Cloud Computing - Mini Project Report
PROJECT-5
April 2023

Submitted By:
Name   ANISH UDUPA. BS ANAVI , BARKHA GOYAL
SRN: PES2UG20CS493, PES2UG20CS500, PES2UG20CS501
VI Semester Section _ H
PES University

Short Description and Scope of the Project

Project Description:

This project involves building a microservices architecture using Docker and Kubernetes for a blogging web app. The architecture will consist of a MongoDB server, a Mongo-Express web service, and a Flask web app. The project requires configuration of the necessary environment variables, creating deployments, and defining services in YAML files to bring up the microservices. Once the microservices are up and running, the records should be inserted into the MongoDB database using a Python script and display them on the homepage of the Flask app.

Project Scope:

The primary objective of this project is to leverage a microservices architecture using Docker and Kubernetes to deploy a complex and scalable blogging web application. In order to accomplish this, the microservices architecture will comprise a MongoDB server, a Mongo-Express web service, and a Flask web application. The project will require a comprehensive understanding of the underlying architecture, as well as a deep knowledge of YAML configuration files and Docker containerization. Additionally, it will involve configuring environment variables, defining secret YAML files, creating deployments, and defining services to establish a seamless integration between the microservices. Once the microservices have been deployed, it will be necessary to insert records into the MongoDB database using a Python script, thereby enabling the storage of large amounts of data. Finally, the Flask application will be launched, with the home page displaying the aforementioned records in a structured and user-friendly manner.

Methodology

Brief Overview of the Tasks:

1.Set up MongoDB server:

- Use the publicly available MongoDB image from DockerHub and configure the necessary environment variables by referring to the image's documentation.
- Create a deployment for the MongoDB server under deployments.yaml, ensuring that you configure the ports and environment variables correctly.
- Define a .yaml file to hold sensitive information like username and password required by the MongoDB server. You can create a secret using a configuration file and use the secret in your deployment as an environment variable.

2.Set up Mongo-Express web service:

- Use the mongo-express image and note down the necessary environment variables from the image's documentation.
- Define a configMap to store the MongoDB server URL and use the configMap to configure the container with environment variables.
- Create a deployment for the mongo-express service under deployments.yaml, configuring the necessary ports and environment variables.
- Define a service for the pod under services.yaml.

3.Set up Flask web app:

- Use the image created from the flask-app-image.dockerfile.
- Create a deployment for the Flask app under deployments.yaml.
- Define a service for the pod in services.yaml

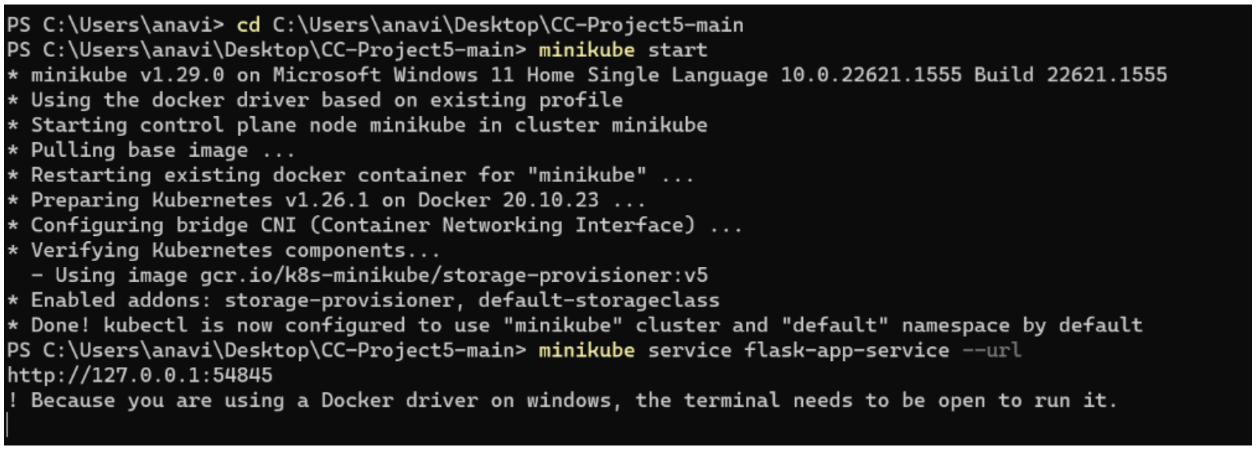
4.Bring it all together:

- Bring up all the microservices.

- Inside the Flask app pod, write and run a Python script to insert records into the MongoDB database. Insert the records into the "blog" database and the "posts" collection.
- 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.

