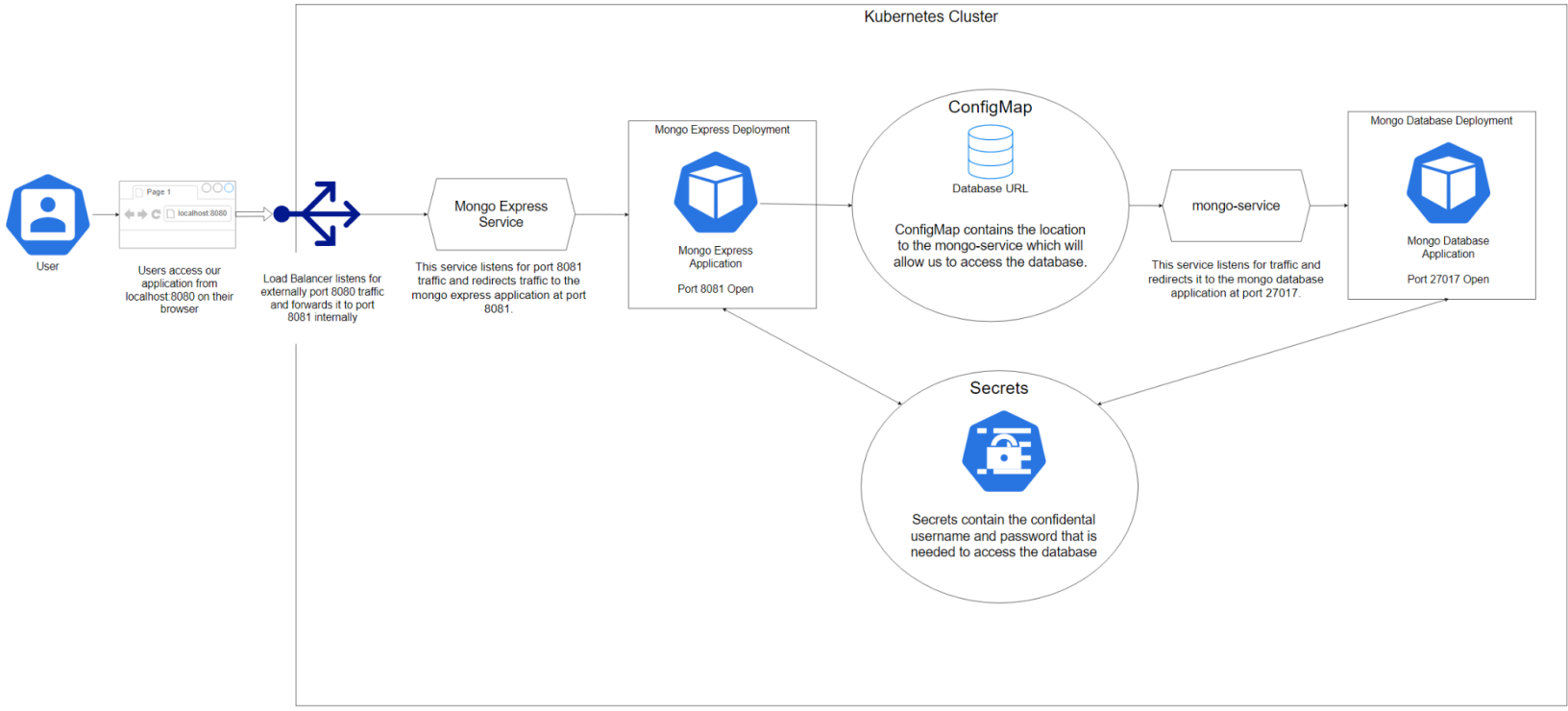
Database Assignment



The objective of this assignment was to implement MongoDB on a Kubernetes architecture. This assignment involves creating an application that is made to communicate to our MongoDB database, which is called Mongo Express. Mongo Express allows users to communicate with an isolated MongoDB database by using a user interface. While doing this assignment, we have to configure Kubernetes ports for clusters and then deploy the MongoDB and service using K3D locally.

The first thing that we have to do is to create a cluster with a load balancer attached. We are tasked to configure the port mapping from the load balancer to the Mongo Express service. Once we set up the load balancer we then have to create the MongoDB application and service files using a YAML file. YAML files are simply infrastructure as code for Kubernetes. We can write everything that we want to create inside one YAML. For this assignment, we will be creating a deployment and a service. A deployment allows us to specify what's inside your pod.

