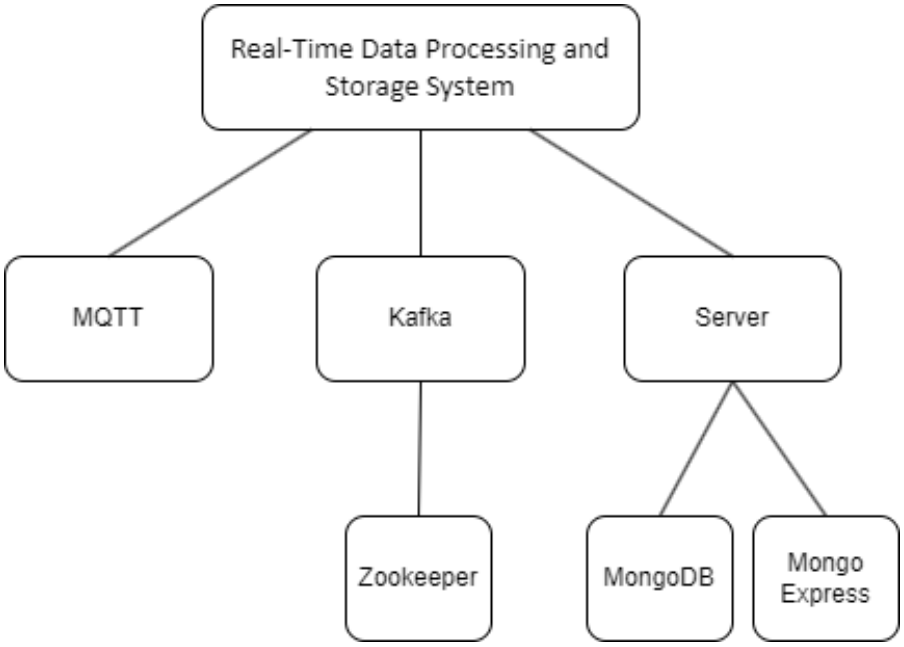
Design and Implementation

1. Architecture



2. Services

2.1 MongoDB Service

The MongoDB service is created using the official MongoDB image. It is crucial for data storage within the system. By defining the root username and password in the environment variables, the service ensures secure access to the database. MongoDB acts as the primary data repository where information collected by the sensor service and processed by the server service is stored for further analysis and retrieval.

2.2 MQTT Service

The MQTT service utilizes the Eclipse Mosquitto image to establish an MQTT broker. This component is essential for handling messaging within the system. By setting up volumes and ports, the service facilitates communication between various components by enabling the exchange of real-time data streams. Mosquitto serves as the messaging backbone, allowing seamless interaction between different services.

2.3 Server Service

The server service, constructed from a Dockerfile in the Server context, acts as the core processing unit within the system. By linking to Mosquitto, MongoDB, and Kafka, it establishes connections to key components for data processing and analysis. The server service processes incoming data streams, conducts real-time analytics, and stores results in the MongoDB database. It serves as the central hub for data manipulation and decision-making.

2.4 Mongo Service

MongoDB Express Service provides a user-friendly web interface for managing the MongoDB database. By configuring the connection settings to the MongoDB service, it allows users to interact with the database visually. This service simplifies database management tasks such as querying, updating, and deleting data, providing a convenient way to monitor and administer the stored information.

2.5 Kafka Service

The Kafka service, based on the Confluent Kafka image, enables real-time data streaming and processing. By connecting to Zookeeper for coordination, Kafka establishes a robust messaging system for handling data streams efficiently. With defined environment variables and port mappings, Kafka ensures high-throughput data transfer and fault-tolerant message processing, supporting the real-time analytics and decision-making processes within the system.

2.6 Zookeeper Service

Zookeeper, deployed using the Confluent Zookeeper image, serves as the distributed coordination platform for the system. By defining specific environment variables and ports, Zookeeper ensures seamless communication and synchronization among the various services. It plays a pivotal role in maintaining consistency and managing distributed resources across the system.

