Task 9.1

Jeremy Pedersen
Student ID: 217593144

Objectives

The main objectives of the project were to:

- Deploy MongoDB within a Kubernetes cluster.
- Secure MongoDB using Kubernetes secrets.
- Configure persistent storage solutions to ensure data reliability.
- Implement proper user authentication and manage permissions.
- Ensure the application could successfully connect and interact with the database.
- Run a query, update and delete on database.
- Set up basic monitoring and optional backup solutions for MongoDB.

Methodology

Deployment of MongoDB

I began by creating a deployment configuration for MongoDB using a Docker container. This setup included defining a deployment YAML file that specifies the MongoDB Docker image, necessary environmental variables, and the required ports.

Configuration of Kubernetes Secrets

To securely handle MongoDB credentials, we utilized Kubernetes secrets. These secrets store the database username and password in an encoded format, ensuring that sensitive information is not exposed in plain text within the deployment configurations.

Persistent Storage

I configured Persistent Volumes (PV) and Persistent Volume Claims (PVC) within Kubernetes to manage data storage for MongoDB. This configuration is essential to ensure that data is not lost when pods are recreated or when the cluster is scaled.

Database Connectivity

The application's deployment manifest was modified to include the MongoDB connection string, ensuring the microservice can connect to the database using the credentials stored in Kubernetes secrets. We tested this connectivity extensively to confirm that the application could perform basic CRUD operations on the database.

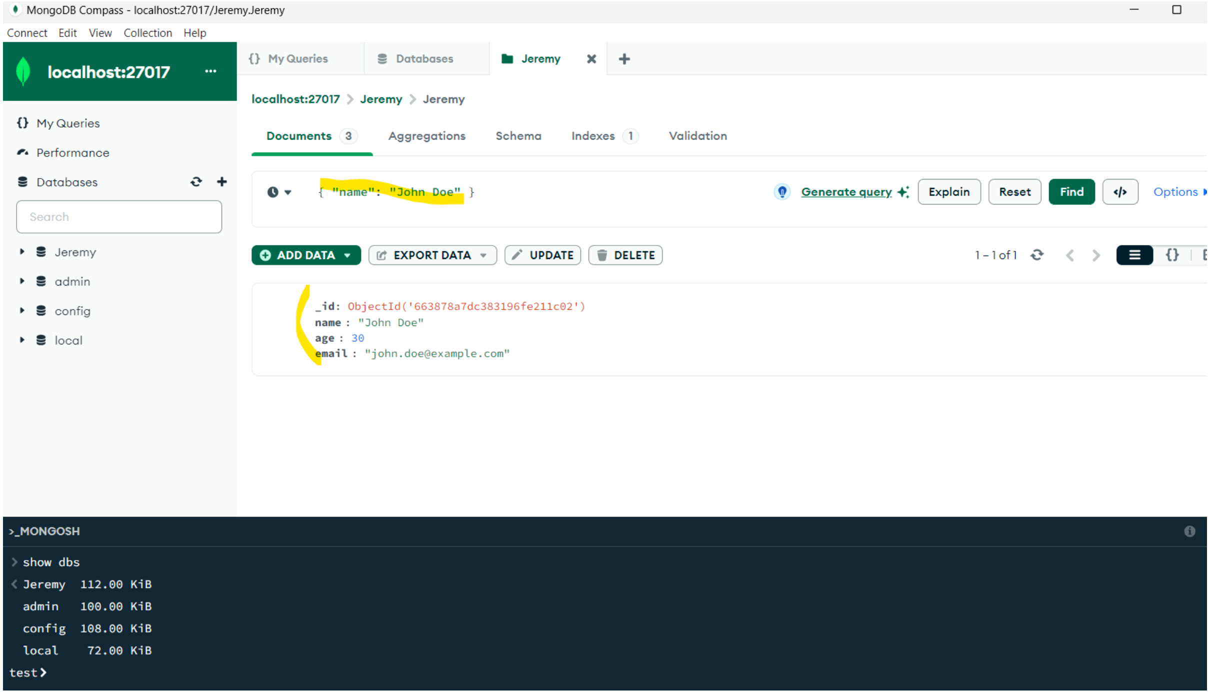
Monitoring

Basic monitoring was implemented to observe the performance and health of the MongoDB instance.

