M7T4 - Monitoring an ingestion service and anomaly detection techniques

Lab Activity 1: Setting Up Prometheus and Grafana for Kafka Monitoring (1 hour)

Objective: Automate monitoring processes for data ingestion services using industry-standard tools.

Problem: Imagine your company, a data analytics firm, needs to monitor Kafka clusters to ensure efficient data ingestion and processing. Your task is to set up Prometheus and Grafana to monitor Kafka metrics.

Step-by-step instructions

Setup:

Install Prometheus:

- **Step 1**: Download the latest Prometheus release from the official website.
- **Step 2**: Extract the downloaded file and navigate to the Prometheus directory.
- **Step 3**: Configure the prometheus.yml file to scrape metrics from Kafka:global:

  ```
    scrape_interval: 15s

  scrape_configs:
    - job_name: 'kafka_brokers'
      static_configs:
        - targets: ['localhost:9090']

    - job_name: 'kafka_exporter'
      static_configs:
        - targets: ['localhost:9308']
  ```

- **Step 4**: Start Prometheus by running the following command:./prometheus --config.file=prometheus.yml

Install Grafana:

- **Step 1**: Download Grafana from the official website.
- **Step 2**: Install Grafana by following the installation instructions for your operating system.

- **Step 3**: Start Grafana by running the following command:sudo service grafana-server start

Configure Grafana:

- **Step 1**: Open your web browser and navigate to http://localhost:3000.
- **Step 2**: Log in with the default credentials (username: admin, password: admin).
- **Step 3**: Add Prometheus as a data source:
  - Go to **Configuration > Data Sources > Add data source**.
  - Select **Prometheus** and enter the URL http://localhost:9090.
  - Click **Save & Test**.

Create Dashboards:

- **Step 1**: Go to **Create > Dashboard**.
- **Step 2**: Add a new panel and select the metrics you want to visualise (e.g., kafka_brokers, kafka_exporter).
- **Step 3**: Customise the dashboard layout and save it.

Set Up Kafka Exporter:

- **Step 1**: Download the Kafka Exporter binary from the official repository.
- **Step 2**: Run the Kafka Exporter with the following command:./kafka_exporter --kafka.server=localhost:9092

Monitor Key Metrics:

- **Step 1**: In Grafana, create panels to monitor key Kafka metrics such as under-replicated partitions, offline partitions, active controller count, request latency, error rates, consumer lag, and consumption rate.
- **Step 2**: Set up alerts for critical metrics using Prometheus Alertmanager.

Expected Outcomes:

- **Prometheus Setup**: Prometheus is installed and configured to scrape Kafka metrics.
- **Grafana Setup**: Grafana is installed and configured to visualise Kafka metrics.
- **Kafka Exporter Setup**: Kafka Exporter is running and providing metrics to Prometheus.
- **Dashboard Creation**: Grafana dashboards are created to monitor key Kafka metrics.
- **Alert Configuration**: Alerts are set up for critical Kafka metrics.

Sample Data:

- **Before Monitoring**:global:

```
  scrape_interval: 15s

scrape_configs:
  - job_name: 'kafka_brokers'
```

```
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'kafka_exporter'
    static_configs:
      - targets: ['localhost:9308']
```

- **After Monitoring**: !Grafana Dashboard
  Lab Activity 2: Implementing Anomaly Detection with Isolation Forest (1 hour)

Objective: Implement forecasting and anomaly detection techniques, including ARIMA, SARIMAX, and other methods.

Problem: Imagine your company, a cybersecurity firm, needs to detect anomalies in network traffic data to identify potential security threats. Your task is to implement anomaly detection using the Isolation Forest algorithm.

Step-by-step instructions

Setup:

Set Up the Environment:

- **Step 1**: Create a new Python virtual environment:python3 -m venv anomaly_detection_env
  source anomaly_detection_env/bin/activate

- **Step 2**: Install the required libraries:pip install pandas scikit-learn matplotlib

Load and Prepare Data:

- **Step 1**: Create a sample dataset with time series data:import pandas as pd
  ```
  import numpy as np

  # Generate sample data
  np.random.seed(42)
  dates = pd.date_range('20230101', periods=100)
  data = pd.DataFrame(np.random.randn(100, 1), index=dates,
  columns=['value'])
  data['value'] += np.linspace(0, 10, 100)
  data['anomaly'] = 0
  data.loc[50, 'value'] = 20  # Inject an anomaly
  data.loc[50, 'anomaly'] = 1
  ```

Train Isolation Forest Model:

- **Step 1**: Import the Isolation Forest model and train it on the dataset:from sklearn.ensemble import IsolationForest

  ```
  # Train Isolation Forest model
  model = IsolationForest(contamination=0.01, random_state=42)
  model.fit(data[['value']])
  data['anomaly_score'] = model.decision_function(data[['value']])
  data['anomaly'] = model.predict(data[['value']])
  data['anomaly'] = data['anomaly'].apply(lambda x: 1 if x == -1 else 0)
  ```

Visualise Anomalies:

- **Step 1**: Plot the data and highlight the detected anomalies:import matplotlib.pyplot as plt

  ```
  plt.figure(figsize=(10, 6))
  plt.plot(data.index, data['value'], label='Value')
  plt.scatter(data.index, data['value'], c=data['anomaly'], cmap='coolwarm',
  label='Anomaly')
  plt.title('Anomaly Detection with Isolation Forest')
  plt.xlabel('Date')
  plt.ylabel('Value')
  plt.legend()
  plt.show()
  ```

Interpret Results:

- **Step 1**: Discuss the results and the effectiveness of the Isolation Forest model in detecting anomalies.
- **Step 2**: Explore how different contamination levels and parameters affect the model's performance.

Expected Outcomes:

- **Environment Setup**: Python virtual environment is created and required libraries are installed.
- **Data Preparation**: Sample dataset with time series data is created and anomalies are injected.
- **Model Training**: Isolation Forest model is trained on the dataset.
- **Anomaly Detection**: Anomalies are detected and highlighted in the dataset.
- **Result Interpretation**: Results are discussed, and the model's performance is evaluated.

Sample Data:

- **Before Anomaly Detection**:dates = pd.date_range('20230101', periods=100)

  ```
  data = pd.DataFrame(np.random.randn(100, 1), index=dates,
  columns=['value'])
  data['value'] += np.linspace(0, 10, 100)
  ```

```
data['anomaly'] = 0
data.loc[50, 'value'] = 20  # Inject an anomaly
data.loc[50, 'anomaly'] = 1
```

- **After Anomaly Detection**:plt.figure(figsize=(10, 6))

```
plt.plot(data.index, data['value'], label='Value')
plt.scatter(data.index, data['value'], c=data['anomaly'], cmap='coolwarm', label='Anomaly')
plt.title('Anomaly Detection with Isolation Forest')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```