The service is used to allow us to interact with our pods. Pods are simply packaged up Docker containers that contain our application.

Once we created the two files we have to create a secret file that will be used to store our credentials. Secrets are objects that contain a small amount of sensitive data such as passwords and keys. Secrets are similar to ConfigMaps but secrets are intended to hold confidential data. For this assignment, we will be using base64 to encode our username and password so users don't have access to plain text.

After we create the secret file, we will deploy the secret.yaml file first and then deploy the mongo deployment(service and deployment of MongoDB). The reason why we deploy the secrets.yaml first is because the mongo deployment needs to reference the secrets.yaml file for environmental variables. This will allow the database application to obtain the username and password.

When we finish that we have to create a config_map which will allow communication between the application and the database. The Config_Map will allow the mongo-express application to find the mongo-service. Once the mongo-express application finds the service, it can connect to the database pods. Config_Maps are similar to secret files because it allows us to store non-confidential data in key-value pairs.

Once the Config_Map is created we will need to create the deployment and service for the mongo express application. For this part, we are tasked with configuring the correct service type, target-port, and ports of the service based on the deployment that was provided. After the file is created we then need to create the config_map first, then the mongo-express deployment and service. This will allow our mongo-express application to consume ConfigMaps as environmental variables.

Finally, everything will be set up and users can access the mongo-express application that will interact with the database. Users can interact with the UI and create new databases, collections, and datasets.

Task

The first thing we need to do is create a cluster and attach a load balancer with port mapping via K3D. K3D is a lightweight tool that is used to interact with kubernetes locally. The load balancer has an external and internal port. Users access our application using the external port and the load balancer uses port forwarding to send the traffic to the internal port.

We then need to create a YAML file that has a kubernetes deployment and service configured in one file. YAML files are simply configuration files. Deployments allow us to specify what is inside our pods. Pods are essentially a bundle of containers needed to run an application. The YAML file also creates a service which allows us to interact with our pods. Once we have set up our YAML file with the specific ports and configuration, we can build our YAML file. This will create all the configurations we specified.

Create a cluster with a load balancer that has port mapping 8080:8081 via K3D. 8081 is for the application.
k3d cluster create mongo -p “8080:8081@loadbalancer”

Create a mongo-deployment.yaml file and paste the following inside. This will create a deployment and service.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo:4.4
          ports:
            - containerPort: 27017
          env:
          - name: MONGO_INITDB_ROOT_USERNAME
            valueFrom:
              secretKeyRef:
                name: mongodb-secret
```

```
            key: mongo-root-username
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-password
---
apiVersion: v1
kind: Service
metadata:
  name: mongo-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

Task 2

Once we created the first YAML, we will need to create a secret.yaml file for the username and password. We will need to encode the string to base64. You will need WSL (Windows) or Mac Terminal to use the following. Or you can use (https://www.base64encode.org/). The secret.yaml file holds our credentials that will allow us to connect to the mongodb database.

In the terminal run the following command
echo -n mongo-root-username | base64
Output -> bW9uZ28tcm9vdC11c2VybmFtZQ==

echo -n mongo-root-passworde | base64
Output -> bW9uZ28tcm9vdC1wYXNzd29yZGU=

We then need to create a secret.yaml file and paste the following credentials into the specified values.

```
apiVersion: v1
kind: Secret
metadata:
   name: mongodb-secret
type: Opaque
data:
   mongo-root-username: bW9uZ28tcm9vdC11c2VybmFtZQ==
   mongo-root-password: bW9uZ28tcm9vdC1wYXNzd29yZA==
```

Task 3

It is finally time to deploy both of these yaml files. You must deploy the secret.yaml file first then the mongo-deployment.yaml file. The mongo-deployment yaml file uses information (credentials) from the secret.yaml file.

kubectl create -f **secret.yaml**
kubectl create -f **mongo-deployment.yaml**

Task 4

Now, we will need to create a config_map.yaml that will allow our application to connect to our database.