Results

The deployment was successful, with MongoDB running stably within the Kubernetes cluster. The application was able to connect to the database seamlessly, demonstrating the capability to manage data effectively. The security measures implemented via Kubernetes secrets performed as expected, safeguarding access credentials.

Database created with query of John



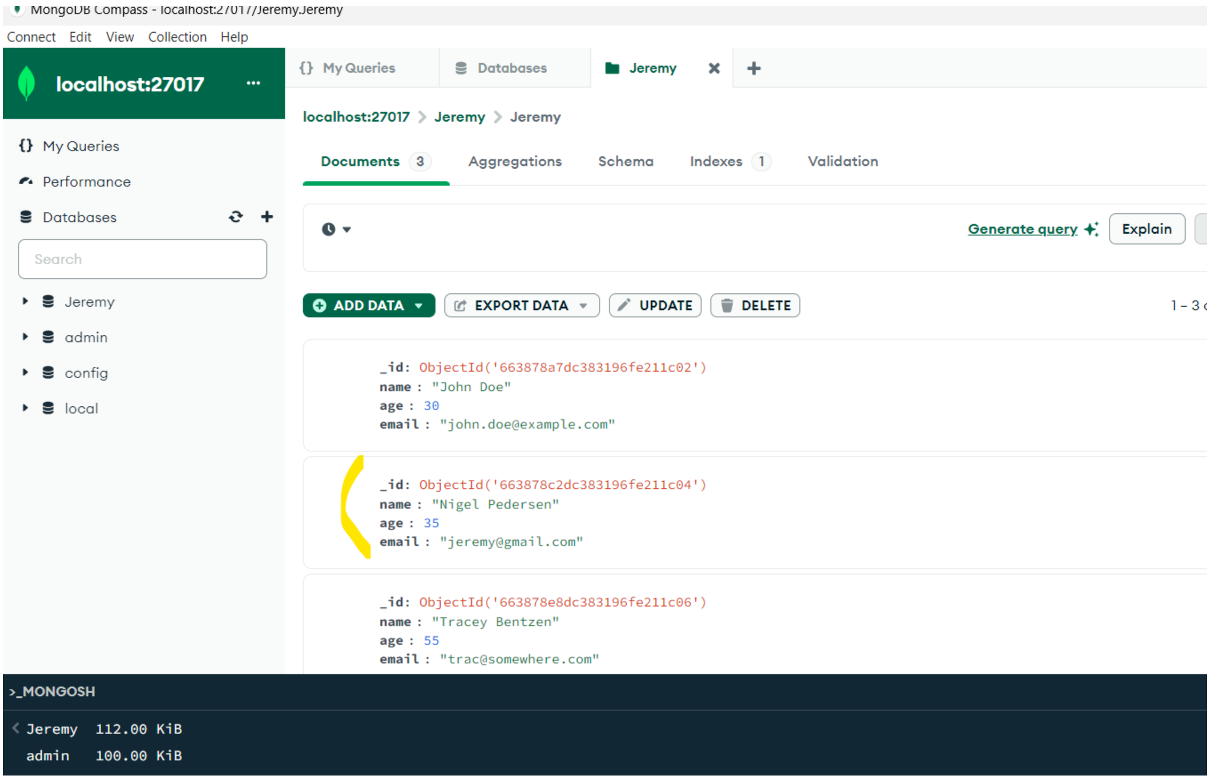
```
{ "name": "John Doe" }
```

```
_id: ObjectId('663878a7dc383196fe211c02')
name : "John Doe"
age : 30
email : "john.doe@example.com"
```

>_MONGOSH

```
> show dbs
< Jeremy   112.00 KiB
  admin    100.00 KiB
  config   108.00 KiB
  local     72.00 KiB
test>
```