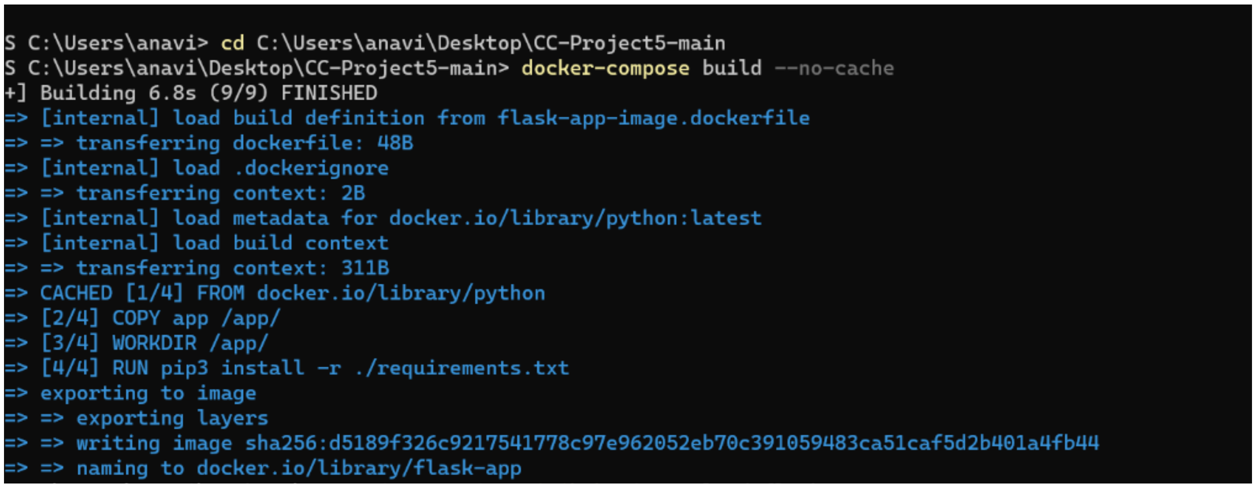
Execution Steps

1. Start Minikube



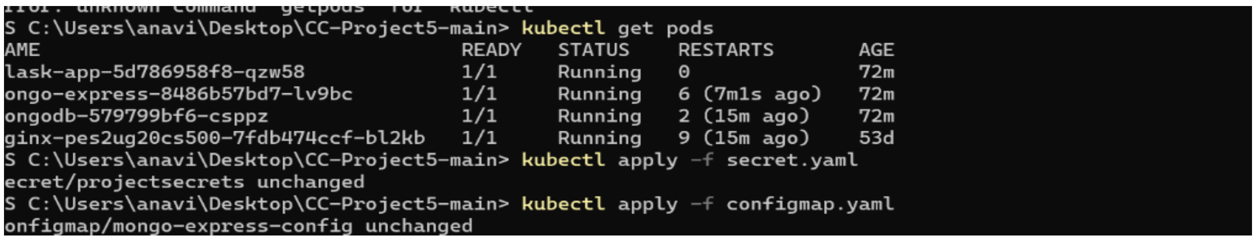
```
PS C:\Users\anavi> cd C:\Users\anavi\Desktop\CC-Project5-main
PS C:\Users\anavi\Desktop\CC-Project5-main> minikube start
* minikube v1.29.0 on Microsoft Windows 11 Home Single Language 10.0.22621.1555 Build 22621.1555
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Restarting existing docker container for "minikube" ...
* Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Users\anavi\Desktop\CC-Project5-main> minikube service flask-app-service --url
http://127.0.0.1:54845
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

2. docker-compose build --no-cache



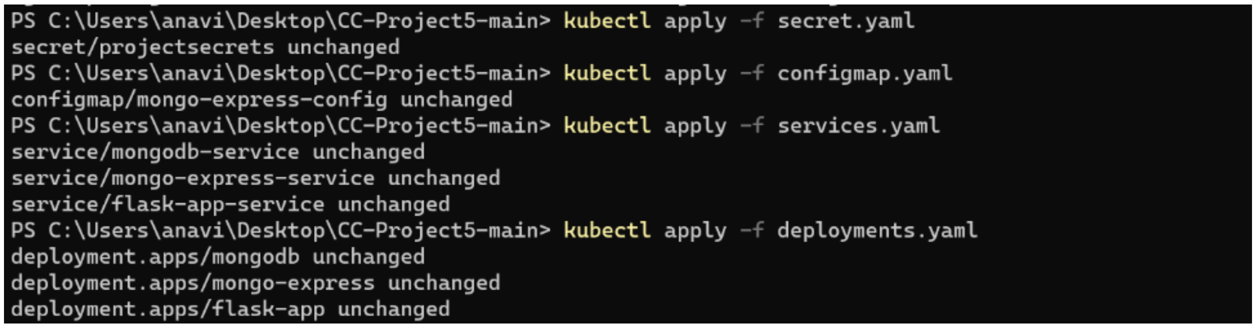
```
S C:\Users\anavi> cd C:\Users\anavi\Desktop\CC-Project5-main
S C:\Users\anavi\Desktop\CC-Project5-main> docker-compose build --no-cache
+] Building 6.8s (9/9) FINISHED
=> [internal] load build definition from flask-app-image.dockerfile
=> => transferring dockerfile: 48B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:latest
=> [internal] load build context
=> => transferring context: 311B
=> CACHED [1/4] FROM docker.io/library/python
=> [2/4] COPY app /app/
=> [3/4] WORKDIR /app/
=> [4/4] RUN pip3 install -r ./requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:d5189f326c9217541778c97e962052eb70c391059483ca51caf5d2b401a4fb44
=> => naming to docker.io/library/flask-app
```

3. **minikube image load flask-app**



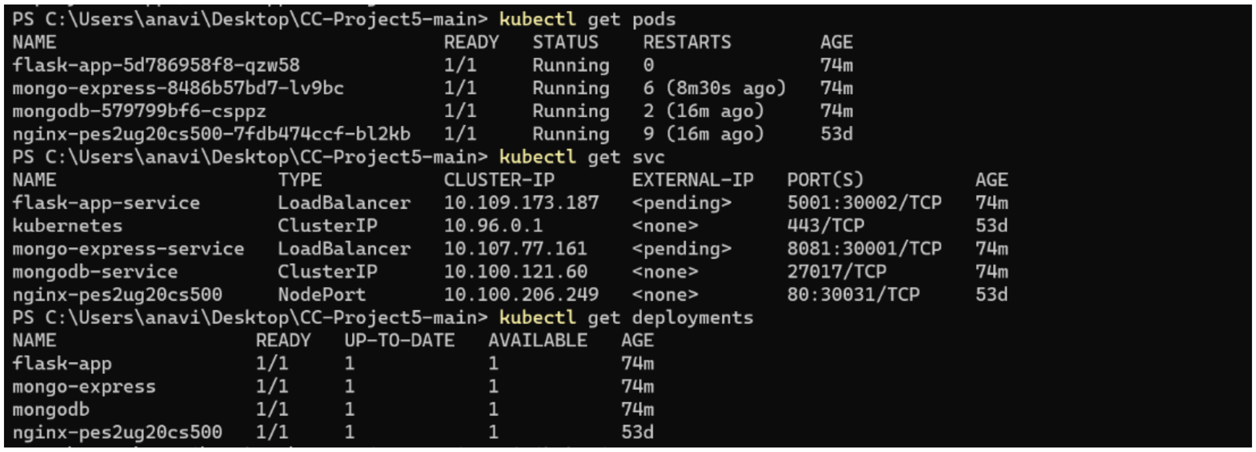
```
rror: unknown command "getpods" for "kubectl"
S C:\Users\anavi\Desktop\CC-Project5-main> kubectl get pods
AME                                     READY   STATUS    RESTARTS       AGE
lask-app-5d786958f8-qzw58               1/1     Running   0              72m
ongo-express-8486b57bd7-lv9bc           1/1     Running   6 (7m1s ago)   72m
ongodb-579799bf6-csppz                  1/1     Running   2 (15m ago)    72m
ginx-pes2ug20cs500-7fdb474ccf-bl2kb     1/1     Running   9 (15m ago)    53d
S C:\Users\anavi\Desktop\CC-Project5-main> kubectl apply -f secret.yaml
ecret/projectsecrets unchanged
S C:\Users\anavi\Desktop\CC-Project5-main> kubectl apply -f configmap.yaml
onfigmap/mongo-express-config unchanged
```

4. **kubectl apply -f secrets.yaml**
5. **kubectl apply -f configmap.yaml**
6. **kubectl apply -f services.yaml**
7. **kubectl apply -f deployment.yaml**



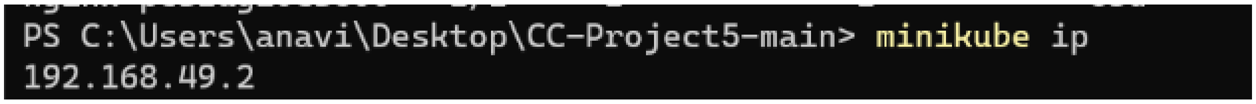
```
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl apply -f secret.yaml
secret/projectsecrets unchanged
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl apply -f configmap.yaml
configmap/mongo-express-config unchanged
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl apply -f services.yaml
service/mongodb-service unchanged
service/mongo-express-service unchanged
service/flask-app-service unchanged
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl apply -f deployments.yaml
deployment.apps/mongodb unchanged
deployment.apps/mongo-express unchanged
deployment.apps/flask-app unchanged
```

8. **kubectl get pods**
9. **kubectl get svc**
10. **kubectl get deployments**



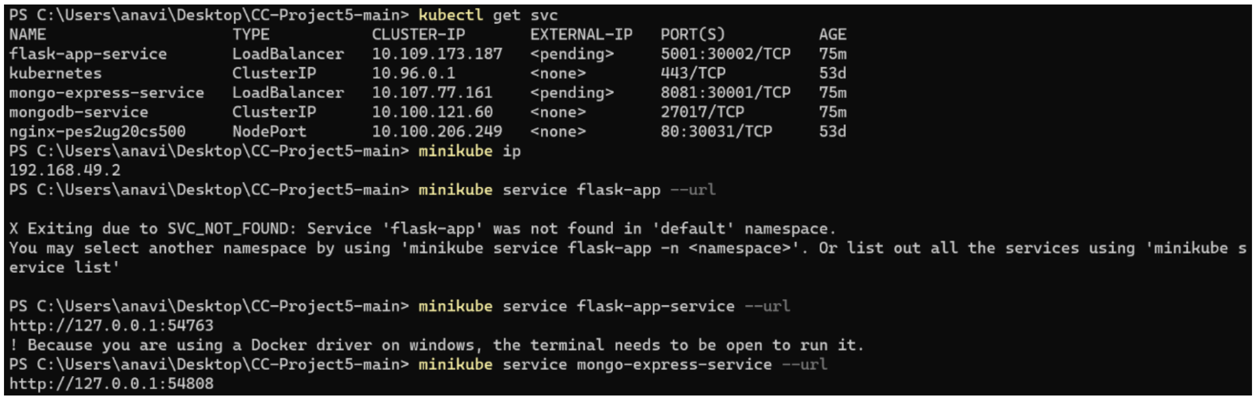
```
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl get pods
NAME                                   READY   STATUS    RESTARTS        AGE
flask-app-5d786958f8-qzw58             1/1     Running   0               74m
mongo-express-8486b57bd7-lv9bc         1/1     Running   6 (8m30s ago)   74m
mongodb-579799bf6-csppz                1/1     Running   2 (16m ago)     74m
nginx-pes2ug20cs500-7fdb474ccf-bl2kb   1/1     Running   9 (16m ago)     53d
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl get svc
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
flask-app-service       LoadBalancer   10.109.173.187   <pending>     5001:30002/TCP   74m
kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP          53d
mongo-express-service   LoadBalancer   10.107.77.161    <pending>     8081:30001/TCP   74m
mongodb-service         ClusterIP      10.100.121.60    <none>        27017/TCP        74m
nginx-pes2ug20cs500     NodePort       10.100.206.249   <none>        80:30031/TCP     53d
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
flask-app             1/1     1            1           74m
mongo-express         1/1     1            1           74m
mongodb               1/1     1            1           74m
nginx-pes2ug20cs500   1/1     1            1           53d
```

11. minikube ip



