Cloud Computing - Mini Project Report
Project - 5
April 2023

Submitted By:
Name: Srihari Adiga BS
SRN: PES1UG20CS691
6th Semester Section L
PES University

Short Description and Scope of the Project

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features. Docker and Kubernetes are almost synonymous to 'microservices' as they help package and manage the different components of a project/ application, thereby easing up the implementation of a microservices architecture.

In this project, we will work with Docker and Kubernetes to make an easily deployable and portable blogging web-app using Flask and MongoDB.

The microservices architecture will deploy a Kubernetes cluster with a mongo dB server pod fronted with a web admin interface and a pod to run the flask app.

Pre-Requisites:

1. Docker
2. Kubernetes

The scope of this project is to develop a blogging web application that uses Flask as the backend framework and MongoDB as the database. The application will be containerized using Docker and deployed using Kubernetes to make it easily deployable and portable across different environments.

The web application should allow users to create, edit, and delete blog posts, as well as view posts created by other users.

The end result of this project will be a fully functional, containerized web application that can be easily deployed and run on any system that supports Docker and Kubernetes.

Methodology

Task Breakdown:

1. MongoDB Server
    i. Using the mongo image publicly available on Docker Hub. Noting down the necessary environment variables to be configured.
    ii. Creating the Deployment for the mongo dB server under deployments.yaml. Configuring the ports and the environment variables correctly.
    iii. Environment variables such as username, password, etc. are sensitive information and are defined as a Secret. Creating a secret.yaml file to hold the sensitive information required by the mongo dB server.
    iv. Creating a Service for the mongo dB server under services.yaml.
2. Mongo-Express Web Service
    i. Use the mongo-express image.
    ii. Creating a configmap to store the mongo dB server url and using the configmap to configure the container with environment variables.
    iii. Creating a Deployment for the mongo-express service under deployments.yaml and configuring the necessary ports and environment variables (drawn from the secret and configmap).
    iv. Defining a service for the pod under services.yaml.
3. Flask WebApp
    i. Use the image created from the flask-app-image.dockerfile.
    ii. Creating a Deployment for the flask app under deployments.yaml.
    iii. Defining a Service for the pod in services.yaml.

Once all the microservices are up and running,

1. Inside the flask-app pod, write and run a python script to insert records into the mongodb database. Insert into: database = 'blog' and collection = 'posts'.

2. Run app.py inside the pod. Visit http://localhost:<port>/ to view the Blog App. The Home Page should display the records inserted into the database in the previous step.

Testing

1. Creation of Deployments and Services

```
E:\Studies\Sem-6\CC\Project\UE20CS351-Cloud-Computing-Problem-Statements\Pruthvi-version>kubectl apply -f "deployments.yaml"
deployment.apps/mongodb-deployment created
deployment.apps/mongo-express-deployment created
deployment.apps/flask-app created

E:\Studies\Sem-6\CC\Project\UE20CS351-Cloud-Computing-Problem-Statements\Pruthvi-version>kubectl apply -f "services.yaml"
service/mongodb-service created
service/mongo-express-service created
service/flask-app-service created
```

2. Verifying the creation of pods

```
E:\Studies\Sem-6\CC\Project\UE20CS351-Cloud-Computing-Problem-Statements\Pruthvi-version>kubectl get pods
NAME                                     READY   STATUS    RESTARTS        AGE
flask-app-86d9976fcb-rngfp               1/1     Running   0               52s
mongo-express-deployment-7c7c688fdb-npcmf   1/1     Running   2 (32s ago)     52s
mongodb-deployment-8489c98595-mj4w6      1/1     Running   0               52s
nginx-pes1ug20cs691-6847fcdf6c-6cnp9     1/1     Running   16 (5m56s ago)   53d
```

