



**Hewlett Packard**  
Enterprise

# **HPCM MONITORING PIPELINE VISUALIZATION**

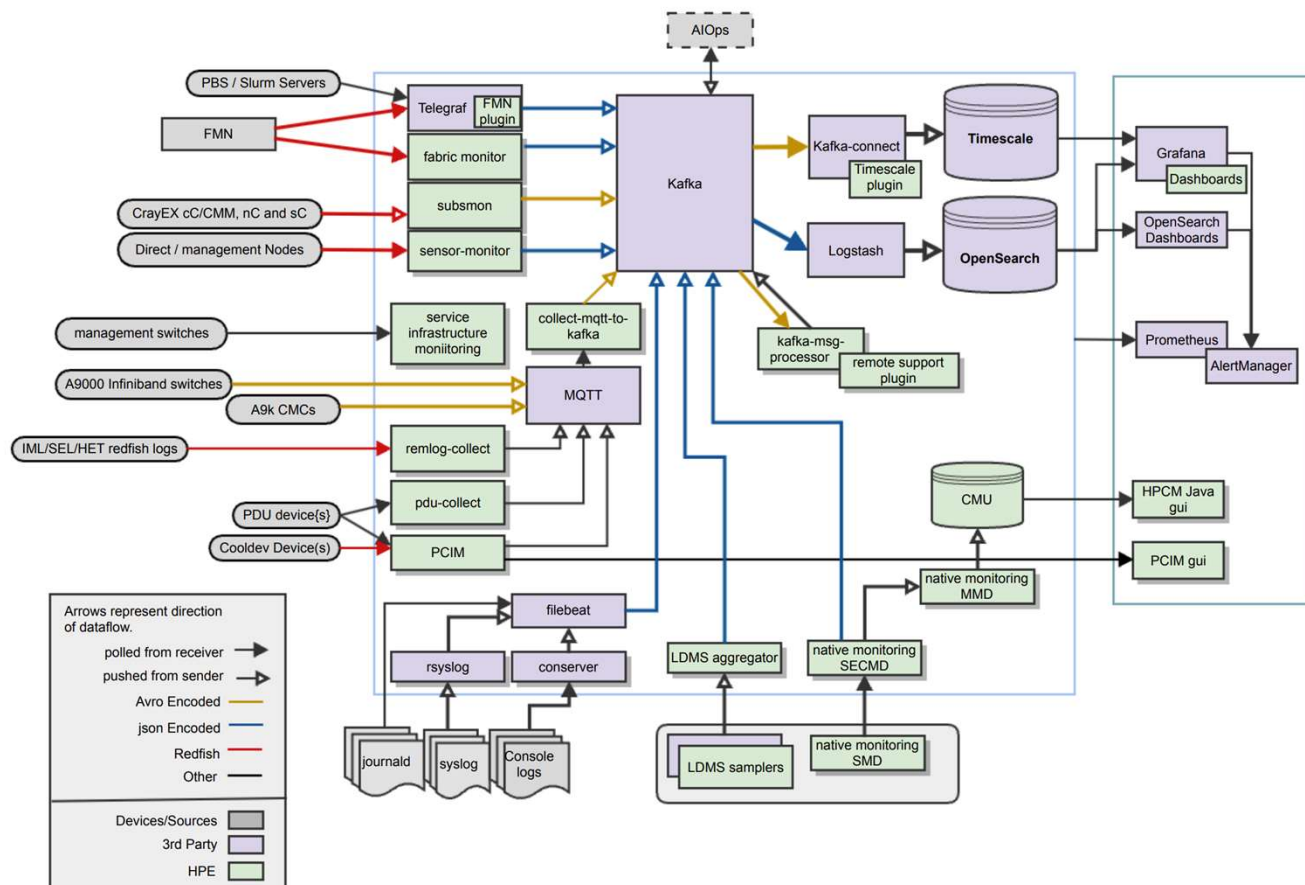
Vidyavardhaka College Of Engineering

CTY Members – Akshay G , Chandana L, Chinmai H K , Abhilash T R , Srivarshini S

Raghul Vasudevan

April 25 2024

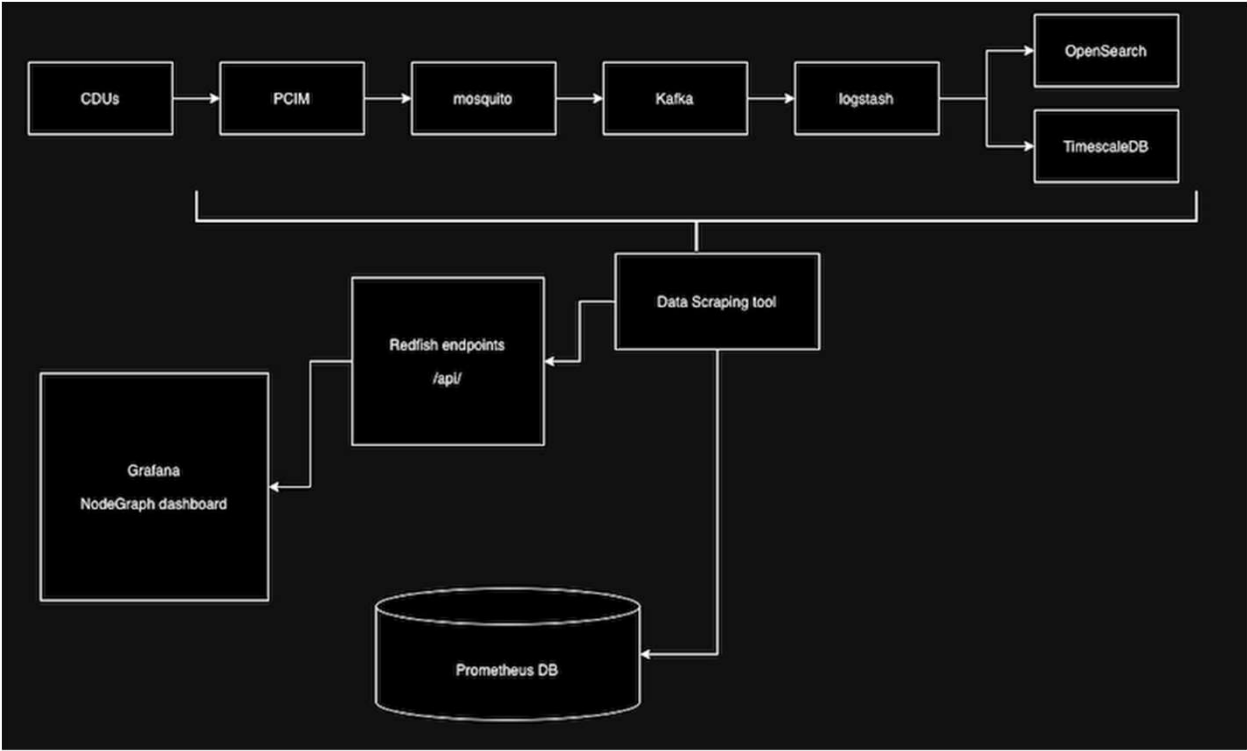
## PROBLEM STATEMENT



- Currently, the user must run different set of commands or mon service status check to debug the respective monitoring pipeline to find the issues.
- The end user/customer finds difficulty in understanding the HPCM monitoring stack and pipelines. (including Support and on-site folks).
- The purpose of this project is to reduce the difficulty in debugging the monitoring stack/components/services and pipelines for the customer and support team.