```
PS C:\Users\anavi\Desktop\CC-Project5-main> minikube ip
192.168.49.2
```

12. kubectl get svc



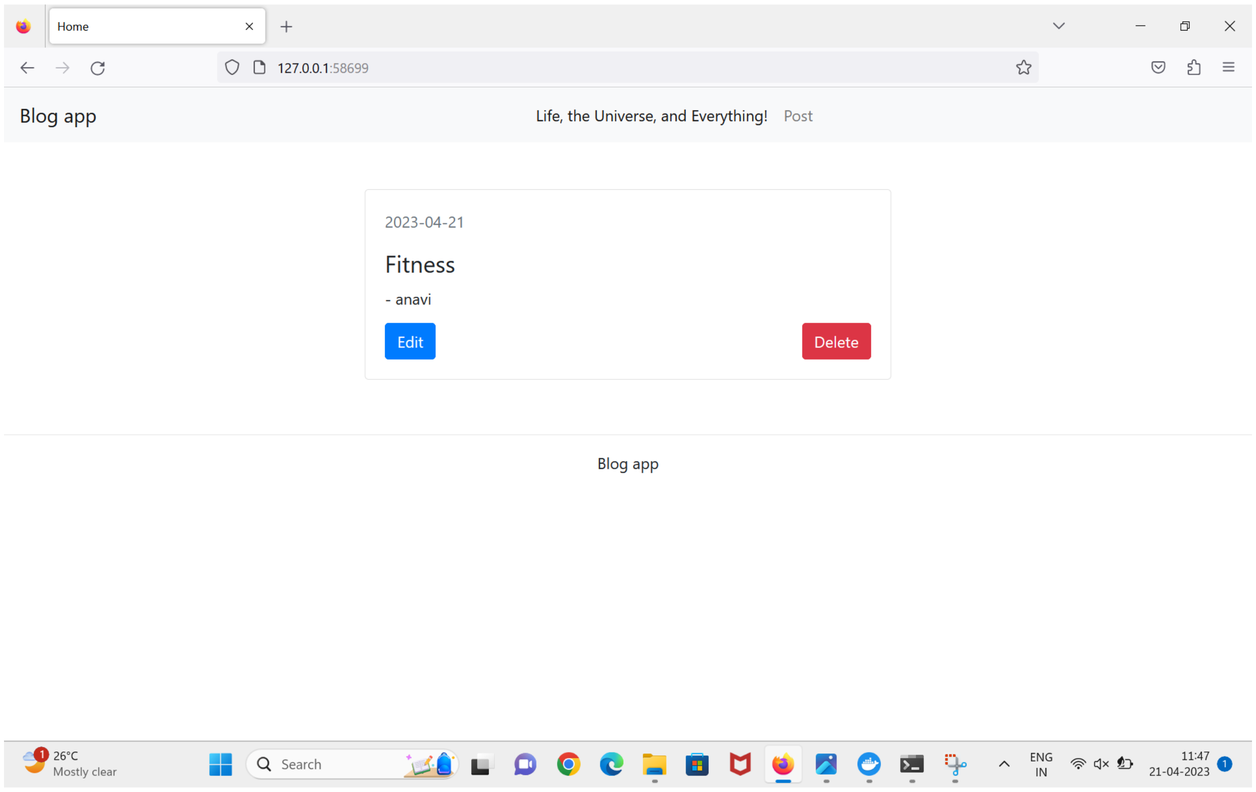
```
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl get svc
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
flask-app-service       LoadBalancer   10.109.173.187   <pending>     5001:30002/TCP   75m
kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP          53d
mongo-express-service   LoadBalancer   10.107.77.161    <pending>     8081:30001/TCP   75m
mongodb-service         ClusterIP      10.100.121.60    <none>        27017/TCP        75m
nginx-pes2ug20cs500     NodePort       10.100.206.249   <none>        80:30031/TCP     53d
PS C:\Users\anavi\Desktop\CC-Project5-main> minikube ip
192.168.49.2
PS C:\Users\anavi\Desktop\CC-Project5-main> minikube service flask-app --url

X Exiting due to SVC_NOT_FOUND: Service 'flask-app' was not found in 'default' namespace.
You may select another namespace by using 'minikube service flask-app -n <namespace>'. Or list out all the services using 'minikube s
ervice list'

