Cloud Computing Technical Report 13-12-2024

James Bacon / jbacon03@student.bbk.ac.uk / 13168233

Below I have included screenshots from the various stages of development (though admittedly I started working on the scripts and then came back to run what I had in the VM).

Installation and deployment of software in virtualised environments

To begin with, I installed node, npm and git, as evidenced below:

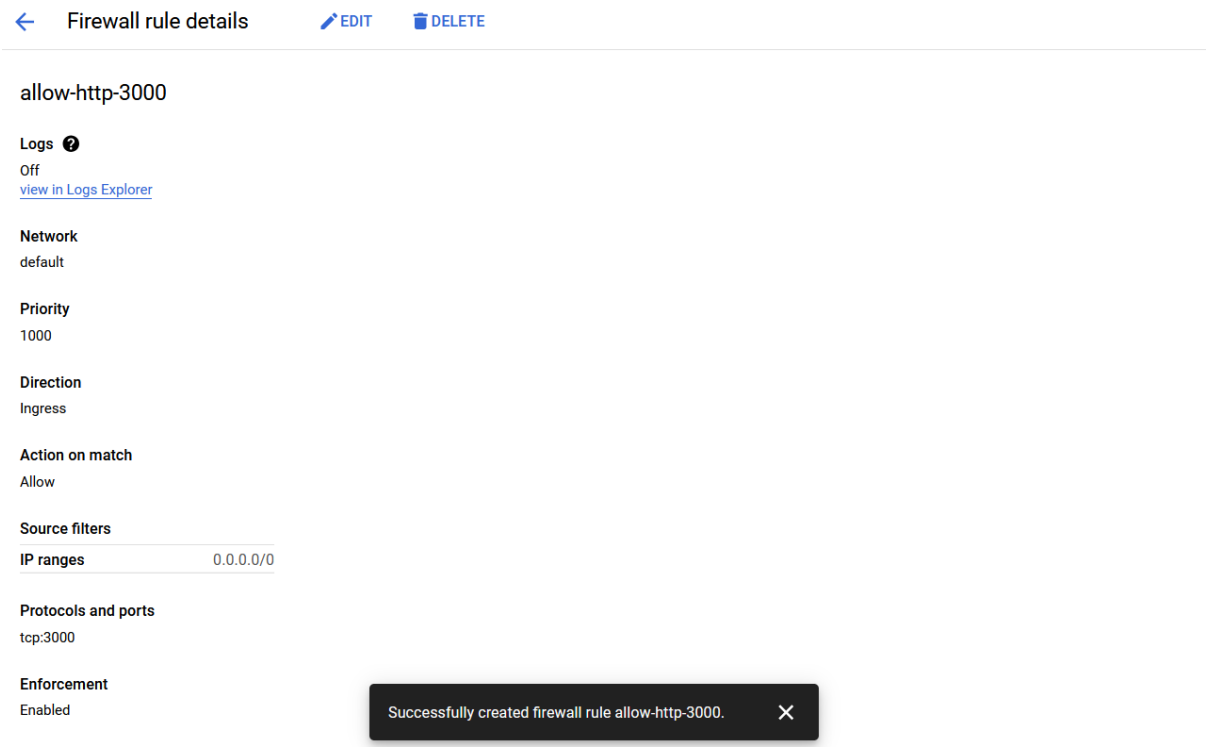
SSH-in-browser UPLOAD FILE DOWNLOAD FILE

```
jaybacon234@piazza-vm:~$ node -v
v16.20.2
jaybacon234@piazza-vm:~$ npm -v
8.19.4
jaybacon234@piazza-vm:~$ git --version
git version 2.25.1
jaybacon234@piazza-vm:~$
```

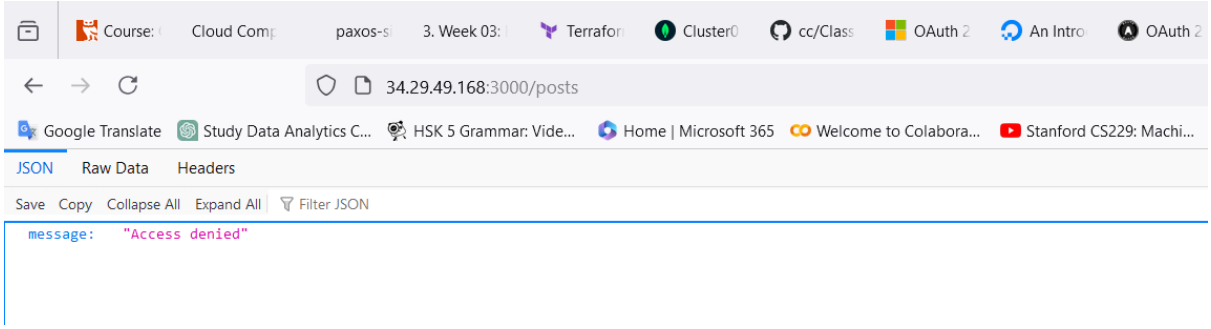
I then cloned the repository from Github using the repository URL and ran the app. As you can see, the server is up and running, the connector is on, and it shows how many posts have expired since I last ran the app.

```
Last login: Sun Nov 24 17:46:19 2024 from 35.235.244.32
jaybacon234@piazza-vm:~$ git clone https://github.com/BayJacon/piazza-coursework.git
Cloning into 'piazza-coursework'...
remote: Enumerating objects: 2571, done.
remote: Counting objects: 100% (2571/2571), done.
remote: Compressing objects: 100% (2096/2096), done.
remote: Total 2571 (delta 386), reused 2560 (delta 375), pack-reused 0 (from 0)
Receiving objects: 100% (2571/2571), 3.56 MiB | 9.51 MiB/s, done.
Resolving deltas: 100% (386/386), done.
jaybacon234@piazza-vm:~$ npm install
npm ERR! code ENOENT
npm ERR! syscall open
npm ERR! path /home/jaybacon234/package.json
npm ERR! errno -2
npm ERR! enoent ENOENT: no such file or directory, open '/home/jaybacon234/package.json'
npm ERR! enoent This is related to npm not being able to find a file.
npm ERR! enoent

npm ERR! A complete log of this run can be found in:
npm ERR!     /home/jaybacon234/.npm/_logs/2024-12-08T13_01_25_365Z-debug-0.log
jaybacon234@piazza-vm:~$ ^C
jaybacon234@piazza-vm:~$ cd piazza-coursework
jaybacon234@piazza-vm:~/piazza-coursework$ npm install

up to date, audited 138 packages in 989ms

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
jaybacon234@piazza-vm:~/piazza-coursework$ node app.js
Server is up and running...
Your MongoDB connector is on...
2 posts updated to "Expired".
```

After struggling to access the app using the external IP address of my VM, I found that I needed to create a firewall rule to allow traffic from all IP ranges:



And finally, I was able to run the app and connect to it as expected using the VM external IP address – I would expect to see access denied here as I've not yet used an auth-token to view it, but it shows that the app is up and running from the VM.



Development of Piazza RESTful APIs

It is apparent from the description in phase C that we would need a scalable database to store posts and user information, as well as our scripts. The structure of the app is similar to the Mini-Post app developed in lab 4, so I took this as my starting template and expand from there. This included of course installing all the necessary packages such as express, mongoose, nodemon and body-parser, and implementing the folder setup with a /routes and /models on top of the root folder containing app.js, which contained the functions and schema of the app respectively.

Posting function in post.js:

To begin with, it was apparent that the post functions for Mini-Post were much more basic, and that as well as creating and storing posts in MongoDB, we would need to be

able to read and update. I therefore expanded this file initially to include functions to add a comment, as well as like and dislike functions. After searching online, I found a template logic for a likes and dislikes counter which I adapted to fit Piazza[1] (which along with the comment function expanded to become more complicated when the expiry function was added).

Expiration:

Trying to solve the problem of expiration was trickier, and required the implementation of a function that would check the status of the tweet each minute, refreshing. I found that cron would be able to schedule a job to check for expired posts automatically[2]. This new functionality required an update to the logic for the posts, disabling comments and likes and updating the status once an expiry time (set by the user at the time of posting) is reached.

Database Design:

To accommodate the wider functionality of the Piazza API, the database schema was expanded to store more complex data. The Post model now includes fields such as title, text, topic, likes, dislikes, comments, and expiration time, along with a status field that registers whether a post is live or expired. A User model was added to store registration details such as username, email, password, and registration date. The MongoDB database was connected using the dotenv package for secure storage of credentials.