# PROJECT ARCHITECTURE, OBJECTIVES AND SKILL

## Architecture ( Diagram below )



### Objectives

1. Data scraping from components of the pipeline.
2. Real-time monitoring.
3. Visualization of status of the pipeline.

### Skill Set

1. Programming language - Python
2. Grafana : Nodegraph API Plugin
3. Linux Operating Environment
4. RedFish End-Point

# PROJECT TIMELINE , MILESTONE , STATUS

<div><div></div>Milestone-1<div></div>Milestone-2<div></div>Milestone-3<div></div>Milestone-4</div>				
<div><div></div>15//03<div></div>dd//mm<div></div>dd//mm<div></div>dd//mm</div>				
Milestone	Milestone Description	Tasks	Status	Next Steps / Issues
Milestone-1	Initial POC with one sample pipeline and minimal metrics	<div><div>1. Investigation</div><div>2. Design</div><div>3. Data Collection / Scrapping tool</div><div>4. Data Persistence method</div><div>5. Grafana NodeGraph dashboard</div></div>	<div><div>1. Completed the investigation on the data scrapping tool from the required components and written a method for data persistence and created a sample POC dashboard with NodeGraph.</div></div>	<div><div>Next:</div><div>1. Explore more on data persistence method with NodeGraph plugin</div><div>2. Further pipelines implementation</div><div>Issues:</div><div><div>• HTTP server hosting for data scrapping</div><div>• DB to Grafana NodeGraph visualization</div></div></div>
Milestone-2	Data flow Integration	<div><div>1. Grafana Upgrade</div><div>2. Data outflow check for each components</div><div>3. Logger</div><div>4. Deployment as service</div></div>	<div><div>1. Completed the Data outflow return functions for Kafka, mqtt, opensearch.</div><div>2. Completed logger as service deployment</div><div>3. Completed data flow edges in Grafana</div></div>	<div><div>1. Next:</div><div>2. Further enhancements and final output Grafana dashboard for Cooling devices pipeline</div></div>
Milestone-3				
Milestone-4				



# AGENDA



- About the Identified monitoring pipeline – CDU and Schema design
- Data collection / scrapping – components involved in pipeline
- Data persistence
- Grafana NodeGraph dashboard
- Demo



## SCHEMA DESIGN – CDU MONITORING PIPELINE

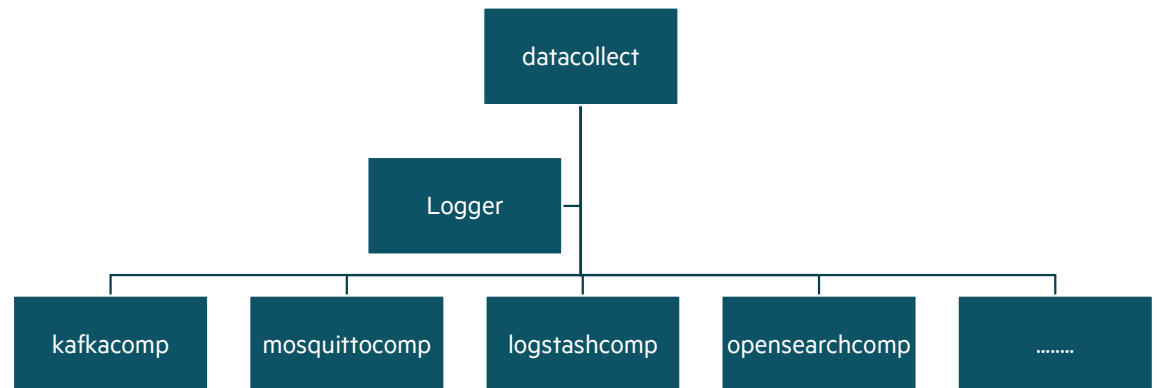
- Chosen metrics has been added in the data collection tool for POC.
- Further metrics will be added for every components later.

SERVICES	STATUS	UPTIME	DATA FLOW
PCIM	Running	Since 2 days	0/1
Mosquitto	Active/Inactive	24hrs	0/1
Kafka	Active/Inactive	Since 10 days, 23hrs	0/1
Logstash	Active/Inactive	Since 12 days, 10hrs	0/1
Opensearch	Active/Inactive	Since 25 days, 5hrs	0/1

# DATA COLLECTION/ SCRAPPING

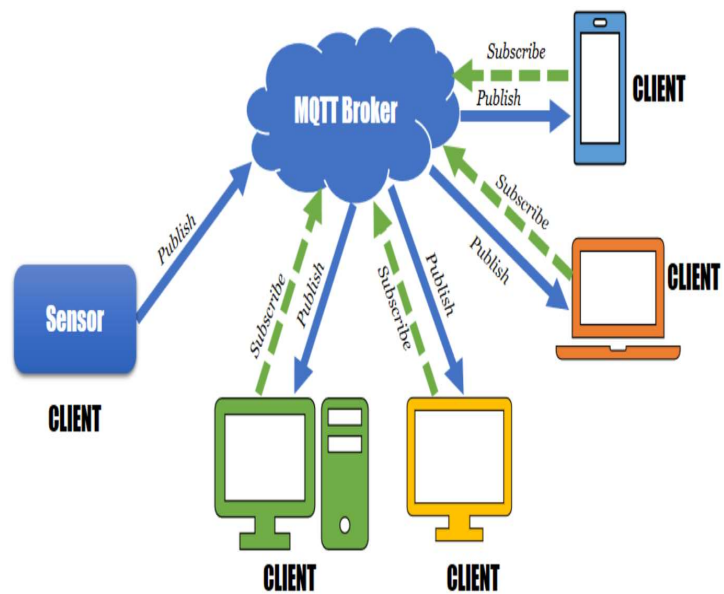
- Mosquitto
- Apache Kafka
- Logstash
- OpenSearch

- Installed the above services and configured to run it as a service
- Extracted the status and uptime using the subprocess module of python
- The extracted status is inserted into a redfish endpoint
- This process is done every 5 seconds to fetch us a time series data structure.