PS C:\Users\anavi\Desktop\CC-Project5-main> minikube service flask-app-service --url
http://127.0.0.1:54763
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
PS C:\Users\anavi\Desktop\CC-Project5-main> minikube service mongo-express-service --url
http://127.0.0.1:54808
```

Project Output:

ADDING DATA FROM FRONTEND



Home

127.0.0.1:58699

Blog app   Life, the Universe, and Everything!   Post

2023-04-21

Fitness

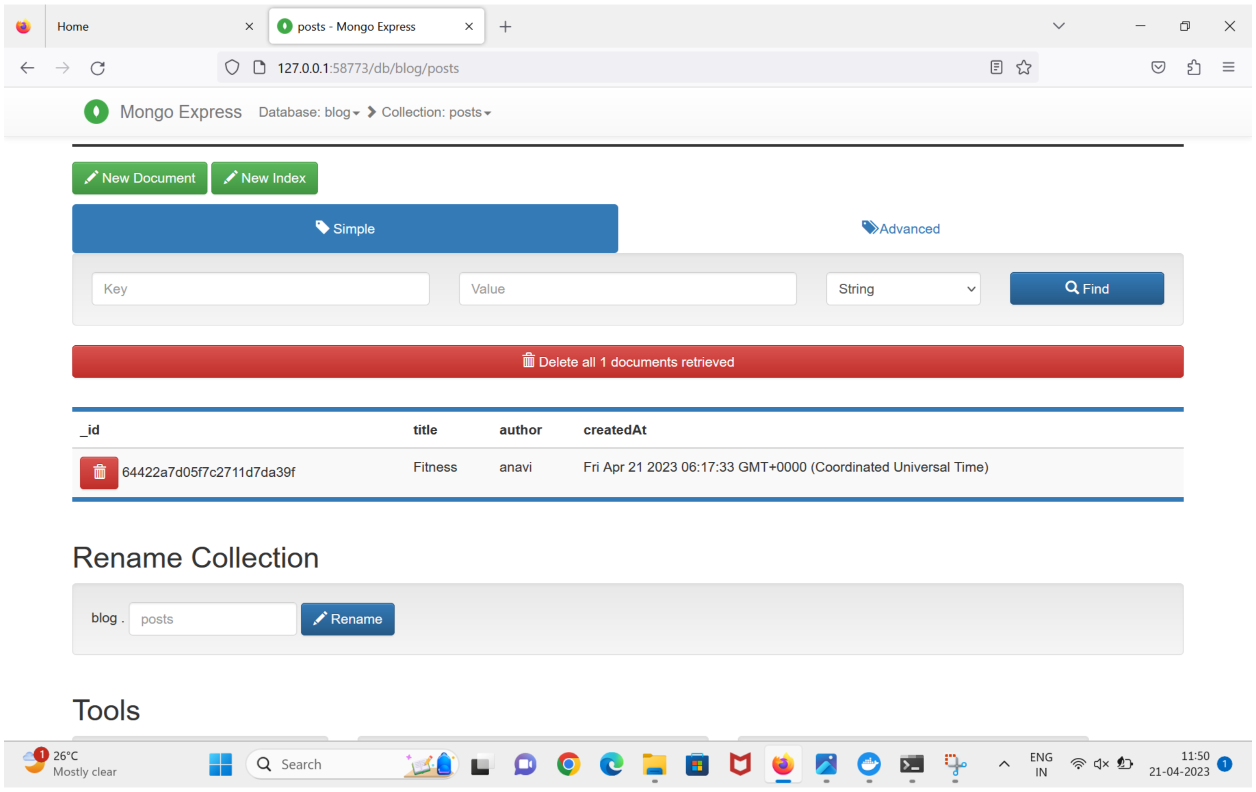
- anavi

Edit   Delete

Blog app

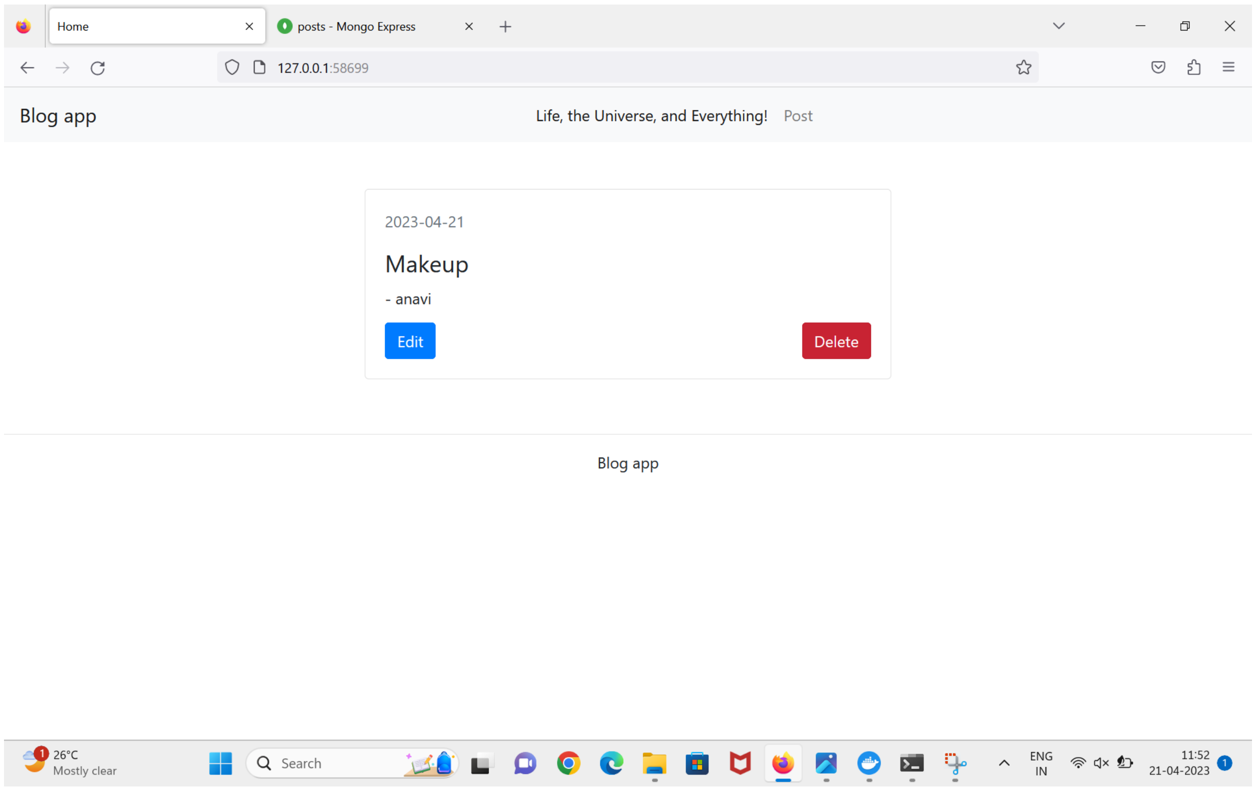
26°C Mostly clear   Search   ENG IN   11:47 21-04-2023

BACKEND IS RECEIVING THE DATA



_id	title	author	createdAt