Create the config_map.yaml and paste the following

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-configmap
data:
  database_url: mongo-service
```

Once we created the config_map we need to create the mongo-express deployment that will be our user interface application. This will allow us to talk to our database. A deployment and service will be made for this yaml file. The service will listen for port 8081 externally, and redirect it to port 8081 internally. Our mongo-express application will listen for port 8081 traffic. Create a mongo-express.yaml file and paste the following

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongoexp-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo-express
  template:
    metadata:
      labels:
        app: mongo-express
```

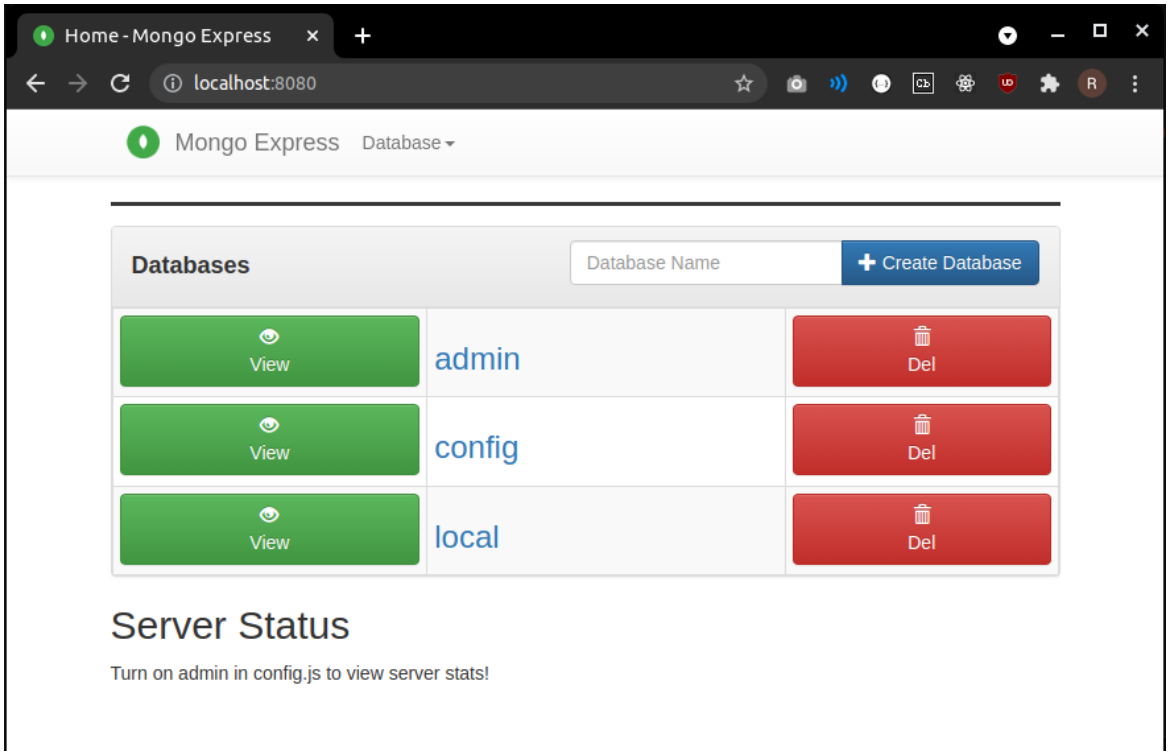
```
    spec:
      containers:
        - name: mongo-express
          image: mongo-express
          ports:
            - containerPort: 8081
          env:
          - name: ME_CONFIG_MONGODB_ADMINUSERNAME
            valueFrom:
              secretKeyRef:
                name: mongodb-secret
                key: mongo-root-username
          - name: ME_CONFIG_MONGODB_ADMINPASSWORD
            valueFrom:
              secretKeyRef:
                name: mongodb-secret
                key: mongo-root-password
          - name: ME_CONFIG_MONGODB_SERVER
            valueFrom:
              configMapKeyRef:
                name: mongodb-configmap
                key: database_url
---
apiVersion: v1
kind: Service
metadata:
  name: mongo-exp-service
spec:
  selector:
    app: mongo-express
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 8081
```

Once we created those files we will need to create the yaml files. First create the config_map and then the mongo-express file.