3. Verifying the creation of service

```
E:\Studies\Sem-6\CC\Project\UE20CS351-Cloud-Computing-Problem-Statements\Pruthvi-version>kubectl get services
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
flask-app-service       LoadBalancer   10.109.203.88    <pending>     5000:30000/TCP   40s
kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP          53d
mongo-express-service   LoadBalancer   10.102.142.140   <pending>     8081:30001/TCP   40s
mongodb-service         ClusterIP      10.106.38.158    <none>        27017/TCP        40s
nginx-pes1ug20cs691     NodePort       10.96.11.227     <none>        80:31744/TCP     53d
```

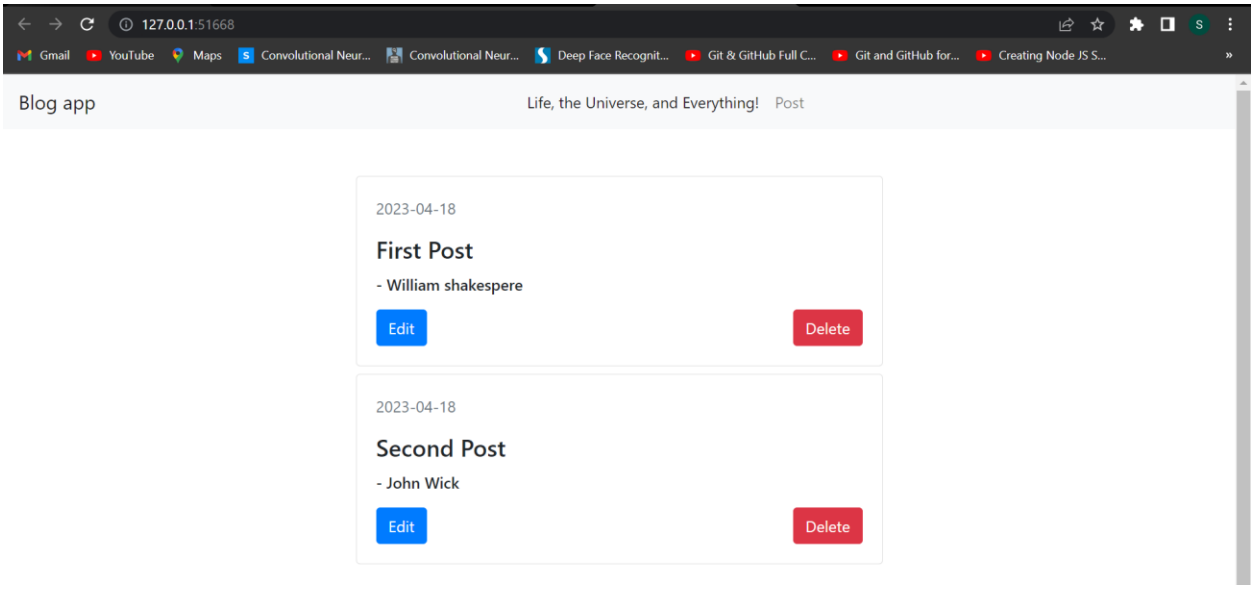
4. Starting flask-app service

```
E:\Studies\Sem-6\CC\Project\UE20CS351-Cloud-Computing-Problem-Statements\Pruthvi-version>minikube service flask-app-service
|-----------|-------------------|-------------|--------------------------|