64422a7d05f7c2711d7da39f	Fitness	anavi	Fri Apr 21 2023 06:17:33 GMT+0000 (Coordinated Universal Time)

MODIFICATION OF THE DATA


Blog app

Life, the Universe, and Everything! Post

2023-04-21

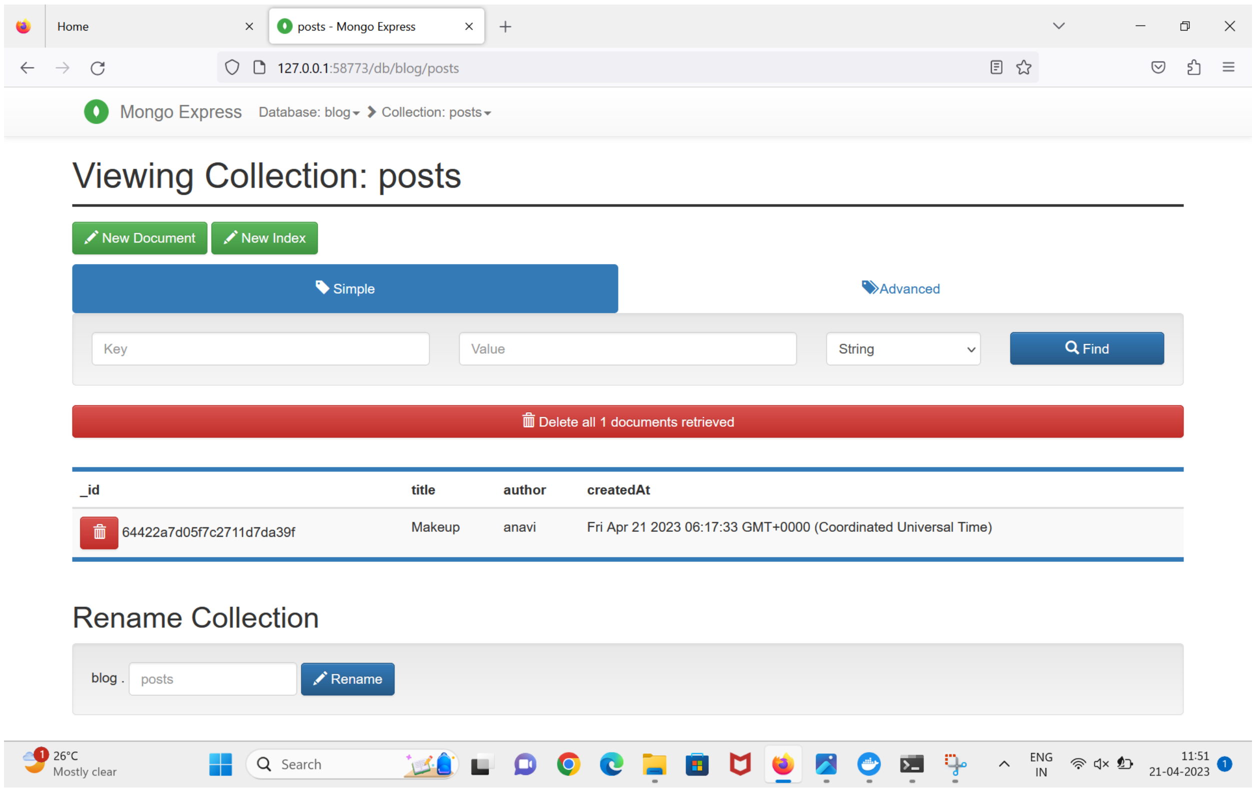
Makeup

- anavi

Edit Delete

Blog app

REFLECTED IN THE DATABASE


Mongo Express Database: blog Collection: posts

Viewing Collection: posts

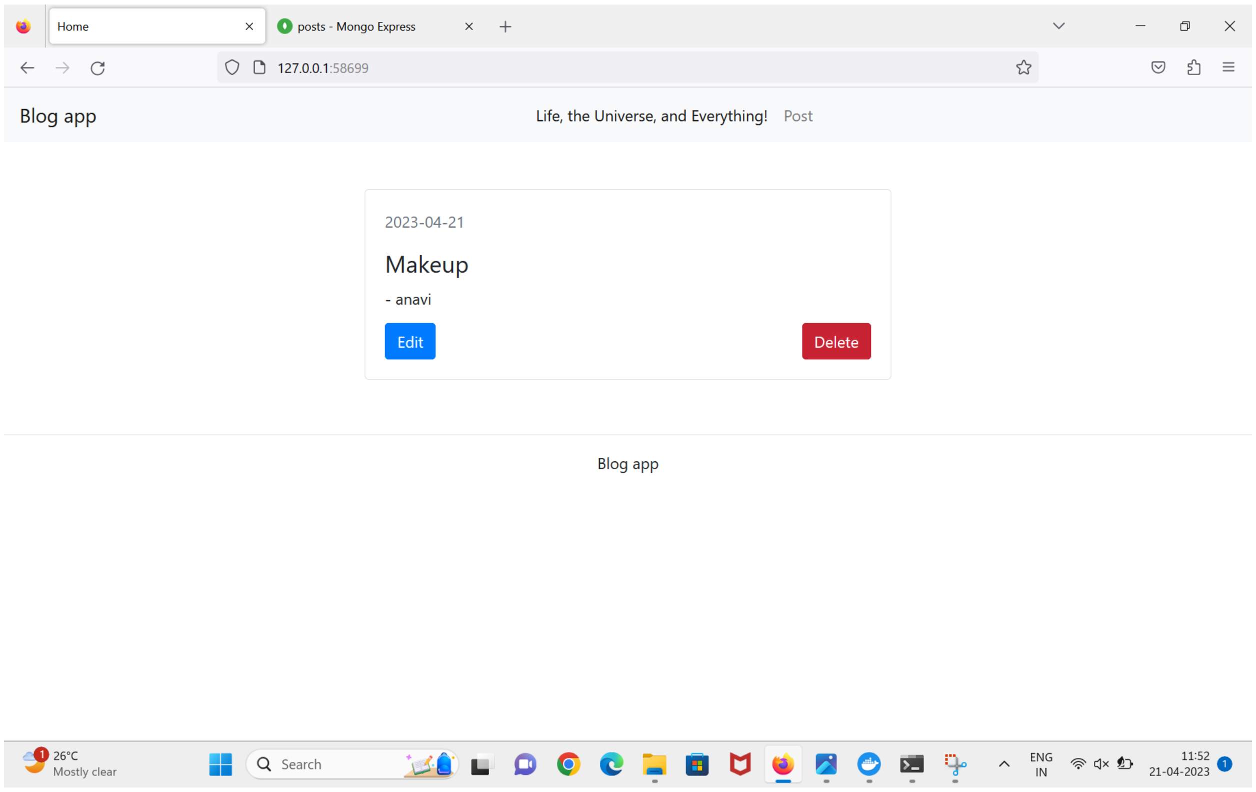
Delete all 1 documents retrieved

_id	title	author	createdAt
64422a7d05f7c2711d7da39f	Makeup	anavi	Fri Apr 21 2023 06:17:33 GMT+0000 (Coordinated Universal Time)

Rename Collection

blog . posts Rename

DELETION OF DATA


Home | posts - Mongo Express

127.0.0.1:58699