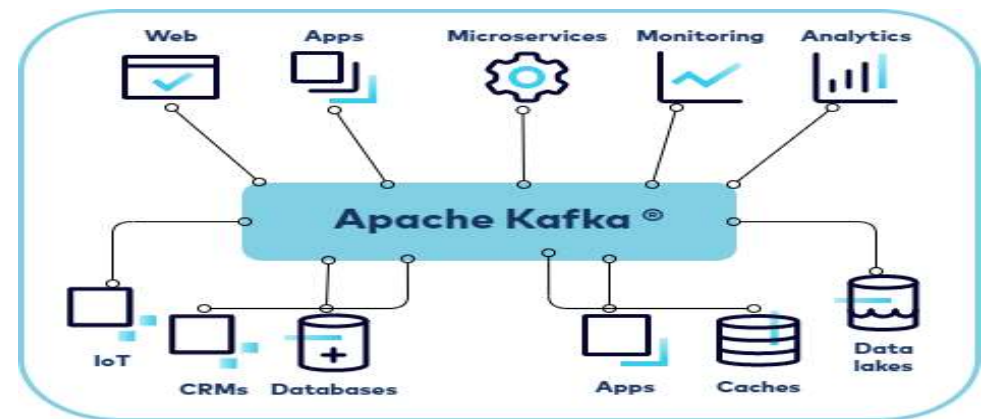


# ABOUT COMPONENTS

## Mosquitto



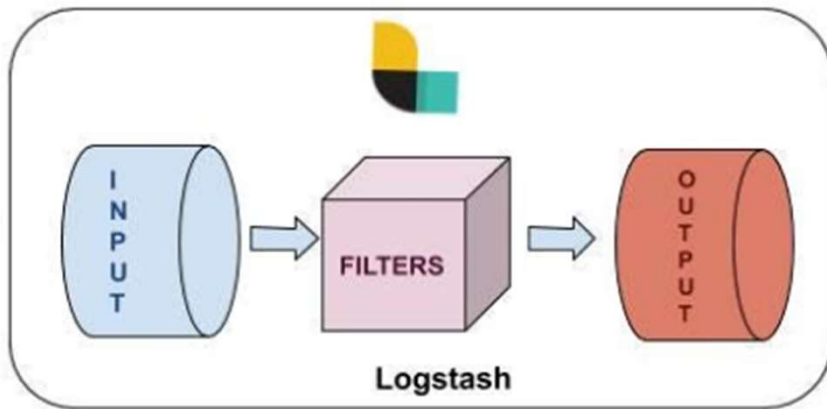
## Apache Kafka



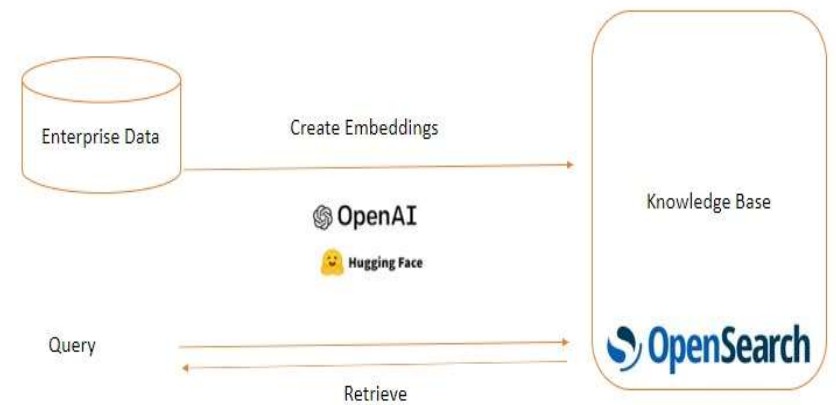


# DATA COLLECTION/ SCRAPPING

## Logstash



## OpenSearch



## OPENSEARCH DATA FLOW CHECK

```
[srivarshini@localhost ~]$ python status.py
Enter your username:admin
Enter your password:admin
OpenSearch service is: Active
[srivarshini@localhost ~]$ python createind.py
enter the index name: pdu
Index created successfully.
[srivarshini@localhost ~]$ python index.py
enter the index name: pdu
Document added successfully.
[srivarshini@localhost ~]$ python dataflow.py
enter index name: pdu
The index 'pdu' exists.
Indexing rate for 'pdu': 1 documents.
[srivarshini@localhost ~]$
```

OpenSearch service status has been extracted and an index named pdu has been created and data has been injected into it dataflow for that index has been verified

## DATA PERSISTENCE - API SERVER

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/api/graph/data`. The browser has several tabs open, including "Edit panel - New dashboa...", "Server Not Found", and several "Red Hat" related pages. The main content area shows a JSON response from the API, with tabs for "JSON", "Raw Data", and "Headers". The JSON data is displayed in a collapsible tree view, showing the following structure:

```
{
  "edges": [
    {
      "id": "1",
      "mainStat": "1",
      "source": "1",
      "target": "2"
    },
    {
      "id": "2",
      "mainStat": "1",
      "source": "2",
      "target": "3"
    }
  ],
  "nodes": [
    {
      "id": "1",
      "mainStat": "Active",
      "title": "Mosquitto",
      "arc_failed": 0.7,
      "arc_passed": 0.3,
      "detail__role": "extrct(IOT)"
    },
    {
      "id": "2",
      "mainStat": "Active",
      "title": "Kafka",
      "arc_failed": 0.5,
      "arc_passed": 0.5,
      "detail__role": "Stream"
    },
    {
      "id": "3",
      "mainStat": "Active",
      "title": "Prometheus",
      "arc_failed": 0.3,
      "arc_passed": 0.7,
      "detail__role": "Load"
    }
  ]
}
```

## GRAFANA - NODEGRAPH



- Grafana Node Graph is a plugin for Grafana that provides a specialized visualization panel for displaying interconnected nodes and edges.
- Grafana Node Graph API, on the other hand, is a separate component or functionality that allows interaction with the Grafana Node Graph plugin programmatically through an API interface.
- Node Graph API is an API interface that allows programmatic interaction with the node graph visualization.

