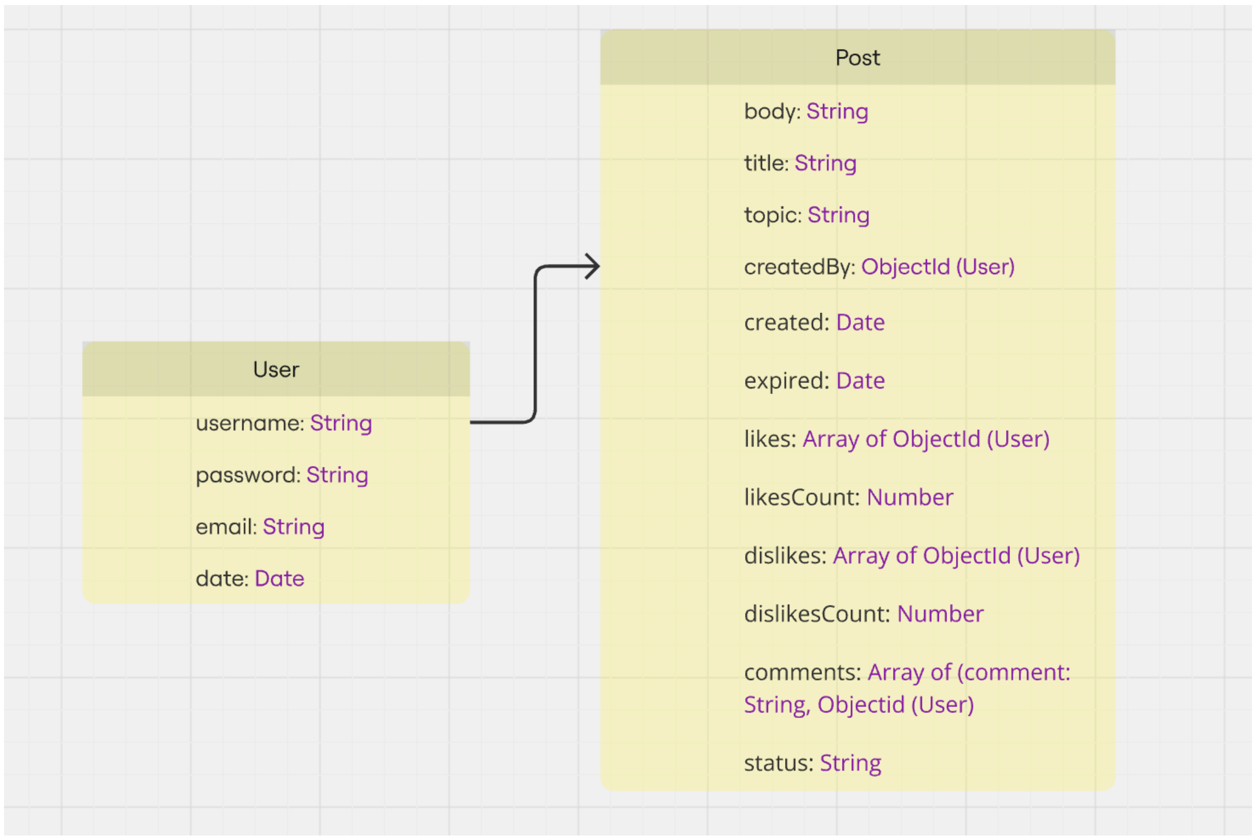
Piazza Replica

Implemented by: Rawan Reda, cloud computing course

Database Model:



createdBy is automatically set when a user creates a post.
created has a default to now
expired has a default a year from now
likesCount and **dislikesCount** have a default of 0
topic is an enum and has to be any of Heath, Sports, Politics, Tech

status is a virtual field that is calculated everytime a post schema is used to create a reponse. If the expired date is before the current date, status is set to *Expired*, otheriwse it is set to *Live*.

Setup description

There are three main folders: models, routes, validations

- models folder has two main schemas: User.js, Post.js
- routes: auth.js and posts.js
- validations: one validation file (register, login, post)

Dependencies installed:

```
  "dependencies": {
    "bcryptjs": "^2.4.3", // for password encryption
    "body-parser": "^1.20.3", // to send request body and parse it
    "dotenv": "^16.4.5", // to create .env file where secrets are stored
    "express": "^4.21.1",
    "joi": "^17.13.3", // for validation of request body
    "jsonwebtoken": "^9.0.2", // for generating auth-token
    "mongoose": "^8.8.1", // for mongodb connection
    "nodemon": "^3.1.7" // to run the app
  }
```

There are two main routes:

/api/user (auth.js file) and ***/api/posts*** (posts.js file)

Requests handled by posts route:

NOTE: all these requests require an authentication token:

GET “/”	topic is used as a query	get all posts if no topic is specified, otherwise get all posts that belong to a topic and none of the posts are expired.
GET “/sort”	topic is passed in the body	gets all ***Live*** posts by topic and sort them in the order of likes and dislikes sum (using a mongodb aggregate a new field “likesDislikesCount” is calculate and returned as part of the response.
GET “/expired”	topic is used as a query	gets all expired posts that belong to a certian topic
POST “/”	body, topic, created, expired are passed in the body	create a new post with default values for created, expired if they are not passed, and the rest of the fields are set accordingly.
PATCH “/:postId”	body has an interactionType and comment (if the the interaction type is COMMENT”)	valid interactionTypes: ● LIKE (this adds user id to the likes list and update the likesCount if it’s the first time that the user likes the point, however, if the user already liked the post and is making another like request, the user is removed from the likes list and likesCount is decremented) ● DISLIKE (same logic applied as Likes except that not it is for dislikes. ● COMMENT (when a user likes a comment, the comment as well a user Id as a reference are added to the comments array) the response returned includes username, interactionvalue, remaining days left to expire, and comment (if any)

Requests handled by users route:

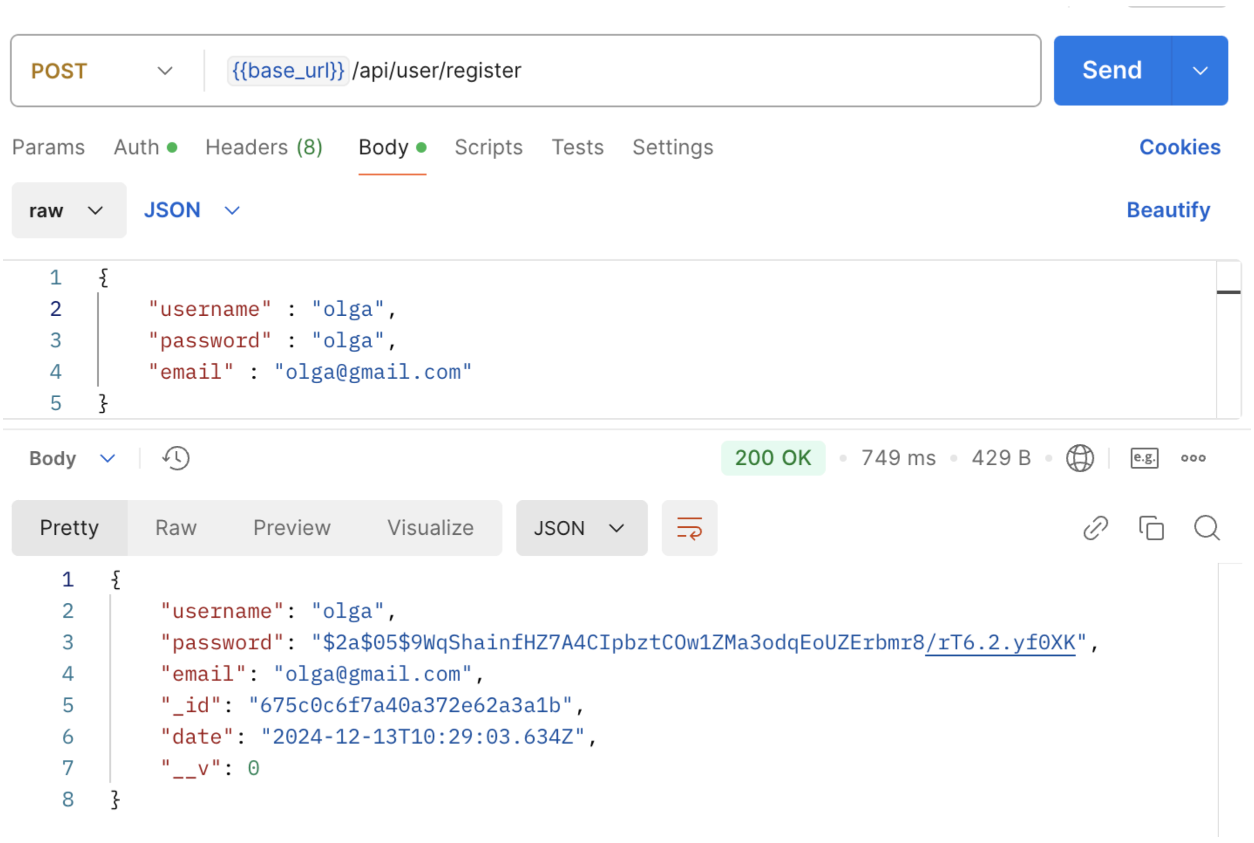
POST “/register”	body containes username, password and email	a new user is registered. the same email can not be used by more than one user
POST “/login”	body contains email and password	an auth-token is generated and returned in the response

Validations applied to requests:

- register: username, email, password are required and size of each entry is set a max and min (with the largest max and min being for the password (3-1024)), also email has to be in the right form.
- login: email and password are required
- post creation: body and topic are requiered, created and expired should be in the form of dates.

Testcases:

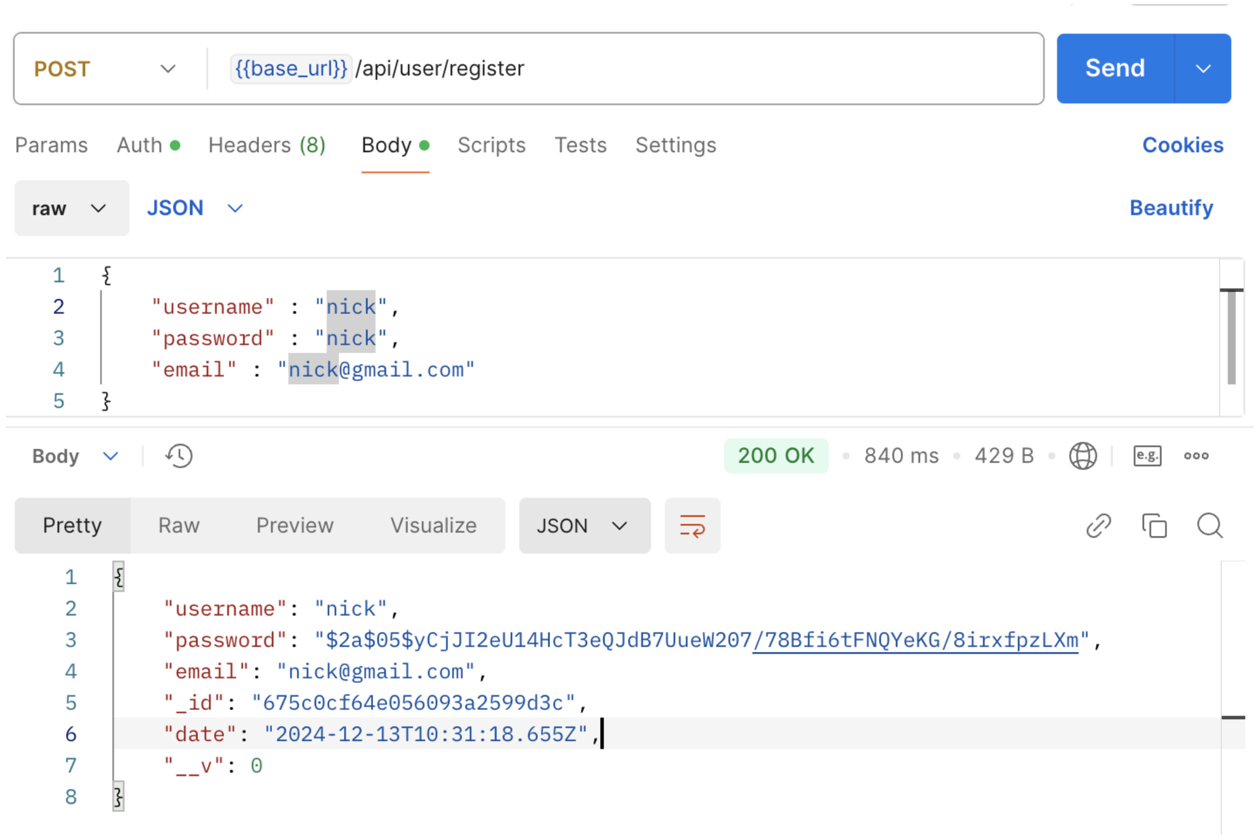
1. Olga, Nick, Mary, and Nestor register and are ready to access the **Piazza** API.



POST {{base_url}} /api/user/register

```
{
    "username" : "olga",
    "password" : "olga",
    "email" : "olga@gmail.com"
}
```

```
{
    "username": "olga",
    "password": "$2a$05$9WqShainfHZ7A4CIpbztCOw1ZMa3odqEoUZErbmr8/rT6.2.yf0XK",
    "email": "olga@gmail.com",
    "_id": "675c0c6f7a40a372e62a3a1b",
    "date": "2024-12-13T10:29:03.634Z",
    "__v": 0
}
```

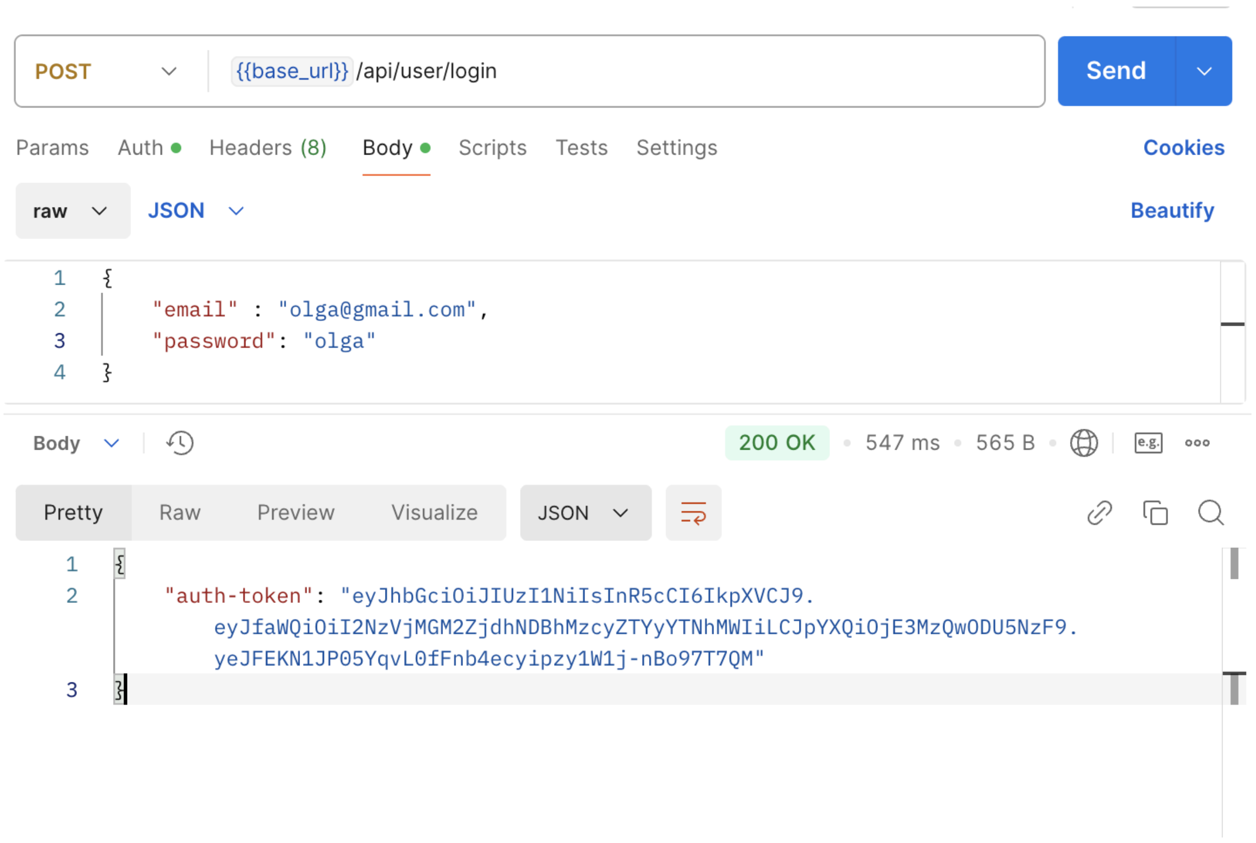


POST {{base_url}} /api/user/register

```
{
    "username" : "nick",
    "password" : "nick",
    "email" : "nick@gmail.com"
}
```

```
{
    "username": "nick",
    "password": "$2a$05$yCjJI2eU14HcT3eQJdB7UueW207/78Bfi6tFNQYeKG/8irxfpzLXm",
    "email": "nick@gmail.com",
    "_id": "675c0cf64e056093a2599d3c",
    "date": "2024-12-13T10:31:18.655Z",
    "__v": 0
}
```

2. Olga, Nick, Mary, and Nestor use the oAuth v2 authorisation service to register and get their tokens.

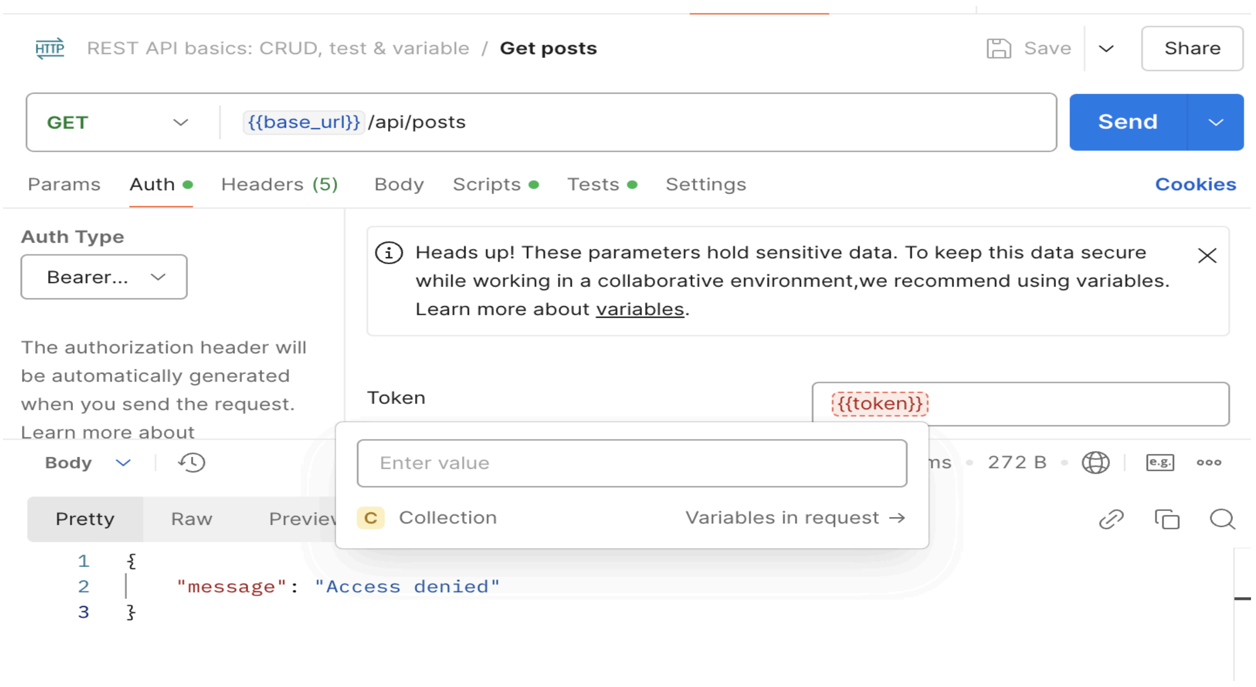


POST {{base_url}} /api/user/login

```
{
    "email" : "olga@gmail.com",
    "password": "olga"
}
```

```
{
    "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJfaWQiOiI2NzVjMGM2ZjdhNDBhMzcyZTYyYTNhMWIiLCJpYXQiOjE3MzQwODU5NzF9.
        yeJFEKN1JP05YqvL0fFnb4ecyipzy1W1j-nBo97T7QM"
}
```

3. Olga makes a call to the API without using her token. This call should be unsuccessful as the user is unauthorised.


REST API basics: CRUD, test & variable / **Get posts**

GET {{base_url}} /api/posts

Auth Type: Bearer...