Post schema (Example):

```
const postSchema = mongoose.Schema({
    user: { type: String, required: true },
    title: { type: String, required: true },
    text: { type: String, required: true },
    topic: { type: String, required: true },
    likes: { type: Number, default: 0 },
    dislikes: { type: Number, default: 0 },
    comments: [{ user: String, text: String, date: Date }],
    expirationTime: { type: Date, required: true },
    status: { type: String, default: 'Live' },
});
```

[1] Pusher, "How to build a real-time likes counter with Node.js," *Pusher Tutorials*, Aug. 21, 2019. [Online]. Available: https://pusher.com/tutorials/realtime-likes-nodejs/#introduction. [Accessed: Dec. 9, 2024].
[2] DigitalOcean, "Node.js Cron Jobs by Examples," *DigitalOcean Tutorials*, Sep. 8, 2021. [Online]. Available: https://www.digitalocean.com/community/tutorials/nodejs-cron-jobs-by-examples. [Accessed: Dec. 9, 2024].

Development of Authorisation functionality:

I next needed to develop the functions in line with the requirements, following OAuth protocol[3], of which a basic starting point was developed in lab 3, which I took as the foundation. This introduced the use of JSON Web Tokens (JWT), generated by a secret key and sent to the client to authorise certain interactions. This meant adding the file verifyToken.js to the root file, and creating a new folder titled validations which stored the logic governing constraints on input such as username, password, comments and registration. The verifyToken middleware was implemented to decode the token and attach the user's details to the request object, enabling secure access control across the API endpoints. I had to install the modules joi, bcryptjs and jsonwebtoken to respectively govern the validations, encrypt the passwords in the database, and issue JWTs. For implementation, I referenced resources such as [4] and [5].

Services:

I have included a short summary of each of the API endpoints and their functionality.

POST /api/user/register:

After the homepage (on port 3000), the first page a new user would land on is the registration page. This allows new piazza users to create login details using a name, email address and password. Credentials are validated against the constraints mentioned in validations.js

POST /api/user/login:

This is required to check user's details against the database of registered users. If the login is successful, a JWT will be issued, which can be attached to the header of a request sent via Postman, ensuring secure interactions with the API.

POST /posts:

There needed to be a functionality to (most importantly) send posts. The validations for this are specified as a title, text, one of four topics and a duration after which the post expires.

---

[3] J. D., "An Introduction to OAuth 2," *DigitalOcean*, Sep. 26, 2023. [Online]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2. [Accessed: Dec. 9, 2024].

[4] M. Agrawal, "Authentication Using JSON Web Token (JWT)," *GeeksforGeeks*, Feb. 5, 2023. [Online]. Available: https://www.geeksforgeeks.org/authentication-using-json-web-token-jwt/. [Accessed: Dec. 9, 2024].

[5] J. Pathak, "How to Build Authentication API with JWT Token in Node.js," *dev.to*, Apr. 8, 2023. [Online]. Available: https://dev.to/jainpathak/how-to-build-authentication-api-with-jwt-token-in-nodejs-2j7b. [Accessed: Dec. 9, 2024].

GET /posts:

This endpoint allowed users to browse (read) posts on the piazza forum. This included the necessary requirements to filter by topic, as well as some logic to allow a search by /top posts, which would show the post with the most interactions based on the data for total interactions. JWT must be attached to GET request header to ensure only authenticated users can view the forum.

POST /posts/:id/comment:

Authenticated users can add comments to a specific post by including the post ID in the endpoint. These record the user's name and a timestamp, and the function prevents expired posts from accepting comments.

POST /posts/:id/like: and POST /posts/:id/dislike:

Allows authenticated users to like or dislike a post. The logic had to include functionality to prevent users from liking or disliking their own posts, and for incrementing the likes, dislikes, and total interactions counter.

Conclusion:

The Piazza API can be considered RESTful because it provides stateless, client-server communication using HTTP methods and standardised URL structures[6]. Resources are accessed via unique endpoints, and it is stateless because each request contains all the necessary information for it to be processed by the server. Furthermore, responses are received in the JSON format, ensuring consistency and scalability.

[6] "REST API Tutorial," RESTful API, 2023. [Online]. Available: https://restfulapi.net/. [Accessed: 08-Dec-2024].

Testing the application

I did originally begin writing pytests for this, asserting whether the code returned by the request matched a good request ("200 or 201 OK"). This proved particularly difficult to emulate the use cases of the app however – though you'll see I have included the first 4 -- so I ultimately tested the endpoint responses using Postman and have included the responses below:

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

http://localhost:3000/api/user/register

POST http://localhost:3000/api/user/register

Params Authorization Headers (8) Body Scripts Settings

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

```
{
    "username": "Olga",
    "email": "olga@example.com",
    "password": "password123"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "username": "Olga",
    "email": "olga@example.com",
    "password": "$2a$05$dz9UPNvrQVPIIr0bptcJV.6/IBAzVkO.ksNhLfbArDVeo4RrSPdPm",
    "_id": "6754a2a20553039ece7e8e0d",
    "date": "2024-12-07T19:31:46.308Z",
    "__v": 0
}
```

http://localhost:3000/api/user/register

POST http://localhost:3000/api/user/register

Params Authorization Headers (8) Body Scripts Settings

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

```
{
    "username": "Nick",
    "email": "nick@example.com",
    "password": "password123"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "username": "Nick",
    "email": "nick@example.com",
    "password": "$2a$05$aVmx50XIc3Xit0qsBcXyHOKpB0BLOrtQH3deyQbHz9pdC6MfqGxR6",
    "_id": "6754a2da0553039ece7e8e11",
    "date": "2024-12-07T19:32:42.177Z",
    "__v": 0
}
```

http://localhost:3000/api/user/register

POST http://localhost:3000/api/user/register

Params Authorization Headers (8) Body Scripts Settings

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

```
{
    "username": "Mary",
    "email": "mary@example.com",
    "password": "password123"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "username": "Mary",
    "email": "mary@example.com",
    "password": "$2a$05$GSBAfx0QSBiImZ73hGA0l.NjgzK54lRVh2YQTYmLoeRK.kDBmJDle",
    "_id": "6754a2dd0553039ece7e8e14",
    "date": "2024-12-07T19:32:45.748Z",
    "__v": 0
}
```


http://localhost:3000/api/user/register

POST http://localhost:3000/api/user/register

Params Authorization Headers (8) Body Scripts Settings

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

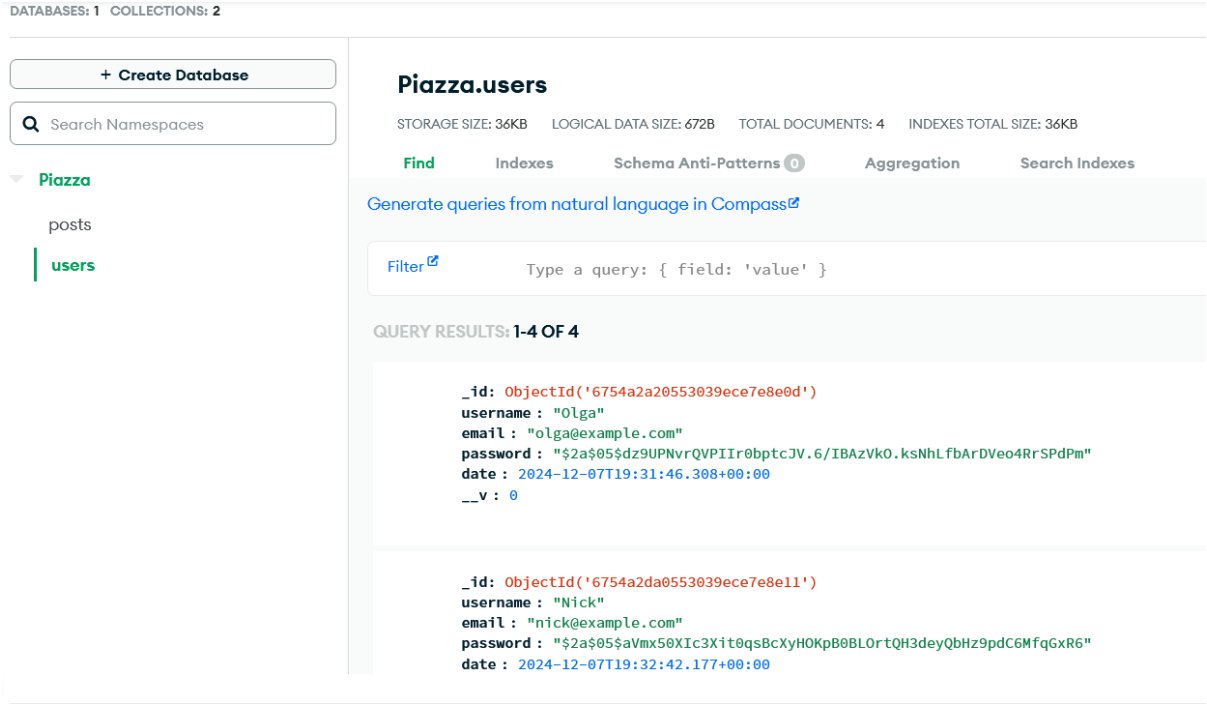
```
{
    "username": "Nestor",
    "email": "nestor@example.com",
    "password": "password123"
}
```