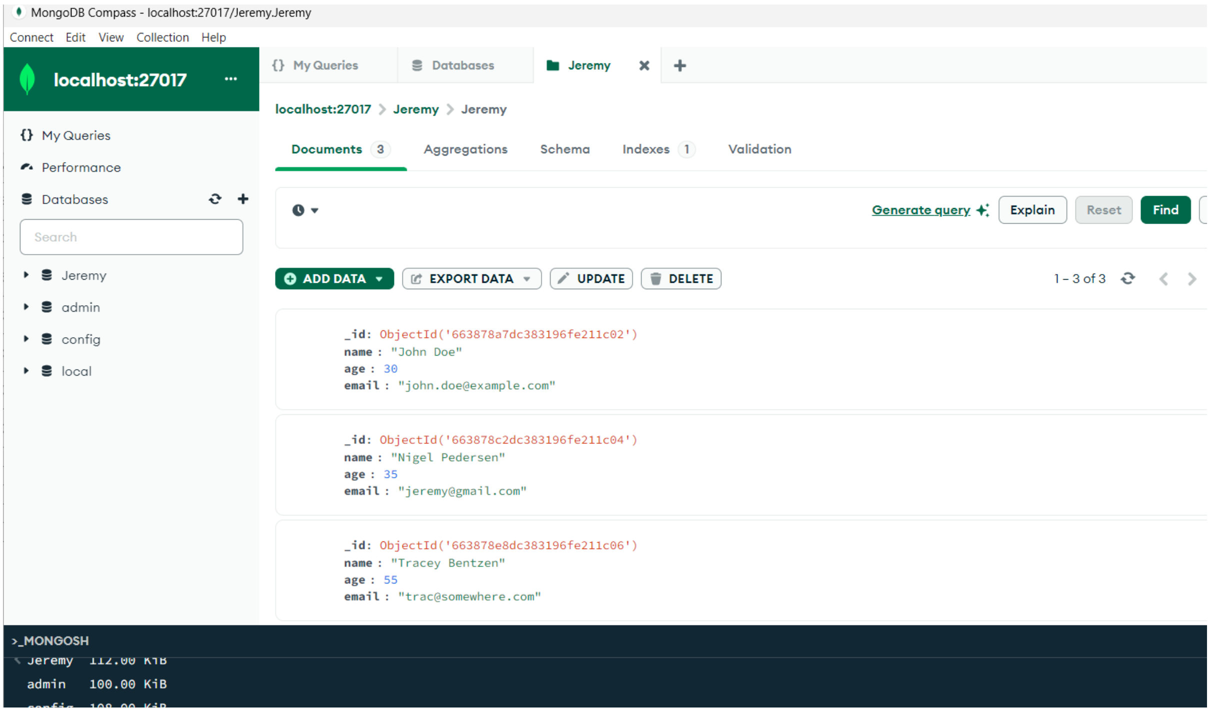
Updated Jeremy Pedersen to Nigel Pedersen



```
_id: ObjectId('663878a7dc383196fe211c02')
name : "John Doe"
age : 30
email : "john.doe@example.com"

_id: ObjectId('663878c2dc383196fe211c04')
name : "Nigel Pedersen"
age : 35
email : "jeremy@gmail.com"

_id: ObjectId('663878e8dc383196fe211c06')
name : "Tracey Bentzen"
age : 55
email : "trac@somewhere.com"
```