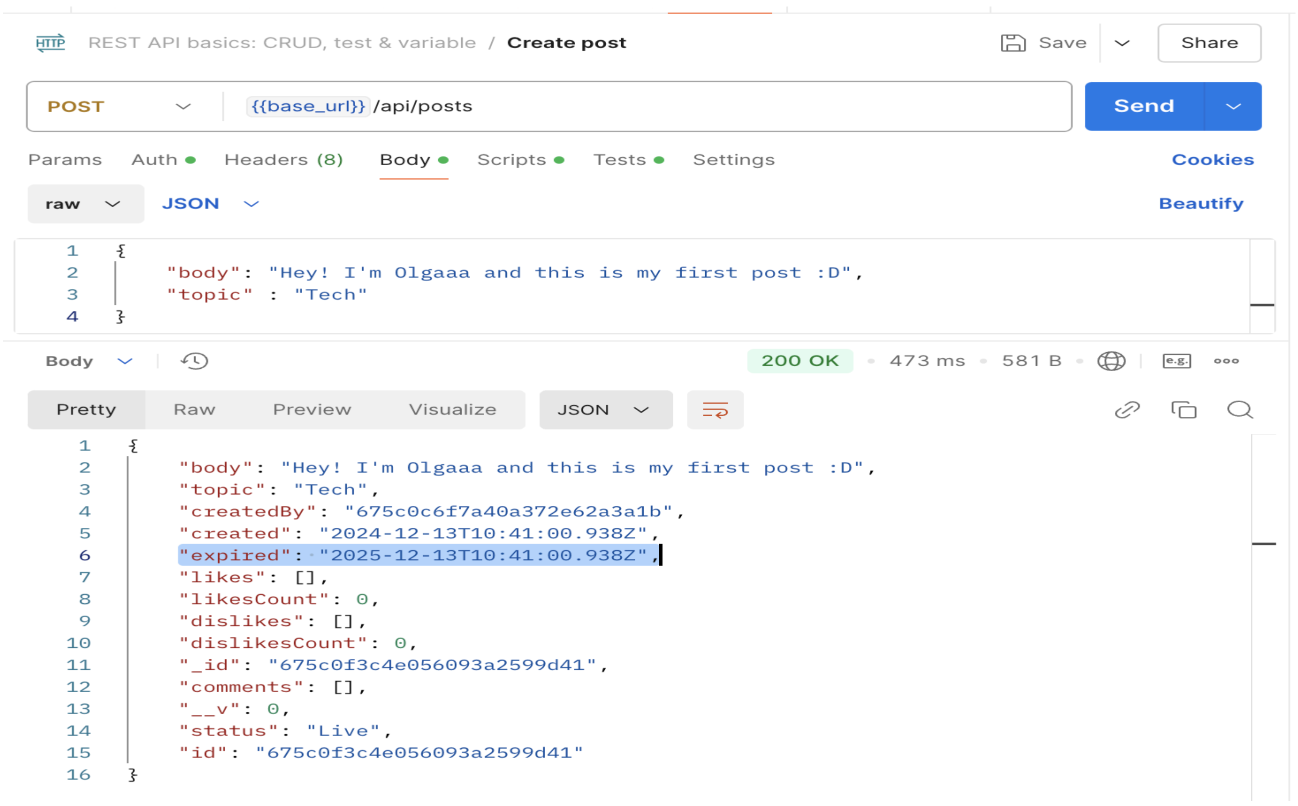
The authorization header will be automatically generated when you send the request. Learn more about

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token: {{token}}

```
{
    "message": "Access denied"
}
```

4. Olga posts a message in the Tech topic with an expiration time (e.g. 5 minutes) using her token. After the end of the expiration time, the message will not accept any further user interactions (likes, dislikes, or comments).


REST API basics: CRUD, test & variable / **Create post**

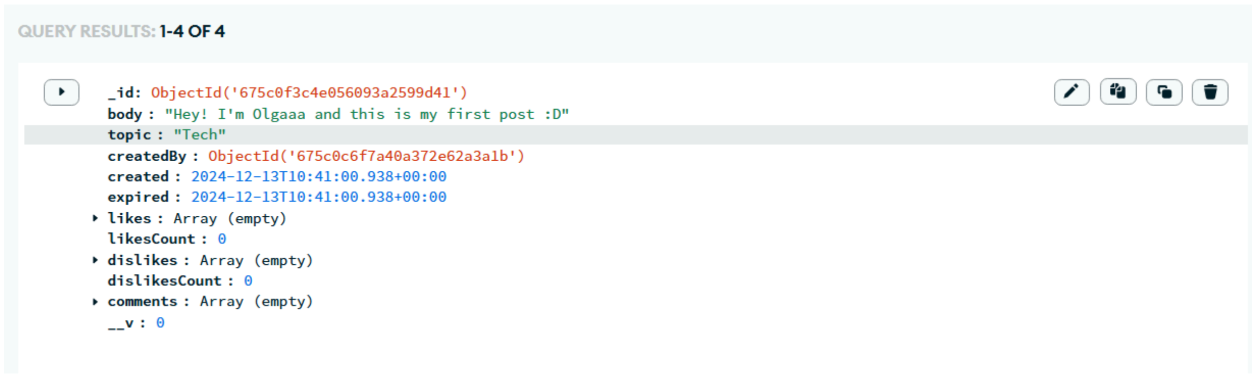
POST {{base_url}} /api/posts

```
{
    "body": "Hey! I'm Olgaaa and this is my first post :D",
    "topic" : "Tech"
}
```

200 OK · 473 ms · 581 B

```
{
    "body": "Hey! I'm Olgaaa and this is my first post :D",
    "topic": "Tech",
    "createdBy": "675c0c6f7a40a372e62a3a1b",
    "created": "2024-12-13T10:41:00.938Z",
    "expired": "2025-12-13T10:41:00.938Z",
    "likes": [],
    "likesCount": 0,
    "dislikes": [],
    "dislikesCount": 0,
    "_id": "675c0f3c4e056093a2599d41",
    "comments": [],
    "__v": 0,
    "status": "Live",
    "id": "675c0f3c4e056093a2599d41"
}
```

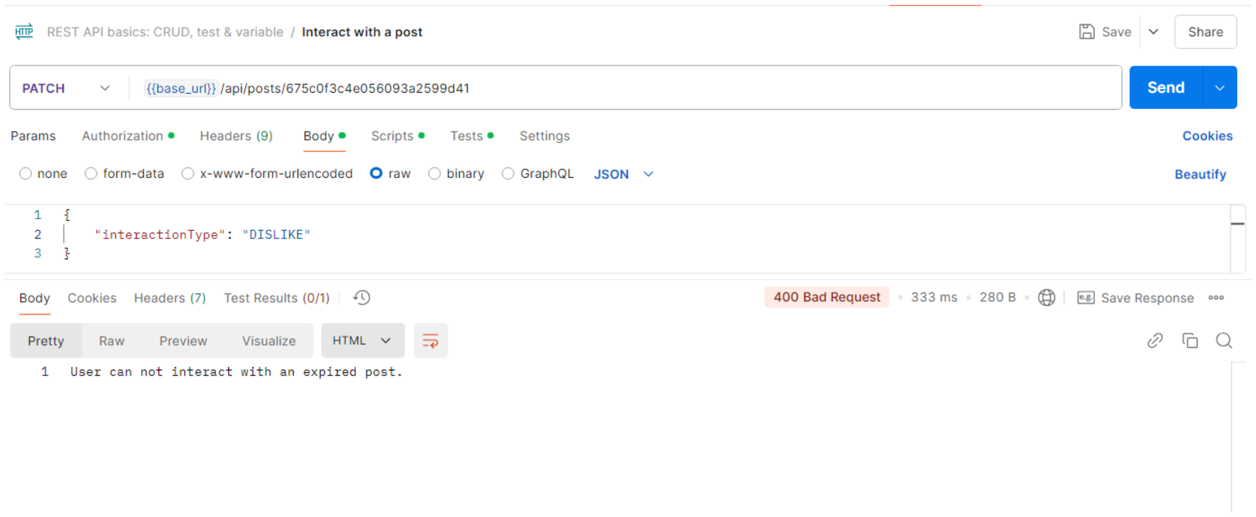
I have updated olga's post expiration date to be able to show that no user can interact afterwards:



QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId('675c0f3c4e056093a2599d41')
body : "Hey! I'm Olgaaa and this is my first post :D"
topic : "Tech"
createdBy : ObjectId('675c0c6f7a40a372e62a3a1b')
created : 2024-12-13T10:41:00.938+00:00
expired : 2024-12-13T10:41:00.938+00:00
likes : Array (empty)
likesCount : 0
dislikes : Array (empty)
dislikesCount : 0
comments : Array (empty)
__v : 0
```

expiration in the above image has been modified



REST API basics: CRUD, test & variable / Interact with a post

PATCH {{base_url}}/api/posts/675c0f3c4e056093a2599d41

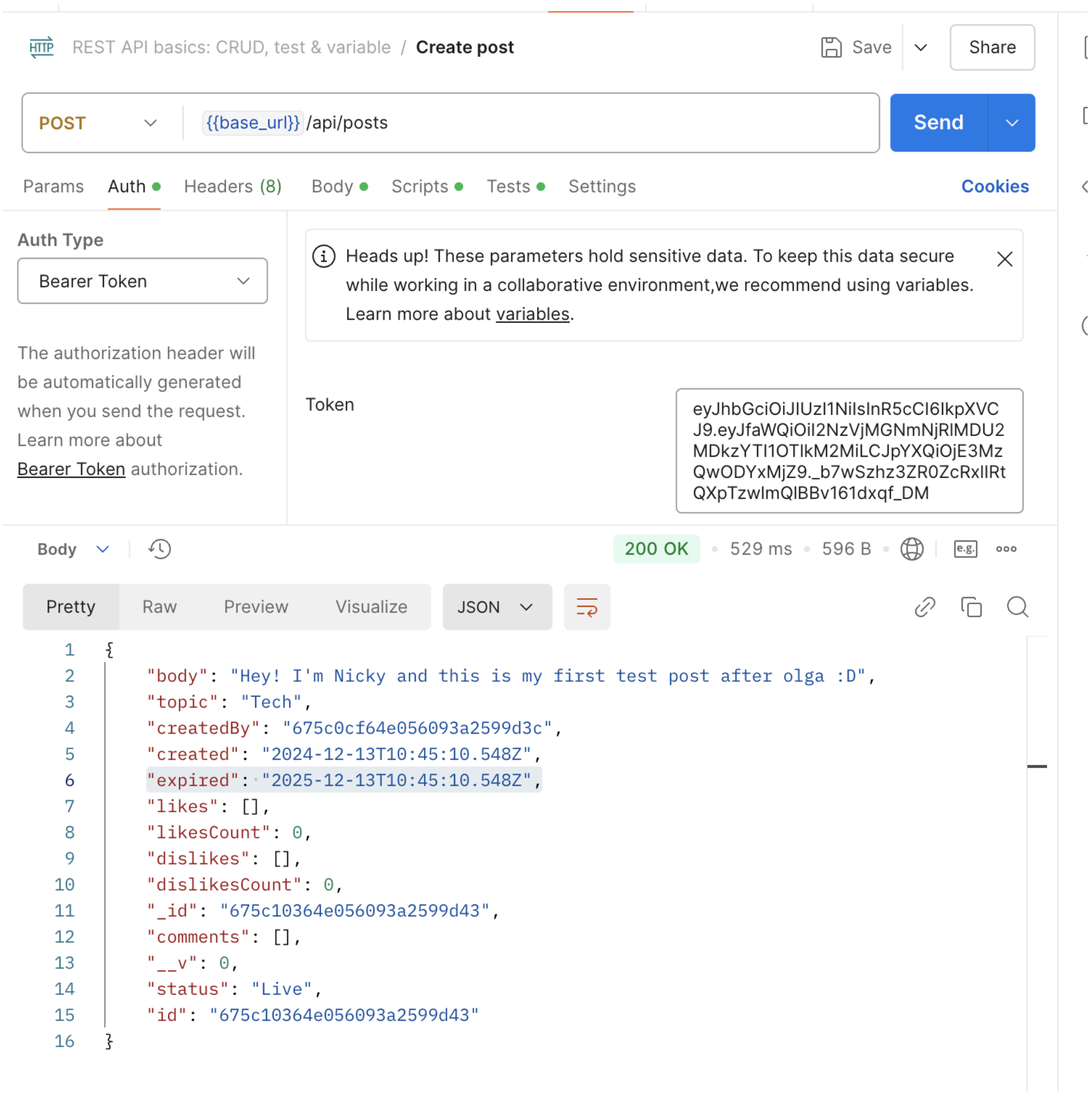
```
{
    "interactionType": "DISLIKE"
}
```

400 Bad Request · 333 ms · 280 B

```
User can not interact with an expired post.
```

same response appears for whoever tries to interact with the post above.

5. Nick posts a message in the Tech topic with an expiration time using his token.


REST API basics: CRUD, test & variable / **Create post**

POST {{base_url}} /api/posts

Params · Auth · Headers (8) · Body · Scripts · Tests · Settings · Cookies

Auth Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

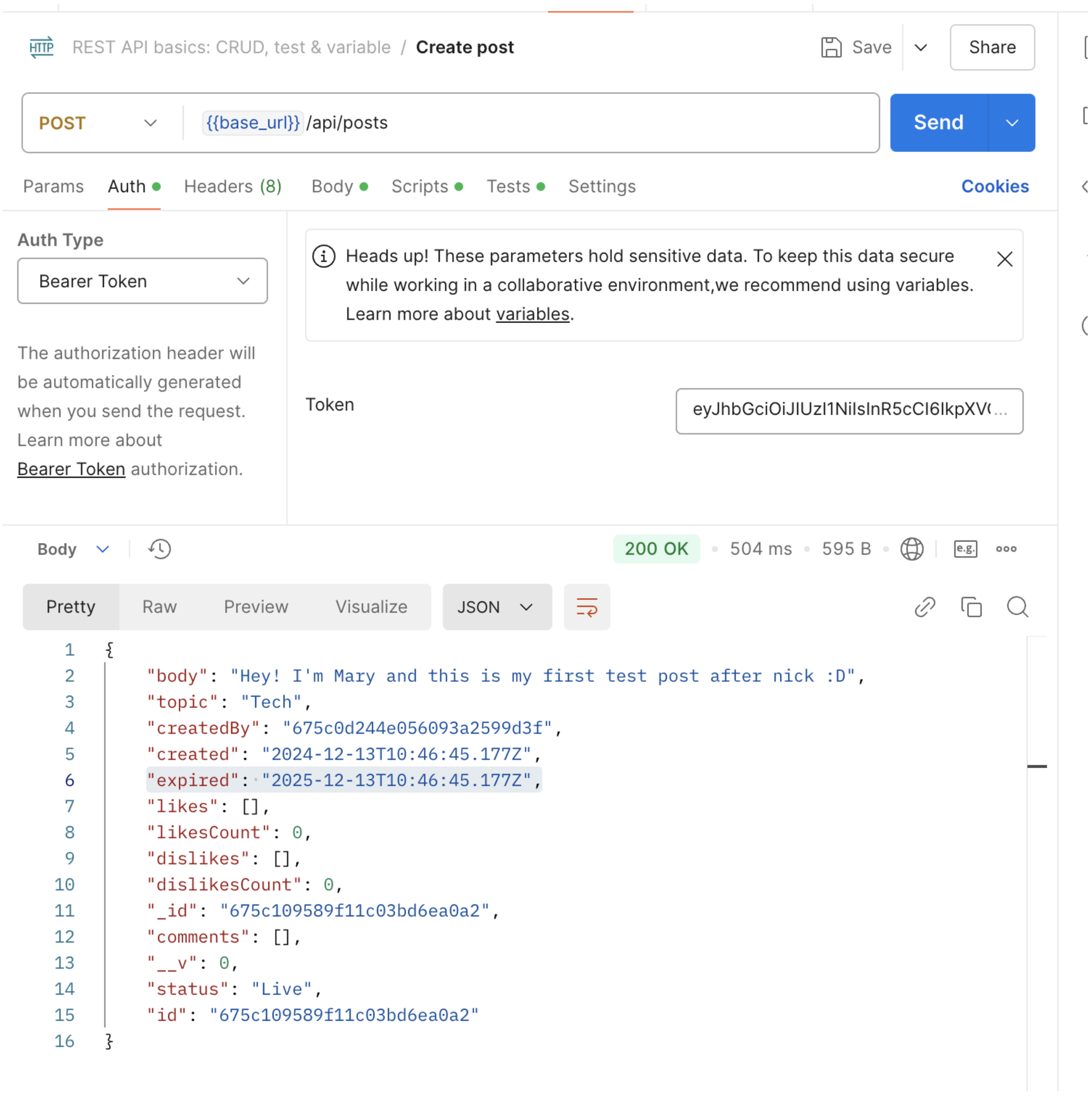
Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzVjMGNmNjRlMDU2MDkzYTI1OTlkM2MiLCJpYXQiOjE3MzQwODYxMjZ9._b7wSzhz3ZR0ZcRxlIRtQXpTzwlmQlBBv161dxqf_DM

Body · 200 OK · 529 ms · 596 B

Pretty · Raw · Preview · Visualize · JSON