kubectl create -f **config_map.yaml**
kubectl create -f **mongo-express.yaml**

Task 5

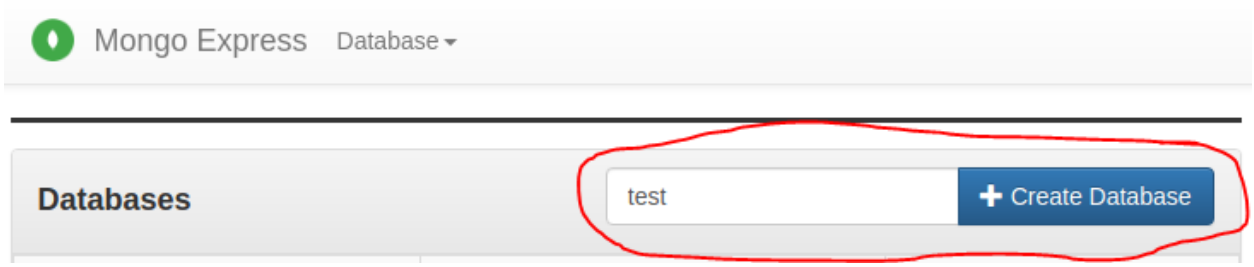
We can finally access our application by going to localhost:8080 on your browser. You should see the following page.



Task 6

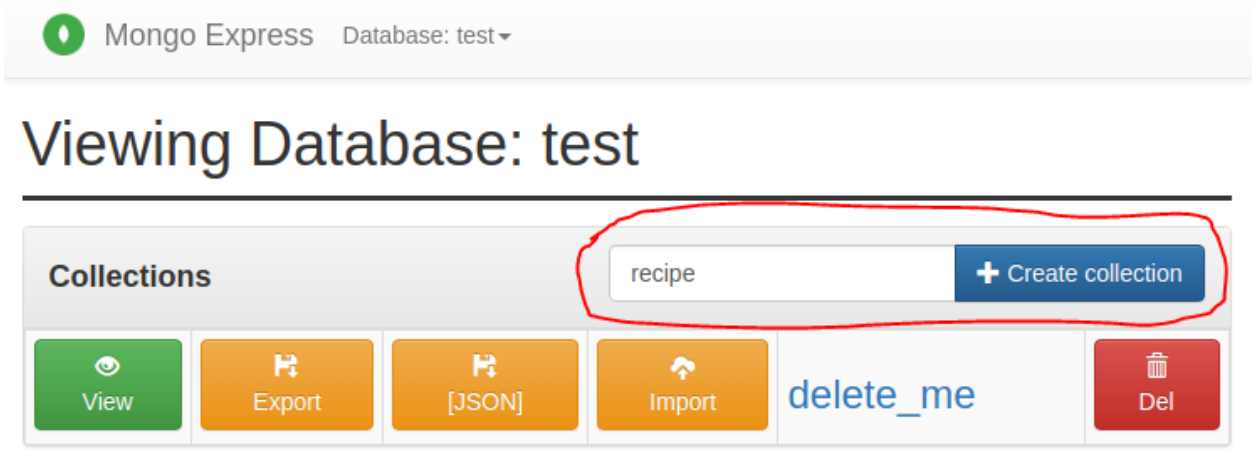
We will need to add some data to our database via the Mongodb-Express application. This is our user interface so we can interact with out database

Create a new database name called “test”