2.7 Sensor Service (for simulation)

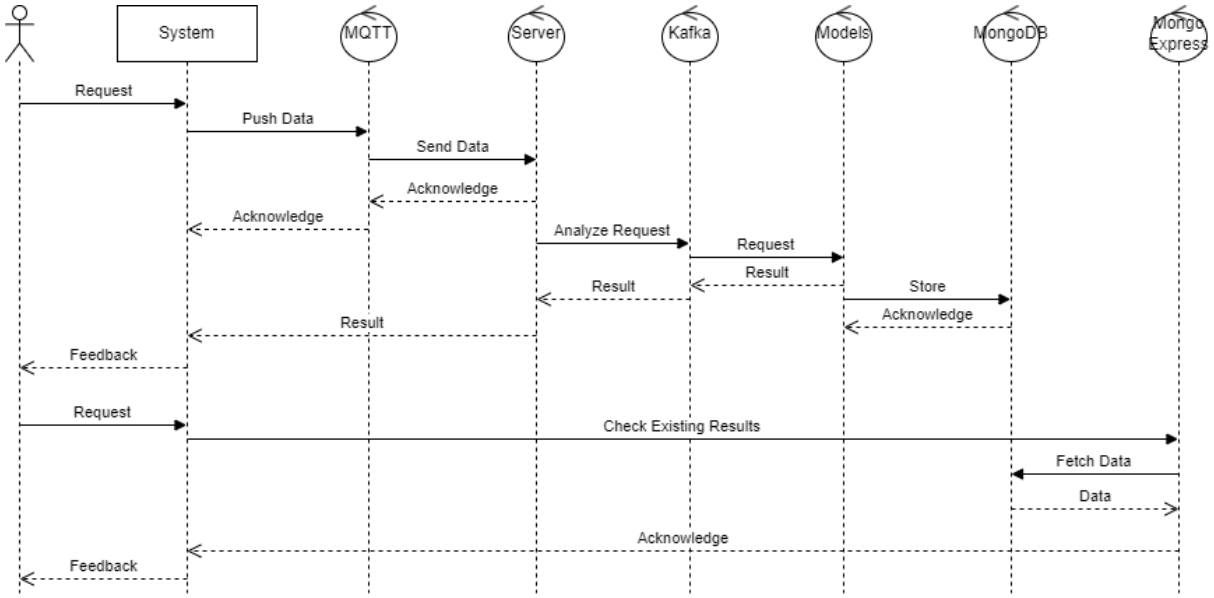
The sensor service is built from a custom Dockerfile within the Data_collection context. It is responsible for collecting real-time data from sensors and devices. By linking to the MQTT service, the sensor service can publish data to specific MQTT topics for consumption by other services. This component plays a vital role in the initial data acquisition stage of the system.

2.8 Model Analysis Service(for simulation)

The model analysis service, built from a Dockerfile in the Models_analysis context, focuses on data analysis and modeling tasks. By depending on Kafka for data ingestion and processing, this service is dedicated to running complex algorithms, statistical models, and machine learning processes on the incoming data streams. It plays a critical role in extracting insights and patterns from the real-time data for decision support.

3. Relationships of Service

3.1 Sequence Diagram



The relationships between the services in the Real-time Data Processing and Storage System are crucial for seamless operation and data flow.

3.2 Between Sensor Service and MQTT Service

The sensor service publishes data to MQTT topics on the Mosquitto broker, enabling other services to subscribe to and process the incoming data streams.

3.3 Between **Server Service** and MQTT, MongoDB, Kafka Services

The server service connects to Mosquitto for data ingestion, MongoDB for data storage, and Kafka for real-time data streaming and processing. It acts as the central processing unit that orchestrates data flow between these services.

3.4 Between Model Analysis Service and Kafka Service

The model analysis service relies on Kafka for receiving real-time data streams, processing them through analytical models, and generating insights. Kafka serves as the messaging backbone for data exchange between the model analysis service and other components.

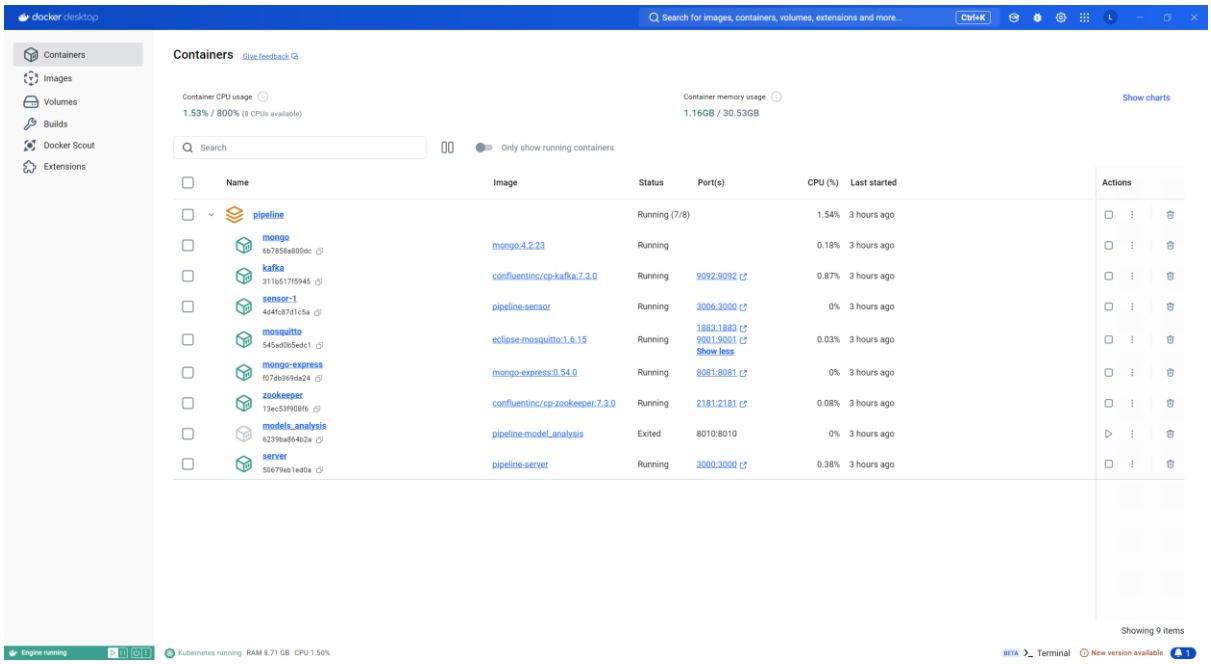
3.5 Between Mongo Express Service and MongoDB Service