Body Cookies Headers (7) Test Results

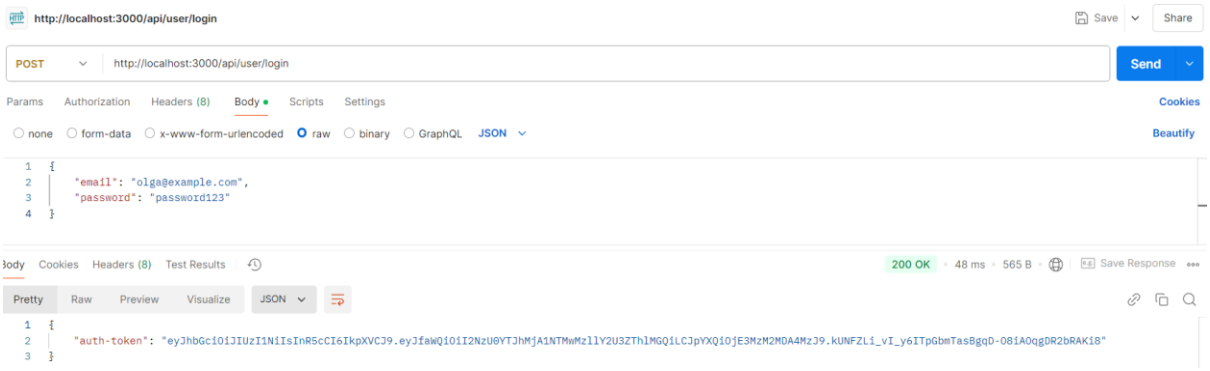
Pretty Raw Preview Visualize JSON

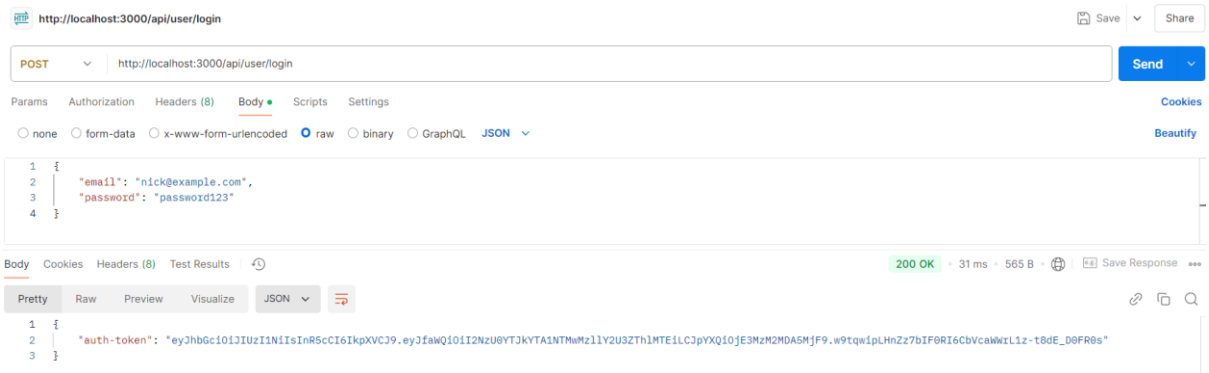

```
{
    "username": "Nestor",
    "email": "nestor@example.com",
    "password": "$2a$05$az27AQrPHZKIzsFdvaq6Hupy8yE1/3bfmNcff4KyLPp9ByUrGsSW.",
    "_id": "6754a53a0553039ece7e8e21",
    "date": "2024-12-07T19:42:50.964Z",
    "__v": 0
}
```


Registered users in MongoDB:



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





http://localhost:3000/api/user/login

POST http://localhost:3000/api/user/login

Send

Params Authorization Headers (8) Body Scripts Settings

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

```
{
    "email": "mary@example.com",
    "password": "password123"
}
```

Body Cookies Headers (8) Test Results 200 OK 26 ms 565 B

Pretty Raw Preview Visualize JSON


```
{
    "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJkZDA1NTMwMzllY2U3ZThlMTQiLCJpYXQiOjE3MzM2MDA5NDh9.YbwBLt9-ikw3YEBc83dSyF2iW9XgGmsX_xRuHzJSnMg"
}
```


http://localhost:3000/api/user/login

POST http://localhost:3000/api/user/login

Send

Params Authorization Headers (8) Body Scripts Settings

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

```
{
    "email": "nestor@example.com",
    "password": "password123"
}
```

Body Cookies Headers (8) Test Results 200 OK 27 ms 565 B

Pretty Raw Preview Visualize JSON


```
{
    "auth-token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTUzYTA1NTMwMzllY2U3ZThlMjEiLCJpYXQiOjE3MzM2MDA5NzR9.xAfLt_CzFiIers_hxSxKWiXKh7MAf8XefB4q8HcCzQo"
}
```


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

http://localhost:3000/posts

POST http://localhost:3000/posts

Send

Params Authorization Headers (8) Body Scripts Settings

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

```
{
    "title": "Frank Grimes",
    "text": "I don't need an auth-token, because I'm Olga Simpson!.",
    "topic": "Tech",
    "duration": 120
}
```

Body Cookies Headers (7) Test Results 401 Unauthorized 6 ms 272 B

Pretty Raw Preview Visualize JSON

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

TC 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).

Using auth-token to login:

http://localhost:3000/posts

POST http://localhost:3000/posts

Send

Params Authorization Headers (9) Body Scripts Settings

Headers 8 hidden

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJhMjA1NT...	
Key	Value	Description

Sending data to post endpoint:

http://localhost:3000/posts

Save | Share

POST http://localhost:3000/posts Send

Params | Authorization | Headers (9) | Body | Scripts | Settings | Cookies

Headers 8 hidden

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJhMjA1NT...	
Key	Value	Description