Blog app   Life, the Universe, and Everything!   Post

2023-04-21

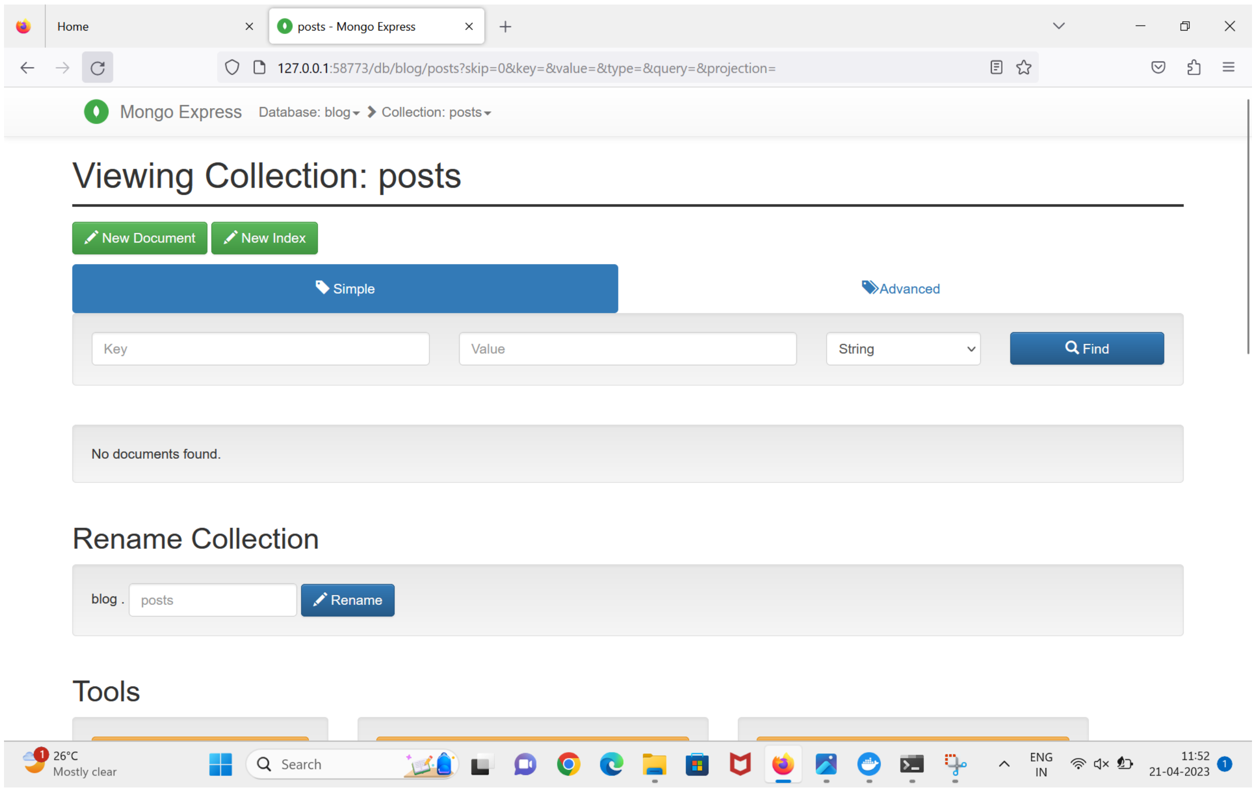
Makeup

- anavi

Edit   Delete

Blog app

26°C Mostly clear   11:52 21-04-2023


Home | posts - Mongo Express

127.0.0.1:58773/db/blog/posts?skip=0&key=&value=&type=&query=&projection=

Mongo Express   Database: blog   Collection: posts

Viewing Collection: posts

New Document   New Index

Simple   Advanced

Key   Value   String   Find

No documents found.

Rename Collection

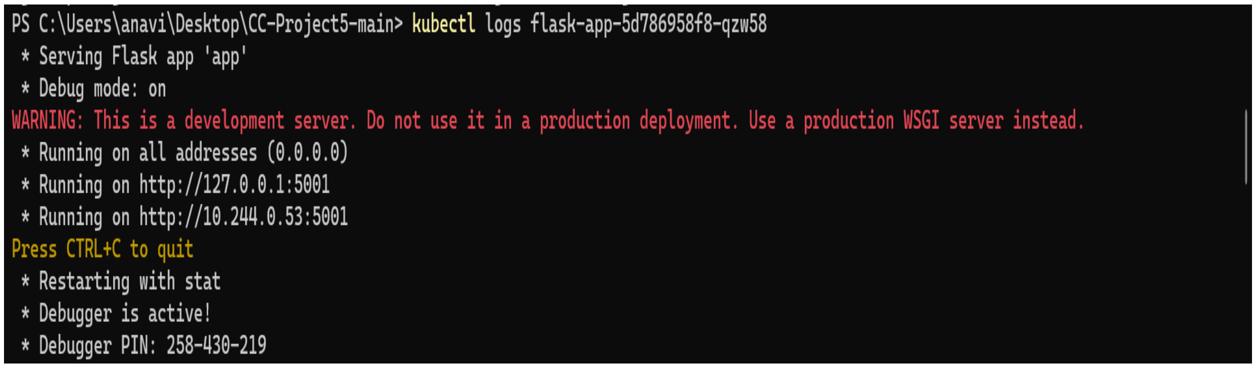
blog . posts   Rename

Tools

26°C Mostly clear   11:52 21-04-2023

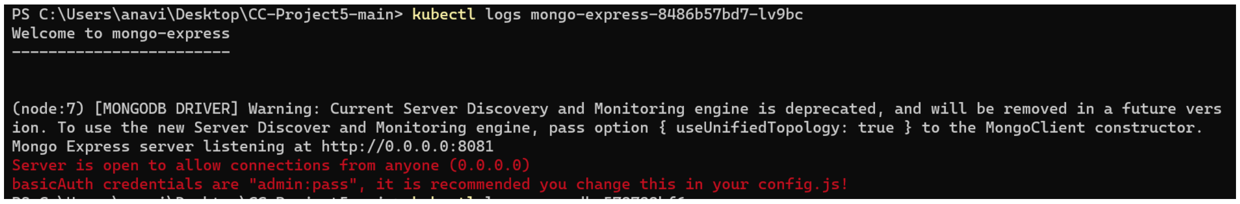
Testing

1.LOGS FOR FLASK APP