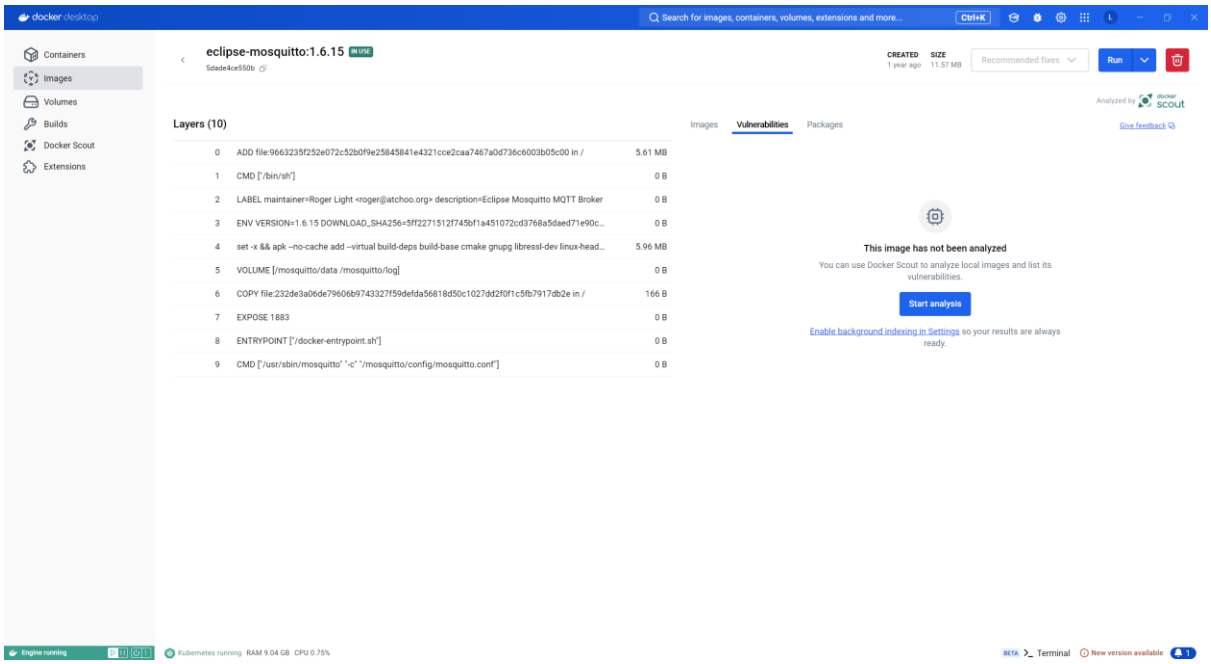
MongoDB Express interfaces with the MongoDB service, providing a visual management tool for interacting with the database. It enhances the usability and accessibility of the MongoDB database for administrators and users.

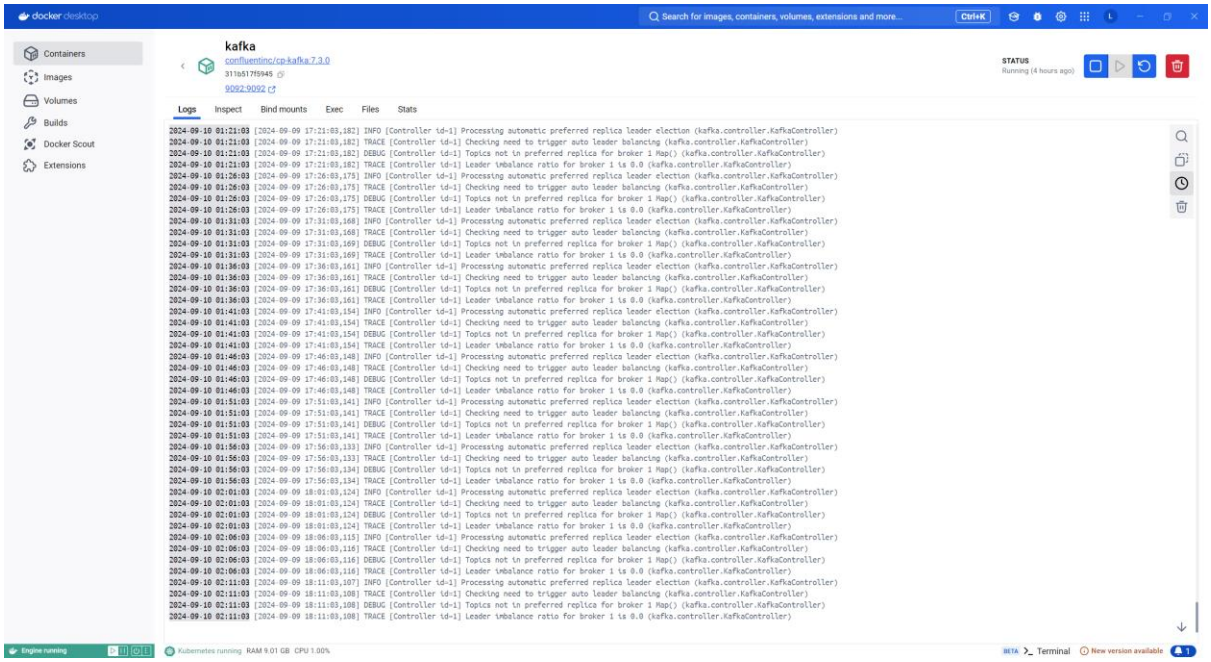
3.6 Between Kafka Service and Zookeeper Service