Body | Cookies | Headers (7) | Test Results — 201 Created · 57 ms · 576 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
{
    "user": "Olga",
    "title": "Cloud Computing",
    "text": "Stelios's Cloud Computing Module is great, I really loved that coursework!",
    "topic": "Tech",
    "expirationTime": "2024-12-07T21:30:03.149Z",
    "status": "Live",
    "likes": 0,
    "dislikes": 0,
    "totalInteractions": 0,
    "_id": "6754bd2f6d3928934568a3a0",
    "comments": [],
    "date": "2024-12-07T21:25:03.152Z",
    "__v": 0
}
```


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

http://localhost:3000/posts

Save | Share

POST http://localhost:3000/posts Send

Params | Authorization | Headers (9) | Body | Scripts | Settings | Cookies

Headers 8 hidden

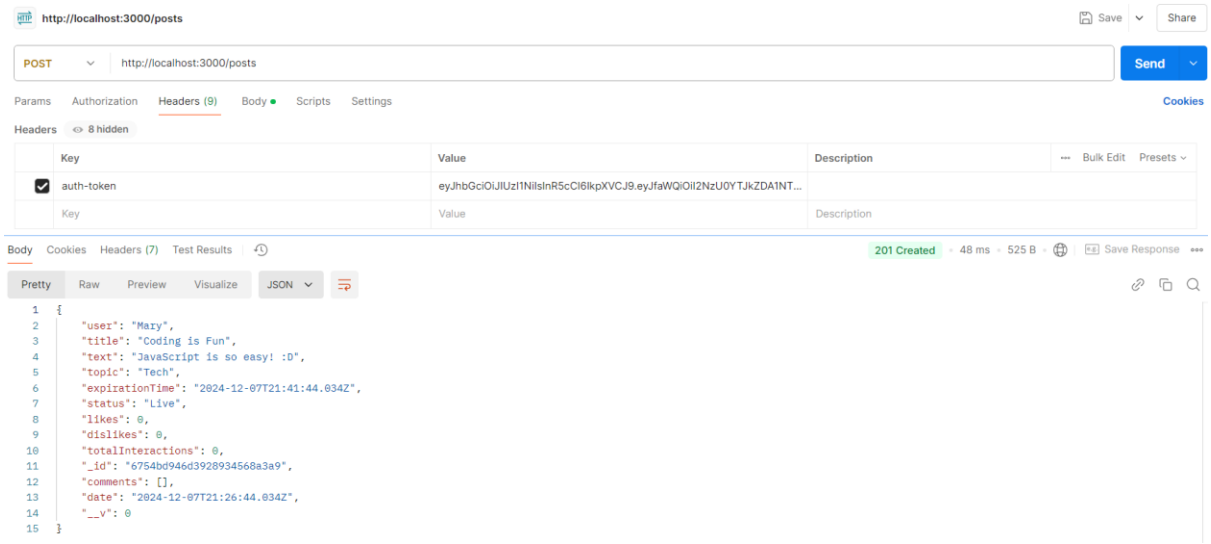
Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJkYTA1NT...	
Key	Value	Description

Body | Cookies | Headers (7) | Test Results — 201 Created · 74 ms · 531 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

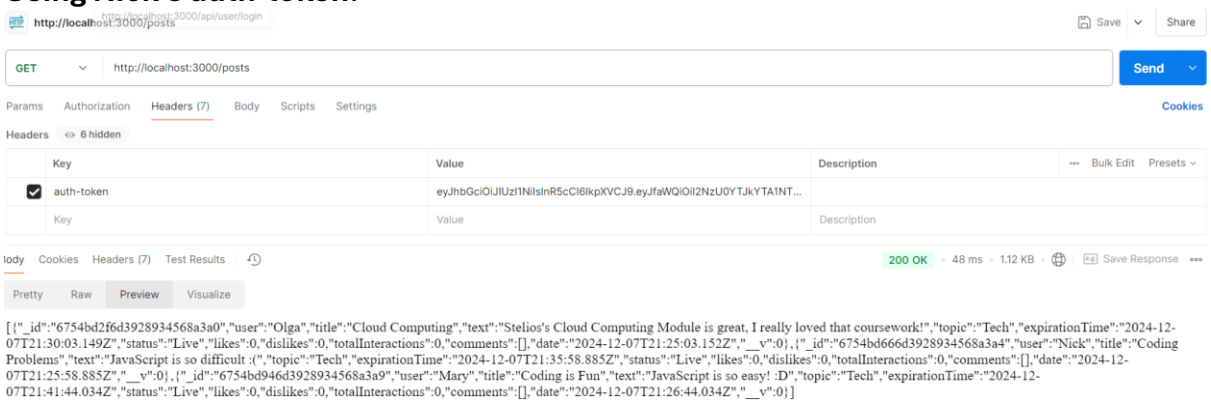

```
{
    "user": "Nick",
    "title": "Coding Problems",
    "text": "JavaScript is so difficult :(",
    "topic": "Tech",
    "expirationTime": "2024-12-07T21:35:58.885Z",
    "status": "Live",
    "likes": 0,
    "dislikes": 0,
    "totalInteractions": 0,
    "_id": "6754bd666d3928934568a3a4",
    "comments": [],
    "date": "2024-12-07T21:25:58.885Z",
    "__v": 0
}
```


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

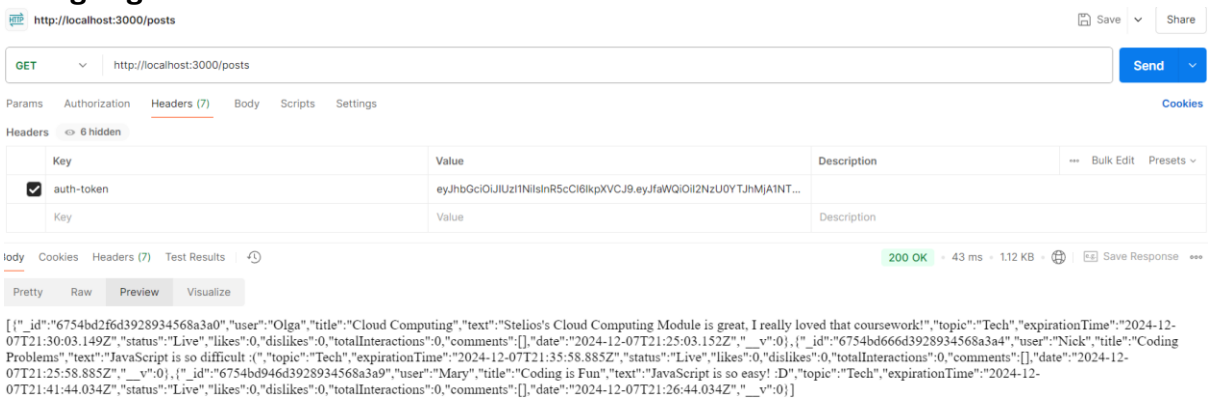


TC 7. Nick and Olga browse all the available posts in the Tech topic; three posts should have zero likes, zero dislikes, and no comments.

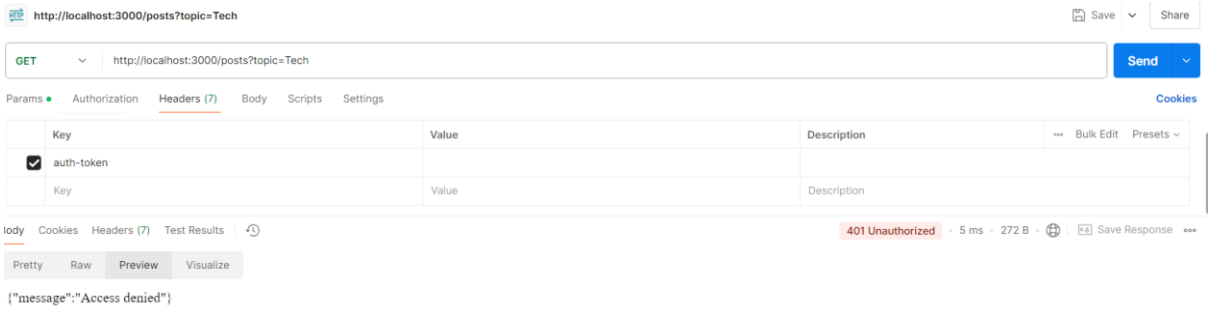
Using Nick's auth-token:



Using Olga's auth-token:

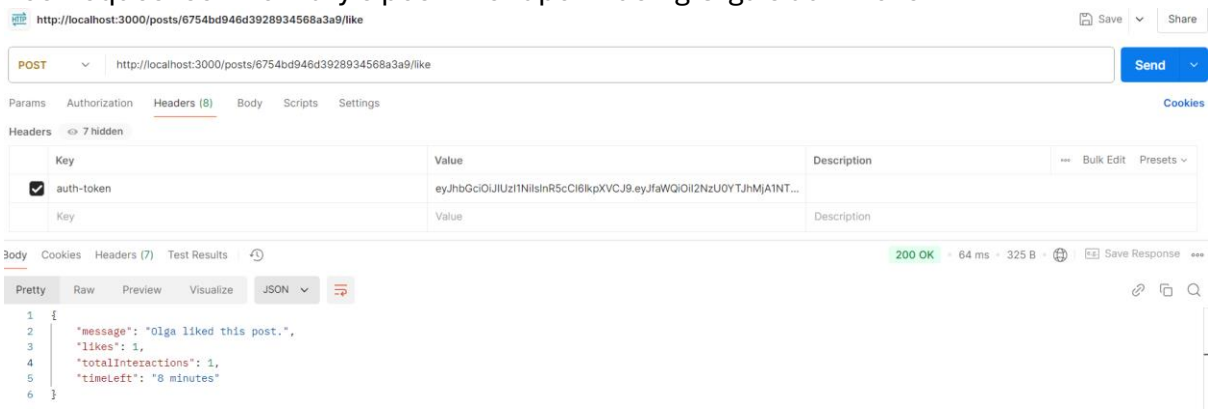


Trying to view posts without auth-token:

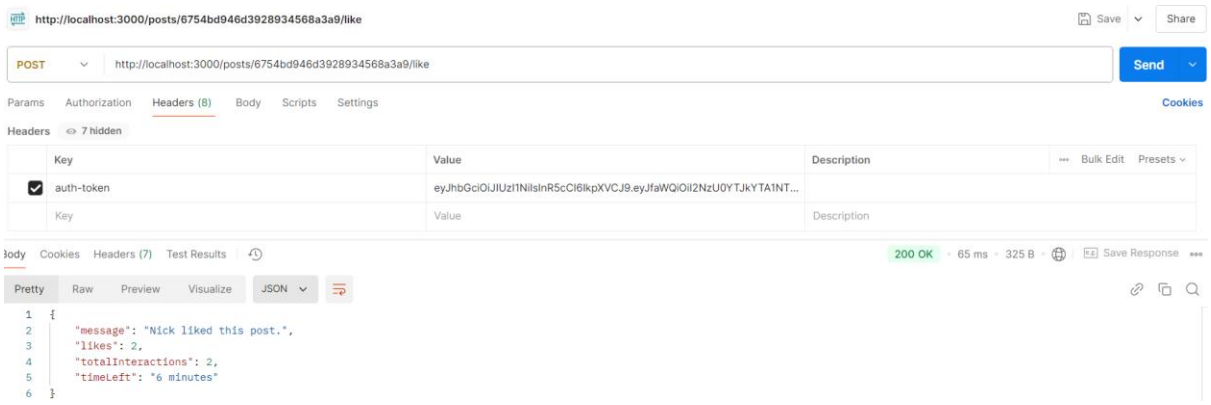


TC 8. Nick and Olga "like" Mary's post on the Tech topic.

Post request sent to Mary's post ID endpoint using Olga's auth-token:



Post request sent to Mary's post ID endpoint using Nick's auth-token:



/posts shows updated likes and interactions:

http://localhost:3000/posts

GET http://localhost:3000/posts

Send

Headers (7) | Key: auth-token | Value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJkYTA1NT...

200 OK · 42 ms · 1.12 KB