```
{
    "body": "Hey! I'm Nicky and this is my first test post after olga :D",
    "topic": "Tech",
    "createdBy": "675c0cf64e056093a2599d3c",
    "created": "2024-12-13T10:45:10.548Z",
    "expired": "2025-12-13T10:45:10.548Z",
    "likes": [],
    "likesCount": 0,
    "dislikes": [],
    "dislikesCount": 0,
    "_id": "675c10364e056093a2599d43",
    "comments": [],
    "__v": 0,
    "status": "Live",
    "id": "675c10364e056093a2599d43"
}
```

6. Mary posts a message in the Tech topic with an expiration time using her token.


REST API basics: CRUD, test & variable / **Create post**

POST {{base_url}}/api/posts

Auth Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

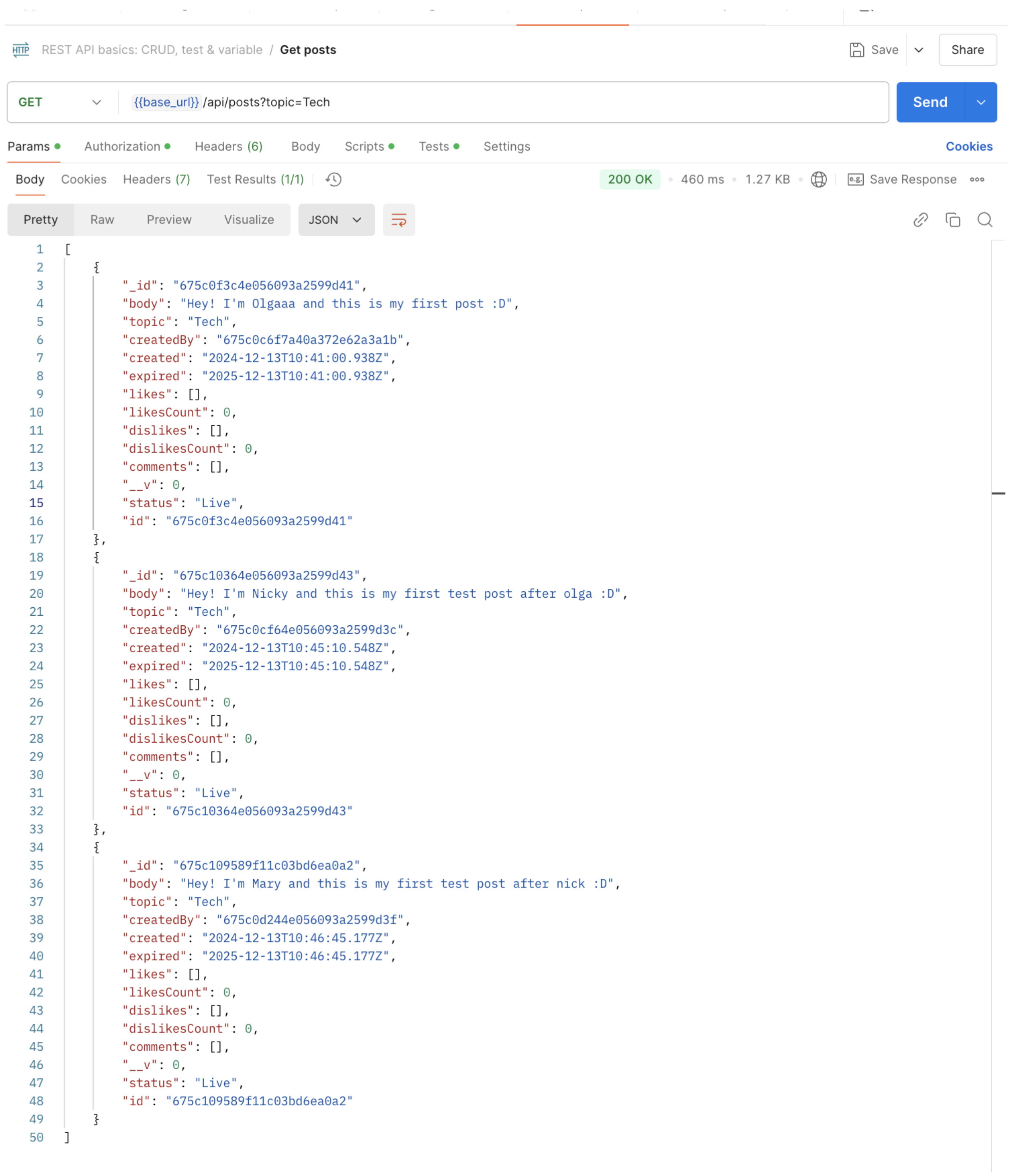
Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV(...

200 OK • 504 ms • 595 B

```
{
    "body": "Hey! I'm Mary and this is my first test post after nick :D",
    "topic": "Tech",
    "createdBy": "675c0d244e056093a2599d3f",
    "created": "2024-12-13T10:46:45.177Z",
    "expired": "2025-12-13T10:46:45.177Z",
    "likes": [],
    "likesCount": 0,
    "dislikes": [],
    "dislikesCount": 0,
    "_id": "675c109589f11c03bd6ea0a2",
    "comments": [],
    "__v": 0,
    "status": "Live",
    "id": "675c109589f11c03bd6ea0a2"
}
```

7. Nick and Olga browse all the available posts in the Tech topic; three posts should have zero likes, zero dislikes, and no comments.

Olga browses:

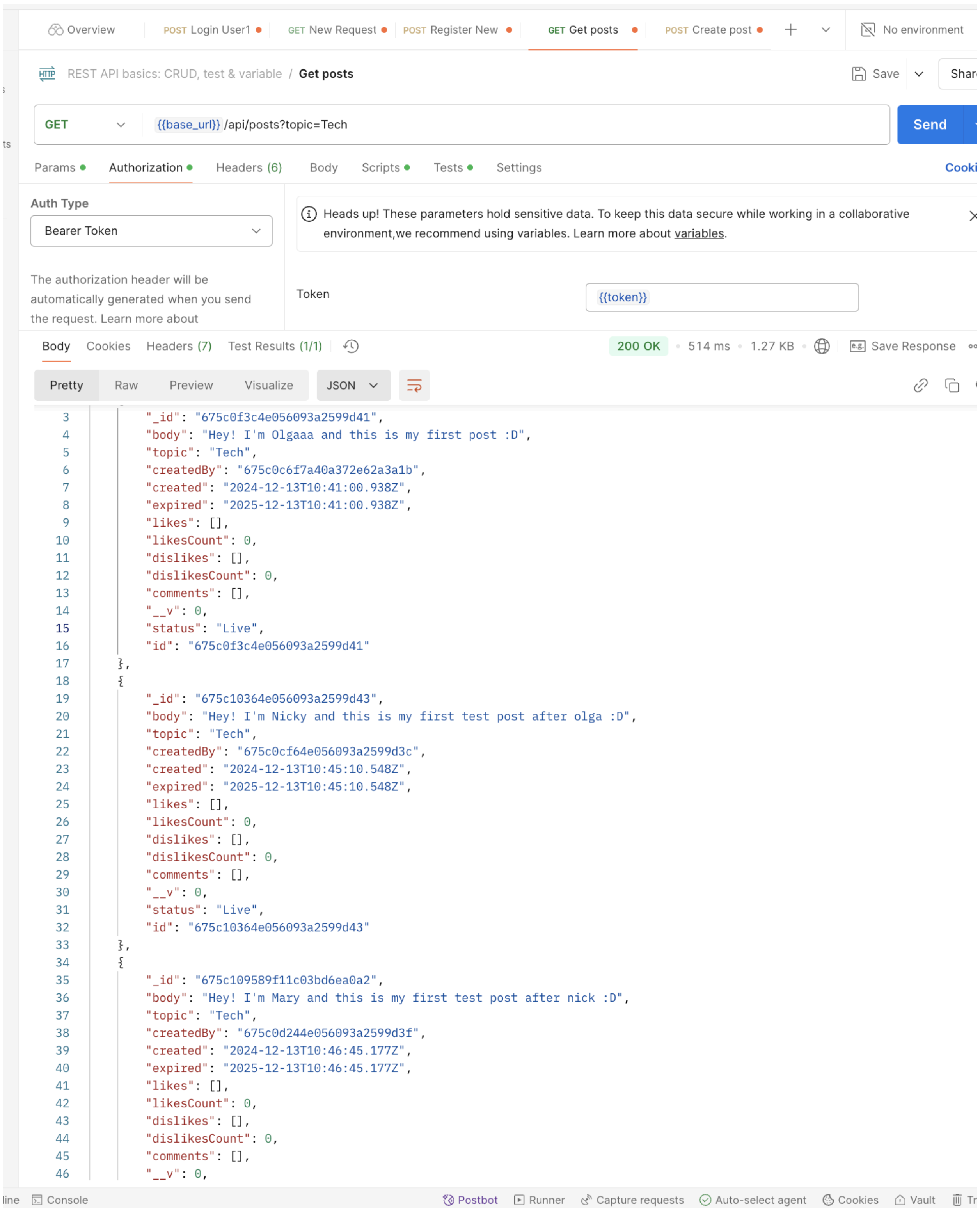


```
GET {{base_url}}/api/posts?topic=Tech
```

200 OK • 460 ms • 1.27 KB

```
[
    {
        "_id": "675c0f3c4e056093a2599d41",
        "body": "Hey! I'm Olgaaa and this is my first post :D",
        "topic": "Tech",
        "createdBy": "675c0c6f7a40a372e62a3a1b",
        "created": "2024-12-13T10:41:00.938Z",
        "expired": "2025-12-13T10:41:00.938Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c0f3c4e056093a2599d41"
    },
    {
        "_id": "675c10364e056093a2599d43",
        "body": "Hey! I'm Nicky and this is my first test post after olga :D",
        "topic": "Tech",
        "createdBy": "675c0cf64e056093a2599d3c",
        "created": "2024-12-13T10:45:10.548Z",
        "expired": "2025-12-13T10:45:10.548Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c10364e056093a2599d43"
    },
    {
        "_id": "675c109589f11c03bd6ea0a2",
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c109589f11c03bd6ea0a2"
    }
]
```

Nick browses:



Overview | POST Login User1 | GET New Request | POST Register New | GET Get posts | POST Create post | No environment

REST API basics: CRUD, test & variable / **Get posts**

GET {{base_url}} /api/posts?topic=Tech

Params | Authorization | Headers (6) | Body | Scripts | Tests | Settings

Auth Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token: {{token}}

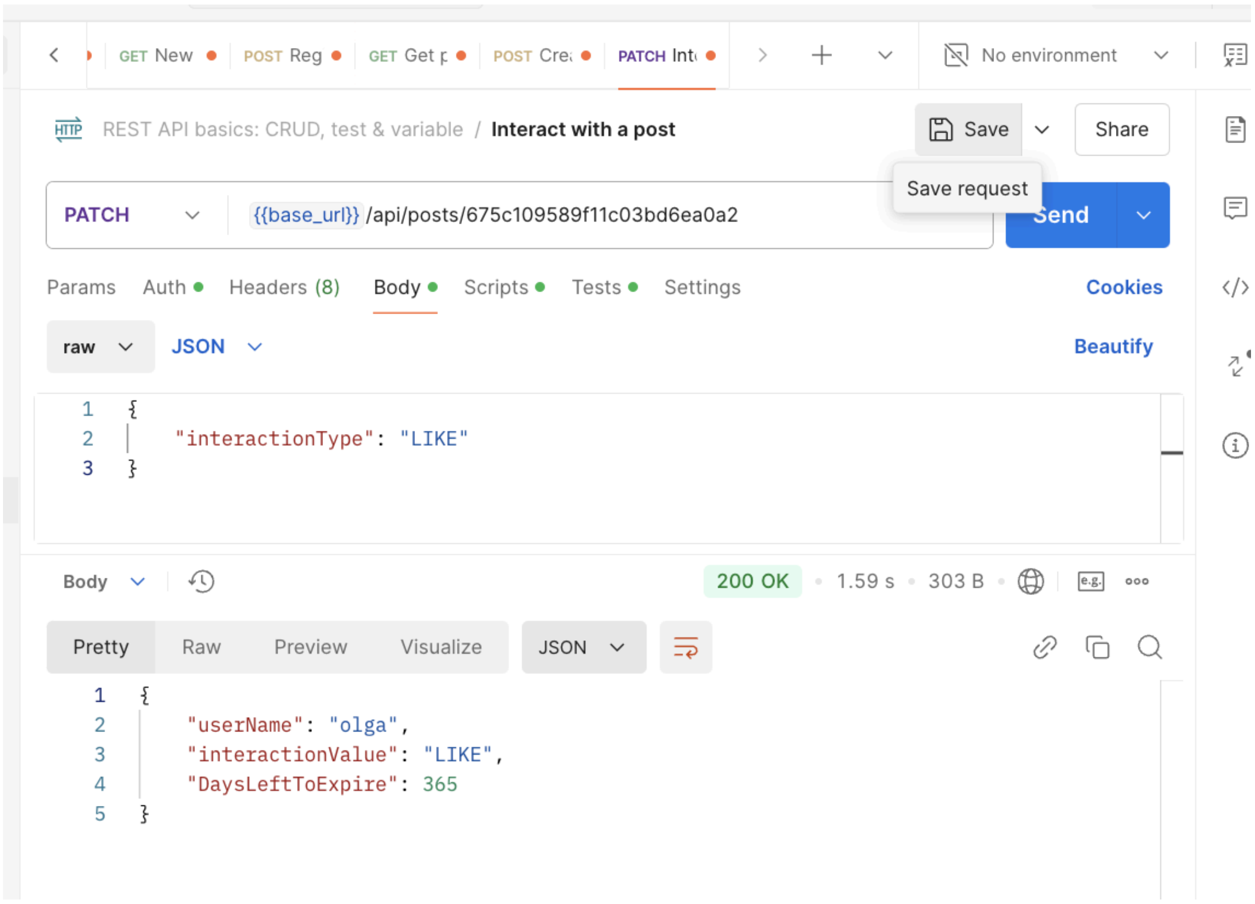
Body | Cookies | Headers (7) | Test Results (1/1) — 200 OK · 514 ms · 1.27 KB

Pretty | Raw | Preview | Visualize | JSON

```
        "_id": "675c0f3c4e056093a2599d41",
        "body": "Hey! I'm Olgaaa and this is my first post :D",
        "topic": "Tech",
        "createdBy": "675c0c6f7a40a372e62a3a1b",
        "created": "2024-12-13T10:41:00.938Z",
        "expired": "2025-12-13T10:41:00.938Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c0f3c4e056093a2599d41"
    },
    {
        "_id": "675c10364e056093a2599d43",
        "body": "Hey! I'm Nicky and this is my first test post after olga :D",
        "topic": "Tech",
        "createdBy": "675c0cf64e056093a2599d3c",
        "created": "2024-12-13T10:45:10.548Z",
        "expired": "2025-12-13T10:45:10.548Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c10364e056093a2599d43"
    },
    {
        "_id": "675c109589f11c03bd6ea0a2",
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
```

Console | Postbot | Runner | Capture requests | Auto-select agent | Cookies | Vault

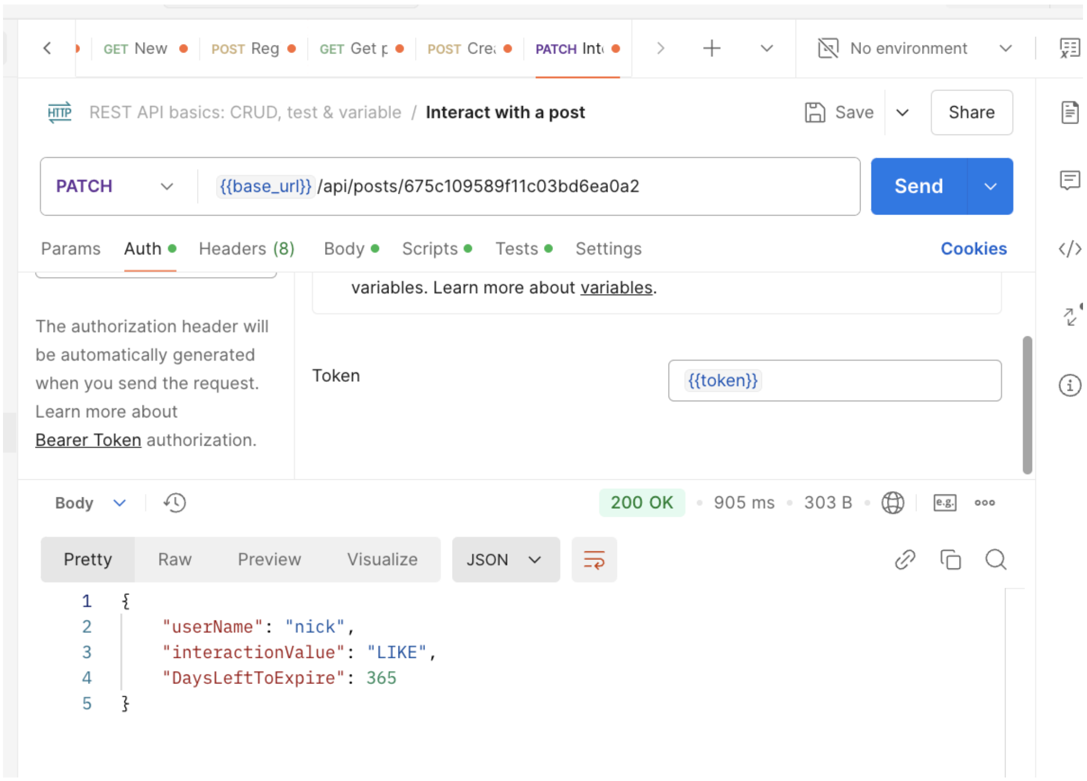
8. Nick and Olga “like” Mary’s post on the Tech topic.



PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

```
{
    "interactionType": "LIKE"
}
```

```
{
    "userName": "olga",
    "interactionValue": "LIKE",
    "DaysLeftToExpire": 365
}
```



PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

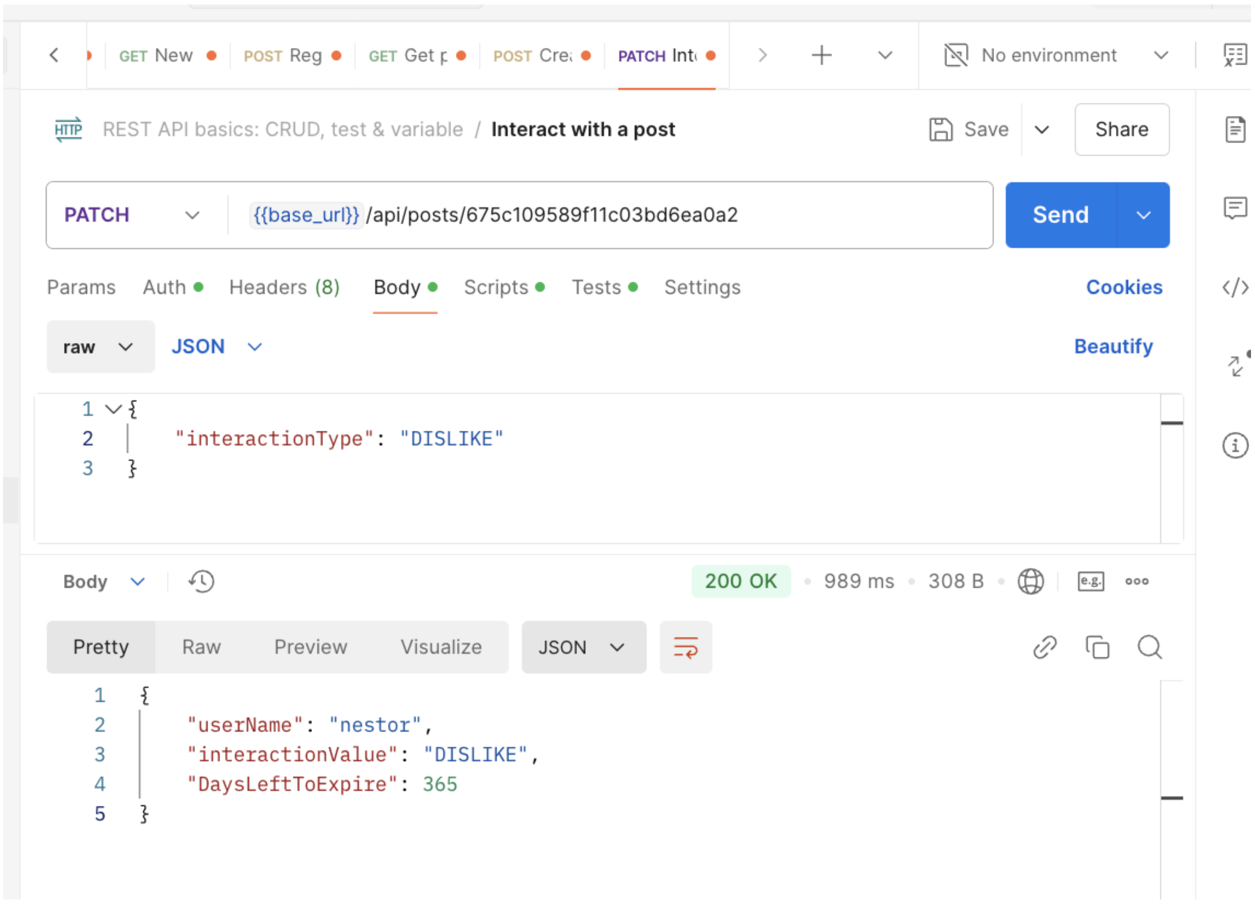
variables. Learn more about variables.

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Token {{token}}