| NAMESPACE |       NAME        | TARGET PORT |           URL            |
|-----------|-------------------|-------------|--------------------------|
| default   | flask-app-service |        5000 | http://192.168.49.2:30000 |
|-----------|-------------------|-------------|--------------------------|
* Starting tunnel for service flask-app-service.
|-----------|-------------------|-------------|------------------------|
| NAMESPACE |       NAME        | TARGET PORT |          URL           |
|-----------|-------------------|-------------|------------------------|
| default   | flask-app-service |             | http://127.0.0.1:51668 |
|-----------|-------------------|-------------|------------------------|
* Opening service default/flask-app-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

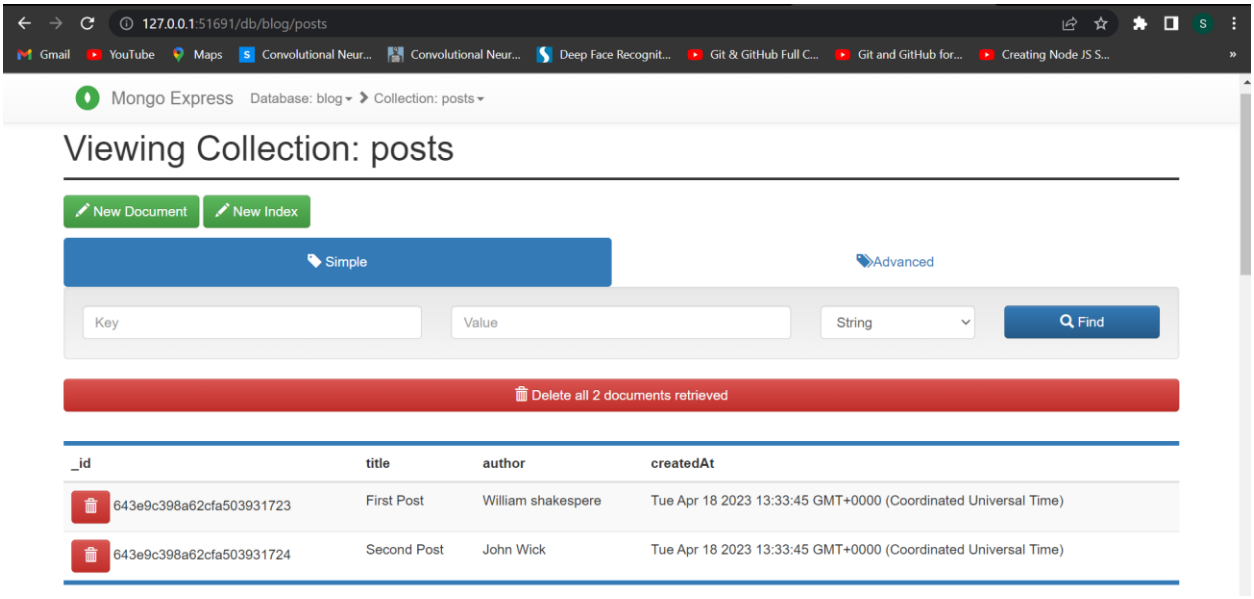
5. Starting mongo-express service