>_MONGOSH

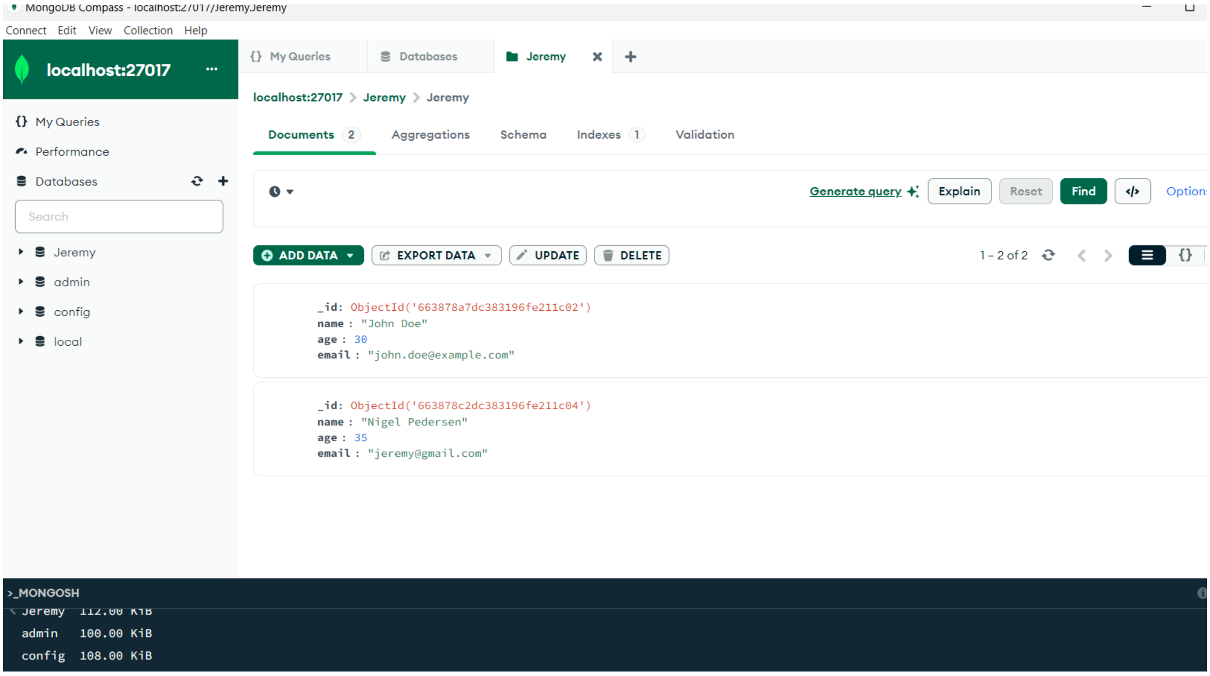
```
< Jeremy   112.00 KiB
  admin    100.00 KiB
```



```
_id: ObjectId('663878a7dc383196fe211c02')
name : "John Doe"
age : 30
email : "john.doe@example.com"

_id: ObjectId('663878c2dc383196fe211c04')
name : "Nigel Pedersen"
age : 35
email : "jeremy@gmail.com"

_id: ObjectId('663878e8dc383196fe211c06')
name : "Tracey Bentzen"
age : 55
email : "trac@somewhere.com"
```

```
  Jeremy   112.00 KiB
  admin    100.00 KiB
```

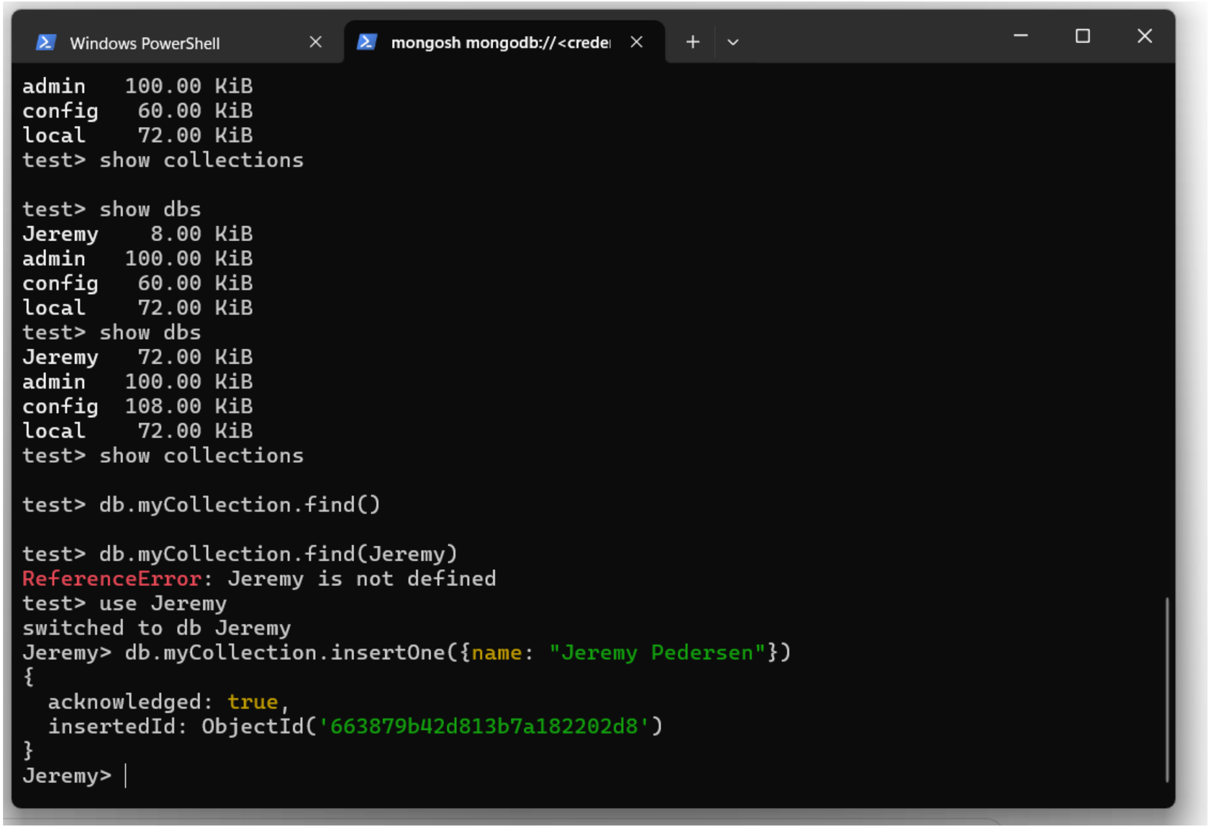
Entry Tracey Bentzen deleted



```
_id: ObjectId('663878a7dc383196fe211c02')
name : "John Doe"
age : 30
email : "john.doe@example.com"

_id: ObjectId('663878c2dc383196fe211c04')
name : "Nigel Pedersen"
age : 35
email : "jeremy@gmail.com"
```