## **DEMO AND NEXT STEPS**

---

### **Demo Objective**

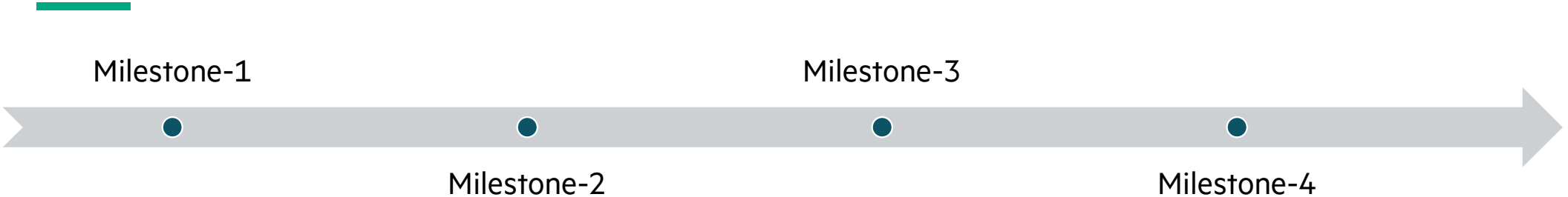
- 1. How status is collected presently**
- 2. Executing the code**
- 3. Presenting the API endpoint**
- 4. Grafana- Visualization of services**

### **Next Steps**

- 1. Checking the status of data flow between the components.**
- 2. Integrating more pipelines.**
- 3. Multiple pipelines visualization.**



# PROJECT PLAN , LEARNINGS , CHALLENGES



## Learnings

- 1. **Linux Operating Environment**
- 2. **Components – Kafka, Mosquitto, Logstash, Opensearch**
- 3. **Prometheus DB**
- 4. **Grafana - Dashboard**

## Challenges

- 1. **Prometheus**
- 2. **Nodegraph vs API plugin**
- 3. **API Endpoints – Multiple and Single**



## **MILESTONE 2 DEMO**

# AGENDA

---

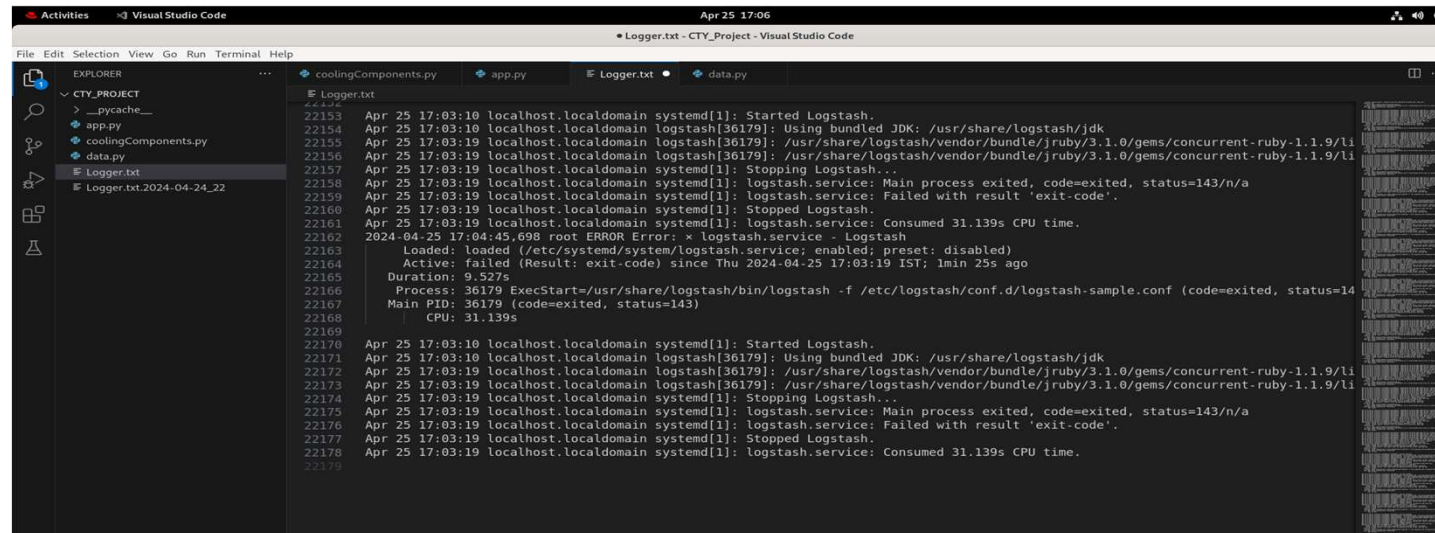
- Logger
- Deploying application as service
- Kafka dataflow
- Data injection to Mqtt
- Mqtt to Kafka
- Kafka to OpenSearch via Logstash
- Grafana dashboard





# LOGGER CLASS

- The logger class is built using the logging module of python
- All the configurations for the logging module have been done in the constructor
- The methods of the class can be used for various types of logging
- There are three different logs identified in the logger class. They are: warning, info, and error
- It is also configured with time rotation handler, so that it creates a backup and keeps rotating the log files so that it will not overload the file.



```
22153 Apr 25 17:03:10 localhost.localdomain systemd[1]: Started Logstash.
22154 Apr 25 17:03:10 localhost.localdomain logstash[36179]: Using bundled JDK: /usr/share/logstash/jdk
22155 Apr 25 17:03:19 localhost.localdomain logstash[36179]: /usr/share/logstash/vendor/bundle/jruby/3.1.0/gems/concurrent-ruby-1.1.9/li
22156 Apr 25 17:03:19 localhost.localdomain logstash[36179]: /usr/share/logstash/vendor/bundle/jruby/3.1.0/gems/concurrent-ruby-1.1.9/li
22157 Apr 25 17:03:19 localhost.localdomain systemd[1]: Stopping Logstash...
22158 Apr 25 17:03:19 localhost.localdomain systemd[1]: logstash.service: Main process exited, code=exited, status=143/n/a
22159 Apr 25 17:03:19 localhost.localdomain systemd[1]: logstash.service: Failed with result 'exit-code'.
22160 Apr 25 17:03:19 localhost.localdomain systemd[1]: Stopped Logstash.
22161 Apr 25 17:03:19 localhost.localdomain systemd[1]: logstash.service: Consumed 31.139s CPU time.
22162 2024-04-25 17:04:45.698 root ERROR Error: * logstash.service - Logstash
22163   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; preset: disabled)
22164   Active: failed (Result: exit-code) since Thu 2024-04-25 17:03:19 IST; 1min 25s ago
22165     Duration: 9.527s
22166   Process: 36179 ExecStart=/usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/logstash-sample.conf (code=exited, status=143)
22167   Main PID: 36179 (code=exited, status=143)
22168     CPU: 31.139s
22169
22170 Apr 25 17:03:10 localhost.localdomain systemd[1]: Started Logstash.
22171 Apr 25 17:03:10 localhost.localdomain logstash[36179]: Using bundled JDK: /usr/share/logstash/jdk
22172 Apr 25 17:03:19 localhost.localdomain logstash[36179]: /usr/share/logstash/vendor/bundle/jruby/3.1.0/gems/concurrent-ruby-1.1.9/li
22173 Apr 25 17:03:19 localhost.localdomain logstash[36179]: /usr/share/logstash/vendor/bundle/jruby/3.1.0/gems/concurrent-ruby-1.1.9/li
22174 Apr 25 17:03:19 localhost.localdomain systemd[1]: Stopping Logstash...
22175 Apr 25 17:03:19 localhost.localdomain systemd[1]: logstash.service: Main process exited, code=exited, status=143/n/a
22176 Apr 25 17:03:19 localhost.localdomain systemd[1]: logstash.service: Failed with result 'exit-code'.
22177 Apr 25 17:03:19 localhost.localdomain systemd[1]: Stopped Logstash.
22178 Apr 25 17:03:19 localhost.localdomain systemd[1]: logstash.service: Consumed 31.139s CPU time.
22179
```

## SERVICE

- We have deployed our application as a service
- The service is named as monitor.service
- This service executes the app.py file using the /usr/bin/python file
- The service will start the application when the command systemctl start monitor is run

```
[[Unit]]
Description=Monitoring Service
After=network.target

[Service]
User=chinmaihk
Group=wheel
WorkingDirectory=/home/chinmaihk/monitoring_system
Environment="PATH=/usr/local/bin"
ExecStart=/usr/bin/python /home/chinmaihk/monitoring_system/app.py
Restart=always
Type=notify

[Install]
WantedBy=multi-user.target
```

## DATA INJECTION TO MQTT:

### 1. Setup:

Configures MQTT broker details (broker\_address, port) and MQTT topic (topic).