```
{
    "userName": "nick",
    "interactionValue": "LIKE",
    "DaysLeftToExpire": 365
}
```

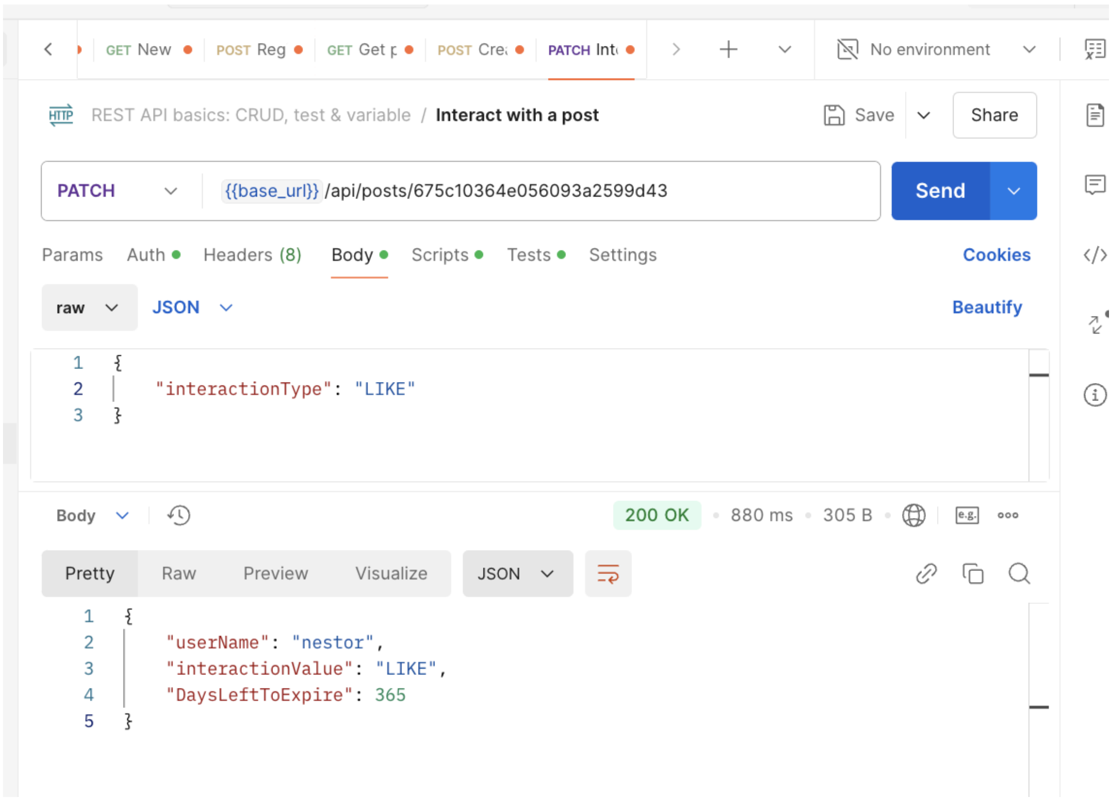
9. Nestor “likes” Nick’s post and “dislikes” Mary’s on the Tech topic.



PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

```json
{
    "interactionType": "DISLIKE"
}
```

```json
{
    "userName": "nestor",
    "interactionValue": "DISLIKE",
    "DaysLeftToExpire": 365
}
```

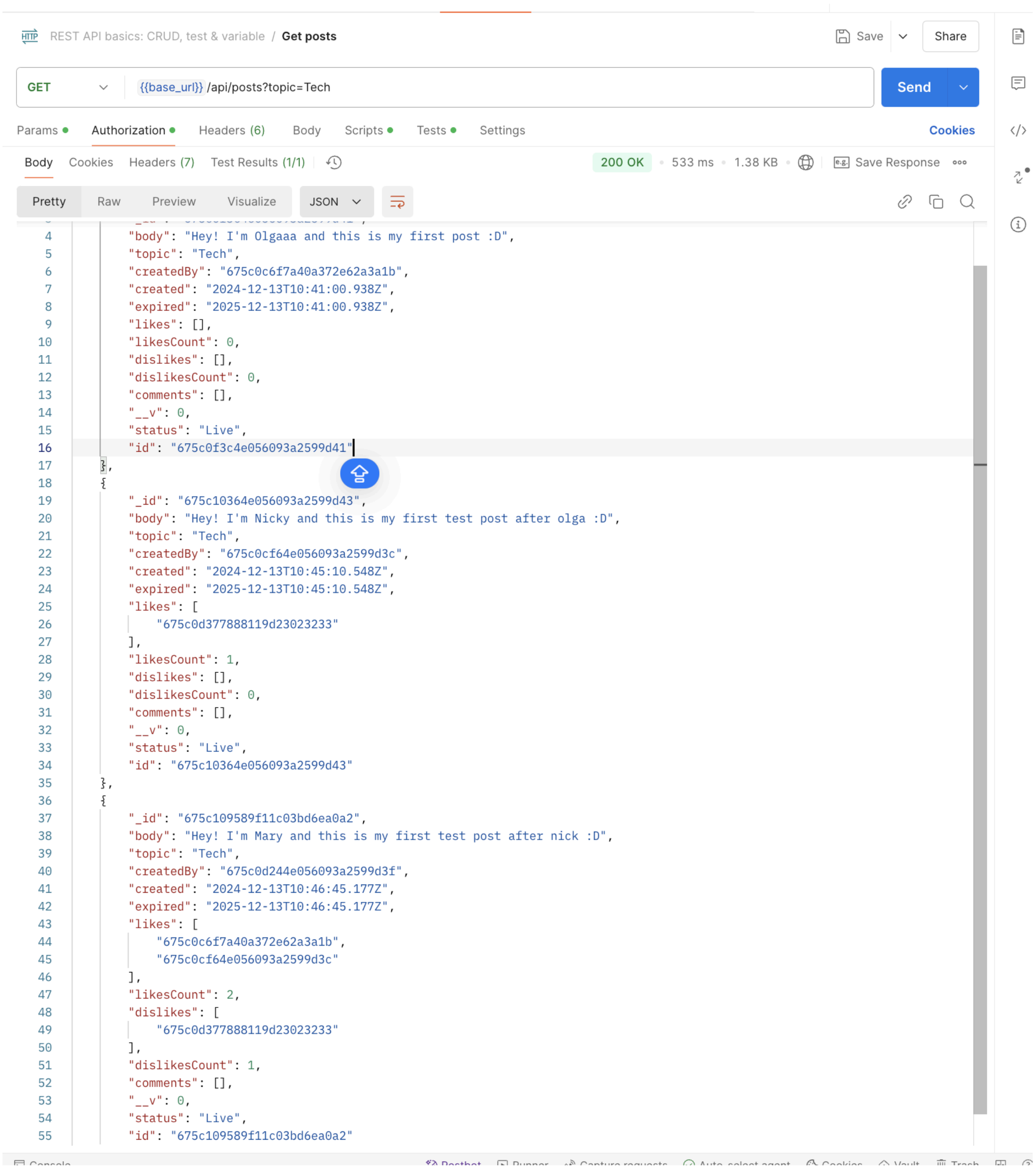


PATCH {{base_url}} /api/posts/675c10364e056093a2599d43

```json
{
    "interactionType": "LIKE"
}
```

```json
{
    "userName": "nestor",
    "interactionValue": "LIKE",
    "DaysLeftToExpire": 365
}
```

10. Nick browses all the available posts on the Tech topic; at this stage, he can see the number of likes and dislikes for each post (Mary has two likes and one dislike, and Nick has one like). No comments have been made yet.



REST API basics: CRUD, test & variable / **Get posts**

GET {{base_url}} /api/posts?topic=Tech

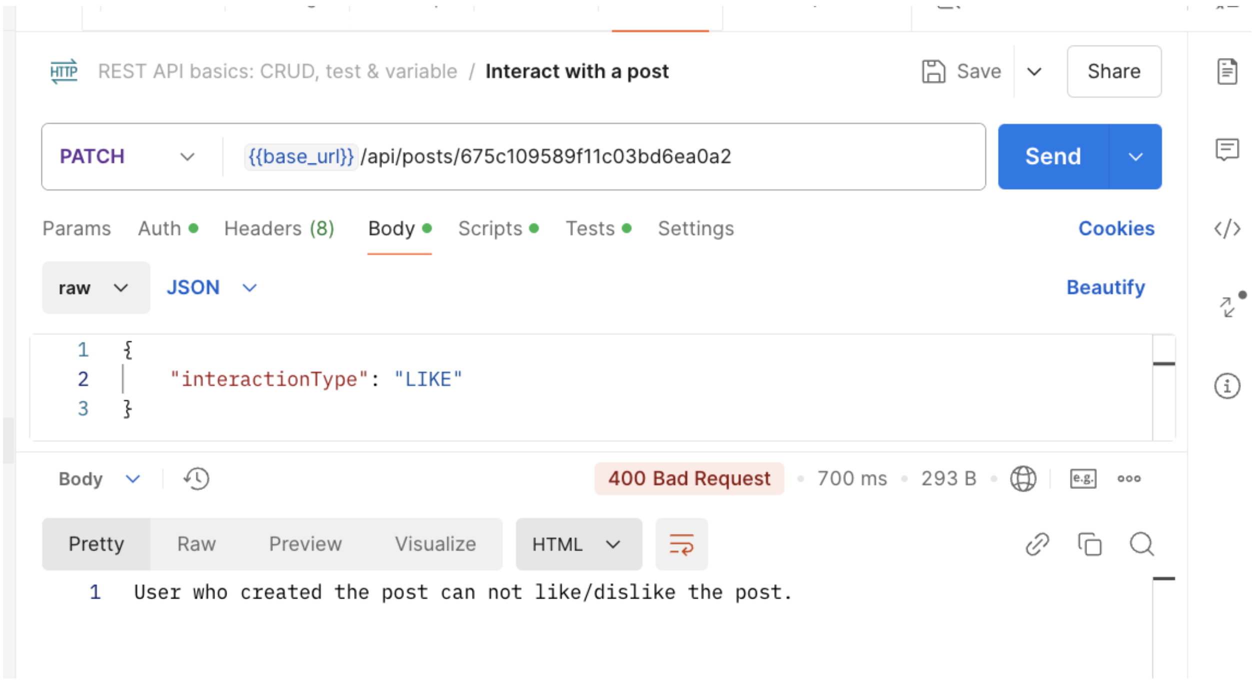
Params ● Authorization ● Headers (6) Body Scripts ● Tests ● Settings Cookies

Body Cookies Headers (7) Test Results (1/1) 200 OK 533 ms 1.38 KB Save Response

Pretty Raw Preview Visualize JSON

```
        "body": "Hey! I'm Olgaaa and this is my first post :D",
        "topic": "Tech",
        "createdBy": "675c0c6f7a40a372e62a3a1b",
        "created": "2024-12-13T10:41:00.938Z",
        "expired": "2025-12-13T10:41:00.938Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c0f3c4e056093a2599d41"
    },
    {
        "_id": "675c10364e056093a2599d43",
        "body": "Hey! I'm Nicky and this is my first test post after olga :D",
        "topic": "Tech",
        "createdBy": "675c0cf64e056093a2599d3c",
        "created": "2024-12-13T10:45:10.548Z",
        "expired": "2025-12-13T10:45:10.548Z",
        "likes": [
            "675c0d377888119d23023233"
        ],
        "likesCount": 1,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c10364e056093a2599d43"
    },
    {
        "_id": "675c109589f11c03bd6ea0a2",
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [
            "675c0c6f7a40a372e62a3a1b",
            "675c0cf64e056093a2599d3c"
        ],
        "likesCount": 2,
        "dislikes": [
            "675c0d377888119d23023233"
        ],
        "dislikesCount": 1,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c109589f11c03bd6ea0a2"
```

11. Mary likes her post on the Tech topic. This call should be unsuccessful; in **Piazza,** a post owner cannot like their messages.

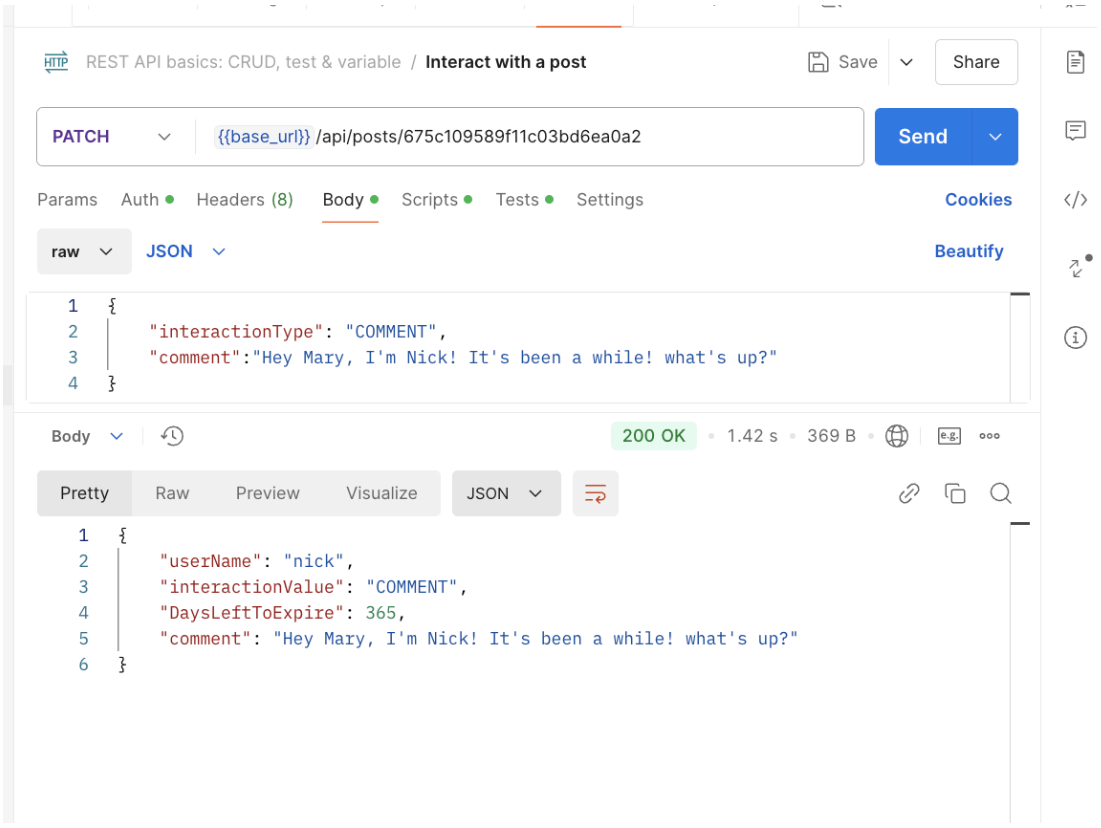


PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

```
{
    "interactionType": "LIKE"
}
```

```
User who created the post can not like/dislike the post.
```

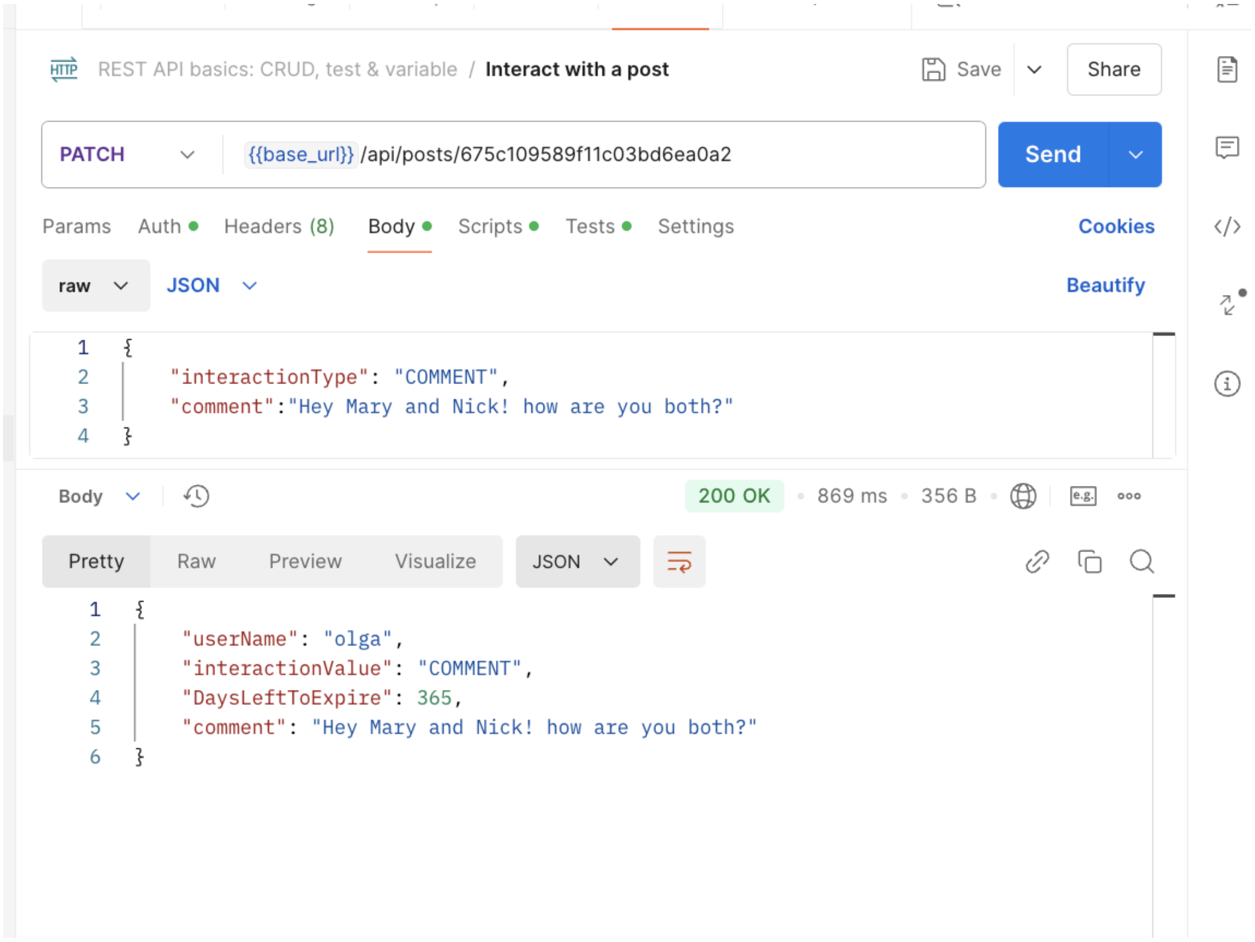
12. Nick and Olga comment on Mary's post on the Tech topic in a round-robin fashion (one after the other, adding at least two comments each).



PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

```
{
    "interactionType": "COMMENT",
    "comment":"Hey Mary, I'm Nick! It's been a while! what's up?"
}
```