```
28        "comments": [],
29        "date": "2024-12-07T21:25:58.885Z",
30        "__v": 0
31    },
32    {
33        "_id": "6754bd946d3928934568a3a9",
34        "user": "Mary",
35        "title": "Coding is Fun",
36        "text": "JavaScript is so easy! :D",
37        "topic": "Tech",
38        "expirationTime": "2024-12-07T21:41:44.034Z",
39        "status": "Live",
40        "likes": 2,
41        "dislikes": 0,
42        "totalInteractions": 2,
43        "comments": [],
44        "date": "2024-12-07T21:26:44.034Z",
45        "__v": 0
46    }
```


TC 9. Nestor "likes" Nick's post and "dislikes" Mary's on the Tech topic.

Nestor sends a "like" request to Nick's post endpoint:

http://localhost:3000/posts/6754c1326d3928934568a3d0/like

POST http://localhost:3000/posts/6754c1326d3928934568a3d0/like

Send

Headers (8) | Key: auth-token | Value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTUzYTA1NT...

200 OK · 67 ms · 329 B


```
1  {
2      "message": "Nestor liked this post.",
3      "likes": 1,
4      "totalInteractions": 1,
5      "timeLeft": "117 minutes"
6  }
```


Nestor sends a "dislike" request to Mary's post endpoint:

http://localhost:3000/posts/6754c1326d3928934568a3d0/dislike

POST http://localhost:3000/posts/6754c1326d3928934568a3d0/dislike

Send

Headers (8) | Key: auth-token | Value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTUzYTA1NT...

200 OK · 61 ms · 336 B


```
1  {
2      "message": "Nestor disliked this post.",
3      "dislikes": 1,
4      "totalInteractions": 2,
5      "timeLeft": "110 minutes"
6  }
```


TC 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.

GET http://localhost:3000/posts Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Body Cookies Headers (7) Test Results 200 OK 43 ms 1.12 KB Save Response

Pretty Raw Preview Visualize JSON


```
18        "_id": "6754c1326d3928934568a3d0",
19        "user": "Nick",
20        "title": "Coding Problems",
21        "text": "JavaScript is so difficult :(",
22        "topic": "Tech",
23        "expirationTime": "2024-12-07T23:42:10.484Z",
24        "status": "Live",
25        "likes": 1,
26        "dislikes": 2,
27        "totalInteractions": 3,
28        "comments": [],
29        "date": "2024-12-07T21:42:10.485Z",
30        "__v": 0
31    },
32    {
33        "_id": "6754c2bc6d3928934568a3e1",
34        "user": "Mary",
35        "title": "Coding is Fun",
36        "text": "JavaScript is so easy! :D",
37        "topic": "Tech",
38        "expirationTime": "2024-12-08T00:48:44.783Z",
39        "status": "Live",
40        "likes": 2,
41        "dislikes": 0,
42        "totalInteractions": 2,
43        "comments": [],
```


(I accidentally left Nick's post ID in when I sent the dislike from Nestor. It still shows that the dislike request was successful, just for Nick's post instead – which incidentally was indeed worse!)

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

POST http://localhost:3000/posts/6754c2bc6d3928934568a3e1/like Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Headers 7 hidden

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJkZDA1NT...	
Key	Value	Description

Body Cookies Headers (7) Test Results 403 Forbidden 58 ms 296 B Save Response

Pretty Raw Preview Visualize JSON


```
1 {
2     "message": "Post owners cannot like their own posts."
3 }
```


TC 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).

POST http://localhost:3000/posts/6754c2bc6d3928934568a3e1/comment Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

Body Cookies Headers (7) Test Results 201 Created 65 ms 914 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "message": "Olga added a comment!",
    "comments": [
        {
            "user": "Nick",
            "text": "You're wrong, I hate JS!",
            "date": "2024-12-07T22:07:58.322Z",
            "_id": "6754c73e9edf56b94fe0ec42"
        },
        {
            "user": "Olga",
            "text": "Haha good point Mary! JavaScript is easy if you study hard.",
            "date": "2024-12-07T22:10:13.869Z",
            "_id": "6754c7c59edf56b94fe0ec4b"
        },
        {
            "user": "Nick",
            "text": "I wish I'd studied harder so I could use JS :'(",
            "date": "2024-12-07T22:11:15.362Z",
            "_id": "6754c8039edf56b94fe0ec52"
        },
        {
            "user": "Olga",
            "text": "Coding isn't for everybody, perhaps you should have studied something else!",
```


TC 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.

GET http://localhost:3000/posts Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Body Cookies Headers (7) Test Results 200 OK 41 ms 1.69 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "_id": "6754c2bc6d3928934568a3e1",
    "user": "Mary",
    "title": "Coding is Fun",
    "text": "JavaScript is so easy! :D",
    "topic": "Tech",
    "expirationTime": "2024-12-08T00:48:44.783Z",
    "status": "Live",
    "likes": 2,
    "dislikes": 0,
    "totalInteractions": 6,
    "comments": [
        {
            "user": "Nick",
            "text": "You're wrong, I hate JS!",
            "date": "2024-12-07T22:07:58.322Z",
            "_id": "6754c73e9edf56b94fe0ec42"
        },
        {
            "user": "Olga",
            "text": "Haha good point Mary! JavaScript is easy if you study hard.",
            "date": "2024-12-07T22:10:13.869Z",
            "_id": "6754c7c59edf56b94fe0ec4b"
        },
        {
            "user": "Nick",
            "text": "I wish I'd studied harder so I could use JS :'(",
```


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

POST http://localhost:3000/posts Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify


```
{
    "title": "Healthy Living",
    "text": "If you want to be healthy, you should study something other than computers! Or even better -- don't study at all...",
    "topic": "Health",
    "duration": 15
}
```


Body Cookies Headers (7) Test Results 201 Created 49 ms 620 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "user": "Nestor",
    "title": "Healthy Living",
    "text": "If you want to be healthy, you should study something other than computers! Or even better -- don't study at all...",
    "topic": "Health",
    "expirationTime": "2024-12-07T22:35:05.743Z",
    "status": "Live",
    "likes": 0,
    "dislikes": 0,
    "totalInteractions": 0,
    "_id": "6754ca159edf56b94fe0ec72",
    "comments": [],
    "date": "2024-12-07T22:20:05.744Z",
    "__v": 0
}
```