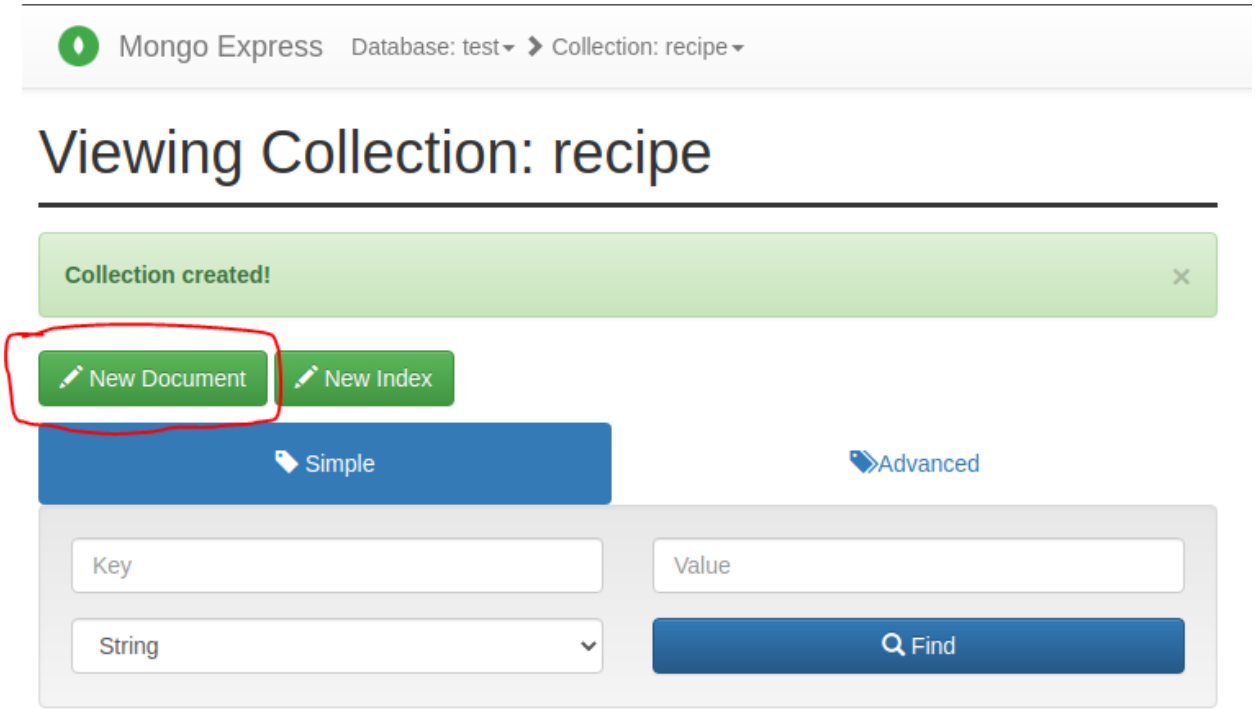


View the database that you created

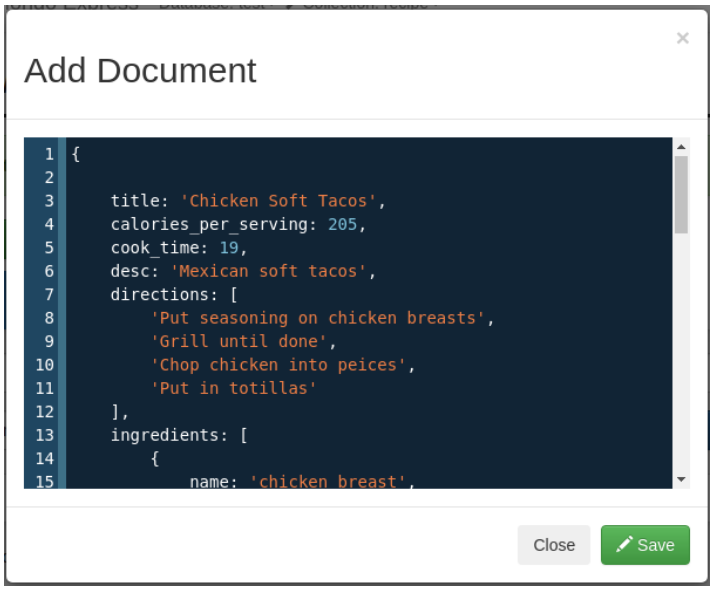
Create a new collection called “recipe”



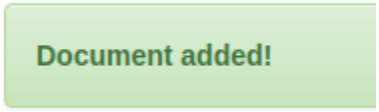
Once the “recipe” collection has been made, click on a new document and paste the following data set.
(https://docs.google.com/document/d/1A7FWSQ5JL108eAtTNQ3KCKutMHjd6Btx-6bA3RnDPKA/edit)



```
{
    title: 'Chicken Soft Tacos',
    calories_per_serving: 205,
    cook_time: 19,
    desc: 'Mexican soft tacos',
    directions: [
        'Put seasoning on chicken breasts',
        'Grill until done',
        'Chop chicken into peices',
        'Put in totillas'
    ],
    ingredients: [
        {
            name: 'chicken breast',
            quantity: {
                amount: 1,
                unit: 'lbs'
            }
        },
        {
            name: 'taco seasoning',
            quantity: {
                amount: 2,
                unit: 'oz'
            }
        },
        {
            name: 'small flour totillas',
            quantity: {
                amount: 12,
                unit: 'oz'
            }
        }
    ],
    likes: [
        261,
        1,
        415
    ],
    likes_count: 3,
    prep_time: 10,
    rating: [
        4,
        4,
        4,
        4,
        2,
        5,
        3
    ],
    rating_avg: 3.71,
    servings: 5,
    tags: [
        'mexican',
        'quick',
        'easy',
        'chicken'
    ],
    type: 'Dinner'
}
```