### 2. Main Loop:

- Initializes an MQTT client and connects to the broker.
- Enters a loop to simulate sensor data publishing:
- Generates simulated sensor readings as a dictionary (message\_dict).
- Converts the dictionary to JSON format (message\_json).
- Publishes the JSON message to the MQTT topic.
- Prints the published message to the console.
- Sleeps for 5 seconds before repeating.

This script is mainly focusing on MQTT client setup and continuous publication of simulated sensor data to an MQTT topic. The main loop efficiently handles the message generation and publishing process while incorporating a brief delay between iterations



```
Activities Terminal Wed Apr 24 10:13 AM 76 %
chandana96k@localhost:~ — python mosquito_data_injection.py
chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x
[chandana96k@localhost ~]$ python mosquito_data_injection.py
Published: {"name": "x9000cdu", "timestamp": 1713967597945, "device_type": "CCDU", "Secondary_Pump_Differential_Pressure": 19, "VFD1_RunTime_Energy_Counter": 34397, "CDU_Voltage_Phase_2_3": 0, "Primary_Facility_Supply_Water_Pressure": 16, "VFD2_DC_Bus_Voltage": 480, "Secondary_Cabinet_Supply_Water_Temperature": 21, "Room_Dew_Point": 2, "Relative_Humidity": 30, "CDU_Voltage_Phase_3_1": 0, "Secondary_Cabinet_Supply_Water_Pressure_2": 32.2, "VFD1_Current": 6.4, "CDU_Current_Phase_1": 2.7, "Secondary_Cabinet_Return_Water_Temperature_2": 25.1, "Secondary_Cabinet_Return_Water_Temperature": 25.1, "PLC_to_VFD_Voltage": 8.5, "Room_Temperature": 22.4, "Primary_Facility_Return_Water_Temperature": 23, "CDU_Current_Phase_3": 0, "PLC_Temperature": 37.7, "Secondary_Cabinet_Flow": 16.9, "VFD1_DC_Bus_Voltage": 489, "Secondary_System_Differential_Pressure": 17.5, "Primary_Facility_Flow": 5.5, "CWV_Valve_Actuator_Voltage": 6.9, "Primary_Facility_Supply_Water_Temperature": 11.5, "CDU_Voltage_Phase_1_2": 210.5, "VFD1_Pump_Speed": 3572, "VFD2_Current": 5.5, "VFD2_Pump_Speed": 3575, "Secondary_Cabinet_Supply_Water_Pressure": 31.9, "Secondary_Pump_Suction_Pressure_2": 13.6, "CDU_Current_Phase_2": 0, "Secondary_Cabinet_Return_Water_Pressure_2": 31, "Primary_Facility_Return_Water_Pressure": 15.8, "CDU_Power": 537, "Secondary_Cabinet_Supply_Water_Temperature_2": 21.8, "Secondary_Cabinet_Return_Water_Pressure": 31.1, "Secondary_Pump_Suction_Pressure_1": 13.6, "Actuator_2_Feedback_Position": 60, "VFD2_RunTime_Energy_Counter": 34591}
-----
Published: {"name": "x9000cdu", "timestamp": 1713967602951, "device_type": "CCDU", "Secondary_Pump_Differential_Pressure": 19, "VFD1_RunTime_Energy_Counter": 34397, "CDU_Voltage_Phase_2_3": 0, "Primary_Facility_Supply_Water_Pressure": 17, "VFD2_DC_Bus_Voltage": 481, "Secondary_Cabinet_Supply_Water_Temperature": 19, "Room_Dew_Point": 3, "Relative_Humidity": 28, "CDU_Voltage_Phase_3_1": 0, "Secondary_Cabinet_Supply_Water_Pressure_2": 32.2, "VFD1_Current": 6.4, "CDU_Current_Phase_1": 2.7, "Secondary_Cabinet_Return_Water_Temperature_2": 25.1, "Secondary_Cabinet_Return_Water_Temperature": 25.1, "PLC_to_VFD_Voltage": 8.5, "Room_Temperature": 22.4, "Primary_Facility_Return_Water_Temperature": 23, "CDU_Current_Phase_3": 0, "PLC_Temperature": 37.7, "Secondary_Cabinet_Flow": 16.9, "VFD1_DC_Bus_Voltage": 489, "Secondary_System_Differential_Pressure": 17.5, "Primary_Facility_Flow": 5.5, "CWV_Valve_Actuator_Voltage": 6.9, "Primary_Facility_Supply_Water_Temperature": 11.5, "CDU_Voltage_Phase_1_2": 210.5, "VFD1_Pump_Speed": 3572, "VFD2_Current": 5.5, "VFD2_Pump_Speed": 3575, "Secondary_Cabinet_Supply_Water_Pressure": 31.9, "Secondary_Pump_Suction_Pressure_2": 13.6, "CDU_Current_Phase_2": 0, "Secondary_Cabinet_Return_Water_Pressure_2": 31, "Primary_Facility_Return_Water_Pressure": 15.8, "CDU_Power": 537, "Secondary_Cabinet_Supply_Water_Temperature_2": 21.8, "Secondary_Cabinet_Return_Water_Pressure": 31.1, "Secondary_Pump_Suction_Pressure_1": 13.6, "Actuator_2_Feedback_Position": 60, "VFD2_RunTime_Energy_Counter": 34591}
```



## MQTT TO KAFKA:

---

### 1. Setup:

Configures an MQTT client to subscribe to a specific topic (`mqtt_topic`).

Initializes a Kafka producer to publish messages to a Kafka topic (`kafka_topic_name`).

### 2. Message Handling:

Defines an `on_message` callback to process incoming MQTT messages.

Publishes MQTT messages to Kafka using the Kafka producer.

### 3. Kafka Consumer (`check_kafka_data`):

Sets up a Kafka consumer to continuously poll for messages from a specified Kafka topic (`kafka_topic_name`).