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

GET http://localhost:3000/posts?topic=Health Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Headers 8 hidden

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJkZDA1NT...	
Key	Value	Description

Bulk Edit Presets

Body Cookies Headers (7) Test Results 200 OK 38 ms 617 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6754ca159edf56b94fe0ec72",
        "user": "Nestor",
        "title": "Healthy Living",
        "text": "If you want to be healthy, you should study something other than computers! Or even better -- don't study at all...",
        "topic": "Health",
        "expirationTime": "2024-12-07T22:35:05.743Z",
        "status": "Live",
        "likes": 0,
        "dislikes": 0,
        "totalInteractions": 0,
        "comments": [],
        "date": "2024-12-07T22:20:05.744Z",
        "__v": 0
    }
]
```


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

POST http://localhost:3000/posts/6754ca159edf56b94fe0ec72/comment Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify


```
{
    "text": "You're so right, my back hurts so much. I hope this degree is worth it!"
}
```


Body Cookies Headers (7) Test Results 201 Created 64 ms 500 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "message": "Mary added a comment!",
    "comments": [
        {
            "user": "Mary",
            "text": "You're so right, my back hurts so much. I hope this degree is worth it!",
            "date": "2024-12-07T22:22:14.215Z",
            "_id": "6754ca969edf56b94fe0ec7b"
        }
    ],
    "totalInteractions": 1,
    "timeLeft": "12 minutes"
}
```


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

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

GET http://localhost:3000/posts?topic=Health Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Headers 8 hidden

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTUzYTA1NT...	

Body Cookies Headers (7) Test Results 200 OK 59 ms 783 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "_id": "6754ca159edf56b94fe0ec72",
        "user": "Nestor",
        "title": "Healthy Living",
        "text": "If you want to be healthy, you should study something other than computers! Or even better -- don't study at all...",
        "topic": "Health",
        "expirationTime": "2024-12-07T22:35:05.743Z",
        "status": "Expired",
        "likes": 0,
        "dislikes": 0,
        "totalInteractions": 1,
        "comments": [
            {
                "user": "Mary",
                "text": "You're so right, my back hurts so much. I hope this degree is worth it!",
                "date": "2024-12-07T22:22:14.215Z",
                "_id": "6754ca969edf56b94fe0ec7b"
            }
        ],
        "date": "2024-12-07T22:20:05.744Z",
        "__v": 1
```


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

GET http://localhost:3000/posts?topic=Sports&status=Expired Send

Params Authorization Headers (9) Body Scripts Settings Cookies

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTJkYTA1NT...	Nick's auth-token
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK 62 ms 235 B Save Response

Pretty Raw Preview Visualize JSON


```
[]
```


TC 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 http://localhost:3000/posts/top?topic=Tech Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Headers 6 hidden

Key	Value	Description
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NzU0YTUzYTA1NT...	Nestor's auth-token

Body Cookies Headers (7) Test Results 200 OK 43 ms 1.07 KB Save Response

Pretty Raw Preview Visualize JSON


```
    "user": "Mary",
    "title": "Coding is Fun",
    "text": "JavaScript is so easy! :D",
    "topic": "Tech",
    "expirationTime": "2024-12-08T00:48:44.783Z",
    "status": "Live",
    "likes": 2,
    "dislikes": 0,
    "totalInteractions": 6,
    "comments": [
        {
            "user": "Nick",
            "text": "You're wrong, I hate JS!",
            "date": "2024-12-07T22:07:58.322Z",
            "_id": "6754c73e9edf56b94fe0ec42"
        },
        {
            "user": "Olga",
            "text": "Haha good point Mary! JavaScript is easy if you study hard.",
            "date": "2024-12-07T22:10:13.869Z",
            "_id": "6754c7c59edf56b94fe0ec4b"
```

Deploying the Piazza project into a VM using Docker

After installing docker and created a new user, I switched user into my new docker user:

```
jaybacon234@piazza-docker:~$ docker -- version
Client:
 Version:           24.0.7
 API version:       1.43
 Go version:        go1.21.1
 Git commit:        24.0.7-0ubuntu2~20.04.1
 Built:             Wed Mar 13 20:29:24 2024
 OS/Arch:           linux/amd64
 Context:           default
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:
 connect: permission denied
jaybacon234@piazza-docker:~$ ^C
jaybacon234@piazza-docker:~$ su - docker-user
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