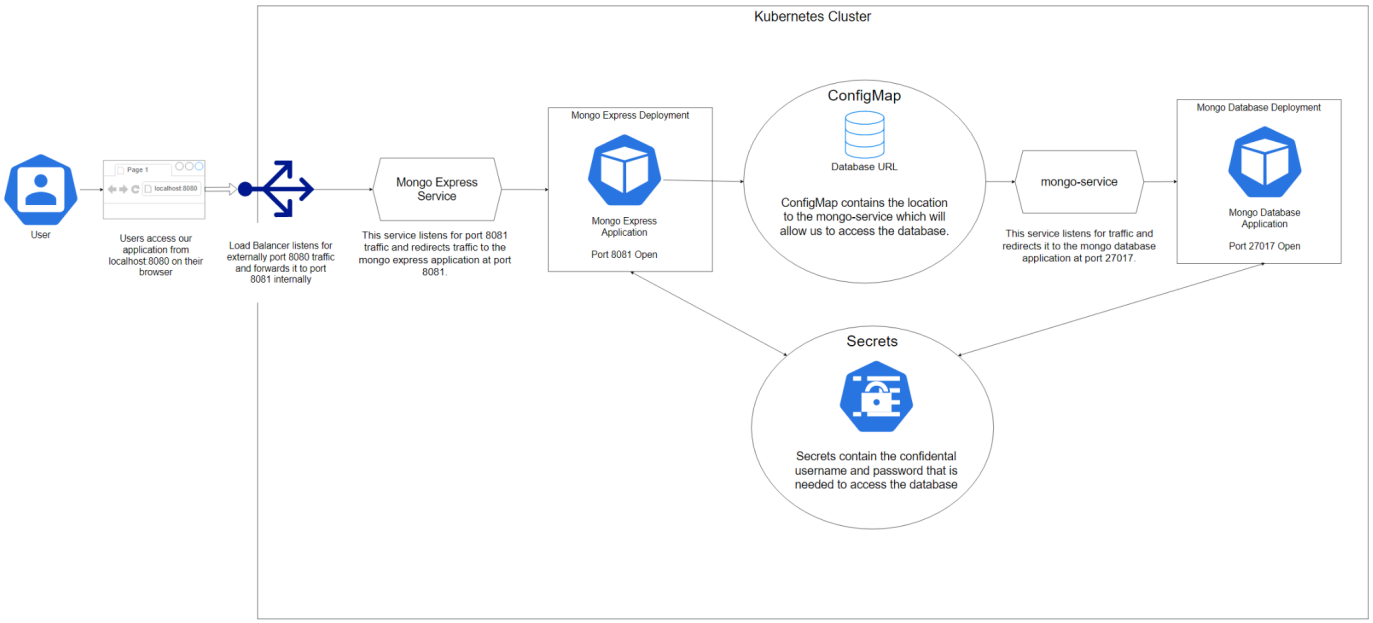
The document has been added once you save it.



You are done!

You can delete the entire cluster using the following
k3d cluster delete mongo

Task 7 - Topology



In the topology, users can access our application on their local browser using “localhost:8080”. The load balancer attached to the cluster will then listen for port 8080 external traffic and redirect traffic to port 8081 internally. Inside the cluster, there is a mongo express service that listens for port 8081 and redirects it to the mongo express pod at port 8081. The mongo express pod has port 8081 open and the mongo express application inside of it. The mongo express application is a simple user interface that allows users to interact with the MongoDB database. The application uses secrets.yaml file to obtain the confidential username and password. Once it has that it will also use a ConfigMap which contains the location to the mongo-service. The mongo-service will then allow us to interact with the mongo database pod which has the actual mongo-database application. The mongo-express application does not talk directly to the database. It uses the ConfigMap as a middle man.

This architecture denies users access to our database directly from the load balancer. Our database must be secured from the public because sensitive data will be leaked. When scaling out these applications, we are creating new pods. It is best so we would have to scale horizontally. Have a database on another node because you don’t necessarily want to scale out the database proportionally to the application. For this architecture, when we scale out these applications, we are creating new pods. It is best to have these applications on separate nodes so we can scale horizontally and have a resilient application. If everything is on one node, it would be inefficient because we don’t need multiple databases. If we want to improve database performance we can implement read replicas that only read from the Controller database. The controller database will only allow writing and the others will have reading permissions. When we want to scale out our application, we will need to create another pod and connect it to the ConfigMap so it knows where the database is located.