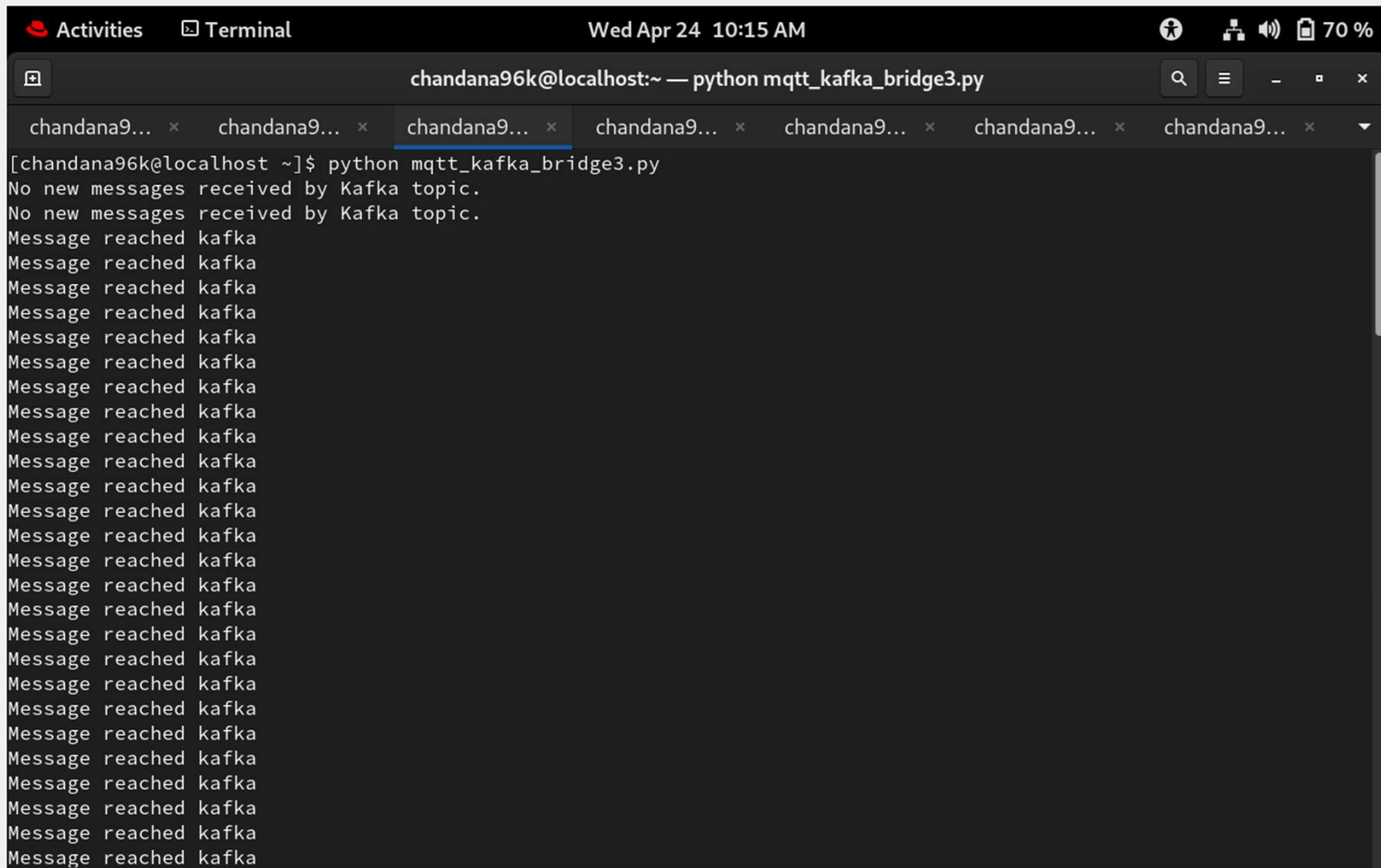
### 4. Main Loop:

Runs the Kafka consumer (`check_kafka_data`) continuously.

The script can be stopped by a `KeyboardInterrupt`, gracefully closing connections.

This script bridges MQTT messages to Kafka, maintaining continuous data flow between the two systems until terminated





The image shows a terminal window titled "Activities Terminal" with the date and time "Wed Apr 24 10:15 AM". The window contains a terminal session for the user "chandana96k@localhost" running the command "python mqtt\_kafka\_bridge3.py". The output of the script shows two initial messages: "No new messages received by Kafka topic." followed by a series of 20 "Message reached kafka" messages. The terminal window has a dark theme and a tab bar at the top with several tabs labeled "chandana9...".

```
chandana96k@localhost:~ — python mqtt_kafka_bridge3.py
[chandana96k@localhost ~]$ python mqtt_kafka_bridge3.py
No new messages received by Kafka topic.
No new messages received by Kafka topic.
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
Message reached kafka
```



## **KAFKA VIA LOGSTASH TO OPENSEARCH:**

### 1. OpenSearch Configuration:

Sets up parameters for connecting to OpenSearch (es\_host, es\_port, es\_user, es\_password, es\_index).

### 2. Function get\_document\_count():

Retrieves the document count of a specified OpenSearch index using an authenticated GET request.

Returns the document count.

### 3. Function monitor\_index\_growth(interval=9):

Continuously monitors the growth of an OpenSearch index by comparing document counts over time.

Prints the change in document count at regular intervals.

### 4. Execution:

Initiates the monitoring process with default settings (monitor\_index\_growth()).

