**Real Time Weather Data Analysis Documentation**

**Architecture:**



**Architecture Overview**

Architecture comprises the following components:

* **Data Producer (Flask Application)**: Fetches real-time weather data from an external API and publishes it to a Kafka topic.
* **Kafka**: Serves as the message broker, facilitating the decoupling of data producers and consumers.
* **Data Consumer**: Consumes weather data from Kafka and stores it in a PostgreSQL database.
* **PostgreSQL**: Stores the structured weather data for querying and analysis..
* **Grafana**: Visualizes data from PostgreSQL and Prometheus, providing insights into weather patterns and system performance.

**Workflow Overview**

* Highlight the workflow:  
  **Data Source OpenWeather → Producer(Python Script) → Kafka Topics → Consumer(Python Script) → PostgreSQL Database → Grafana for Visualization/Monitoring.**

**Configuration Details**

**Application Server (VM1)**

* Hosts data producer and consumer scripts (Python).
* Lightweight web framework (Flask/Django) for live production simulation.
* Recommended OS: Ubuntu Server 22.04.

**Kafka Server (VM2)**

* Hosts Kafka and Zookeeper.
* Separate disks for Kafka logs to avoid disk I/O bottlenecks.
* Recommended OS: Ubuntu Server 22.04.
* Run Kafka and Zookeeper in Docker containers using Docker Compose for better isolation.

**PostgreSQL Server (VM3)**

* Dedicated VM for PostgreSQL to ensure database performance.
* Enable backups and point-in-time recovery.
* Recommended OS: Ubuntu Server 22.04.
* Optionally, run PostgreSQL in a Docker container for easy management.

**Monitoring Server (VM4)**

* Hosts Prometheus for metrics collection and Grafana for visualization.
* Lightweight VM to handle monitoring traffic.
* Install Node Exporter for system metrics.
* Recommended OS: Ubuntu Server 22.04.

**Workflow**

1. **User Access:**

* Use tools like Kafka UI to monitor Kafka topics and processes.

2. **Data Flow:**

* Producers on the Application Server (VM1) send data to Kafka.
* Kafka brokers (VM2) store and distribute messages.
* Consumers on VM1 process data and store it in PostgreSQL (VM3).

3.**Monitoring:**

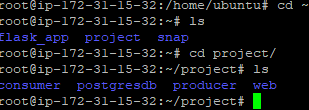
* Prometheus (VM4) scrapes metrics from Kafka, PostgreSQL, and the application.
* Grafana visualizes metrics for performance monitoring.

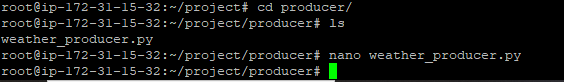
**Project Structured Steps:**

**1. Fetch Real-Time Weather Data**

* **Obtain API Access**: Register with a weather data provider like OpenWeatherMap to acquire an API key.
* **Develop the Producer Script**: Create a Python script (weather\_producer.py) that fetches weather data for selected cities and sends it to your Kafka topic.

Snapshots





import json

import time

import requests

from kafka import KafkaProducer

# Kafka Configuration

KAFKA\_BROKER = '43.204.166.47:9092' # Replace with your Kafka VM's IP address

KAFKA\_TOPIC = 'weather-data'

# OpenWeatherMap Configuration

API\_KEY = 'efc3ed98c7a21d70415d0a0cbe9970aa' # Replace with your OpenWeatherMap API key

CITIES = ["Mumbai", "Dubai", "Abu Dhabi", "Delhi", "Dhaka", "Rome", "Paris", "Sharjah", "New York", "Miami", "Chicago", "Tokyo", "Las Vegas", "Seattle", "Hong Kong", ">

API\_URL = "http://api.openweathermap.org/data/2.5/weather"

# Kafka Producer Initialization

producer = KafkaProducer(

bootstrap\_servers=KAFKA\_BROKER,

value\_serializer=lambda v: json.dumps(v).encode('utf-8') # Serialize data to JSON

)

def fetch\_weather(city):

"""Fetch weather data for a specific city."""

params = {

'q': city,

'appid': API\_KEY,

'units': 'metric' # Fetch data in Celsius

}

try:

response = requests.get(API\_URL, params=params)

response.raise\_for\_status()

data = response.json()

return {

'city': city,

'temperature': data['main']['temp'],

'humidity': data['main']['humidity'],

'weather': data['weather'][0]['description'],

'timestamp': time.strftime('%Y-%m-%d %H:%M:%S')

}

except Exception as e:

print(f"Error fetching data for {city}: {e}")

return None

def send\_to\_kafka():

"""Fetch weather data for each city and send it to Kafka."""

print("Starting Weather Data Producer...")

while True: # Infinite loop to fetch data at intervals

for city in CITIES:

weather\_data = fetch\_weather(city)

if weather\_data:

print(f"Sending data to Kafka: {weather\_data}")

producer.send(KAFKA\_TOPIC, value=weather\_data)

print("Batch of data sent. Waiting for 10 seconds...\n")