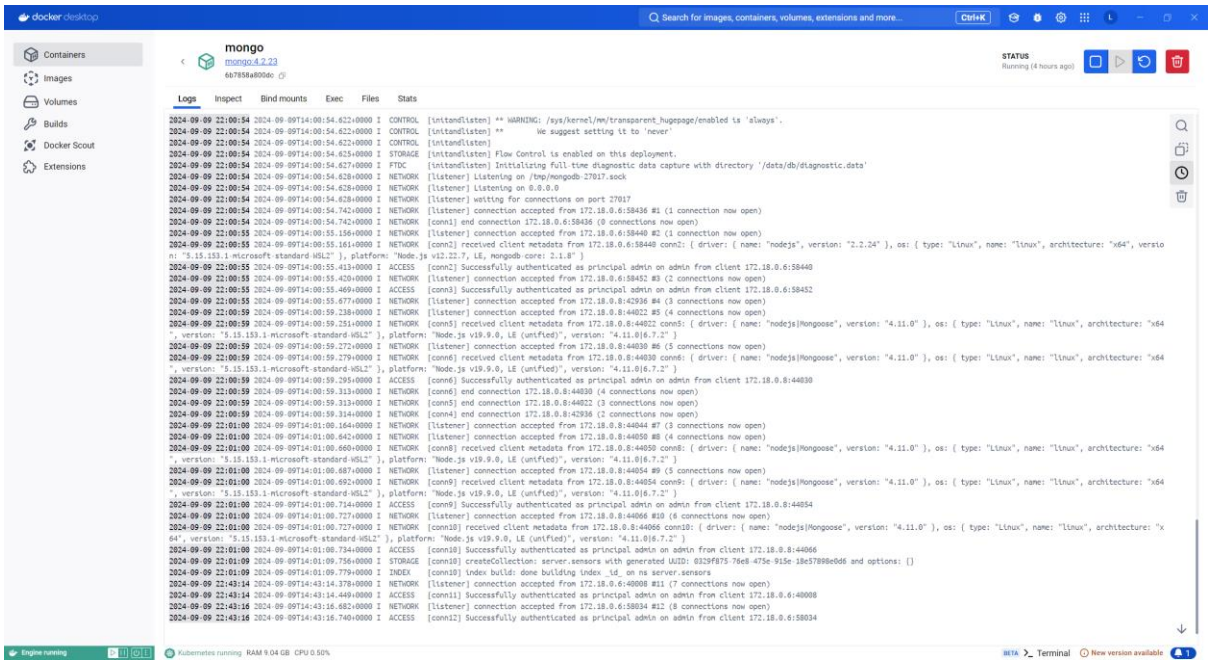
Kafka depends on Zookeeper for distributed coordination and management of topics, partitions, and offsets. Zookeeper ensures the reliability and consistency of Kafka's messaging system, facilitating seamless data streaming and processing.

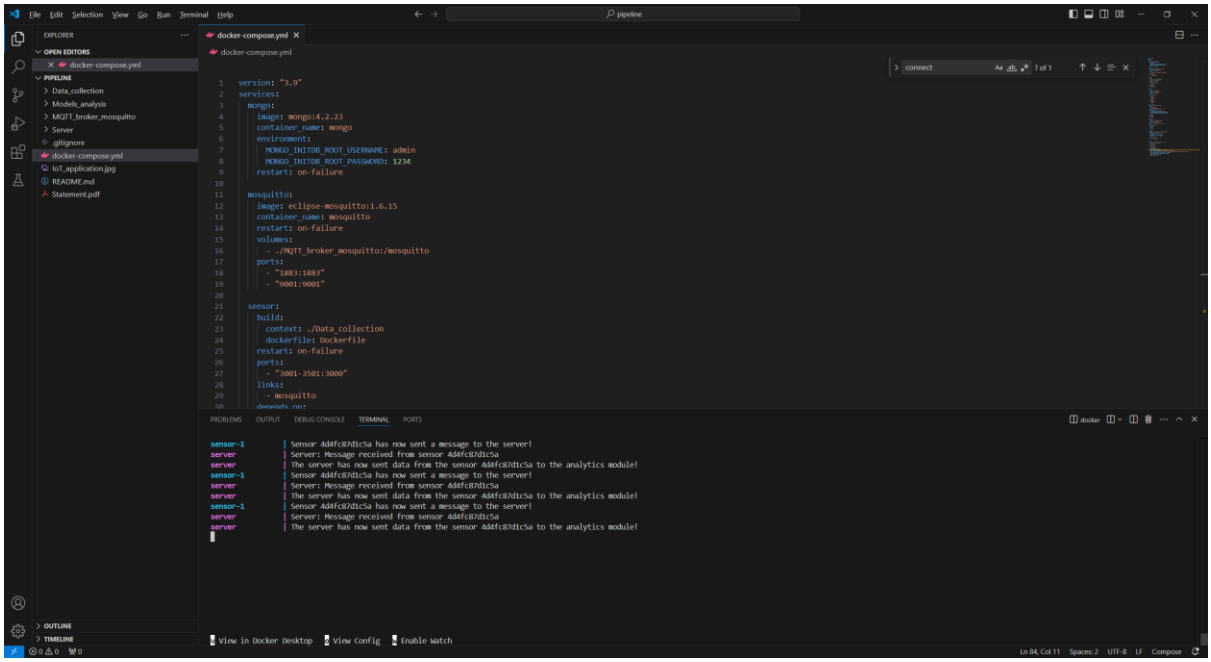
4. Implementation Results and Demonstration