```
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl logs flask-app-5d786958f8-qzw58
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.244.0.53:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 258-430-219
```

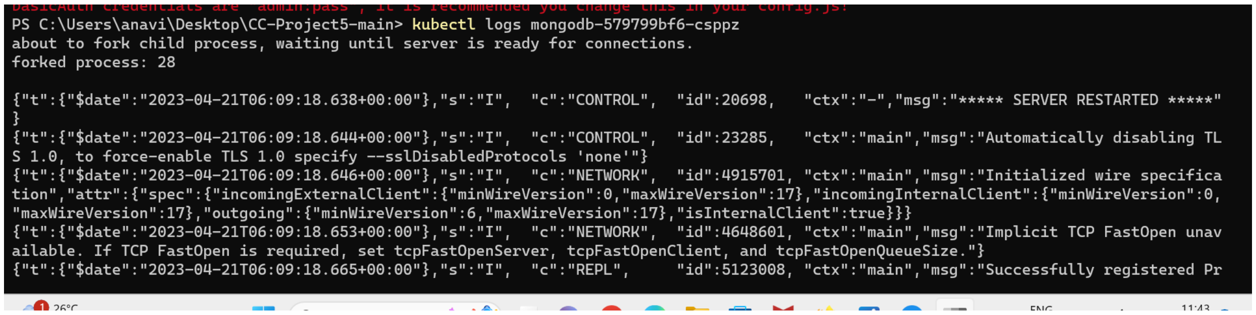
2.LOGS FOR MONGO-EXPRESS