docker-user@piazza-docker:~$ docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu2~20.04.1
docker-user@piazza-docker:~$
```

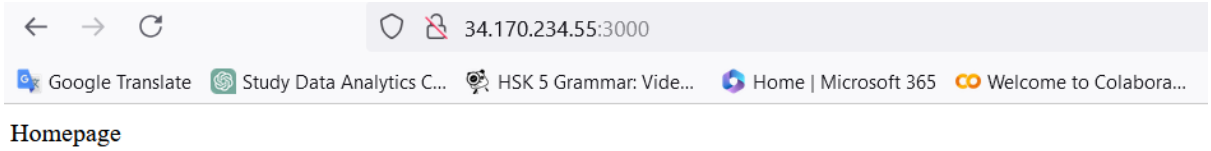
I cloned the Github repo into the new VM I set up for running a docker container in:

SSH-in-browser

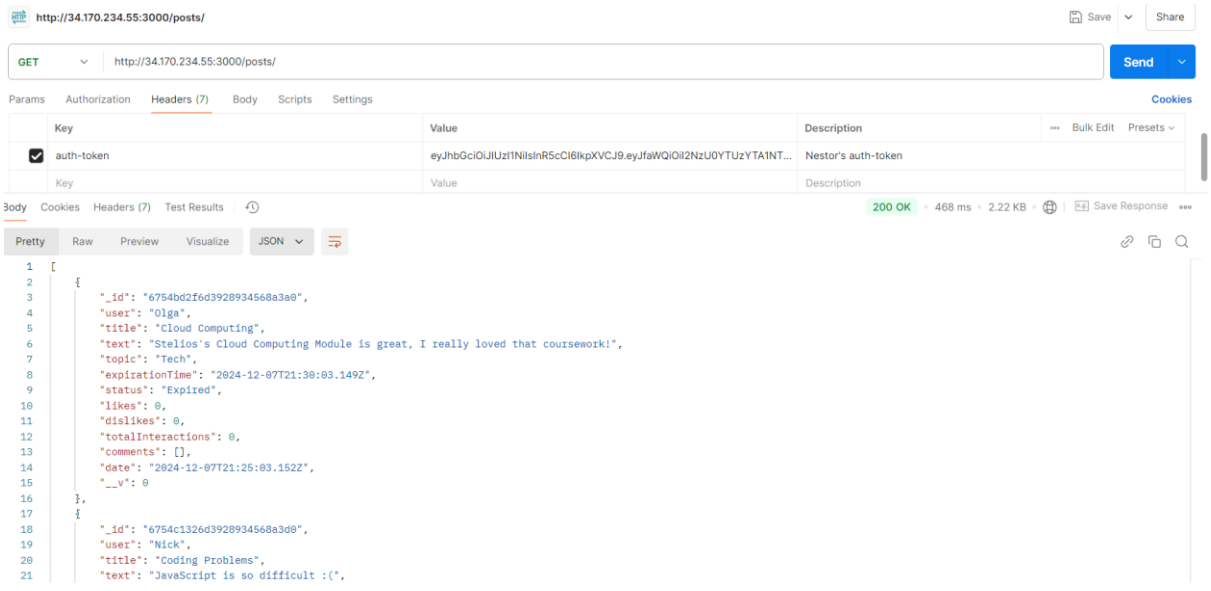
```
docker-user@piazza-docker:~$ cd piazza-coursework
-bash: cd: piazza-coursework: No such file or directory
docker-user@piazza-docker:~$ pwd
/home/docker-user
docker-user@piazza-docker:~$ ls
docker-user@piazza-docker:~$ git clone https://github.com/BayJacon/piazza-coursework.git
Cloning into 'piazza-coursework'...
remote: Enumerating objects: 2575, done.
remote: Counting objects: 100% (2575/2575), done.
remote: Compressing objects: 100% (2098/2098), done.
remote: Total 2575 (delta 388), reused 2564 (delta 377), pack-reused 0 (from 0)
Receiving objects: 100% (2575/2575), 3.56 MiB | 12.78 MiB/s, done.
Resolving deltas: 100% (388/388), done.
docker-user@piazza-docker:~$ cd piazza-coursework
docker-user@piazza-docker:~/piazza-coursework$
```

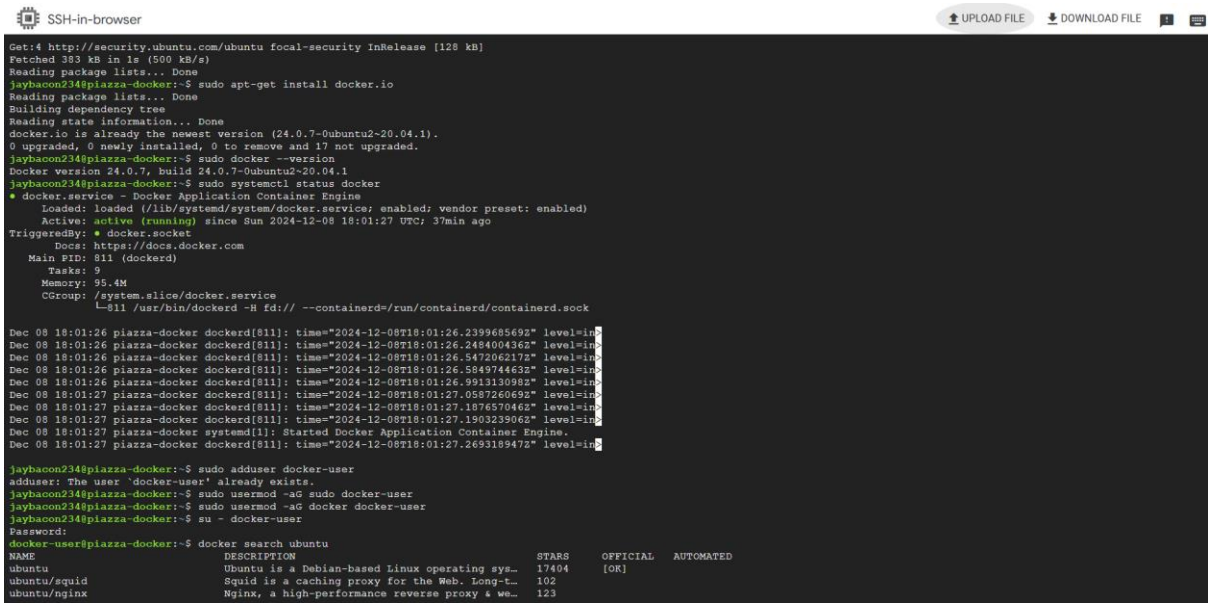
```
jaybacon234@piazza-docker:~$ git clone --branch master https://BayJacon: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@github.com/BayJacon/piazza-coursework.git
Cloning into 'piazza-coursework'...
fatal: Remote branch master not found in upstream origin
jaybacon234@piazza-docker:~$ git clone --branch main https://BayJacon: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@github.com/BayJacon/piazza-coursework.git
Cloning into 'piazza-coursework'...
remote: Enumerating objects: 2575, done.
remote: Counting objects: 100% (2575/2575), done.
remote: Compressing objects: 100% (2098/2098), done.
remote: Total 2575 (delta 388), reused 2564 (delta 377), pack-reused 0 (from 0)
Receiving objects: 100% (2575/2575), 3.56 MiB | 13.30 MiB/s, done.
Resolving deltas: 100% (388/388), done.
jaybacon234@piazza-docker:~$ cd piazza-coursework
jaybacon234@piazza-docker:~/piazza-coursework$ git pull
Already up to date.
jaybacon234@piazza-docker:~/piazza-coursework$ pwd
/home/jaybacon234/piazza-coursework
jaybacon234@piazza-docker:~/piazza-coursework$ ls -ld /home/jaybacon234/piazza-coursework
drwxrwxr-x 7 jaybacon234 jaybacon234 4096 Dec  8 14:19 /home/jaybacon234/piazza-coursework
jaybacon234@piazza-docker:~/piazza-coursework$ pico Dockerfile
jaybacon234@piazza-docker:~/piazza-coursework$ sudo docker image build -t piazza-app-image: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  20.88MB
Step 1/6 : FROM alpine
latest: Pulling from library/alpine
38a8310d387e: Pull complete
```

The docker image is here and running and should now be visible in my browser using the VM's external IP:



As shown above I can see that the docker image is running in the VM and I'm able to access the homepage in the browser. As shown below, sending a GET request to the external IP of the Docker VM using the auth-token for Olga returns all the posts:



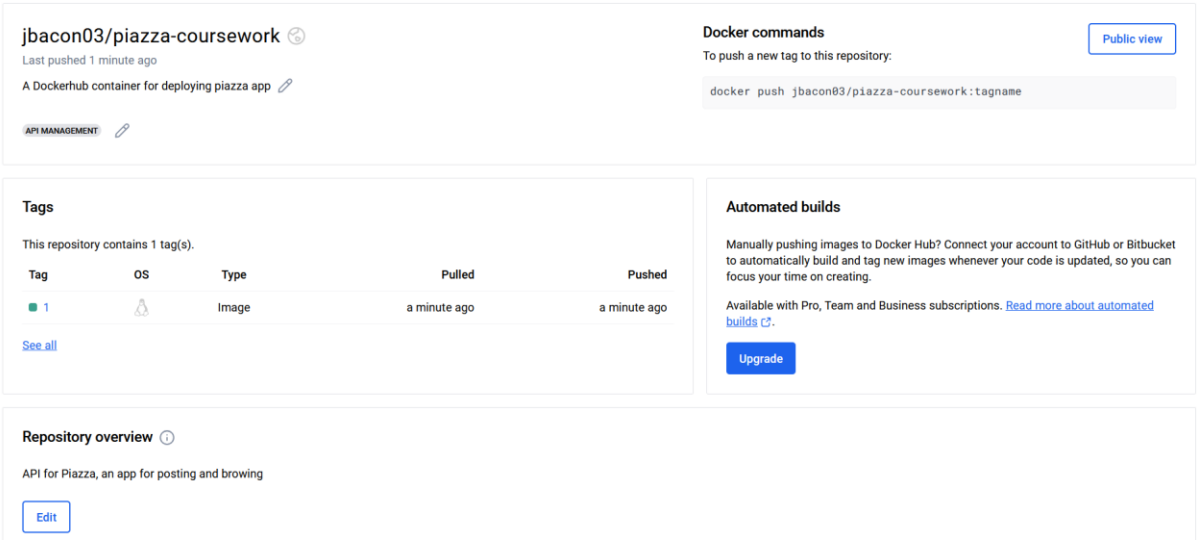


```
docker-user@piazza-docker:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
de44b265507a: Pull complete
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
docker-user@piazza-docker:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED       SIZE
piazza-app-image    1         6557cb7e387f   4 hours ago   102MB
alpine              latest    4048db5d3672   3 days ago    7.84MB
ubuntu              latest    b1d9df8ab815   2 weeks ago   78.1MB
docker-user@piazza-docker:~$ docker ps -all
CONTAINER ID   IMAGE                 COMMAND            CREATED       STATUS                     PORTS     NAMES
4279a9bfa139   piazza-app-image:1    "node ./app.js"    4 hours ago   Exited (137) 4 hours ago             piazza-container
docker-user@piazza-docker:~$ cd piazza-coursework
docker-user@piazza-docker:~/piazza-coursework$ pico Dockerfile
docker-user@piazza-docker:~/piazza-coursework$ docker image build -t piazza-image: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  20.88MB
Step 1/6 : FROM alpine
 ---> 4048db5d3672
Step 2/6 : RUN apk add --update nodejs npm
 ---> Using cache
 ---> f665a70ca23c
Step 3/6 : COPY . /src
 ---> 5be179623ca5
Step 4/6 : WORKDIR /src
 ---> Running in cbc42fa492f2
Removing intermediate container cbc42fa492f2
 ---> 879a924dd414
Step 5/6 : EXPOSE 3000
 ---> Running in 4d54a48bdc04
Removing intermediate container 4d54a48bdc04
 ---> 1766546a4f18
Step 6/6 : ENTRYPOINT ["node", "./app.js"]
 ---> Running in 2f68d50cc016
Removing intermediate container 2f68d50cc016
 ---> 06f92646a6fe
Successfully built 06f92646a6fe
Successfully tagged piazza-image:1
docker-user@piazza-docker:~/piazza-coursework$
```

After logging in, I managed to push it to DockerHub:

```
docker-user@piazza-docker:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED       SIZE
piazza-app-image    1         6557cb7e387f   4 hours ago   102MB
alpine              latest    4048db5d3672   3 days ago    7.84MB
ubuntu              latest    b1d9df8ab815   2 weeks ago   78.1MB
docker-user@piazza-docker:~$ docker ps -all
CONTAINER ID   IMAGE                 COMMAND            CREATED       STATUS                     PORTS     NAMES
4279a9bfa139   piazza-app-image:1    "node ./app.js"    4 hours ago   Exited (137) 4 hours ago             piazza-container
docker-user@piazza-docker:~$ cd piazza-coursework
docker-user@piazza-docker:~/piazza-coursework$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
piazza-image        1         06f92646a6fe   8 minutes ago  102MB
piazza-app-image    1         6557cb7e387f   4 hours ago    102MB
alpine              latest    4048db5d3672   3 days ago     7.84MB
ubuntu              latest    b1d9df8ab815   2 weeks ago    78.1MB
docker-user@piazza-docker:~/piazza-coursework$ docker rmi piazza-app-image:1
Error response from daemon: conflict: unable to remove repository reference "piazza-app-image:1" (must force) - container 4279a9bfa139 is using its referenced image 6557cb7e387f
docker-user@piazza-docker:~/piazza-coursework$ docker tag piazza-image:1 jbacon03/piazza-coursework:1
docker-user@piazza-docker:~/piazza-coursework$ docker push jbacon03/piazza-coursework:1
The push refers to repository [docker.io/jbacon03/piazza-coursework]
f5773c477fa6: Pushed
2b49e6b71b85: Pushed
9a0181bd79cd: Mounted from library/alpine
1: digest: sha256:18bd3a820d8b7f415cdee3d868b0bfa0d87b79683ffcebe00439ce6494c59d4e size: 951
docker-user@piazza-docker:~/piazza-coursework$
```

And you can see it here in the repository I created:



and finally searching using docker search <username> shows it in the repository in my VM terminal:

```
Search Docker Hub for images
docker-user@piazza-docker:~/piazza-coursework$ docker search jbacon03
NAME                          DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
jbacon03/piazza-coursework    A Dockerhub container for deploying piazza a…   0
docker-user@piazza-docker:~/piazza-coursework$
```

Deploying the application in Kubernetes

After setting up my cluster configuration on GCP, I created my piazza-pod, and checked it was running.

```
gke-piazza-gke-cluster-default-pool-4b7906f8-t885   Ready    <none>   21m   v1.30.5-gke.1699000
gke-piazza-gke-cluster-default-pool-4b7906f8-zrjq   Ready    <none>   21m   v1.30.5-gke.1699000
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl cluster-info
Kubernetes control plane is running at https://35.188.50.159
GLBCDefaultBackend is running at https://35.188.50.159/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
KubeDNS is running at https://35.188.50.159/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://35.188.50.159/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl run piazza-pod --image=jbacon03/piazza-image:1
pod/piazza-pod created
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
piazza-pod   1/1     Running   0          12s
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl describe pods
Name:             piazza-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             gke-piazza-gke-cluster-default-pool-4b7906f8-zrjq/10.128.0.12
Start Time:       Sun, 08 Dec 2024 21:25:31 +0000
Labels:           run=piazza-pod
Annotations:      <none>
Status:           Running
IP:               10.64.0.5
IPs:
  IP:  10.64.0.5
Containers:
  piazza-pod:
    Container ID:   containerd://170efe8daf73ce6598ef1b44ab798ab32a7b2d23e9c19d84a6905342c61bf81b
    Image:          jbacon03/piazza-image:1
    Image ID:       docker.io/jbacon03/piazza-image@sha256:8b6ac7ec49b76fc912a0178a740b7979a5d60817cb2ec914bc81788e03c95b14
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sun, 08 Dec 2024 21:25:37 +0000
    Ready:          True
    Restart Count:  0
```

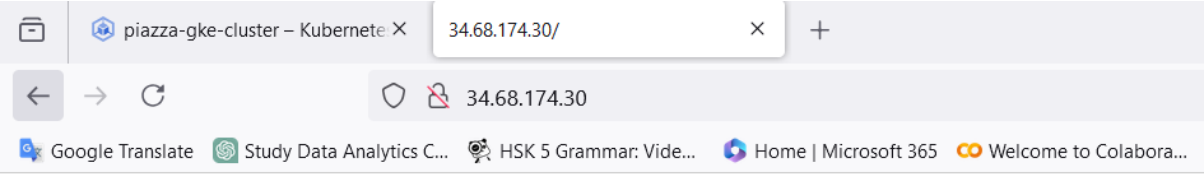
I created the file piazza-deployment with 5 replicas.

```
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ pico piazza-deployment.yaml
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ cat piazza-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: piazza-deployment
  labels:
    app: piazza
spec:
  replicas: 5
  selector:
    matchLabels:
      app: piazza
  template:
    metadata:
      labels:
        app: piazza
    spec:
      containers:
      - name: piazza-container
        image: jbacon03/piazza-image:1
        ports:
        - containerPort: 3000
        imagePullPolicy: Always
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$
```

After applying them, I can see that these are running. I then created and deployed the load balancer following the configuration as below:

```
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
piazza-deployment-777f67bf45-7mwdb      1/1     Running   0          2m44s
piazza-deployment-777f67bf45-8fjpw      1/1     Running   0          2m44s
piazza-deployment-777f67bf45-jqgmz      1/1     Running   0          2m44s
piazza-deployment-777f67bf45-k912c      1/1     Running   0          2m44s
piazza-deployment-777f67bf45-xxksk      1/1     Running   0          2m44s
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ pico piazza-service.yaml
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ cat piazza-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: piazza-service
  labels:
    app: piazza
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 3000
    protocol: TCP
  selector:
    app: piazza
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl apply -f piazza-service.yaml
service/piazza-service created
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl get services
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
kubernetes       ClusterIP      34.118.224.1     <none>          443/TCP        48m
piazza-service   LoadBalancer   34.118.230.113   34.68.174.30    80:31264/TCP   72s
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl get services
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
kubernetes       ClusterIP      34.118.224.1     <none>          443/TCP        51m
piazza-service   LoadBalancer   34.118.230.113   34.68.174.30    80:31264/TCP   3m47s
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$
```

Following the external IP in my browser shows my piazza app homepage:



Homepage

I started my deployment with 5 replicas, but I wanted to update the version and deploy more replicas so I edited the yaml configuration as required, and you can see below that I now have 15 replicas running.

```
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
piazza-deployment-777f67bf45-7mwdb      1/1     Running   0          29m
piazza-deployment-777f67bf45-8fjpw      1/1     Running   0          29m
piazza-deployment-777f67bf45-ffldd      1/1     Running   0          14s
piazza-deployment-777f67bf45-gj7rb      1/1     Running   0          13s
piazza-deployment-777f67bf45-jqgmz      1/1     Running   0          29m
piazza-deployment-777f67bf45-k912c      1/1     Running   0          29m
piazza-deployment-777f67bf45-kqqpt      1/1     Running   0          9m38s
piazza-deployment-777f67bf45-lg6x2      1/1     Running   0          9m37s
piazza-deployment-777f67bf45-m76lf      1/1     Running   0          13s
piazza-deployment-777f67bf45-pvd5g      1/1     Running   0          13s
piazza-deployment-777f67bf45-qcsdk      1/1     Running   0          9m38s
piazza-deployment-777f67bf45-rbzx7      1/1     Running   0          14s
piazza-deployment-777f67bf45-xn2fj      1/1     Running   0          14s
piazza-deployment-777f67bf45-xpqxp      1/1     Running   0          14s
piazza-deployment-777f67bf45-xxksk      1/1     Running   0          29m
jaybacon234@cloudshell:~ (cloud-computing-project-437518)$
```