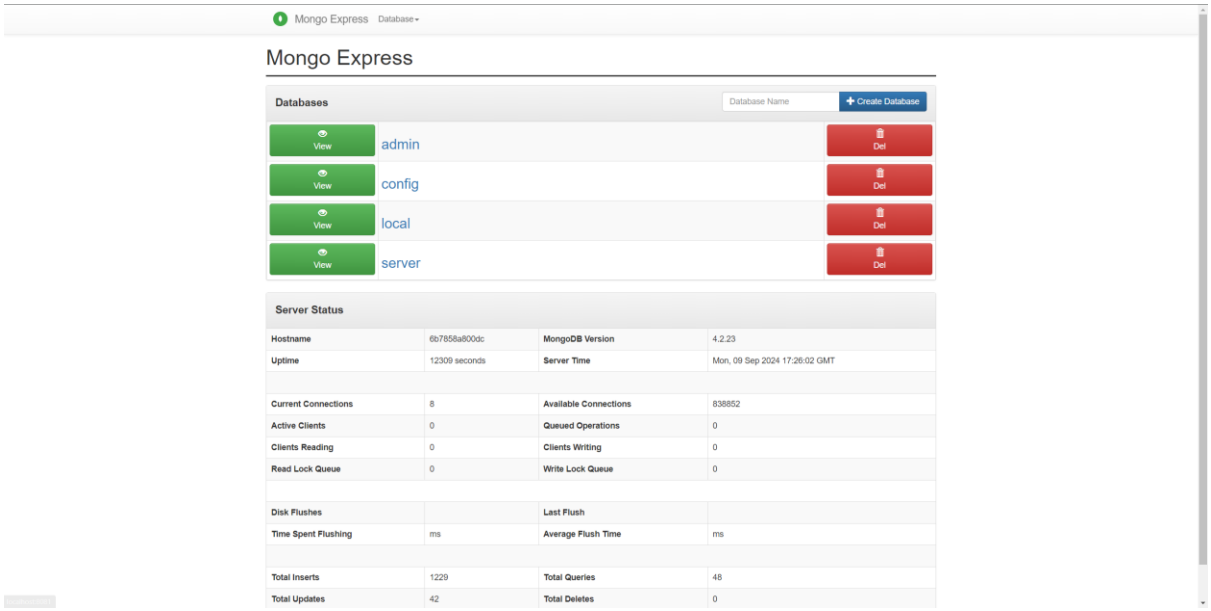


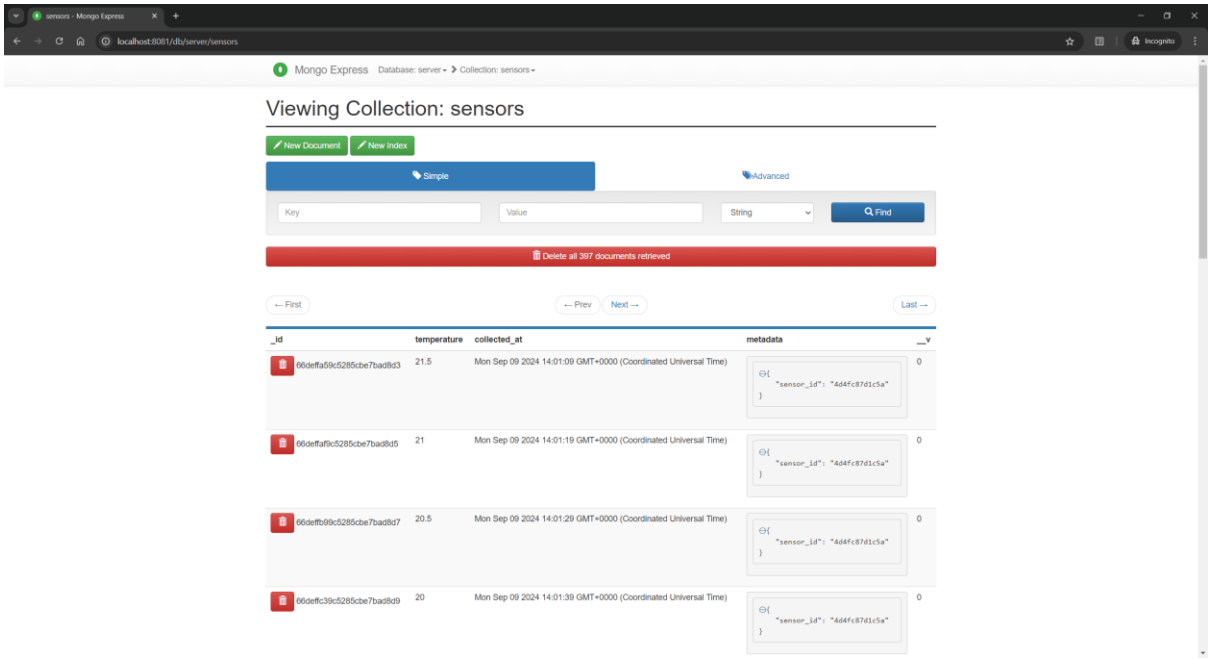



docker desktop
Search for images, containers, volumes, extensions and more...
Ctrl+K
Containers
Images
Volumes
Builds
Docker Scout
Extensions
kafka
confluentinc/cp-kafka:7.3.0
311b5177f5945
9092:9092
STATUS
Running (4 hours ago)
Logs
Inspect
Bind mounts
Exec
Files
Stats
2024-09-10 01:21:03 [2024-09-09 17:21:03,182] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:21:03 [2024-09-09 17:21:03,182] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:21:03 [2024-09-09 17:21:03,182] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:21:03 [2024-09-09 17:21:03,182] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:26:03 [2024-09-09 17:26:03,175] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:26:03 [2024-09-09 17:26:03,175] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:26:03 [2024-09-09 17:26:03,175] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:26:03 [2024-09-09 17:26:03,175] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:31:03 [2024-09-09 17:31:03,168] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:31:03 [2024-09-09 17:31:03,168] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:31:03 [2024-09-09 17:31:03,169] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:31:03 [2024-09-09 17:31:03,169] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:36:03 [2024-09-09 17:36:03,161] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:36:03 [2024-09-09 17:36:03,161] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:36:03 [2024-09-09 17:36:03,161] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:36:03 [2024-09-09 17:36:03,161] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:41:03 [2024-09-09 17:41:03,154] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:41:03 [2024-09-09 17:41:03,154] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:41:03 [2024-09-09 17:41:03,154] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:41:03 [2024-09-09 17:41:03,154] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:46:03 [2024-09-09 17:46:03,148] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:46:03 [2024-09-09 17:46:03,148] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:46:03 [2024-09-09 