The script efficiently tracks changes in an OpenSearch index's document count over time





```
Activities Terminal Wed Apr 24 10:15 AM
chandana96k@localhost:~ — sudo /opt/logstash-8.12.1/bin/logstash -f /etc/logstash/conf.d/dataflow3.conf
chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x
re\": 17.5, \"Primary_Facility_Flow\": 5.5, \"CWV_Valve_Actuator_Voltage\": 6.9, \"Primary_Facility_Supply_Water_T
emperature\": 11.5, \"CDU_Voltage_Phase_1_2\": 210.5, \"VFD1_Pump_Speed\": 3572, \"VFD2_Current\": 5.5, \"VFD2_Pum
p_Speed\": 3575, \"Secondary_Cabinet_Supply_Water_Pressure\": 31.9, \"Secondary_Pump_Suction_Pressure_2\": 13.6, \"
CDU_Current_Phase_2\": 0, \"Secondary_Cabinet_Return_Water_Pressure_2\": 31, \"Primary_Facility_Return_Water_Pres
sure\": 15.8, \"CDU_Power\": 537, \"Secondary_Cabinet_Supply_Water_Temperature_2\": 21.8, \"Secondary_Cabinet_Retu
rn_Water_Pressure\": 31.1, \"Secondary_Pump_Suction_Pressure_1\": 13.6, \"Actuator_2_Feedback_Position\": 60, \"VF
D2_RunTime_Energy_Counter\": 34591}
  },
    \"CDU_Current_Phase_3\" => 0,
    \"CWV_Valve_Actuator_Voltage\" => 6.9,
    \"VFD1_Pump_Speed\" => 3572,
    \"Secondary_Pump_Suction_Pressure_2\" => 13.6,
    \"Secondary_Pump_Differential_Pressure\" => 19,
    \"CDU_Current_Phase_1\" => 2.7,
    \"Room_Temperature\" => 22.4,
    \"VFD2_RunTime_Energy_Counter\" => 34591,
    \"VFD1_RunTime_Energy_Counter\" => 34397,
    \"Secondary_Cabinet_Supply_Water_Temperature\" => 23,
    \"VFD1_Current\" => 6.4,
    \"VFD1_DC_Bus_Voltage\" => 489,
    \"@timestamp\" => 2024-04-24T14:15:30.647375743Z,
    \"Relative_Humidity\" => 31,
    \"Room_Dew_Point\" => 4,
    \"CDU_Voltage_Phase_1_2\" => 210.5,
    \"timestamp\" => 1713968128567,
    \"Secondary_Cabinet_Return_Water_Pressure_2\" => 31
  }
}
```





```
Activities Terminal Wed Apr 24 10:15 AM
chandana96k@localhost:~ — python kafka_opensearch_dataflow3.py
chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x chandana9... x
[chandana96k@localhost ~]$ python kafka_opensearch_dataflow3.py
Document count: 5983, Growth: 1
Document count: 5985, Growth: 2
Document count: 5987, Growth: 2
Document count: 5989, Growth: 2
Document count: 5991, Growth: 2
Document count: 5992, Growth: 1
Document count: 5994, Growth: 2
Document count: 5996, Growth: 2
Document count: 5998, Growth: 2
Document count: 6000, Growth: 2
Document count: 6002, Growth: 2
Document count: 6003, Growth: 1
Document count: 6004, Growth: 1
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
No growth
```



## MONITORING KAFKA DATAFLOW

### 1. Kafka Consumer Setup:

- The code sets up a Kafka consumer that connects to a Kafka broker (localhost:9092) and subscribes to a specific topic (CoolDev).

### 2. Message Consumption:

- The consumer continuously polls the Kafka topic for new messages every 1 minute (timeout=60 seconds).

### 3. Message Handling:

- If a message is received, it decodes and prints the message content.
- Handles scenarios like no new messages or errors gracefully.

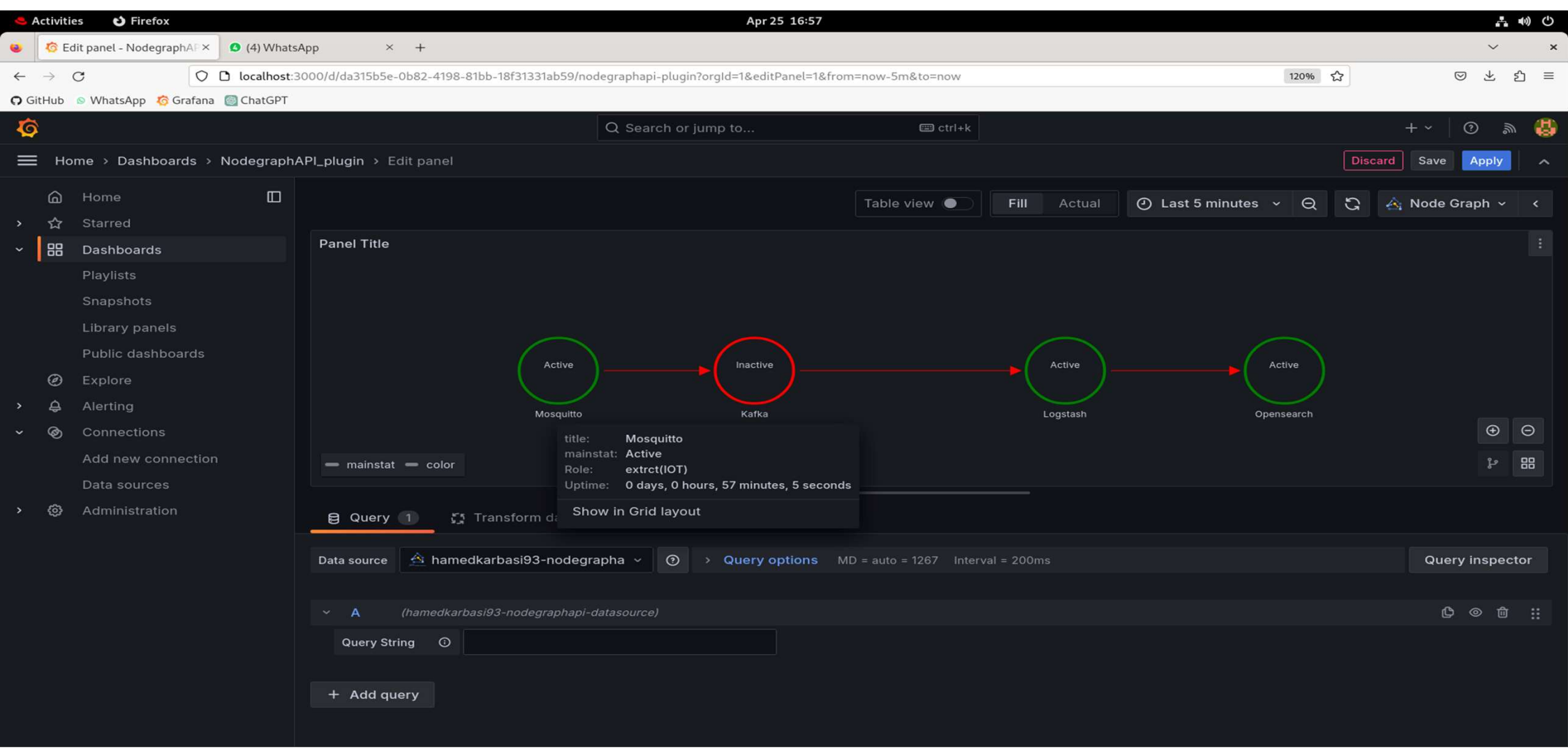
### 4. Graceful Termination:

- The consumer can be stopped by interrupting the script (e.g., pressing Ctrl+C), which closes the Kafka consumer gracefully.