```
E:\Studies\Sem-6\CC\Project\UE20CS351-Cloud-Computing-Problem-Statements\Pruthvi-version>minikube service mongo-express-
service
|-----------|----------------------|--------------------|--------------------------|
| NAMESPACE |         NAME         |    TARGET PORT     |           URL            |
|-----------|----------------------|--------------------|--------------------------|
| default   | mongo-express-service | mongo-express/8081 | http://192.168.49.2:30001 |
|-----------|----------------------|--------------------|--------------------------|
* Starting tunnel for service mongo-express-service.
|-----------|----------------------|-------------|------------------------|
| NAMESPACE |         NAME         | TARGET PORT |          URL           |
|-----------|----------------------|-------------|------------------------|
| default   | mongo-express-service |             | http://127.0.0.1:51691 |
|-----------|----------------------|-------------|------------------------|
* Opening service default/mongo-express-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

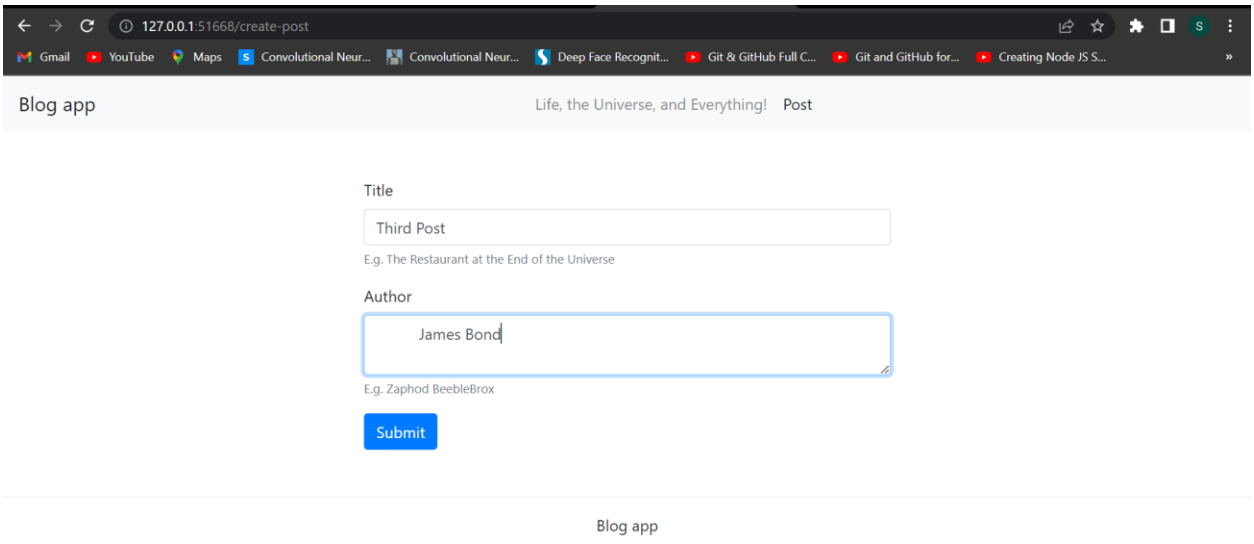
6. Records inserted into the database are loaded onto the home page



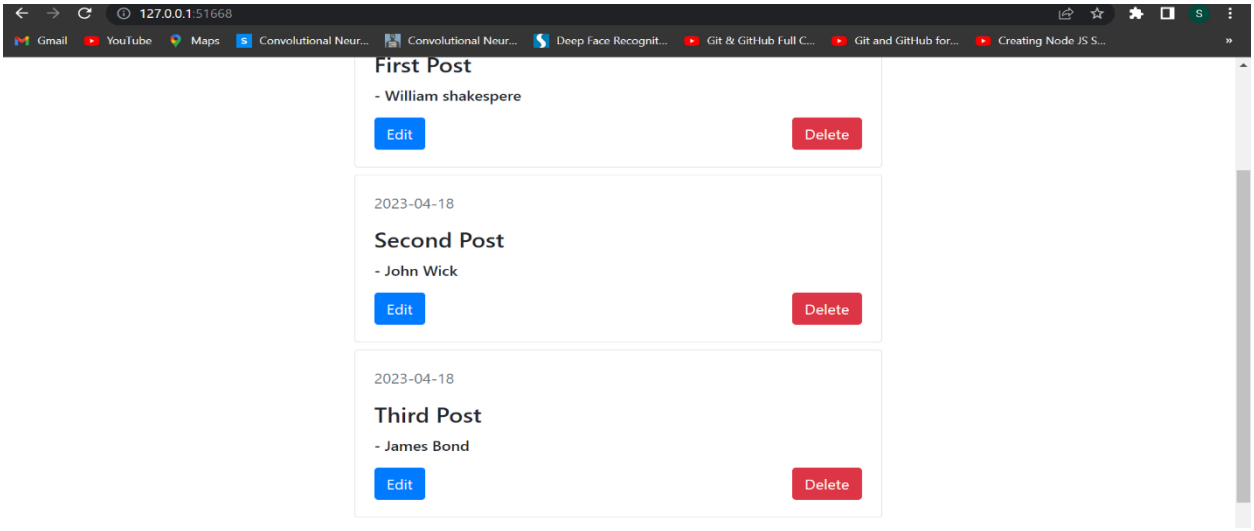
7. **Verifying the creation of records in mongodb through mongo-express interface**