17:46:03,148] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:46:03 [2024-09-09 17:46:03,148] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:51:03 [2024-09-09 17:51:03,141] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:51:03 [2024-09-09 17:51:03,141] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:51:03 [2024-09-09 17:51:03,141] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:51:03 [2024-09-09 17:51:03,141] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 01:56:03 [2024-09-09 17:56:03,133] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 01:56:03 [2024-09-09 17:56:03,133] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 01:56:03 [2024-09-09 17:56:03,134] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 01:56:03 [2024-09-09 17:56:03,134] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 02:01:03 [2024-09-09 18:01:03,124] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 02:01:03 [2024-09-09 18:01:03,124] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 02:01:03 [2024-09-09 18:01:03,124] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 02:01:03 [2024-09-09 18:01:03,124] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 02:06:03 [2024-09-09 18:06:03,115] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 02:06:03 [2024-09-09 18:06:03,116] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 02:06:03 [2024-09-09 18:06:03,116] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 02:06:03 [2024-09-09 18:06:03,116] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
2024-09-10 02:11:03 [2024-09-09 18:11:03,107] INFO [Controller id=1] Processing automatic preferred replica leader election (kafka.controller.KafkaController)
2024-09-10 02:11:03 [2024-09-09 18:11:03,108] TRACE [Controller id=1] Checking need to trigger auto leader balancing (kafka.controller.KafkaController)
2024-09-10 02:11:03 [2024-09-09 18:11:03,108] DEBUG [Controller id=1] Topics not in preferred replica for broker 1 Map() (kafka.controller.KafkaController)
2024-09-10 02:11:03 [2024-09-09 18:11:03,108] TRACE [Controller id=1] Leader imbalance ratio for broker 1 is 0.0 (kafka.controller.KafkaController)
Engine running
Kubernetes running
RAM 9.01 GB
CPU 1.00%
Terminal
New version available