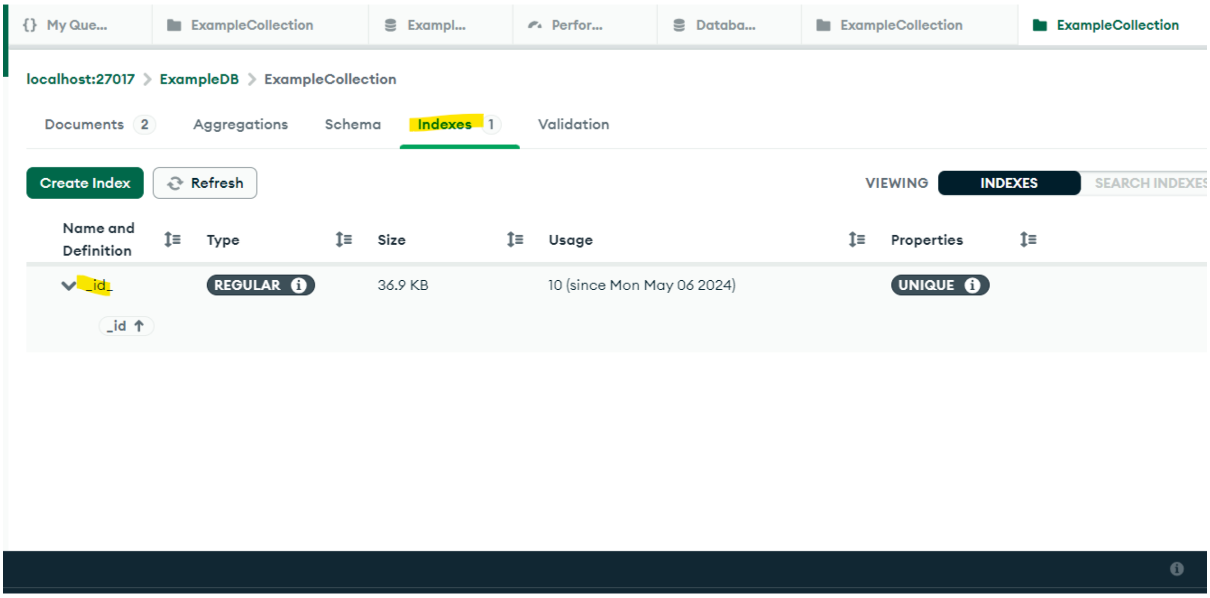
```
  Jeremy   112.00 KiB
  admin    100.00 KiB
  config   108.00 KiB
```