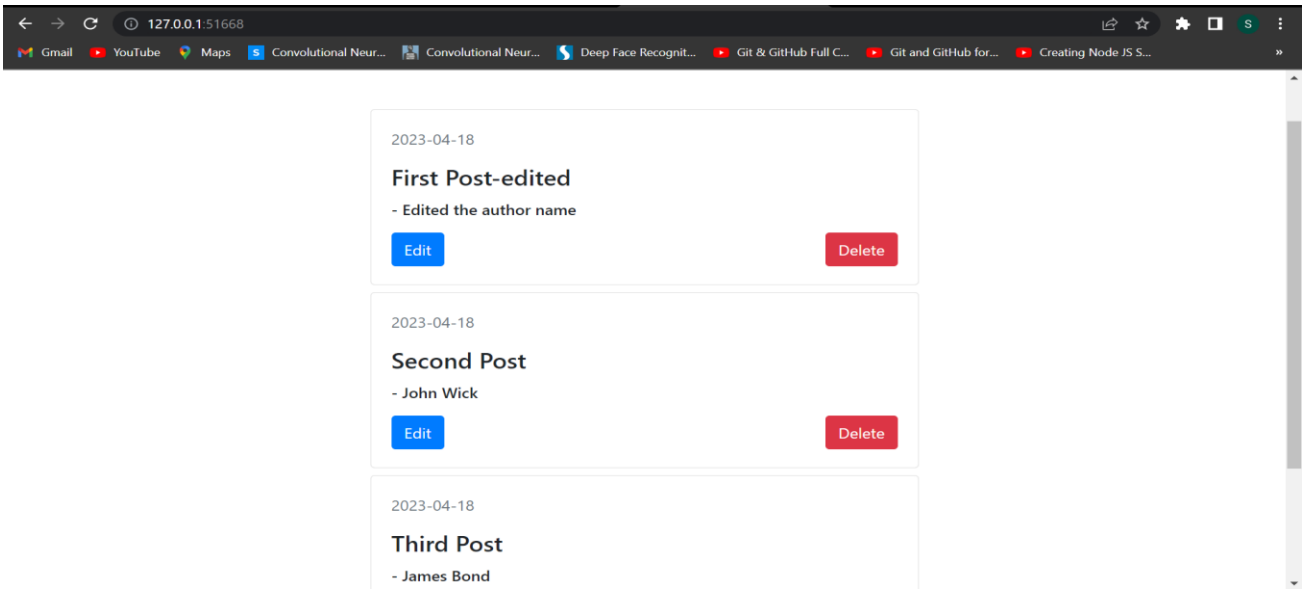
8. **Creation of new post**



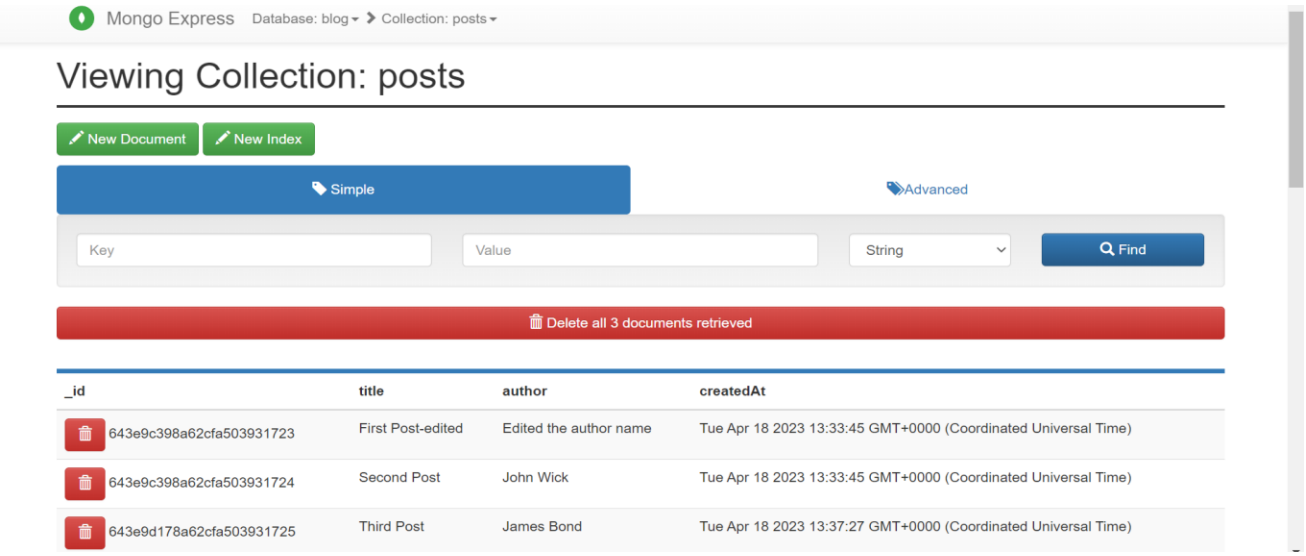
9.