```
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl logs mongo-express-8486b57bd7-lv9bc
Welcome to mongo-express
------------------------


(node:7) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future vers
ion. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Mongo Express server listening at http://0.0.0.0:8081
Server is open to allow connections from anyone (0.0.0.0)
basicAuth credentials are "admin:pass", it is recommended you change this in your config.js!
```

3.LOGS FOR MONGODB



```
basicAuth credentials are "admin:pass", it is recommended you change this in your config.js!
PS C:\Users\anavi\Desktop\CC-Project5-main> kubectl logs mongodb-579799bf6-csppz
about to fork child process, waiting until server is ready for connections.
forked process: 28

{"t":{"$date":"2023-04-21T06:09:18.638+00:00"},"s":"I",  "c":"CONTROL",  "id":20698,   "ctx":"-","msg":"***** SERVER RESTARTED *****"
}
{"t":{"$date":"2023-04-21T06:09:18.644+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TL
S 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2023-04-21T06:09:18.646+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initialized wire specifica
tion","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInternalClient":{"minWireVersion":0,
"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient":true}}}
{"t":{"$date":"2023-04-21T06:09:18.653+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unav
ailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
{"t":{"$date":"2023-04-21T06:09:18.665+00:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Successfully registered Pr
```

Results and Conclusions

This Project successfully demonstrates the working of a simple blogging application and dynamically handles any error encountered. The deployment occurs on a docker container via a kubernetes cluster and we see that the three applications work simultaneously without any exceptions. The records are inserted into the database and all the CRUD operations can be performed. Hence we can conclude that the integration of the three frameworks into a single microservice architecture has been successfully achieved in this project. The project highlighted the flexibility and scalability of microservices architectures, allowing for the seamless integration of multiple components into a cohesive system.