docker desktop
Search for images, containers, volumes, extensions and more...
Ctrl+K
Containers
Images
Volumes
Builds
Docker Scout
Extensions
mongo
mongo:4.2.23
6b7858a800dc
STATUS
Running (4 hours ago)
Logs
Inspect
Bind mounts
Exec
Files
Stats
2024-09-09 22:00:54 2024-09-09T14:00:54.622+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2024-09-09 22:00:54 2024-09-09T14:00:54.622+0000 I CONTROL [initandlisten] ** We suggest setting it to 'never'
2024-09-09 22:00:54 2024-09-09T14:00:54.622+0000 I CONTROL [initandlisten]
2024-09-09 22:00:54 2024-09-09T14:00:54.625+0000 I STORAGE [initandlisten] Flow Control is enabled on this deployment.
2024-09-09 22:00:54 2024-09-09T14:00:54.627+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/data/db/diagnostic.data'
2024-09-09 22:00:54 2024-09-09T14:00:54.628+0000 I NETWORK [listener] Listening on /tmp/mongodb-27017.sock
2024-09-09 22:00:54 2024-09-09T14:00:54.628+0000 I NETWORK [listener] Listening on 0.0.0.0
2024-09-09 22:00:54 2024-09-09T14:00:54.628+0000 I NETWORK [listener] waiting for connections on port 27017
2024-09-09 22:00:54 2024-09-09T14:00:54.742+0000 I NETWORK [listener] connection accepted from 172.18.0.6:58436 #1 (1 connection now open)
2024-09-09 22:00:54 2024-09-09T14:00:54.742+0000 I NETWORK [conn1] end connection 172.18.0.6:58436 (0 connections now open)
2024-09-09 22:00:55 2024-09-09T14:00:55.156+0000 I NETWORK [listener] connection accepted from 172.18.0.6:58440 #2 (1 connection now open)
2024-09-09 22:00:55 2024-09-09T14:00:55.161+0000 I NETWORK [conn2] received client metadata from 172.18.0.6:58440 conn2: { driver: { name: "nodejs", version: "2.2.24" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "5.15.153.1-microsoft-standard-WSL2" }, platform: "Node.js v12.22.7, LE, mongodb-core: 2.1.8" }
2024-09-09 22:00:55 2024-09-09T14:00:55.413+0000 I ACCESS [conn2] Successfully authenticated as principal admin on admin from client 172.18.0.6:58440
2024-09-09 22:00:55 2024-09-09T14:00:55.420+0000 I NETWORK [listener] connection accepted from 172.18.0.6:58452 #3 (2 connections now open)
2024-09-09 22:00:55 2024-09-09T14:00:55.469+0000 I ACCESS [conn3] Successfully authenticated as principal admin on admin from client 172.18.0.6:58452
2024-09-09 22:00:55 2024-09-09T14:00:55.677+0000 I NETWORK [listener] connection accepted from 172.18.0.8:42936 #4 (3 connections now open)
2024-09-09 22:00:59 2024-09-09T14:00:59.238+0000 I NETWORK [listener] connection accepted from 172.18.0.8:44022 #5 (4 connections now open)
2024-09-09 22:00:59 2024-09-09T14:00:59.251+0000 I NETWORK [conn5] received client metadata from 172.18.0.8:44022 conn5: { driver: { name: "nodejs|Mongoose", version: "4.11.0" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "5.15.153.1-microsoft-standard-WSL2" }, platform: "Node.js v19.9.0, LE (unified)", version: "4.11.0|6.7.2" }
2024-09-09 22:00:59 2024-09-09T14:00:59.272+0000 I NETWORK [listener] connection accepted from 172.18.0.8:44030 #6 (5 connections now open)
2024-09-09 22:00:59 2024-09-09T14:00:59.279+0000 I NETWORK [conn6] received client metadata from 172.18.0.8:44030 conn6: { driver: { name: "nodejs|Mongoose", version: "4.11.0" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "5.15.153.1-microsoft-standard-WSL2" }, platform: "Node.js v19.9.0, LE (unified)", version: "4.11.0|6.7.2" }
2024-09-09 22:00:59 2024-09-09T14:00:59.295+0000 I ACCESS [conn6] Successfully authenticated as principal admin on admin from client 172.18.0.8:44030
2024-09-09 22:00:59 2024-09-09T14:00:59.313+0000 I NETWORK [conn6] end connection 172.18.0.8:44030 (4 connections now open)
2024-09-09 22:00:59 2024-09-09T14:00:59.313+0000 I NETWORK [conn5] end connection 172.18.0.8:44022 (3 connections now open)
2024-09-09 22:00:59 2024-09-09T14:00:59.314+0000 I NETWORK [conn4] end connection 172.18.0.8:42936 (2 connections now open)
2024-09-09 22:01:00 2024-09-09T14:01:00.164+0000 I NETWORK [listener] connection accepted from 172.18.0.8:44044 #7 (3 connections now open)
2024-09-09 22:01:00 2024-09-09T14:01:00.642+0000 I NETWORK [listener] connection accepted from 172.18.0.8:44050 #8 (4 connections now open)
2024-09-09 22:01:00 2024-09-09T14:01:00.660+0000 I NETWORK [conn8] received client metadata from 172.18.0.8:44050 conn8: { driver: { name: "nodejs|Mongoose", version: "4.11.0" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "5.15.153.1-microsoft-standard-WSL2" }, platform: "Node.js v19.9.0, LE (unified)", version: "4.11.0|6.7.2" }
2024-09-09 22:01:00 2024-09-09T14:01:00.687+0000 I NETWORK [listener] connection accepted from 172.18.0.8:44054 #9 (5 connections now open)
2024-09-09 22:01:00 2024-09-09T14:01:00.692+0000 I NETWORK [conn9] received client metadata from 172.18.0.8:44054 conn9: { driver: { name: "nodejs|Mongoose", version: "4.11.0" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "5.15.153.1-microsoft-standard-WSL2" }, platform: "Node.js v19.9.0, LE (unified)", version: "4.11.0|6.7.2" }
2024-09-09 22:01:00 2024-09-09T14:01:00.714+0000 I ACCESS [conn9] Successfully authenticated as principal admin on admin from client 172.18.0.8:44054
2024-09-09 22:01:00 2024-09-09T14:01:00.727+0000 I NETWORK [listener] connection accepted from 172.18.0.8:44066 #10 (6 connections now open)
2024-09-09 22:01:00 2024-09-09T14:01:00.727+0000 I NETWORK [conn10] received client metadata from 172.18.0.8:44066 conn10: { driver: { name: "nodejs|Mongoose", version: "4.11.0" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "5.15.153.1-microsoft-standard-WSL2" }, platform: "Node.js v19.9.0, LE (unified)", version: "4.11.0|6.7.2" }
2024-09-09 22:01:00 2024-09-09T14:01:00.734+0000 I ACCESS [conn10] Successfully authenticated as principal admin on admin from client 172.18.0.8:44066
2024-09-09 22:01:09 2024-09-09T14:01:09.756+0000 I STORAGE [conn10] createCollection: server.sensors with generated UUID: 0329f875-76e8-475e-915e-18e57898e0d6 and options: {}
2024-09-09 22:01:09 2024-09-09T14:01:09.779+0000 I INDEX [conn10] index build: done building index _id_ on ns server.sensors
2024-09-09 22:43:14 2024-09-09T14:43:14.378+0000 I NETWORK [listener] connection accepted from 172.18.0.6:40008 #11 (7 connections now open)
2024-09-09 22:43:14 2024-09-09T14:43:14.449+0000 I ACCESS [conn11] Successfully authenticated as principal admin on admin from client 172.18.0.6:40008
2024-09-09 22:43:16 2024-09-09T14:43:16.682+0000 I NETWORK [listener] connection accepted from 172.18.0.6:58034 #12 (8 connections now open)
2024-09-09 22:43:16 2024-09-09T14:43:16.740+0000 I ACCESS [conn12] Successfully authenticated as principal admin on admin from client 172.18.0.6:58034
Engine running
Kubernetes running
RAM 9.04 GB
CPU 0.50%
Terminal
New version available