Connected to database Jeremy and found record (document) Jeremy Pedersen



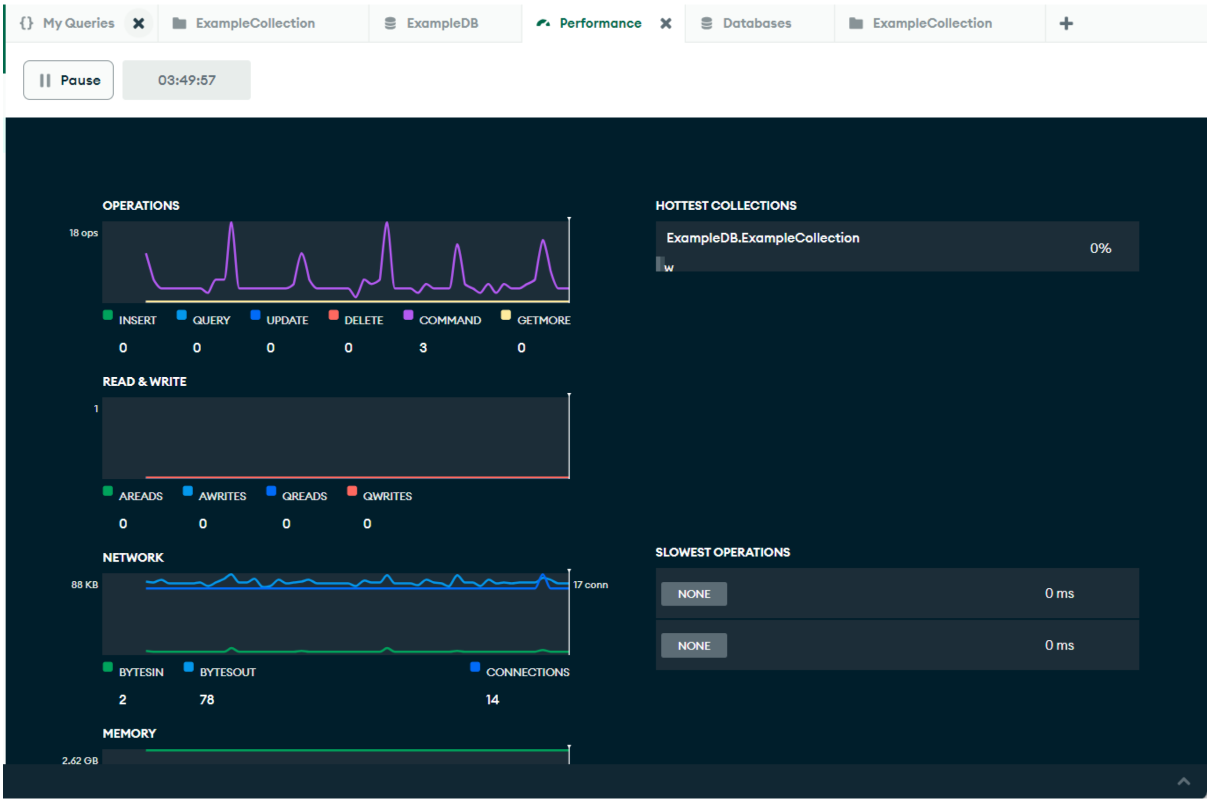
```
admin   100.00 KiB
config   60.00 KiB
local    72.00 KiB
test> show collections

test> show dbs
Jeremy    8.00 KiB
admin   100.00 KiB
config   60.00 KiB
local    72.00 KiB
test> show dbs
Jeremy   72.00 KiB
admin   100.00 KiB
config  108.00 KiB
local    72.00 KiB
test> show collections

test> db.myCollection.find()

test> db.myCollection.find(Jeremy)
ReferenceError: Jeremy is not defined
test> use Jeremy
switched to db Jeremy
Jeremy> db.myCollection.insertOne({name: "Jeremy Pedersen"})
{
  acknowledged: true,
  insertedId: ObjectId('663879b42d813b7a182202d8')
}
Jeremy> 
```