```
{
    "userName": "nick",
    "interactionValue": "COMMENT",
    "DaysLeftToExpire": 365,
    "comment": "Hey Mary, I'm Nick! It's been a while! what's up?"
}
```

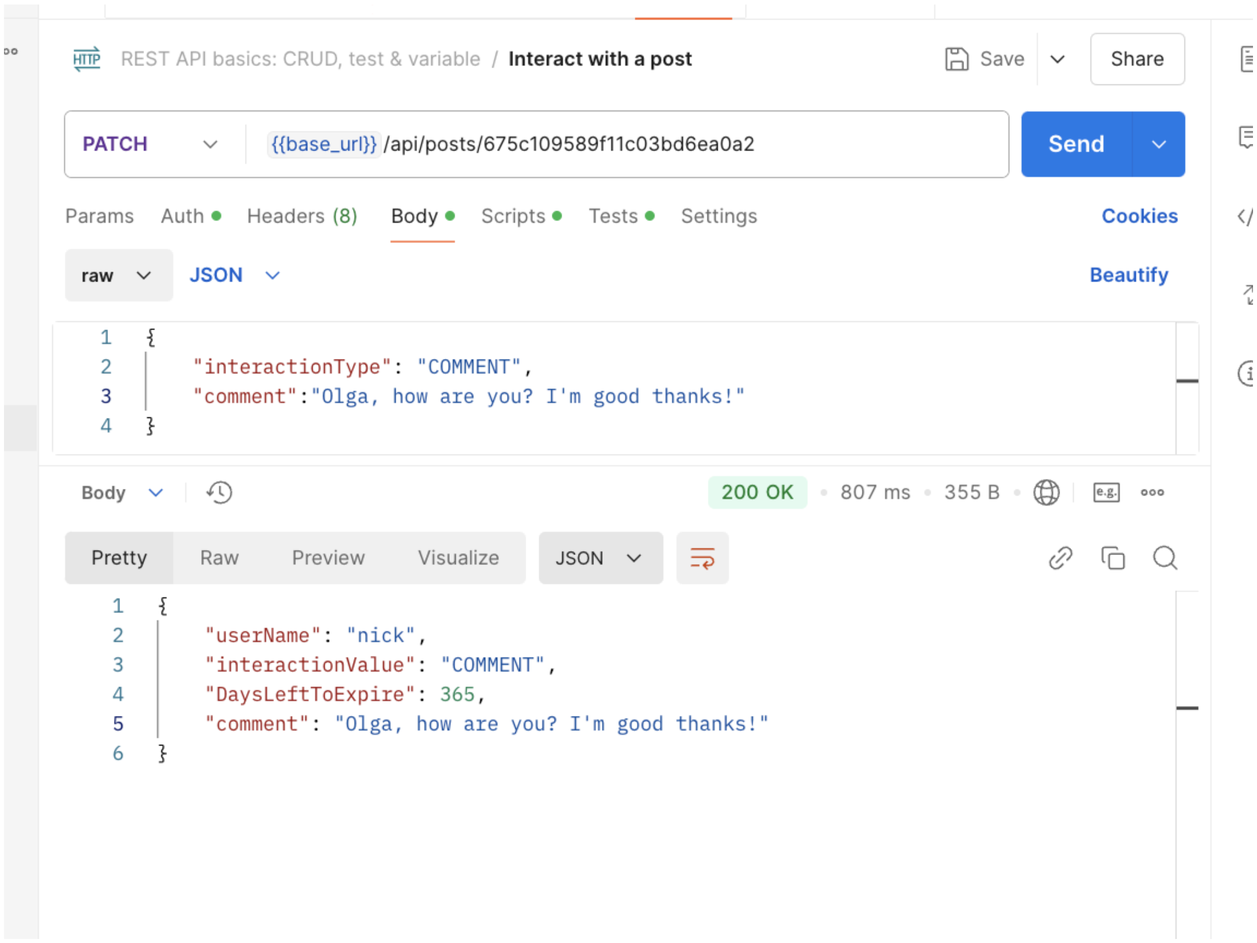

REST API basics: CRUD, test & variable / **Interact with a post**

PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

```
{
    "interactionType": "COMMENT",
    "comment":"Hey Mary and Nick! how are you both?"
}
```

200 OK • 869 ms • 356 B

```
{
    "userName": "olga",
    "interactionValue": "COMMENT",
    "DaysLeftToExpire": 365,
    "comment": "Hey Mary and Nick! how are you both?"
}
```

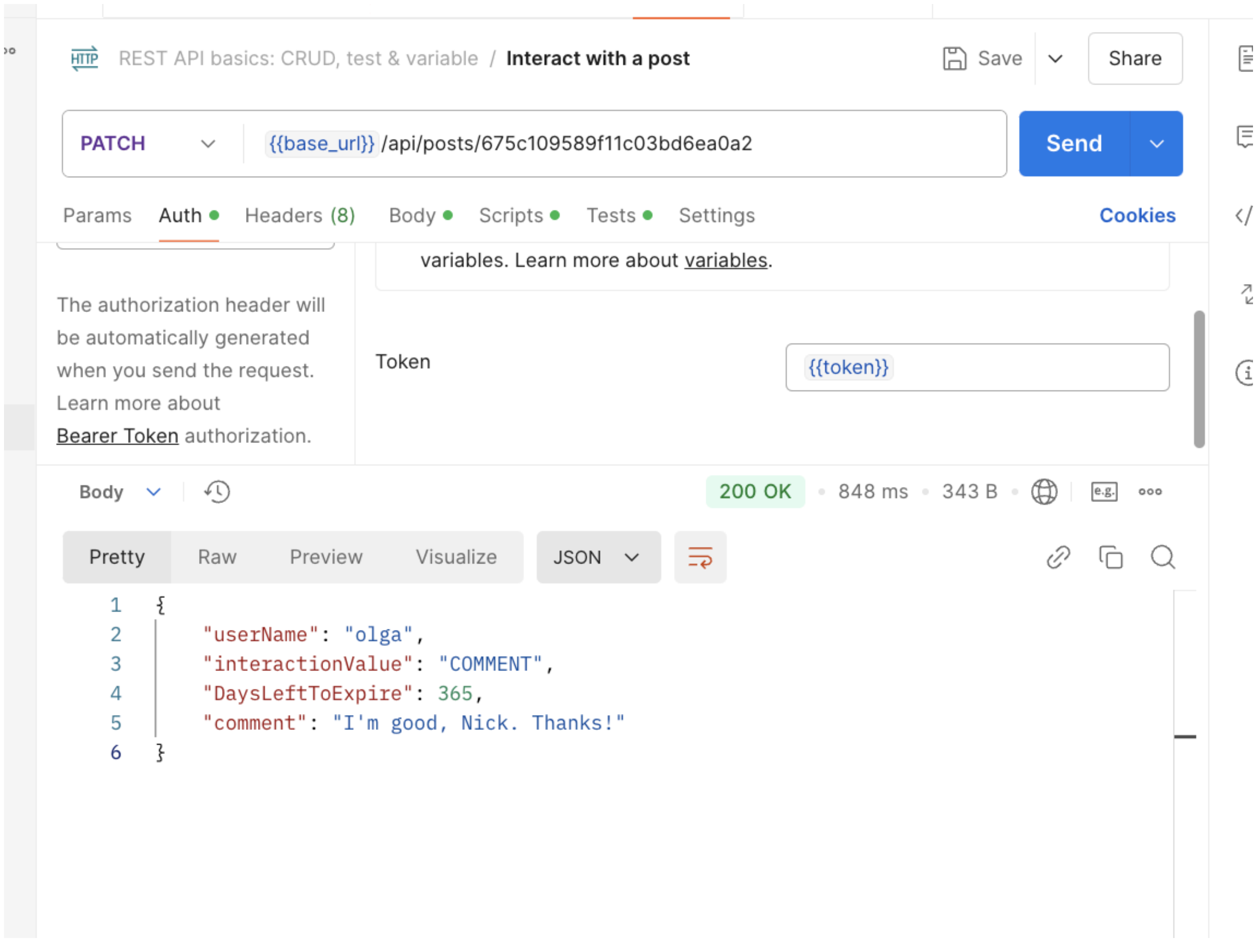

REST API basics: CRUD, test & variable / **Interact with a post**

PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

```
{
    "interactionType": "COMMENT",
    "comment":"Olga, how are you? I'm good thanks!"
}
```

200 OK • 807 ms • 355 B

```
{
    "userName": "nick",
    "interactionValue": "COMMENT",
    "DaysLeftToExpire": 365,
    "comment": "Olga, how are you? I'm good thanks!"
}
```


REST API basics: CRUD, test & variable / **Interact with a post**

PATCH {{base_url}} /api/posts/675c109589f11c03bd6ea0a2

variables. Learn more about variables.

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

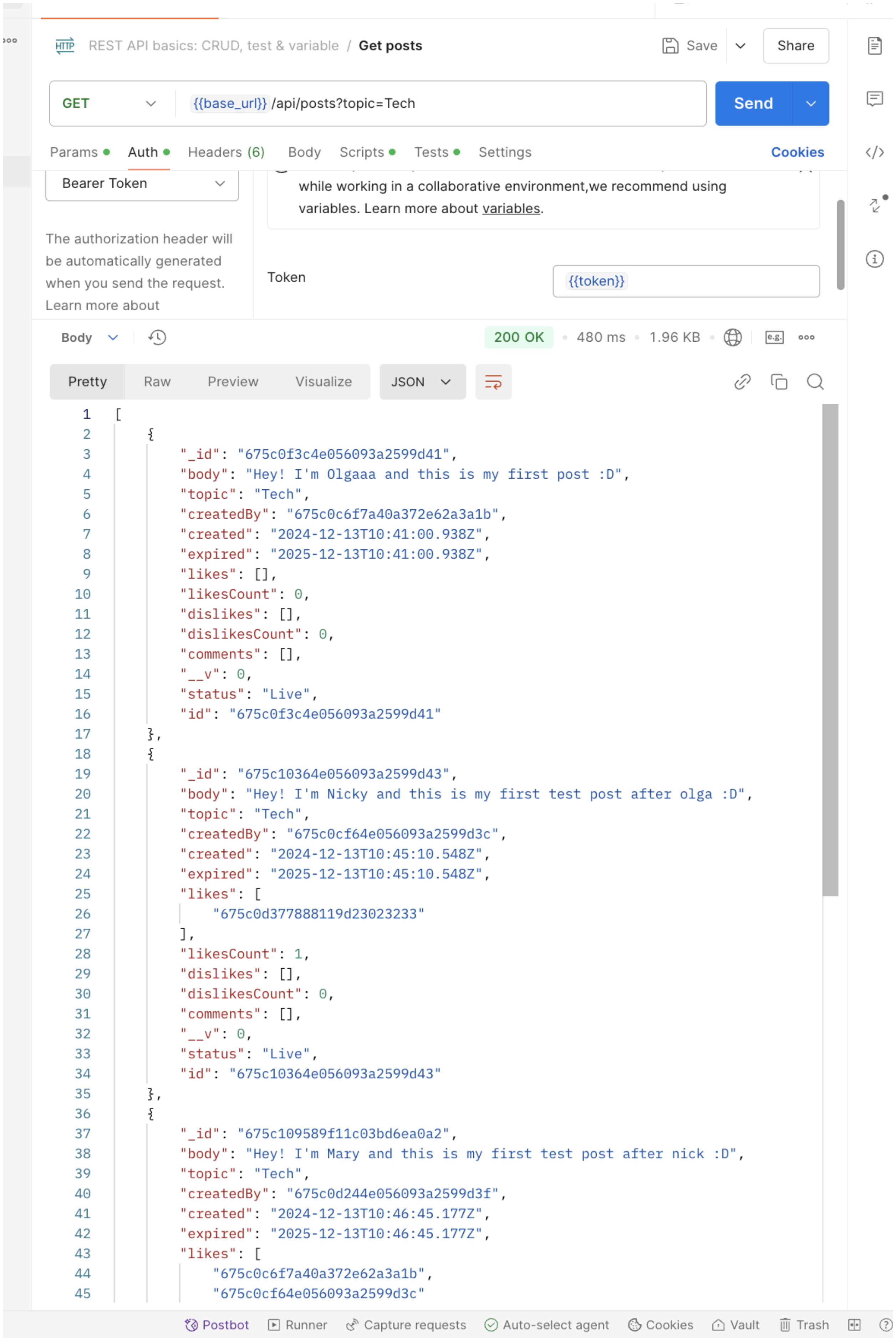
Token {{token}}

200 OK • 848 ms • 343 B

```json
{
    "userName": "olga",
    "interactionValue": "COMMENT",
    "DaysLeftToExpire": 365,
    "comment": "I'm good, Nick. Thanks!"
}
```

13. Nick browses all the available posts in the Tech topic; at this stage, he can see the number of likes and dislikes of each post and the comments made.

(check next page)


REST API basics: CRUD, test & variable / **Get posts**

GET {{base_url}}/api/posts?topic=Tech

Auth: Bearer Token

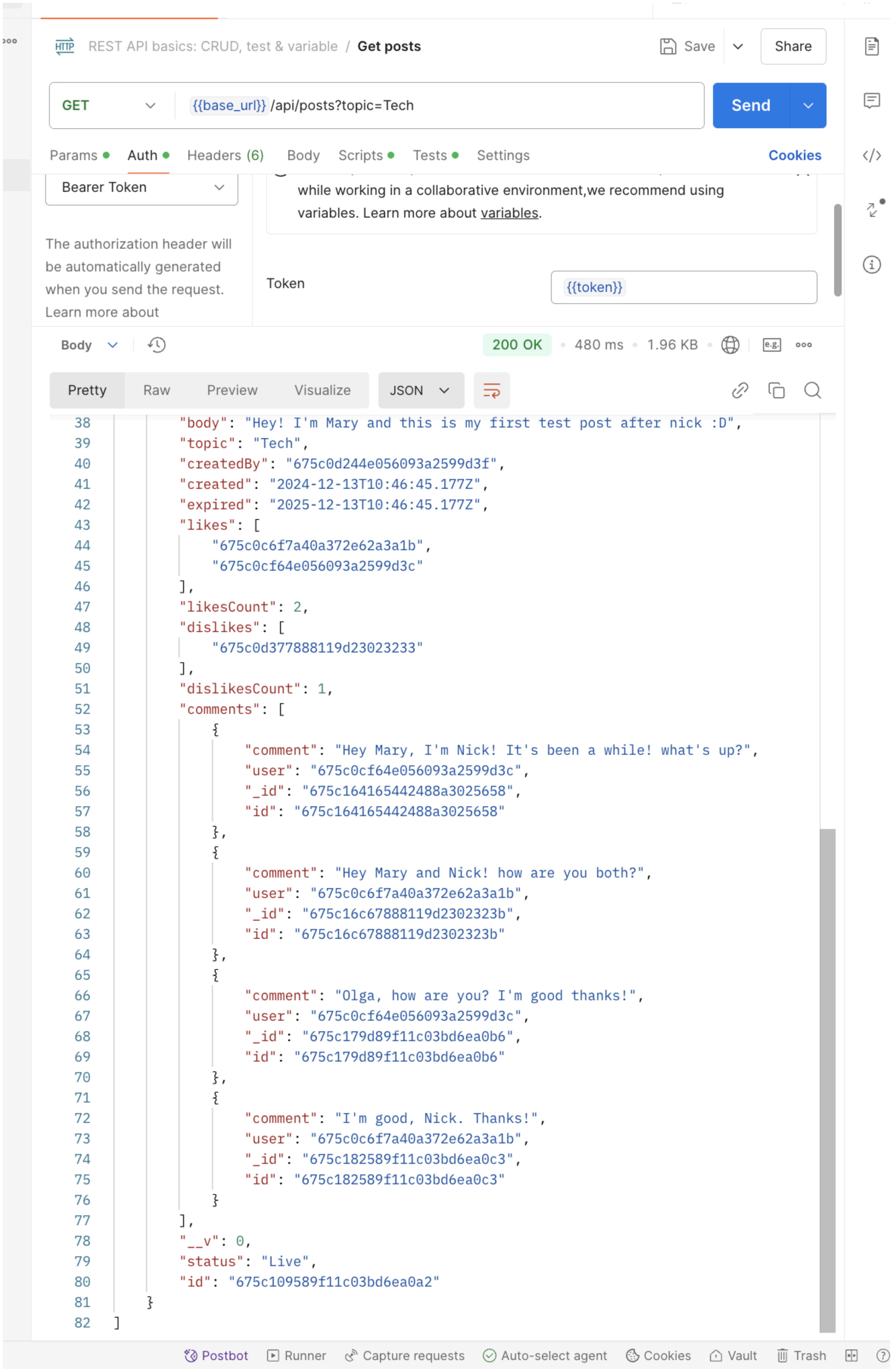
The authorization header will be automatically generated when you send the request. Learn more about

while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token {{token}}

200 OK • 480 ms • 1.96 KB

```
[
    {
        "_id": "675c0f3c4e056093a2599d41",
        "body": "Hey! I'm Olgaaa and this is my first post :D",
        "topic": "Tech",
        "createdBy": "675c0c6f7a40a372e62a3a1b",
        "created": "2024-12-13T10:41:00.938Z",
        "expired": "2025-12-13T10:41:00.938Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c0f3c4e056093a2599d41"
    },
    {
        "_id": "675c10364e056093a2599d43",
        "body": "Hey! I'm Nicky and this is my first test post after olga :D",
        "topic": "Tech",
        "createdBy": "675c0cf64e056093a2599d3c",
        "created": "2024-12-13T10:45:10.548Z",
        "expired": "2025-12-13T10:45:10.548Z",
        "likes": [
            "675c0d377888119d23023233"
        ],
        "likesCount": 1,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c10364e056093a2599d43"
    },
    {
        "_id": "675c109589f11c03bd6ea0a2",
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [
            "675c0c6f7a40a372e62a3a1b",
            "675c0cf64e056093a2599d3c"
```



REST API basics: CRUD, test & variable / **Get posts**

Save | Share

GET {{base_url}} /api/posts?topic=Tech Send

Params ● | Auth ● | Headers (6) | Body | Scripts ● | Tests ● | Settings | Cookies

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about

while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token {{token}}

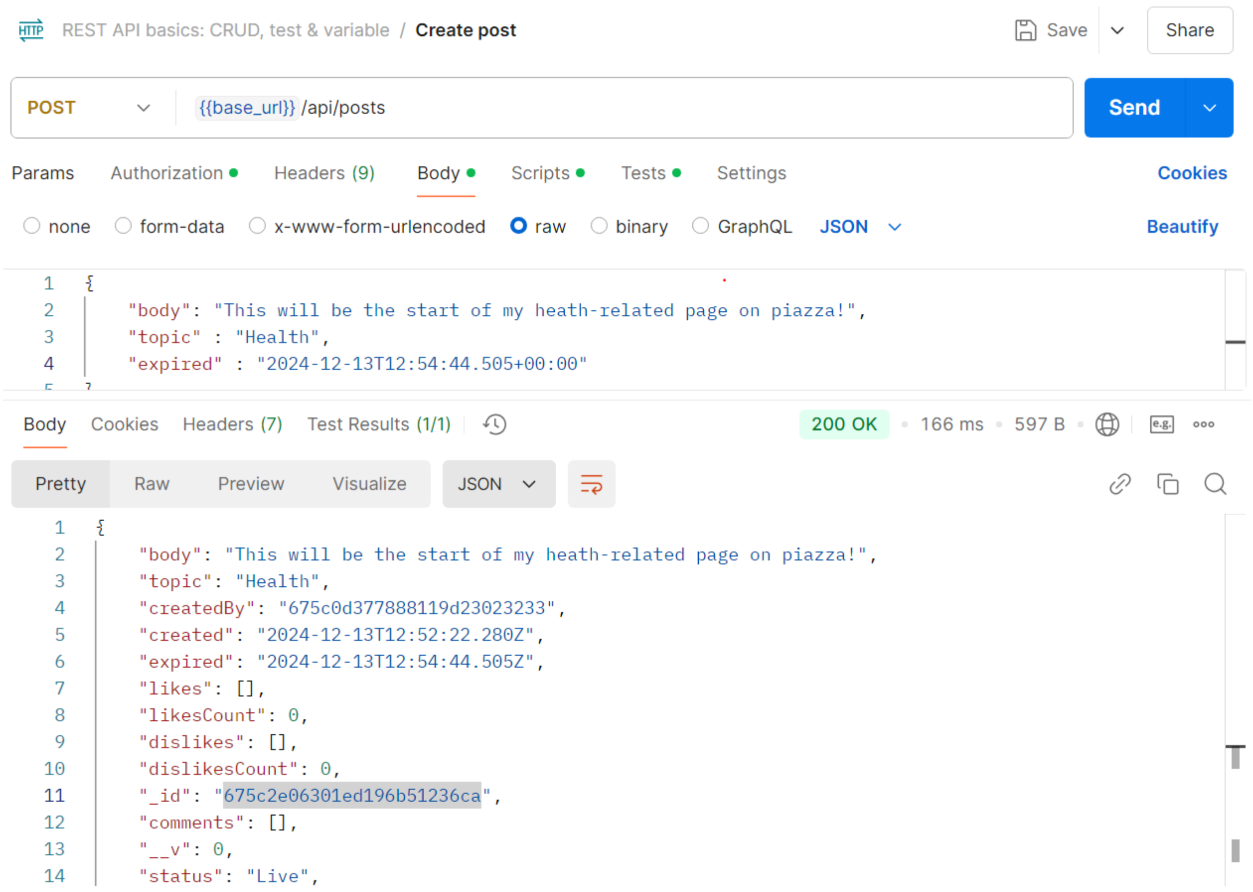
Body | 200 OK · 480 ms · 1.96 KB

Pretty | Raw | Preview | Visualize | JSON