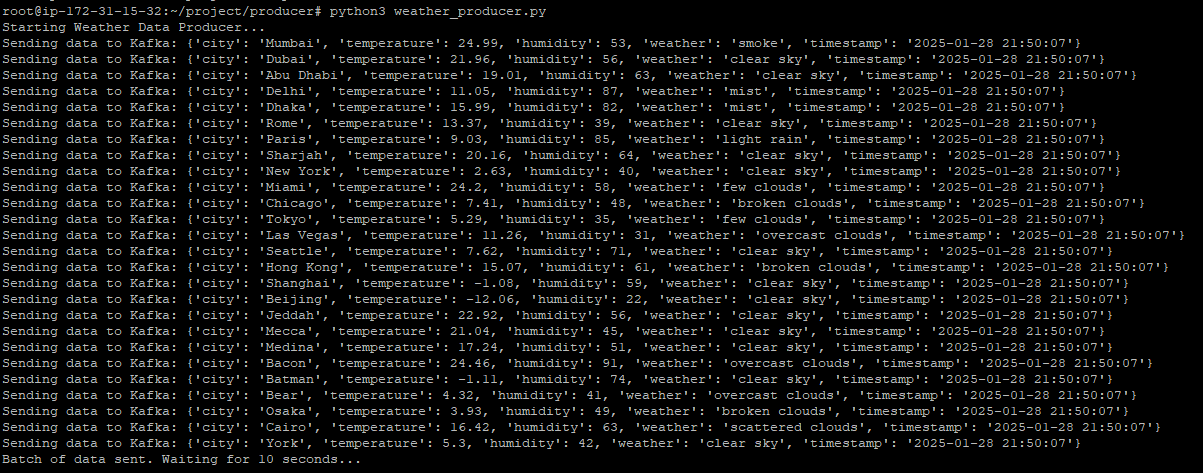
time.sleep(10) # Fetch data every 10 seconds

if \_\_name\_\_ == "\_\_main\_\_":

send\_to\_kafka()



sends Data it to your Kafka topic



**2. Consume Data and Store in PostgreSQL**

* **Set Up PostgreSQL**: Ensure your PostgreSQL database is configured to accept connections.
* **Develop the Consumer Script**: Create a Python script (weather\_consumer.py) that reads data from the Kafka topic and inserts it into PostgreSQL.



from kafka import KafkaConsumer

import psycopg2

import json

def insert\_weather\_data(connection, data):

try:

with connection.cursor() as cursor:

insert\_query = """

INSERT INTO weather\_data (city, temperature, humidity, weather, timestamp)

VALUES (%s, %s, %s, %s, %s)

"""

cursor.execute(insert\_query, (

data['city'],

data['temperature'],

data['humidity'],

data['weather'],

data['timestamp']

))

connection.commit()

print(f"Data inserted into PostgreSQL: {data}")

except Exception as e:

print(f"Error inserting data into PostgreSQL: {e}")

def consume\_weather\_data():

try:

# Connect to PostgreSQL

connection = psycopg2.connect(

dbname="postgres",

user="postgres",

password="Shama@2406",

host="43.205.39.152",

port="5432"

)

print("Connected to PostgreSQL database")

# Kafka Consumer

consumer = KafkaConsumer(

'weather-data',

bootstrap\_servers=['43.204.166.47:9092'],

auto\_offset\_reset='earliest',

enable\_auto\_commit=True,

value\_deserializer=lambda x: json.loads(x.decode('utf-8'))

)

print("Starting Weather Data Consumer...")

for message in consumer:

print(f"Received message: {message.value}")

weather\_data = message.value

insert\_weather\_data(connection, weather\_data)

except Exception as e:

print(f"Error: {e}")

finally:

if 'connection' in locals() and connection:

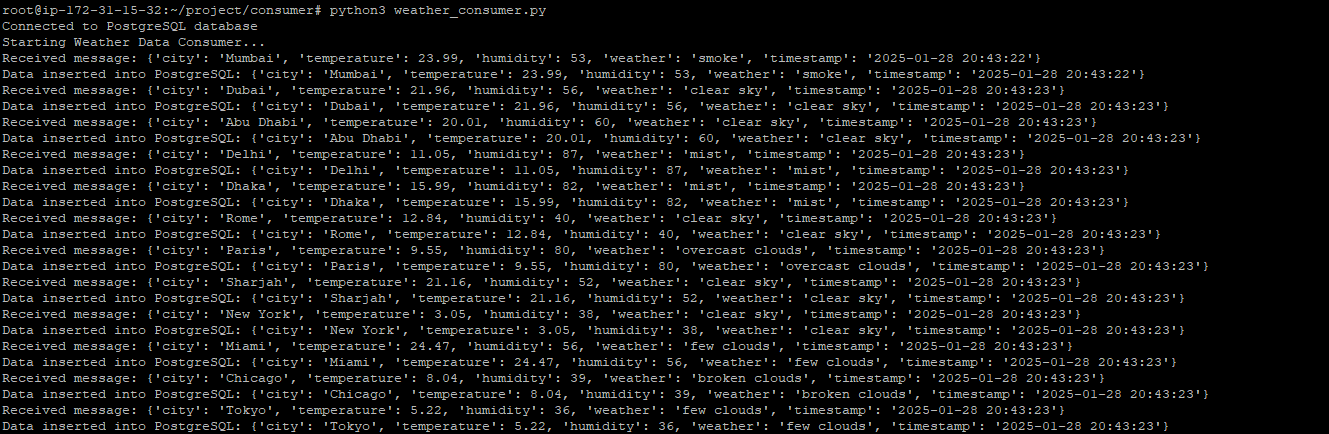
connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

consume\_weather\_data()



reads data from the Kafka topic and inserts it into PostgreSQL



**3. Visualize Data with Grafana**

* **Configure Grafana**: Set up Grafana to connect to your PostgreSQL database as a data source.
* **Create Dashboards**: Design dashboards to visualize weather metrics such as temperature and humidity over time.