Index added to ID



Name and Definition	Type	Size	Usage	Properties
_id _id ↑	REGULAR ⓘ	36.9 KB	10 (since Mon May 06 2024)	UNIQUE ⓘ

Performance Monitor



OPERATIONS

INSERT 0
QUERY 0
UPDATE 0
DELETE 0
COMMAND 3
GETMORE 0

READ & WRITE

AREADS 0
AWRITES 0
QREADS 0
QWRITES 0

NETWORK

BYTESIN 2
BYTESOUT 78
CONNECTIONS 14

MEMORY

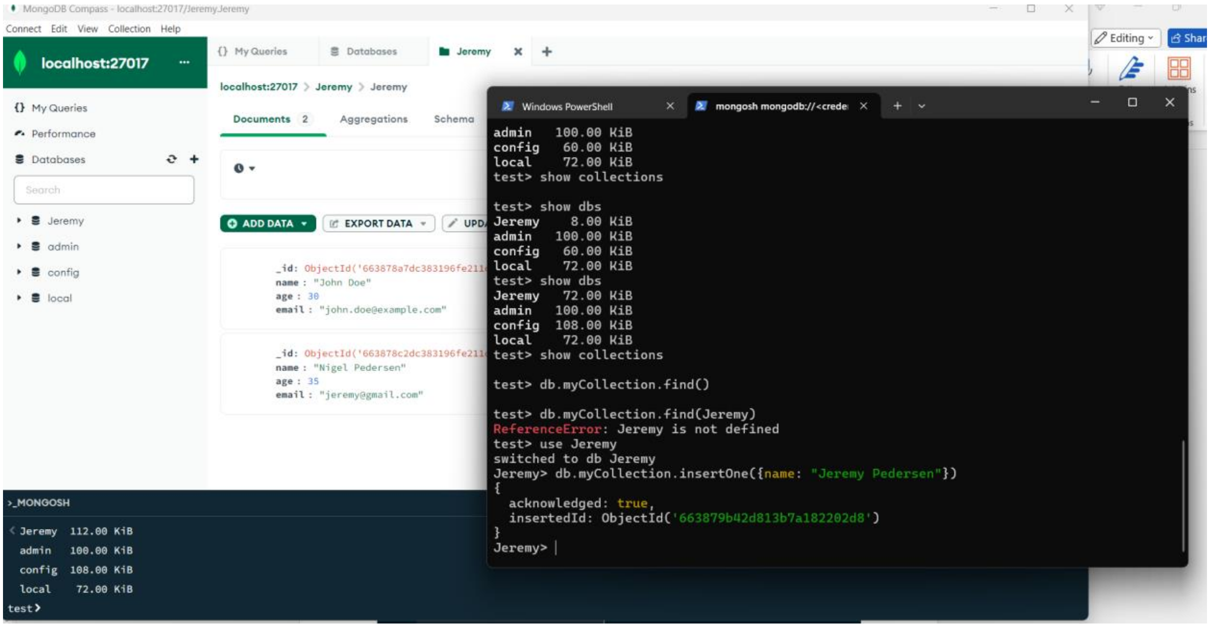
HOTTEST COLLECTIONS

ExampleDB.ExampleCollection 0%
w

SLOWEST OPERATIONS

NONE 0 ms
NONE 0 ms

You can see my connections

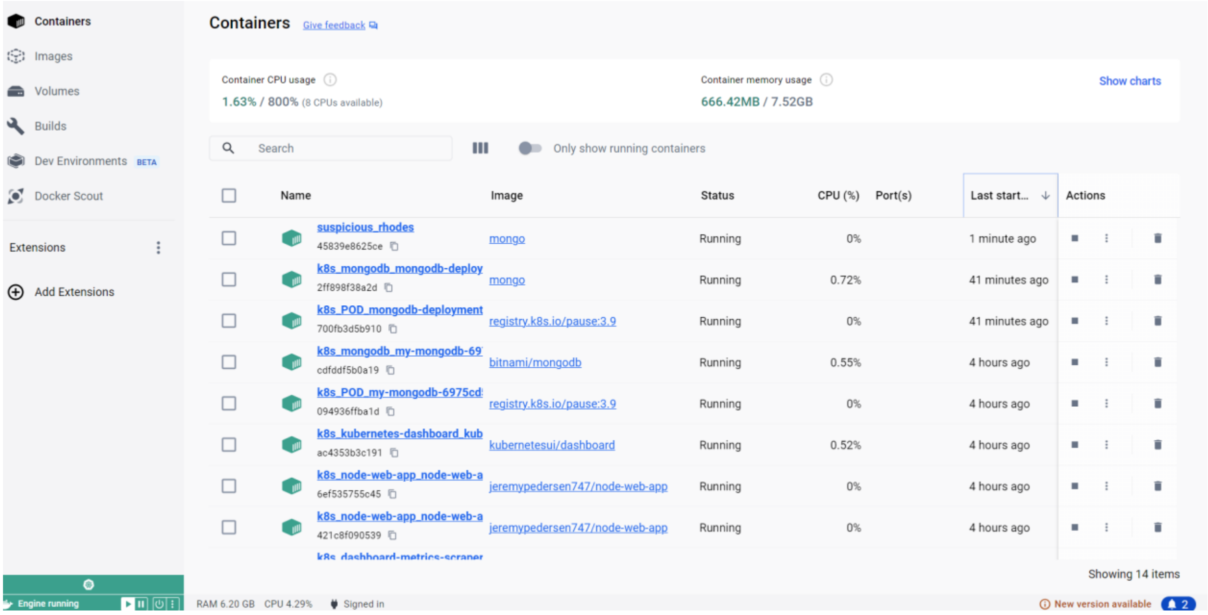


```
_id: ObjectId('663878a7dc383196fe211
name : "John Doe"
age : 30
email : "john.doe@example.com"

_id: ObjectId('663878c2dc383196fe211
name : "Nigel Pedersen"
age : 35
email : "jeremy@gmail.com"
```

```
>_MONGOSH
< Jeremy   112.00 KiB
  admin    100.00 KiB
  config   108.00 KiB
  local     72.00 KiB
test>
```

```
admin   100.00 KiB
config   60.00 KiB
local    72.00 KiB
test> show collections

test> show dbs
Jeremy    8.00 KiB
admin   100.00 KiB
config   60.00 KiB
local    72.00 KiB
test> show dbs
Jeremy   72.00 KiB
admin   100.00 KiB
config  108.00 KiB
local    72.00 KiB
test> show collections

test> db.myCollection.find()

test> db.myCollection.find(Jeremy)
ReferenceError: Jeremy is not defined
test> use Jeremy