```
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [
            "675c0c6f7a40a372e62a3a1b",
            "675c0cf64e056093a2599d3c"
        ],
        "likesCount": 2,
        "dislikes": [
            "675c0d377888119d23023233"
        ],
        "dislikesCount": 1,
        "comments": [
            {
                "comment": "Hey Mary, I'm Nick! It's been a while! what's up?",
                "user": "675c0cf64e056093a2599d3c",
                "_id": "675c164165442488a3025658",
                "id": "675c164165442488a3025658"
            },
            {
                "comment": "Hey Mary and Nick! how are you both?",
                "user": "675c0c6f7a40a372e62a3a1b",
                "_id": "675c16c67888119d2302323b",
                "id": "675c16c67888119d2302323b"
            },
            {
                "comment": "Olga, how are you? I'm good thanks!",
                "user": "675c0cf64e056093a2599d3c",
                "_id": "675c179d89f11c03bd6ea0b6",
                "id": "675c179d89f11c03bd6ea0b6"
            },
            {
                "comment": "I'm good, Nick. Thanks!",
                "user": "675c0c6f7a40a372e62a3a1b",
                "_id": "675c182589f11c03bd6ea0c3",
                "id": "675c182589f11c03bd6ea0c3"
            }
        ],
        "__v": 0,
        "status": "Live",
        "id": "675c109589f11c03bd6ea0a2"
    }
]
```

Postbot | Runner | Capture requests | Auto-select agent | Cookies | Vault | Trash

14. Nestor posts a message on the Health topic with an expiration time using her token.



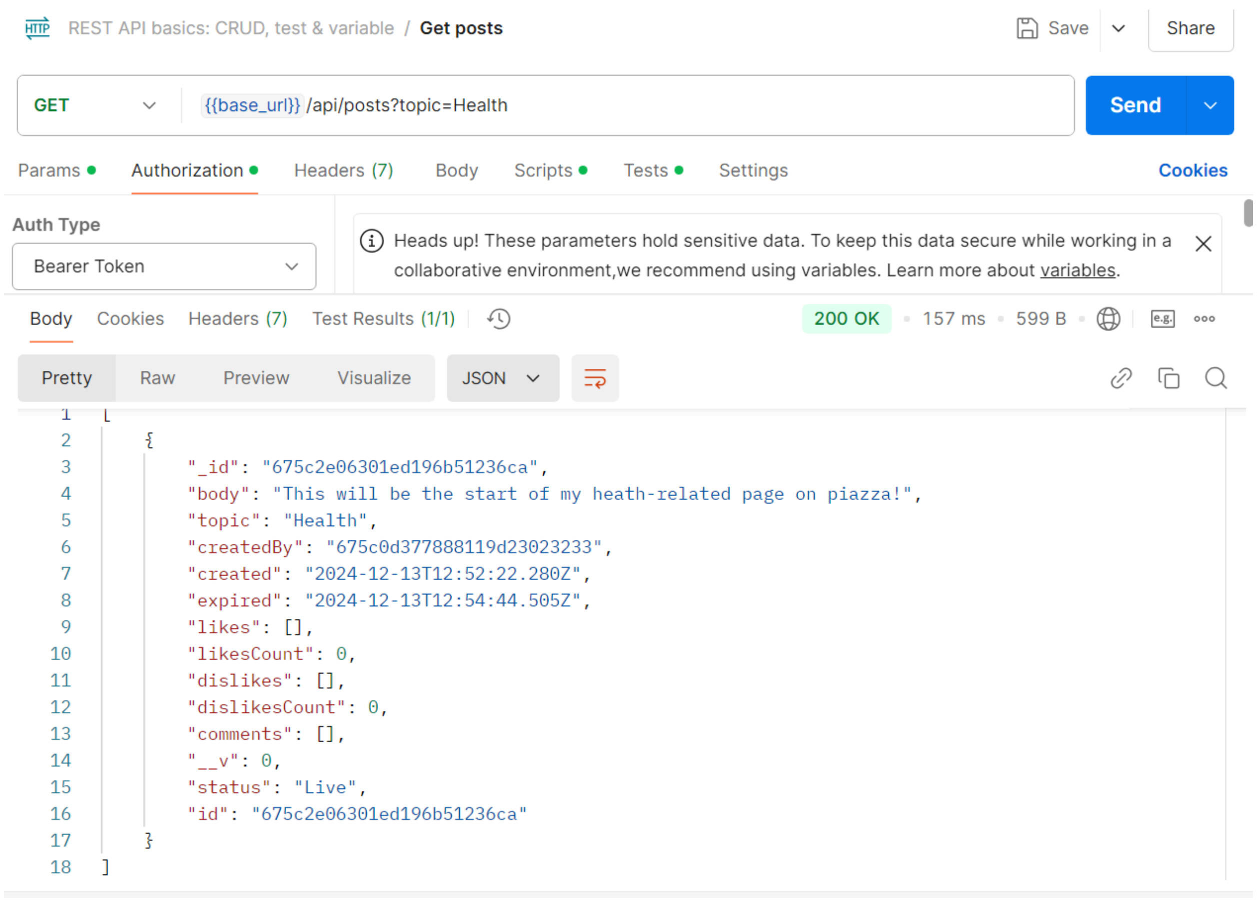
POST {{base_url}}/api/posts

```
{
    "body": "This will be the start of my heath-related page on piazza!",
    "topic" : "Health",
    "expired" : "2024-12-13T12:54:44.505+00:00"
```

200 OK • 166 ms • 597 B

```
{
    "body": "This will be the start of my heath-related page on piazza!",
    "topic": "Health",
    "createdBy": "675c0d377888119d23023233",
    "created": "2024-12-13T12:52:22.280Z",
    "expired": "2024-12-13T12:54:44.505Z",
    "likes": [],
    "likesCount": 0,
    "dislikes": [],
    "dislikesCount": 0,
    "_id": "675c2e06301ed196b51236ca",
    "comments": [],
    "__v": 0,
    "status": "Live",
```

15. Mary browses all the available posts on the Health topic; at this stage, she can see only Nestor's post.



GET {{base_url}} /api/posts?topic=Health

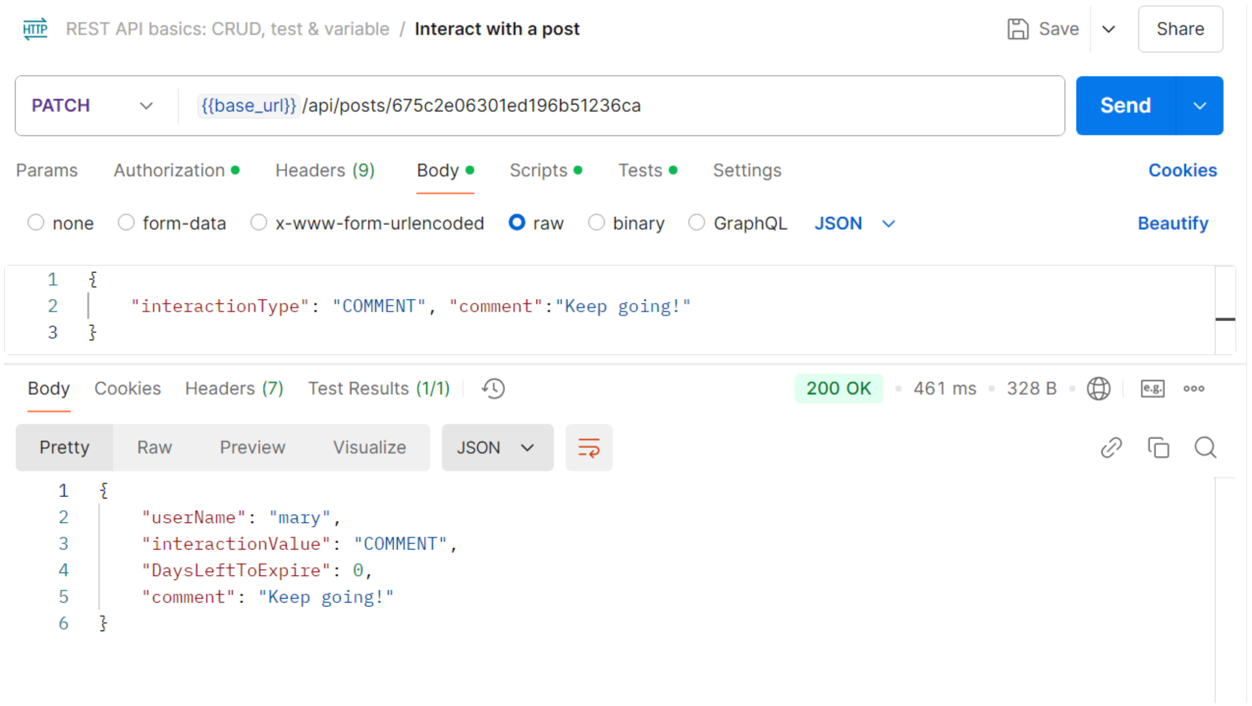
Auth Type: Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

200 OK • 157 ms • 599 B

```
[
    {
        "_id": "675c2e06301ed196b51236ca",
        "body": "This will be the start of my heath-related page on piazza!",
        "topic": "Health",
        "createdBy": "675c0d377888119d23023233",
        "created": "2024-12-13T12:52:22.280Z",
        "expired": "2024-12-13T12:54:44.505Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "status": "Live",
        "id": "675c2e06301ed196b51236ca"
    }
]
```

16. Mary posts a comment in Nestor's message on the Health topic.



PATCH {{base_url}} /api/posts/675c2e06301ed196b51236ca

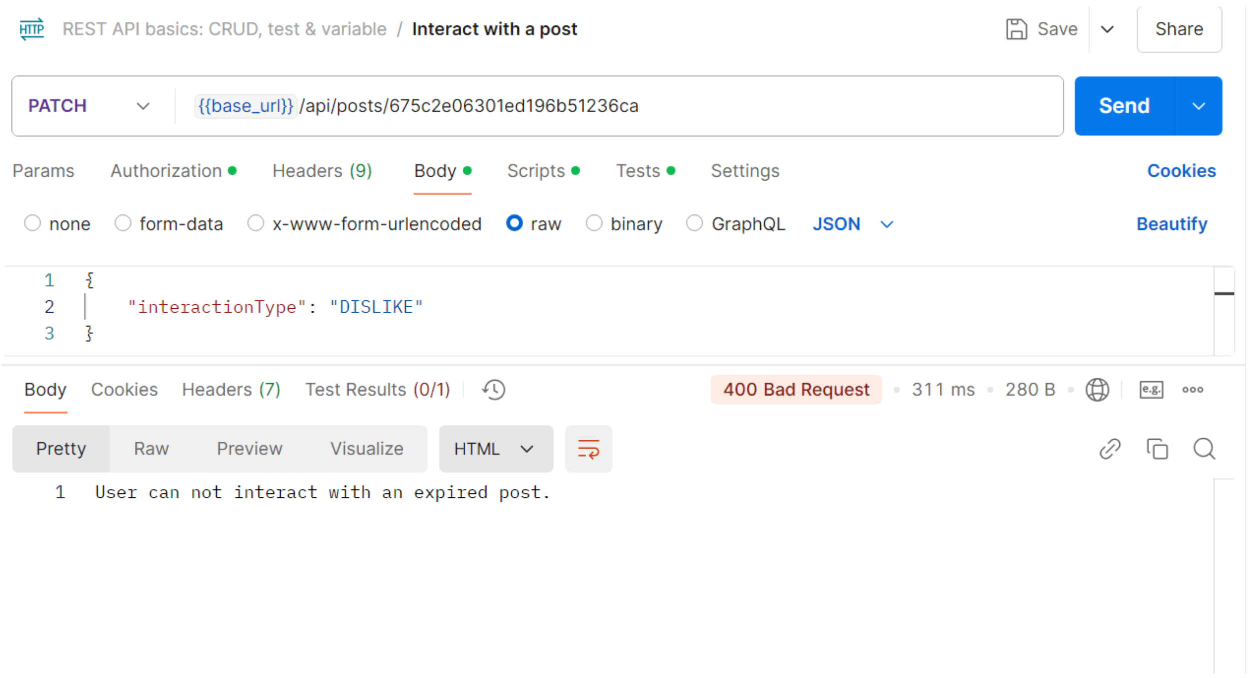
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "interactionType": "COMMENT", "comment":"Keep going!"
}
```

200 OK • 461 ms • 328 B

```
{
    "userName": "mary",
    "interactionValue": "COMMENT",
    "DaysLeftToExpire": 0,
    "comment": "Keep going!"
}
```

17. Mary dislikes Nestor’s message on the Health topic after the end of post-expiration time. This should fail.



REST API basics: CRUD, test & variable / **Interact with a post**

PATCH {{base_url}} /api/posts/675c2e06301ed196b51236ca

Params | Authorization ● | Headers (9) | Body ● | Scripts ● | Tests ● | Settings

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

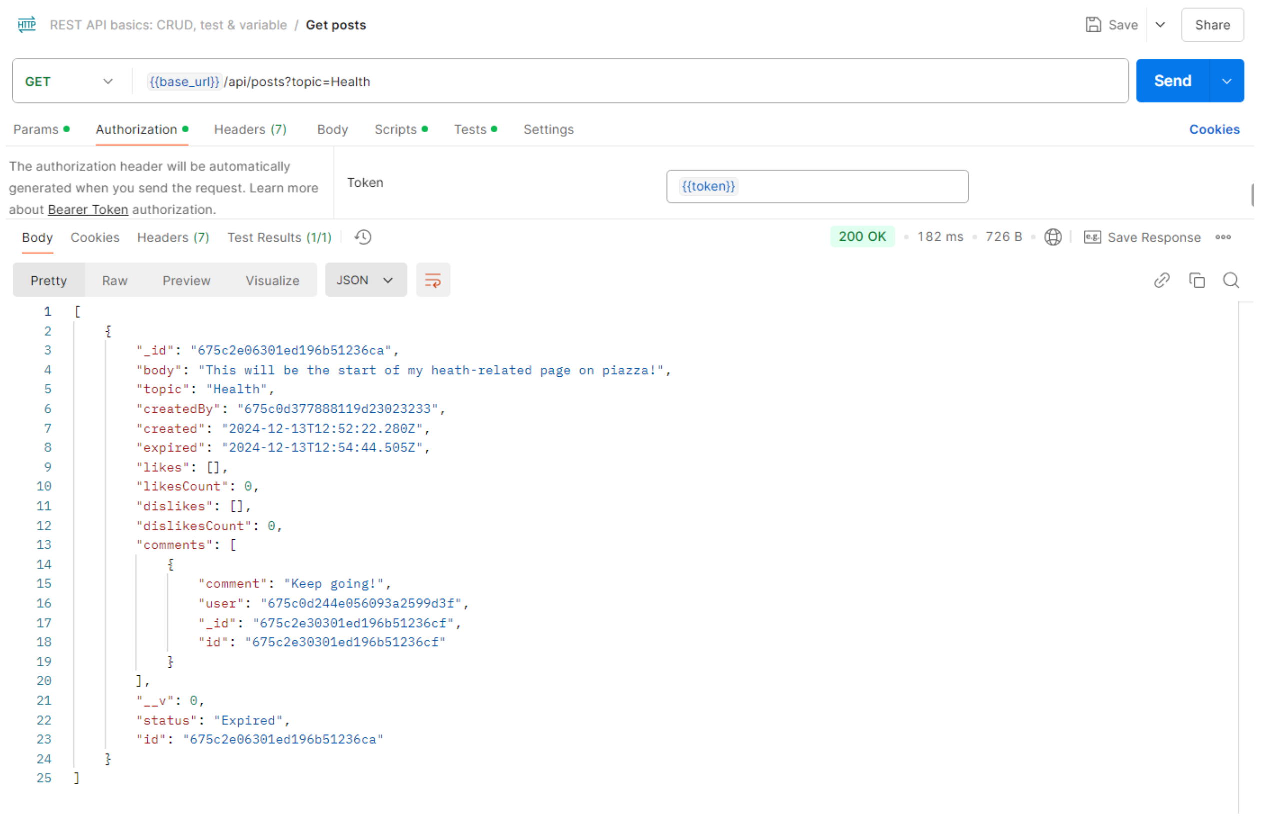
```
{
    "interactionType": "DISLIKE"
}
```

Body | Cookies | Headers (7) | Test Results (0/1) — 400 Bad Request • 311 ms • 280 B

Pretty | Raw | Preview | Visualize | HTML

```
User can not interact with an expired post.
```

18. Nestor browses all the messages on the Health topic. There should be only one post (his own) with one comment (Mary’s).