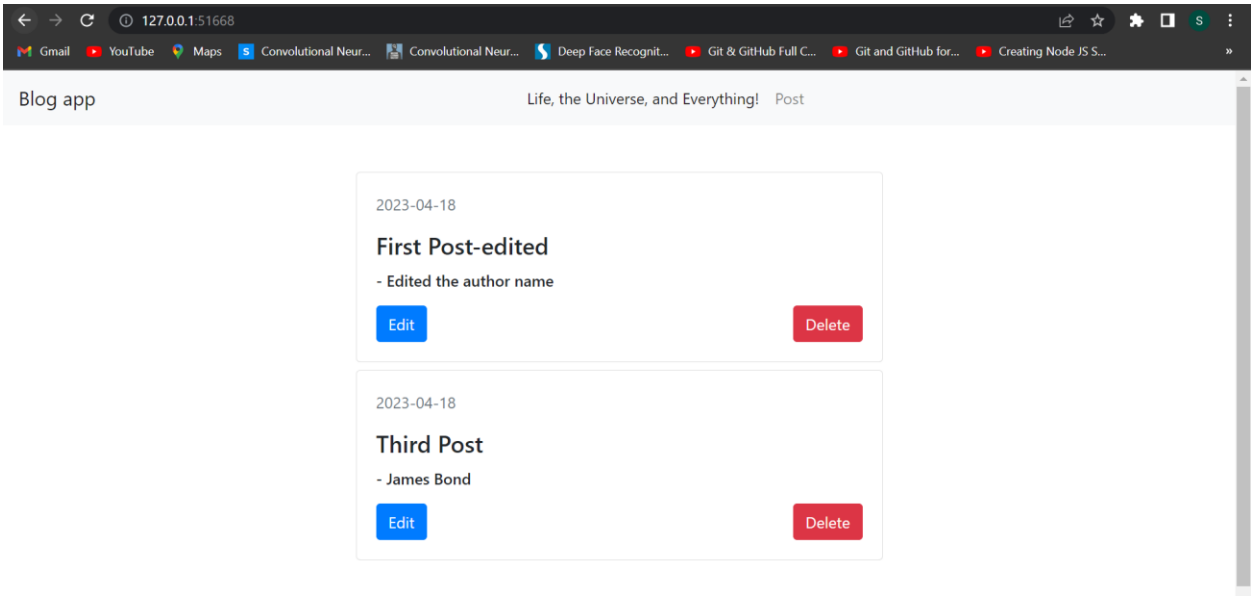
10. Editing the post



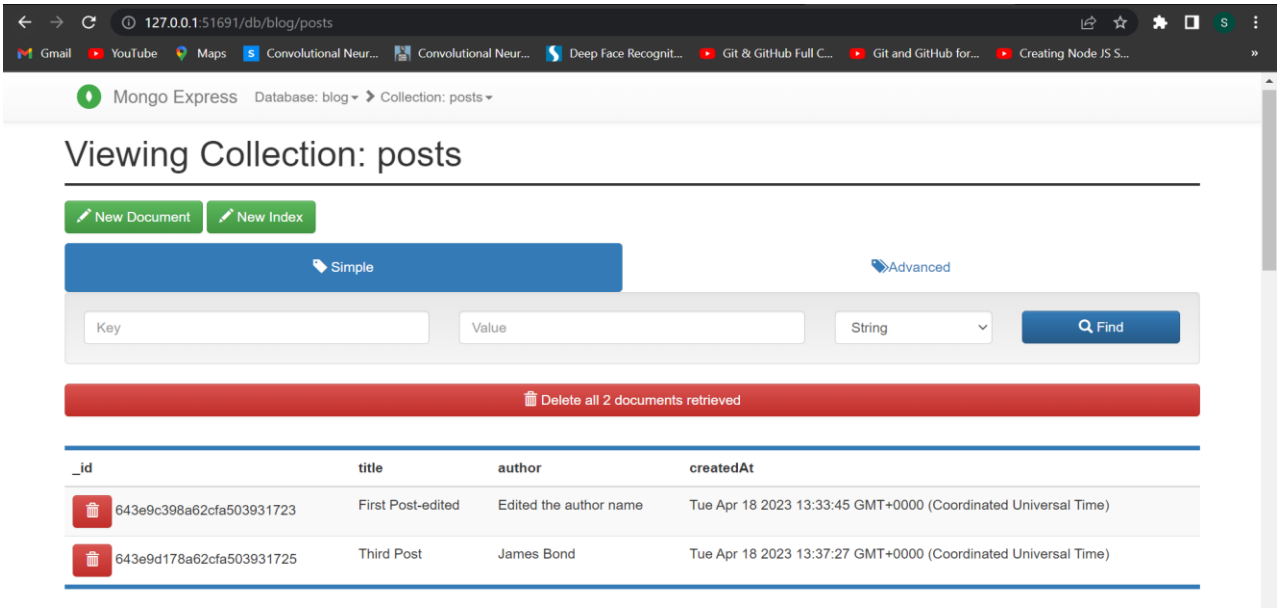
11. Verifying the edit operation in the mongodb database



12. Deleting the post



13. Verifying the delete operation in the mongodb database


127.0.0.1:51691/db/blog/posts
Gmail
YouTube
Maps
Convolutional Neur...
Convolutional Neur...
Deep Face Recognit...
Git & GitHub Full C...
Git and GitHub for...
Creating Node JS S...
Mongo Express
Database: blog
Collection: posts
Viewing Collection: posts
New Document
New Index
Simple
Advanced
Key
Value
String
Find
Delete all 2 documents retrieved
_id
title
author
createdAt
643e9c398a62cfa503931723
First Post-edited
Edited the author name
Tue Apr 18 2023 13:33:45 GMT+0000 (Coordinated Universal Time)
643e9d178a62cfa503931725
Third Post
James Bond
Tue Apr 18 2023 13:37:27 GMT+0000 (Coordinated Universal Time)

Results and Conclusion

In conclusion, using Docker and Kubernetes to create a portable and easily deployable blogging web-app using Flask and MongoDB can be a powerful solution for modern web development. By Dockerising the Flask app and deploying it in a Kubernetes cluster, you can easily scale the application and handle large amounts of traffic.

Additionally, using MongoDB as the database for the web-app allows for flexible and scalable data storage, making it an ideal choice for web applications that require frequent updates and dynamic content.

Overall, this project provides a practical example of how containerization and orchestration technologies can be used to simplify the deployment and management of complex web applications.