File Edit Selection View Go Run Terminal Help
pipeline
EXPLORER
OPEN EDITORS
docker-compose.yml
PIPELINE
Data_collection
Models_analysis
MQTT_broker_mosquitto
Server
.gitignore
docker-compose.yml
IoT_application.jpg
README.md
Statement.pdf
docker-compose.yml
connect
version: "3.9"
services:
mongo:
image: mongo:4.2.23
container_name: mongo
environment:
MONGO_INITDB_ROOT_USERNAME: admin
MONGO_INITDB_ROOT_PASSWORD: 1234
restart: on-failure
mosquitto:
image: eclipse-mosquitto:1.6.15
container_name: mosquitto
restart: on-failure
volumes:
- ./MQTT_broker_mosquitto:/mosquitto
ports:
- "1883:1883"
- "9001:9001"
sensor:
build:
context: ./Data_collection
dockerfile: Dockerfile
restart: on-failure
ports:
- "3001-3501:3000"
links:
- mosquitto
depends_on:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker
sensor-1 | Sensor 4d4fc87d1c5a has now sent a message to the server!
server | Server: Message received from sensor 4d4fc87d1c5a
server | The server has now sent data from the sensor 4d4fc87d1c5a to the analytics module!
sensor-1 | Sensor 4d4fc87d1c5a has now sent a message to the server!
server | Server: Message received from sensor 4d4fc87d1c5a
server | The server has now sent data from the sensor 4d4fc87d1c5a to the analytics module!
sensor-1 | Sensor 4d4fc87d1c5a has now sent a message to the server!
server | Server: Message received from sensor 4d4fc87d1c5a
server | The server has now sent data from the sensor 4d4fc87d1c5a to the analytics module!
OUTLINE
TIMELINE
View in Docker Desktop
View Config
Enable Watch
Ln 84, Col 11 Spaces: 2 UTF-8 LF Compose


Mongo Express
Database
Mongo Express
Databases
Database Name
Create Database
View
admin
Del
View
config
Del
View
local
Del
View
server
Del
Server Status
Hostname 6b7858a800dc
MongoDB Version 4.2.23
Uptime 12309 seconds
Server Time Mon, 09 Sep 2024 17:26:02 GMT
Current Connections 8
Available Connections 838852
Active Clients 0
Queued Operations 0
Clients Reading 0
Clients Writing 0
Read Lock Queue 0
Write Lock Queue 0
Disk Flushes
Last Flush
Time Spent Flushing ms
Average Flush Time ms
Total Inserts 1229
Total Queries 48
Total Updates 42
Total Deletes 0


sensors - Mongo Express
localhost:8081/db/server/sensors
Incognito
Mongo Express
Database: server
Collection: sensors
Viewing Collection: sensors
New Document
New Index
Simple
Advanced
Key
Value
String
Find
Delete all 397 documents retrieved
← First
← Prev
Next →
Last →
_id
temperature
collected_at
metadata
__v
66deffa69c5285cbe7bad8d3
21.5
Mon Sep 09 2024 14:01:09 GMT+0000 (Coordinated Universal Time)
{ "sensor_id": "4d4fc87d1c5a" }
0
66deffaf9c5285cbe7bad8d5
21
Mon Sep 09 2024 14:01:19 GMT+0000 (Coordinated Universal Time)
{ "sensor_id": "4d4fc87d1c5a" }
0
66deffb89c5285cbe7bad8d7
20.5
Mon Sep 09 2024 14:01:29 GMT+0000 (Coordinated Universal Time)
{ "sensor_id": "4d4fc87d1c5a" }
0
66deffc39c5285cbe7bad8d9
20
Mon Sep 09 2024 14:01:39 GMT+0000 (Coordinated Universal Time)
{ "sensor_id": "4d4fc87d1c5a" }
0