REST API basics: CRUD, test & variable / **Get posts**

GET {{base_url}} /api/posts?topic=Health

Params ● | Authorization ● | Headers (7) | Body | Scripts ● | Tests ● | Settings

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

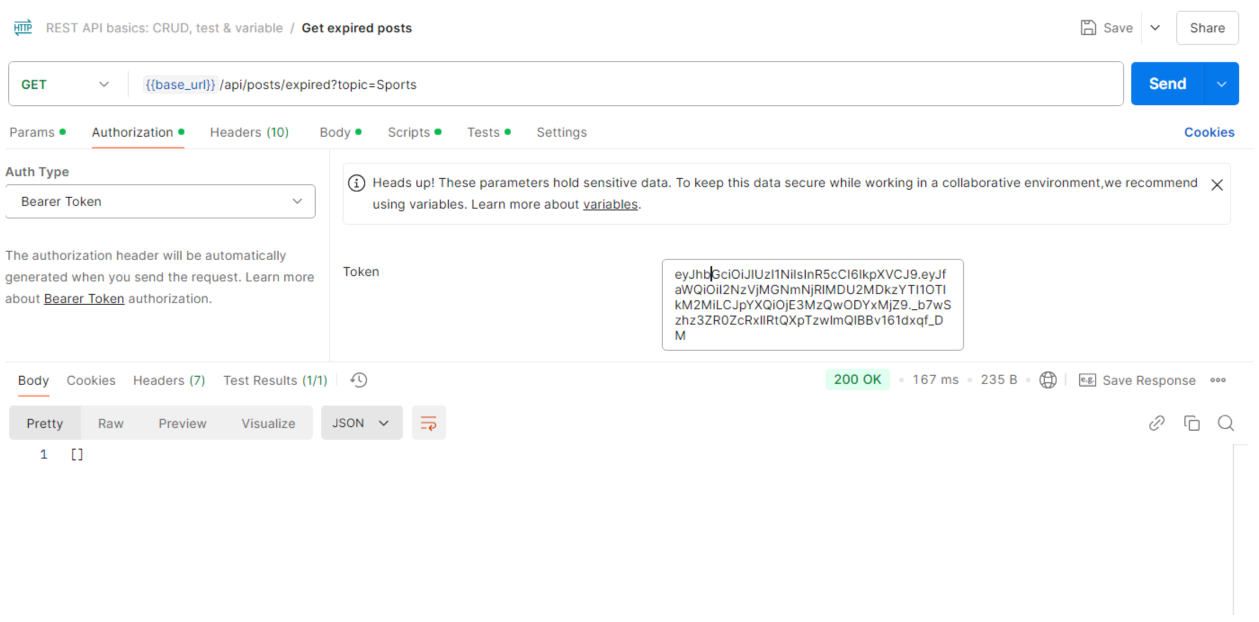
Token {{token}}

Body | Cookies | Headers (7) | Test Results (1/1) — 200 OK • 182 ms • 726 B

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "_id": "675c2e06301ed196b51236ca",
        "body": "This will be the start of my heath-related page on piazza!",
        "topic": "Health",
        "createdBy": "675c0d377888119d23023233",
        "created": "2024-12-13T12:52:22.280Z",
        "expired": "2024-12-13T12:54:44.505Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [
            {
                "comment": "Keep going!",
                "user": "675c0d244e056093a2599d3f",
                "_id": "675c2e30301ed196b51236cf",
                "id": "675c2e30301ed196b51236cf"
            }
        ],
        "__v": 0,
        "status": "Expired",
        "id": "675c2e06301ed196b51236ca"
    }
]
```

19. Nick browses all the expired messages on the Sports topic. These should be empty.



GET {{base_url}} /api/posts/expired?topic=Sports

Auth Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

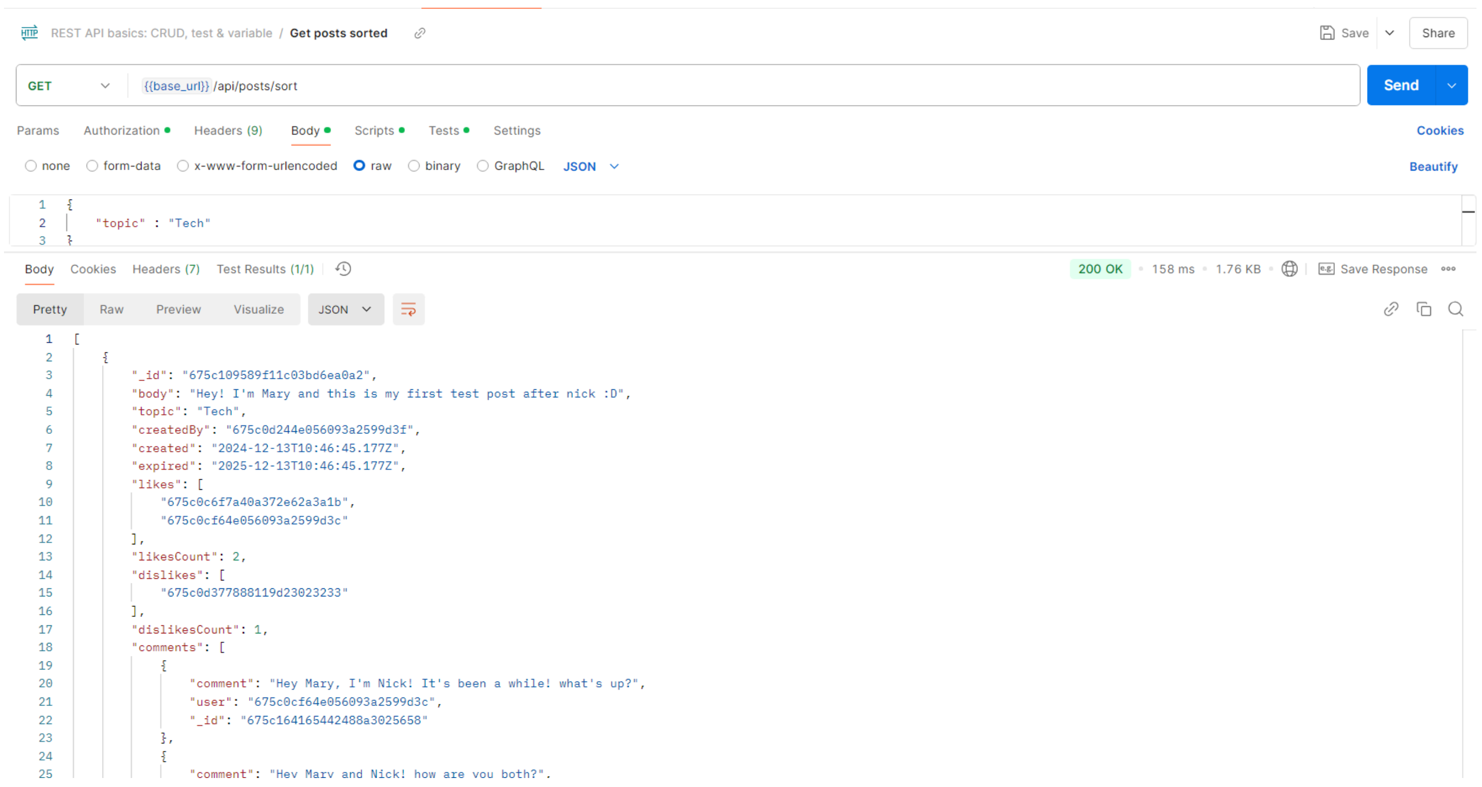
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzVjMGNmNjRlMDU2MDkzYTI1OTIkM2MiLCJpYXQiOjE3MzQwODYxMjZ9._b7wSzhz3ZR0ZcRxIlRtQXpTzwImQIBBv161dxqf_DM

200 OK • 167 ms • 235 B

```
[]
```

20. Nestor queries for an active post with the highest interest (maximum number of likes and dislikes) in the Tech topic. This should be Mary's post.



GET {{base_url}} /api/posts/sort

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

```
{
    "topic" : "Tech"
}
```

200 OK • 158 ms • 1.76 KB

```
[
    {
        "_id": "675c109589f11c03bd6ea0a2",
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [
            "675c0c6f7a40a372e62a3a1b",
            "675c0cf64e056093a2599d3c"
        ],
        "likesCount": 2,
        "dislikes": [
            "675c0d377888119d23023233"
        ],
        "dislikesCount": 1,
        "comments": [
            {
                "comment": "Hey Mary, I'm Nick! It's been a while! what's up?",
                "user": "675c0cf64e056093a2599d3c",
                "_id": "675c164165442488a3025658"
            },
            {
                "comment": "Hey Mary and Nick! how are you both?",
```

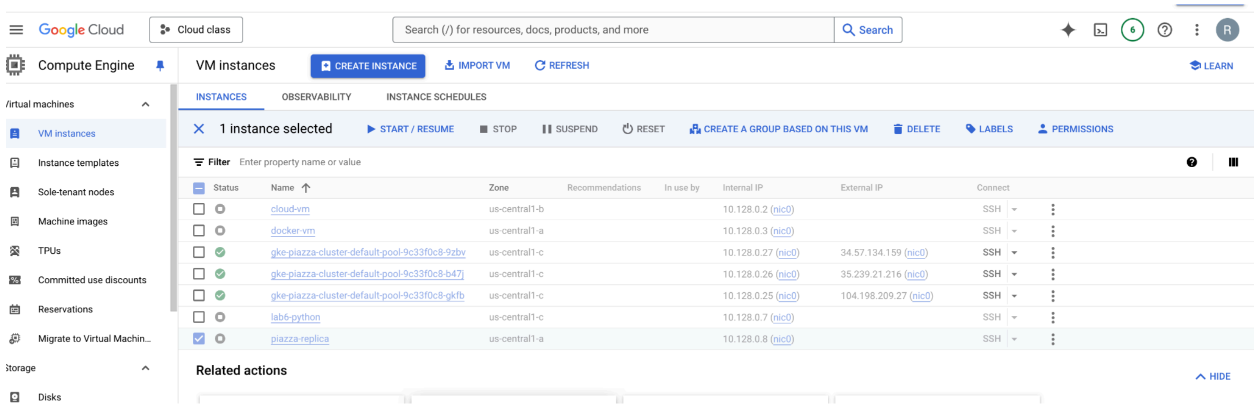
Full response: sorted by descending order of the sum of likes and dislikes:

```
[
  {
    "_id": "675c109589f11c03bd6ea0a2",
        "body": "Hey! I'm Mary and this is my first test post after nick :D",
        "topic": "Tech",
        "createdBy": "675c0d244e056093a2599d3f",
        "created": "2024-12-13T10:46:45.177Z",
        "expired": "2025-12-13T10:46:45.177Z",
        "likes": [
            "675c0c6f7a40a372e62a3a1b",
            "675c0cf64e056093a2599d3c"
        ],
        "likesCount": 2,
        "dislikes": [
            "675c0d377888119d23023233"
        ],
        "dislikesCount": 1,
        "comments": [
            {
                "comment": "Hey Mary, I'm Nick! It's been a while! what's up?",
                "user": "675c0cf64e056093a2599d3c",
                "_id": "675c164165442488a3025658"
            },
            {
                "comment": "Hey Mary and Nick! how are you both?",
                "user": "675c0c6f7a40a372e62a3a1b",
                "_id": "675c16c67888119d2302323b"
            },
            {
                "comment": "Olga, how are you? I'm good thanks!",
                "user": "675c0cf64e056093a2599d3c",
                "_id": "675c179d89f11c03bd6ea0b6"
            },
            {
                "comment": "I'm good, Nick. Thanks!",
                "user": "675c0c6f7a40a372e62a3a1b",
                "_id": "675c182589f11c03bd6ea0c3"
            }
        ],
        "__v": 0,
        "likesDislikesCount": 3
    },
    {
        "_id": "675c10364e056093a2599d43",
        "body": "Hey! I'm Nicky and this is my first test post after olga :D",
        "topic": "Tech",
        "createdBy": "675c0cf64e056093a2599d3c",
        "created": "2024-12-13T10:45:10.548Z",
        "expired": "2025-12-13T10:45:10.548Z",
        "likes": [
            "675c0d377888119d23023233"
        ],
        "likesCount": 1,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "likesDislikesCount": 1
    },
    {
        "_id": "675c0f3c4e056093a2599d41",
        "body": "Hey! I'm Olgaaa and this is my first post :D",
        "topic": "Tech",
        "createdBy": "675c0c6f7a40a372e62a3a1b",
        "created": "2024-12-13T10:41:00.938Z",
        "expired": "2025-12-13T10:41:00.938Z",
        "likes": [],
        "likesCount": 0,
        "dislikes": [],
        "dislikesCount": 0,
        "comments": [],
        "__v": 0,
        "likesDislikesCount": 0
    }
]
```

The next step would be to deply my project into a VM using docker

- I have uploaded my code to a github repo here https://github.com/RawanReda/Piazza-replica
- then I deployed it in google cloud vm

I have first cloned the repo in my VM. The repo was private at the time so I used a token, then I created a Dockerfile to build the image.

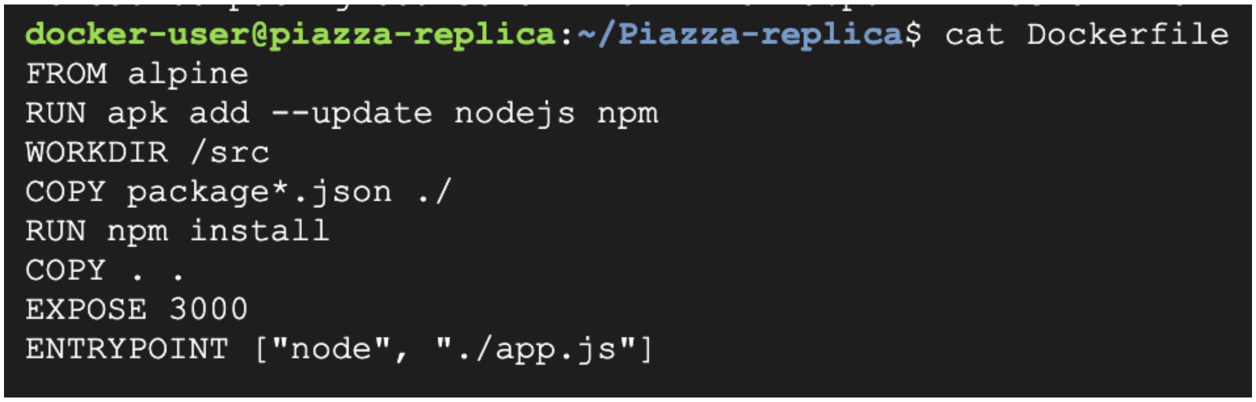


1 instance selected

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
○	cloud-vm	us-central1-b			10.128.0.2 (nic0)		SSH
○	docker-vm	us-central1-a			10.128.0.3 (nic0)		SSH
✓	gke-piazza-cluster-default-pool-9c33f0c8-9zbv	us-central1-c			10.128.0.27 (nic0)	34.57.134.159 (nic0)	SSH
✓	gke-piazza-cluster-default-pool-9c33f0c8-b47j	us-central1-c			10.128.0.26 (nic0)	35.239.21.216 (nic0)	SSH
✓	gke-piazza-cluster-default-pool-9c33f0c8-gkfb	us-central1-c			10.128.0.25 (nic0)	104.198.209.27 (nic0)	SSH
○	lab6-python	us-central1-c			10.128.0.7 (nic0)		SSH
○	piazza-replica	us-central1-a			10.128.0.8 (nic0)		SSH

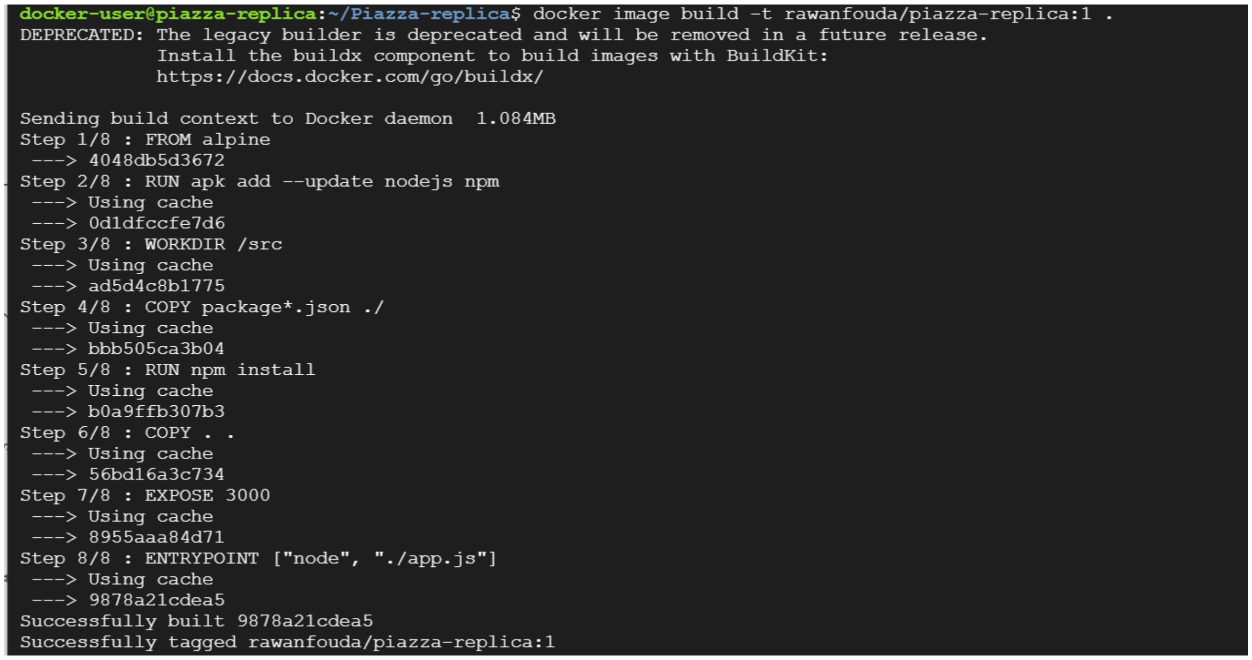
Related actions

This is the docker file:

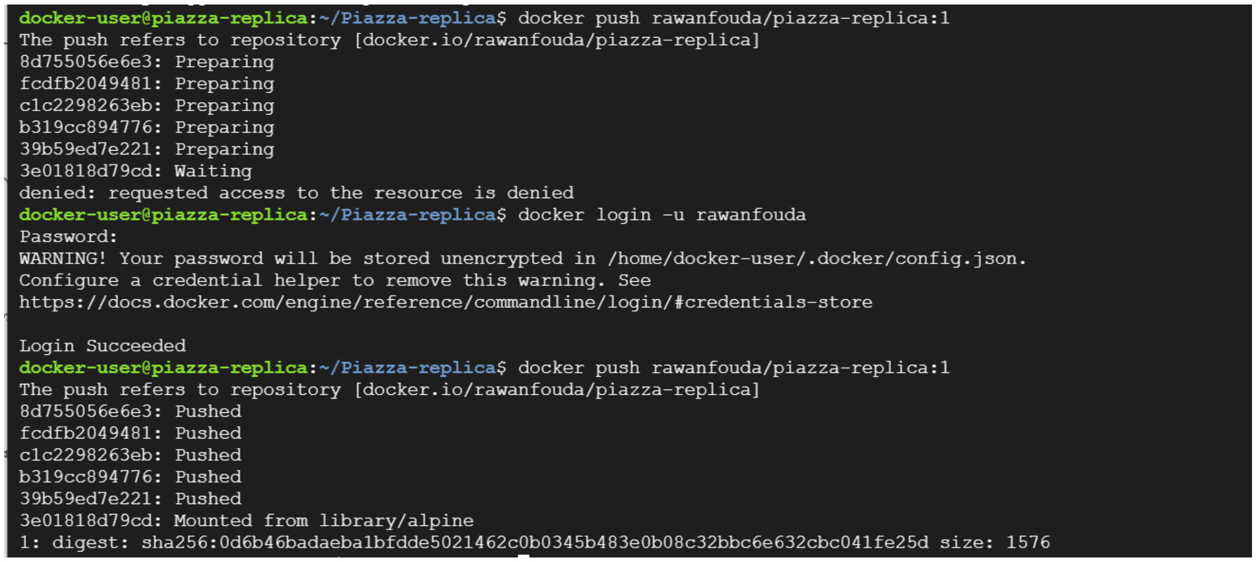


```
docker-user@piazza-replica:~/Piazza-replica$ cat Dockerfile
FROM alpine
RUN apk add --update nodejs npm
WORKDIR /src
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
ENTRYPOINT ["node", "./app.js"]
```

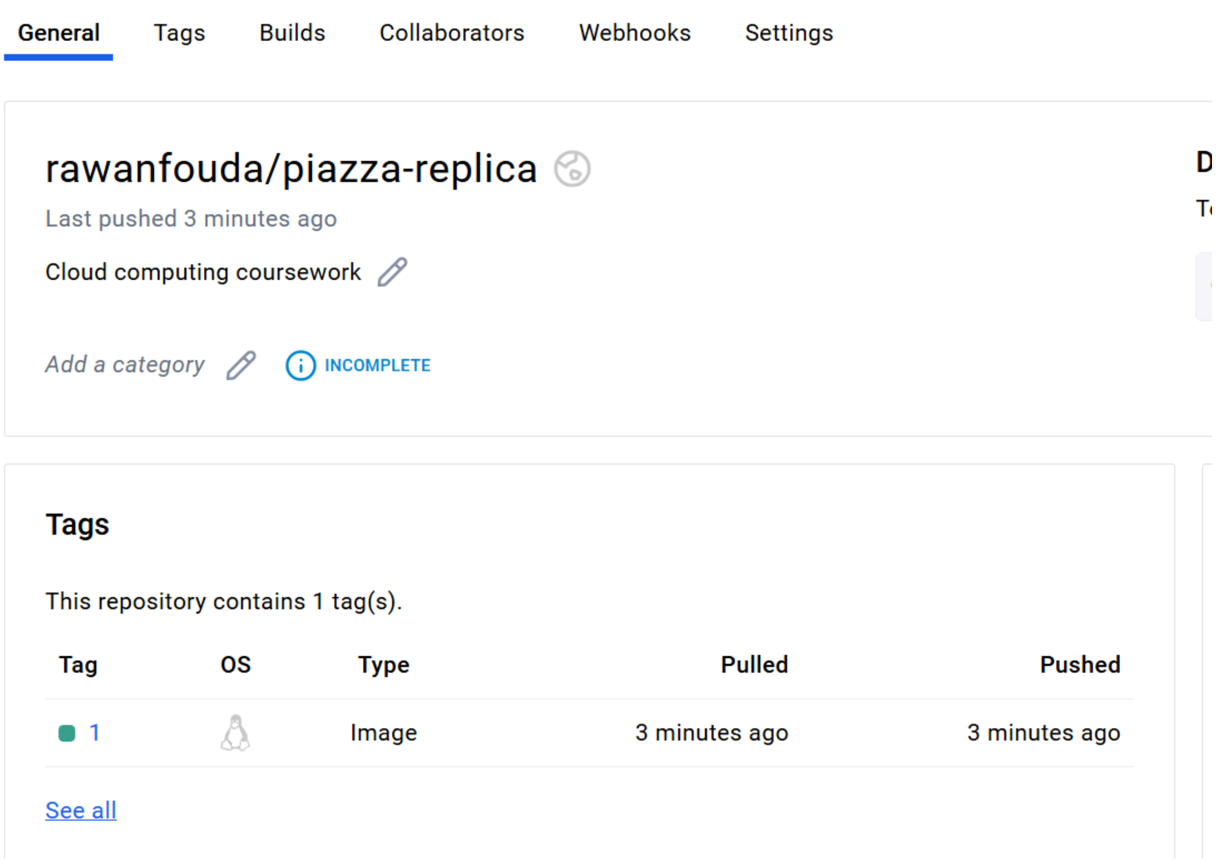
I have first copied package*.json files and then ran npm install, to make sure all the needed dependencies would be available and the app would be running, then I built the image and pushed it to dockerhub as shown below.