switched to db Jeremy
Jeremy> db.myCollection.insertOne({name: "Jeremy Pedersen"})
{
  acknowledged: true,
  insertedId: ObjectId('663879b42d813b7a182202d8')
}
Jeremy> |
```

Container in Docker



Containers Give feedback

Container CPU usage
1.63% / 800% (8 CPUs available)

Container memory usage
666.42MB / 7.52GB

Show charts

Search

Only show running containers

□	Name	Image	Status	CPU (%)	Port(s)	Last start...	Actions
□	suspicious_rhodes 45839e8625ce	mongo	Running	0%		1 minute ago	
□	k8s_mongodb_mongodb-deploy 2ff898f38a2d	mongo	Running	0.72%		41 minutes ago	
□	k8s_POD_mongodb-deployment 700fb3d5b910	registry.k8s.io/pause:3.9	Running	0%		41 minutes ago	
□	k8s_mongodb_my-mongodb-69 cdfddf5b0a19	bitnami/mongodb	Running	0.55%		4 hours ago	
□	k8s_POD_my-mongodb-6975cd 094936ffba1d	registry.k8s.io/pause:3.9	Running	0%		4 hours ago	
□	k8s_kubernetes-dashboard_kub ac4353b3c191	kubernetesui/dashboard	Running	0.52%		4 hours ago	
□	k8s_node-web-app_node-web-a 6ef535755c45	jeremypedersen747/node-web-app	Running	0%		4 hours ago	
□	k8s_node-web-app_node-web-a 421c8f090539	jeremypedersen747/node-web-app	Running	0%		4 hours ago	
	k8s_dashboard-metrics-scraper						

Showing 14 items

RAM 6.20 GB CPU 4.29% Signed in

New version available 2