In summary, this code creates a simple Kafka consumer that listens for messages from a specified topic and prints the message content when received, all while handling potential errors and interruptions gracefully

```
srivarshini@localhost:~$ python reachkaf.py
Message reached kafka: {"name": "x9000cdu", "timestamp": 1713976074558, "device_type": "CCDU", "Secondary_Pump_Differential_Pressure": 18, "VFD1_RunTime_Energy_Counter": 34397, "CDU_Voltage_Phase_2_3": 0, "Primary_Facility_Supply_Water_Pressure": 17, "VFD2_DC_Bus_Voltage": 480, "Secondary_Cabinet_Supply_Water_Temperature": 21, "Room_Dew_Point": 3, "Relative_Humidity": 30, "CDU_Voltage_Phase_1_1": 0, "Secondary_Cabinet_Supply_Water_Pressure_2": 32.2, "VFD1_Current": 6.4, "CDU_Current_Phase_1": 2.7, "Secondary_Cabinet_Return_Water_Temperature_2": 25.1, "Secondary_Cabinet_Return_Water_Temperature": 25.1, "PLC_to_VFD_Voltage": 8.5, "Room_Temperature": 22.4, "Primary_Facility_Return_Water_Temperature": 23, "CDU_Current_Phase_3": 0, "PLC_Temperature": 37.7, "Secondary_Cabinet_Flow": 16.9, "VFD1_DC_Bus_Voltage": 489, "Secondary_System_Differential_Pressure": 17.5, "Primary_Facility_Flow": 5.5, "CMV_Valve_Actuator_Voltage": 6.9, "Primary_Facility_Supply_Water_Temperature": 11.5, "CDU_Voltage_Phase_1_2": 210.5, "VFD1_Pump_Speed": 3572, "VFD2_Current": 5.5, "VFD2_Pump_Speed": 3575, "Secondary_Cabinet_Supply_Water_Pressure": 31.9, "Secondary_Pump_Suction_Pressure_2": 13.6, "CDU_Current_Phase_2": 0, "Secondary_Cabinet_Return_Water_Pressure_2": 31, "Primary_Facility_Return_Water_Pressure": 15.8, "CDU_Power": 537, "Secondary_Cabinet_Supply_Water_Temperature_2": 21.8, "Secondary_Cabinet_Return_Water_Pressure": 31.1, "Secondary_Pump_Suction_Pressure_1": 13.6, "Actuator_2_Feedback_Position": 60, "VFD2_RunTime_Energy_Counter": 34591}
Message reached kafka: {"name": "x9000cdu", "timestamp": 1713976124608, "device_type": "CCDU", "Secondary_Pump_Differential_Pressure": 19, "VFD1_RunTime_Energy_Counter": 34397, "CDU_Voltage_Phase_2_3": 0, "Primary_Facility_Supply_Water_Pressure": 17, "VFD2_DC_Bus_Voltage": 482, "Secondary_Cabinet_Supply_Water_Temperature": 19, "Room_Dew_Point": 2, "Relative_Humidity": 30, "CDU_Voltage_Phase_1_1": 0, "Secondary_Cabinet_Supply_Water_Pressure_2": 32.2, "VFD1_Current": 6.4, "CDU_Current_Phase_1": 2.7, "Secondary_Cabinet_Return_Water_Temperature_2": 25.1, "Secondary_Cabinet_Return_Water_Temperature": 25.1, "PLC_to_VFD_Voltage": 8.5, "Room_Temperature": 22.4, "Primary_Facility_Return_Water_Temperature": 23, "CDU_Current_Phase_3": 0, "PLC_Temperature": 37.7, "Secondary_Cabinet_Flow": 16.9, "VFD1_DC_Bus_Voltage": 489, "Secondary_System_Differential_Pressure": 17.5, "Primary_Facility_Flow": 5.5, "CMV_Valve_Actuator_Voltage": 6.9, "Primary_Facility_Supply_Water_Temperature": 11.5, "CDU_Voltage_Phase_1_2": 210.5, "VFD1_Pump_Speed": 3572, "VFD2_Current": 5.5, "VFD2_Pump_Speed": 3575, "Secondary_Cabinet_Supply_Water_Pressure": 31.9, "Secondary_Pump_Suction_Pressure_2": 13.6, "CDU_Current_Phase_2": 0, "Secondary_Cabinet_Return_Water_Pressure_2": 31, "Primary_Facility_Return_Water_Pressure": 15.8, "CDU_Power": 537, "Secondary_Cabinet_Supply_Water_Temperature_2": 21.8, "Secondary_Cabinet_Return_Water_Pressure": 31.1, "Secondary_Pump_Suction_Pressure_1": 13.6, "Actuator_2_Feedback_Position": 60, "VFD2_RunTime_Energy_Counter": 34591}
No new messages received by Kafka topic.
No new messages received by Kafka topic.
Message reached kafka: {"name": "x9000cdu", "timestamp": 1713976280046, "device_type": "CCDU", "Secondary_Pump_Differential_Pressure": 18, "VFD1_RunTime_Energy_Counter": 34397, "CDU_Voltage_Phase_2_3": 0, "Primary_Facility_Supply_Water_Pressure": 16, "VFD2_DC_Bus_Voltage": 482, "Secondary_Cabinet_Supply_Water_Temperature": 23, "Room_Dew_Point": 2, "Relative_Humidity": 29, "CDU_Voltage_Phase_1_1": 0, "Secondary_Cabinet_Supply_Water_Pressure_2": 32.2, "VFD1_Current": 6.4, "CDU_Current_Phase_1": 2.7, "Secondary_Cabinet_Return_Water_Temperature_2": 25.1, "Secondary_Cabinet_Return_Water_Temperature": 25.1, "PLC_to_VFD_Voltage": 8.5, "Room_Temperature": 22.4, "Primary_Facility_Return_Water_Temperature": 23, "CDU_Current_Phase_3": 0, "PLC_Temperature": 37.7, "Secondary_Cabinet_Flow": 16.9, "VFD1_DC_Bus_Voltage": 489, "Secondary_System_Differential_Pressure": 17.5, "Primary_Facility_Flow": 5.5, "CMV_Valve_Actuator_Voltage": 6.9, "Primary_Facility_Supply_Water_Temperature": 11.5, "CDU_Voltage_Phase_1_2": 210.5, "VFD1_Pump_Speed": 3572, "VFD2_Current": 5.5, "VFD2_Pump_Speed": 3575, "Secondary_Cabinet_Supply_Water_Pressure": 31.9, "Secondary_Pump_Suction_Pressure_2": 13.6, "CDU_Current_Phase_2": 0, "Secondary_Cabinet_Return_Water_Pressure_2": 31, "Primary_Facility_Return_Water_Pressure": 15.8, "CDU_Power": 537, "Secondary_Cabinet_Supply_Water_Temperature_2": 21.8, "Secondary_Cabinet_Return_Water_Pressure": 31.1, "Secondary_Pump_Suction_Pressure_1": 13.6, "Actuator_2_Feedback_Position": 60, "VFD2_RunTime_Energy_Counter": 34591}
^CStopping Kafka consumer.
[srivarshini@localhost ~]$
```

# GRAFANA DASHBOARD SNAPSHOT



# THANK YOU