```
docker-user@piazza-replica:~/Piazza-replica$ docker image build -t rawanfouda/piazza-replica:1 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  1.084MB
Step 1/8 : FROM alpine
 ---> 4048db5d3672
Step 2/8 : RUN apk add --update nodejs npm
 ---> Using cache
 ---> 0d1dfccfe7d6
Step 3/8 : WORKDIR /src
 ---> Using cache
 ---> ad5d4c8b1775
Step 4/8 : COPY package*.json ./
 ---> Using cache
 ---> bbb505ca3b04
Step 5/8 : RUN npm install
 ---> Using cache
 ---> b0a9ffb307b3
Step 6/8 : COPY . .
 ---> Using cache
 ---> 56bd16a3c734
Step 7/8 : EXPOSE 3000
 ---> Using cache
 ---> 8955aaa84d71
Step 8/8 : ENTRYPOINT ["node", "./app.js"]
 ---> Using cache
 ---> 9878a21cdea5
Successfully built 9878a21cdea5
Successfully tagged rawanfouda/piazza-replica:1
```



```
docker-user@piazza-replica:~/Piazza-replica$ docker push rawanfouda/piazza-replica:1
The push refers to repository [docker.io/rawanfouda/piazza-replica]
8d755056e6e3: Preparing
fcdfb2049481: Preparing
c1c2298263eb: Preparing
b319cc894776: Preparing
39b59ed7e221: Preparing
3e01818d79cd: Waiting
denied: requested access to the resource is denied
docker-user@piazza-replica:~/Piazza-replica$ docker login -u rawanfouda
Password:
WARNING! Your password will be stored unencrypted in /home/docker-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
docker-user@piazza-replica:~/Piazza-replica$ docker push rawanfouda/piazza-replica:1
The push refers to repository [docker.io/rawanfouda/piazza-replica]
8d755056e6e3: Pushed
fcdfb2049481: Pushed
c1c2298263eb: Pushed
b319cc894776: Pushed
39b59ed7e221: Pushed
3e01818d79cd: Mounted from library/alpine
1: digest: sha256:0d6b46badaeba1bfdde5021462c0b0345b483e0b08c32bbc6e632cbc041fe25d size: 1576
```



rawanfouda/piazza-replica

Last pushed 3 minutes ago

Cloud computing coursework

Add a category INCOMPLETE

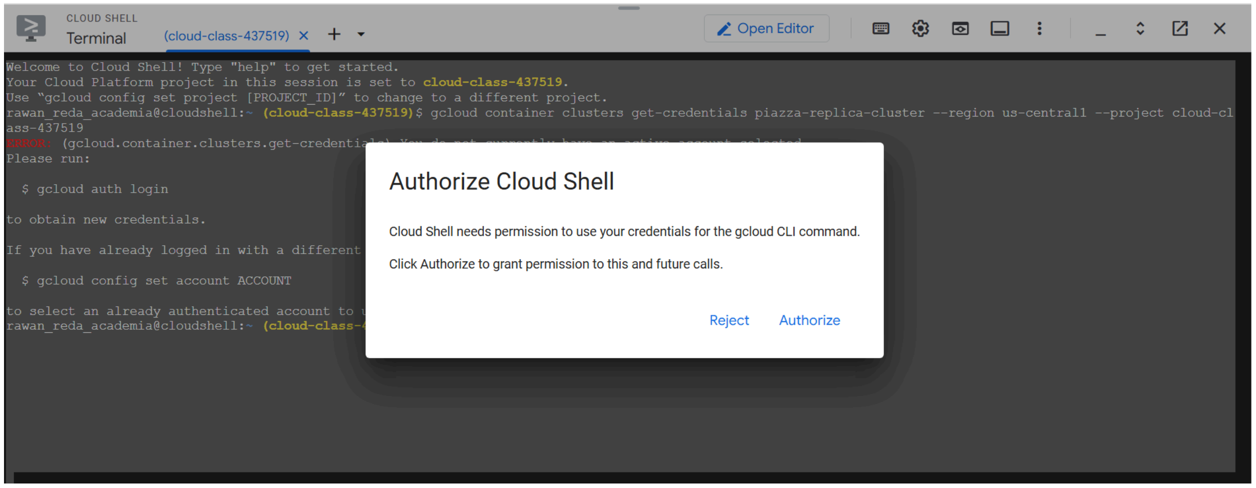
Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
🟩 1		Image	3 minutes ago	3 minutes ago

See all

Afterwards, I deployed the app in kubernetes where we create a kubernetes cluster on google cloud and deploy. I have also applied the load balancing and replication requirements, and test the running pods by using the external IP generate by kubernetes service.



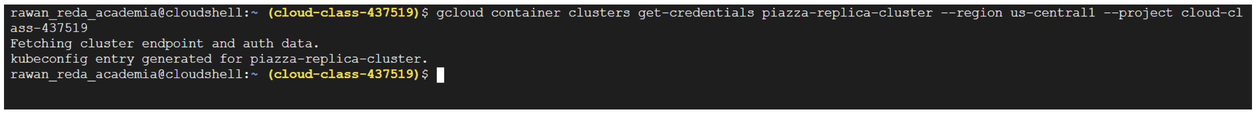
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cloud-class-437519.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ gcloud container clusters get-credentials piazza-replica-cluster --region us-central1 --project cloud-cl
ass-437519
ERROR: (gcloud.container.clusters.get-credenti
Please run:

  $ gcloud auth login

to obtain new credentials.

If you have already logged in with a different

  $ gcloud config set account ACCOUNT

to select an already authenticated account to
rawan_reda_academia@cloudshell:~ (cloud-class-
```

Authorize Cloud Shell

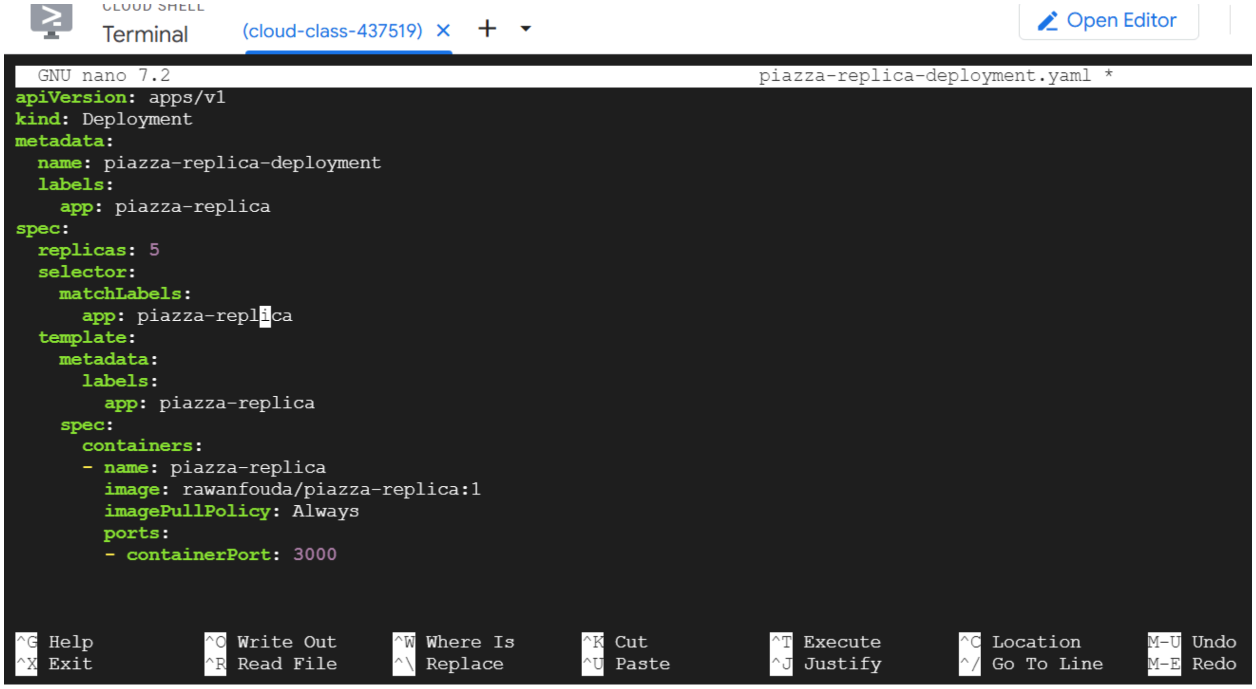
Cloud Shell needs permission to use your credentials for the gcloud CLI command.

Click Authorize to grant permission to this and future calls.

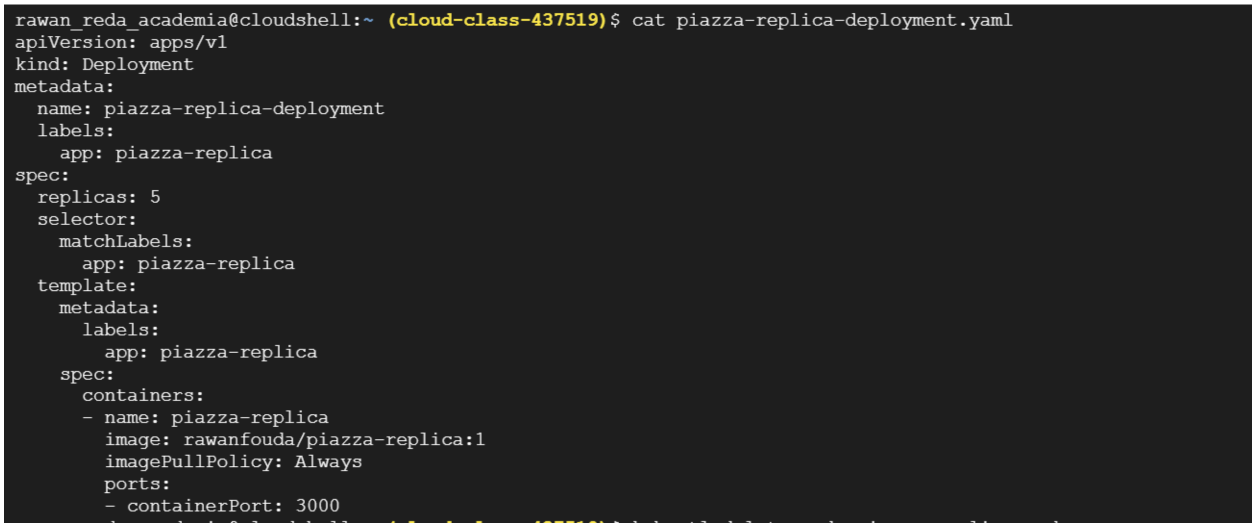
Reject Authorize



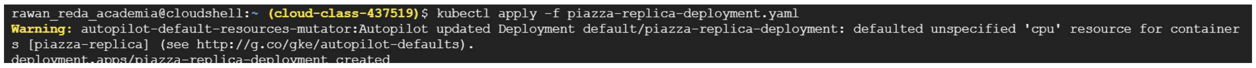
```
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ gcloud container clusters get-credentials piazza-replica-cluster --region us-central1 --project cloud-cl
ass-437519
Fetching cluster endpoint and auth data.
kubeconfig entry generated for piazza-replica-cluster.
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$
```



```
  GNU nano 7.2                                          piazza-replica-deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: piazza-replica-deployment
  labels:
    app: piazza-replica
spec:
  replicas: 5
  selector:
    matchLabels:
      app: piazza-replica
  template:
    metadata:
      labels:
        app: piazza-replica
    spec:
      containers:
      - name: piazza-replica
        image: rawanfouda/piazza-replica:1
        imagePullPolicy: Always
        ports:
        - containerPort: 3000



^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo
```

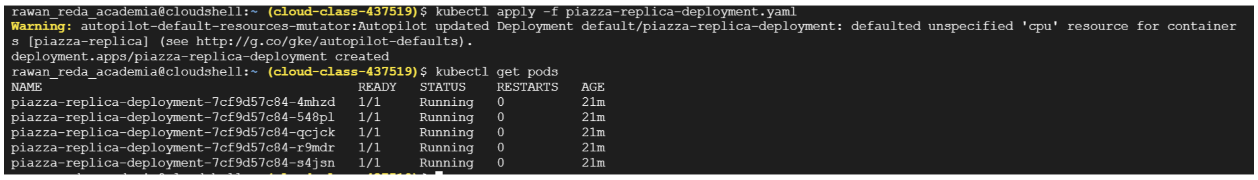


```
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ cat piazza-replica-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: piazza-replica-deployment
  labels:
    app: piazza-replica
spec:
  replicas: 5
  selector:
    matchLabels:
      app: piazza-replica
  template:
    metadata:
      labels:
        app: piazza-replica
    spec:
      containers:
      - name: piazza-replica
        image: rawanfouda/piazza-replica:1
        imagePullPolicy: Always
        ports:
        - containerPort: 3000
```



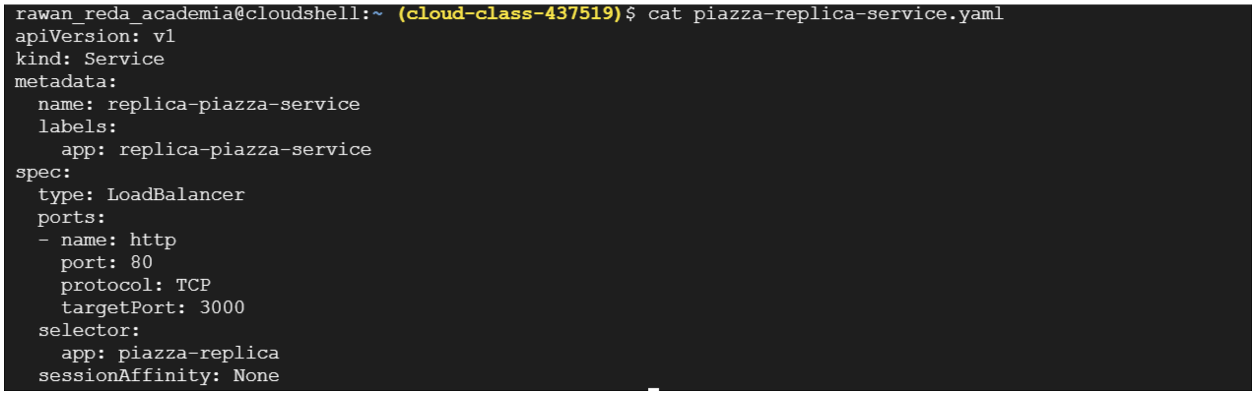
```
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ kubectl apply -f piazza-replica-deployment.yaml
Warning: autopilot-default-resources-mutator:Autopilot updated Deployment default/piazza-replica-deployment: defaulted unspecified 'cpu' resource for container
s [piazza-replica] (see http://g.co/gke/autopilot-defaults).
deployment.apps/piazza-replica-deployment created
```

made sure all the pods are running:

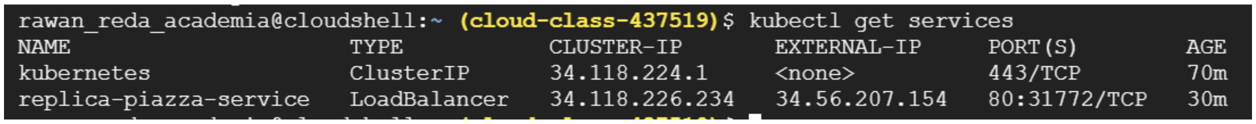


```
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ kubectl apply -f piazza-replica-deployment.yaml
Warning: autopilot-default-resources-mutator:Autopilot updated Deployment default/piazza-replica-deployment: defaulted unspecified 'cpu' resource for container
s [piazza-replica] (see http://g.co/gke/autopilot-defaults).
deployment.apps/piazza-replica-deployment created
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ kubectl get pods
NAME                                         READY   STATUS    RESTARTS   AGE
piazza-replica-deployment-7cf9d57c84-4mhzd   1/1     Running   0          21m
piazza-replica-deployment-7cf9d57c84-548pl   1/1     Running   0          21m
piazza-replica-deployment-7cf9d57c84-qcjck   1/1     Running   0          21m
piazza-replica-deployment-7cf9d57c84-r9mdr   1/1     Running   0          21m
piazza-replica-deployment-7cf9d57c84-s4jsn   1/1     Running   0          21m
```

Then, I implemented the load balancer setup:

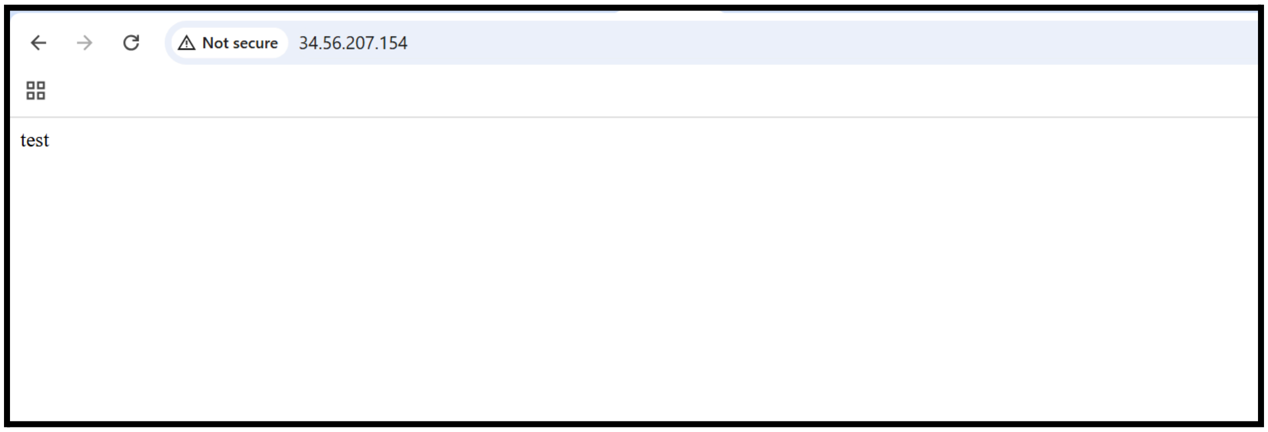


```
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ cat piazza-replica-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: replica-piazza-service
  labels:
    app: replica-piazza-service
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app: piazza-replica
  sessionAffinity: None
```



```
rawan_reda_academia@cloudshell:~ (cloud-class-437519)$ kubectl get services
NAME                     TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
kubernetes               ClusterIP      34.118.224.1     <none>          443/TCP        70m
replica-piazza-service   LoadBalancer   34.118.226.234   34.56.207.154   80:31772/TCP   30m
```

and this was just a simple test to ensure my app was running on kubernetes:



test

Future work:

- create CI/CD pipeline
- create UI for piazza and perhaps have more user experience scenarios