Kafka monitoring using Prometheus and Grafana

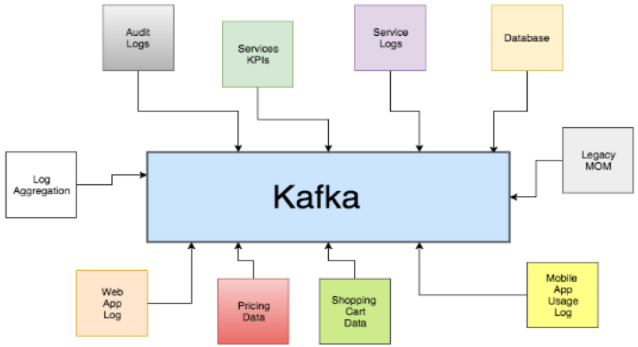
Dr. Soheila Dehghanzadeh
Denso Automotive Deutschland

Agenda

- Introduction to Kafka, Prometheus and Grafana
- Install Kafka
- Install Prometheus
- Monitor Kafka from Prometheus
- Install Grafana
- Monitor Kafka from Grafana
- Prometheus Rules

Introduction to Kafka

- Distributed streaming platform
  - Publish and subscribe to streams of records
  - Fault tolerant storage
    - Replicate topic log partitions to multiple servers
  - Application can process records as they appear in kafka
  - Fast and efficient IO by batching and compressing records
- Decouples data streams (consumers do not need to know about producers)
- Ingest data into data lakes, applications and real-time analytics systems
- Usecases
  - IoT applications
  - Microservices
  - Distributed applications

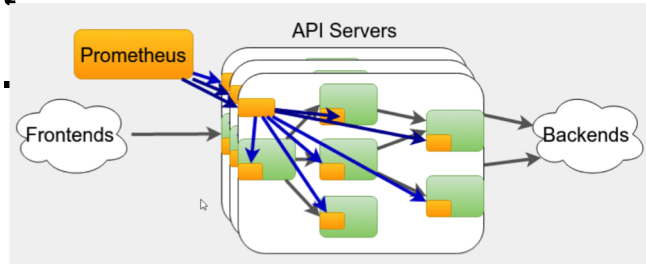


Introduction to Zookeeper

- A distributed, open-source coordination system for distributed applications
  - **Configuration** : sharing config info across all nodes
  - **Synchronization: locks, barriers and queus**
  - **Naming registry** find a machine in a cluster of 1000 of services
  - **Group services**: leader election

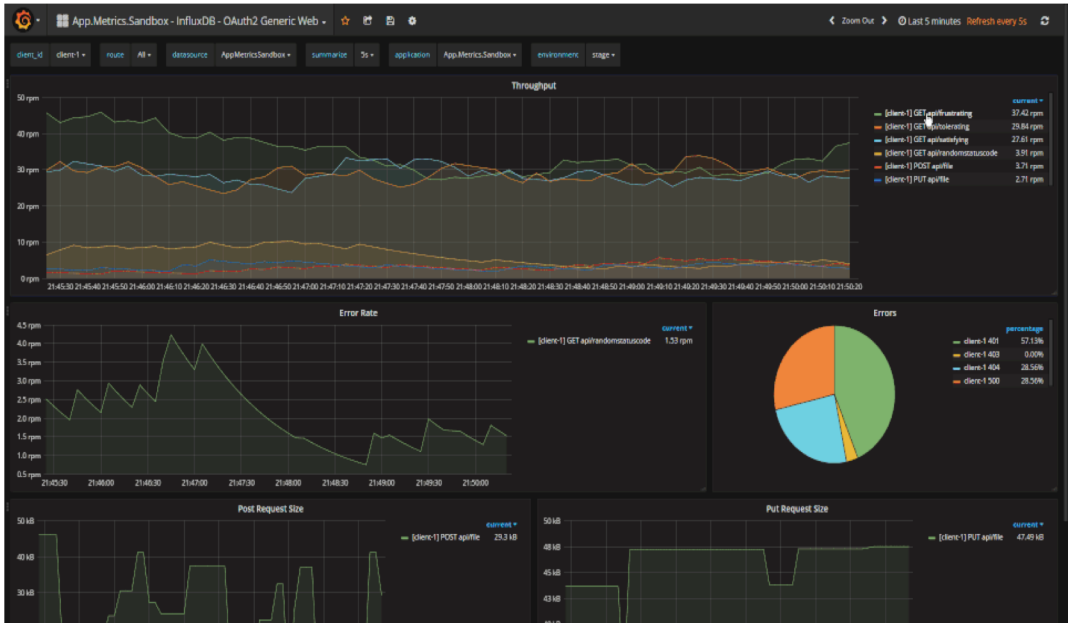
Introduction to Prometheus

- Prometheus is a metrics-based time series database
- Metric types
  - Counter: an increasing value
  - Gauge: an arbitrary value that can go up or down
  - Histogram: sampled observations counted in buckets
- Prometheus consists of two parts:
  - An exporter exposing metrics at a port
    - Node exporter, MySQL exporter,...
  - A Prometheus server



Introduction to Grafana

- Receive time-series data and visualize it with customized dashboards
  - Imports from
    - Graphite
    - Prometheus
    - InfluxDB
    - Elasticsearch
    - Google Stackdriver
    - AWS CloudWatch
    - Azure Monitor
    - Loki
    - MySQL
    - PostgreSQL
    - Microsoft SQL Server (MSSQL)
    - OpenTSDB
    - Testdata
    - Mixed



Scenario

1. In this scenario, we aim to run a kafka cluster and monitor it via
    a. Prometheus
    b. Grafana
2. To continue with the handson create a folder to keep the tutorial materials
    a. mkdir summerSchool
3. Prerequisites
    a. Linux (Debian/Ubuntu/Mint)
    b. Java

```
sudo apt update
sudo apt install default-jdk
```

Install Kafka

1. Install kafka from https://tecadmin.net/install-apache-kafka-ubuntu/
    a. wget http://www-us.apache.org/dist/kafka/2.2.1/kafka_2.12-2.2.1.tgz
    b. tar xzf kafka_2.12-2.2.1.tgz
    c. mv kafka_2.12-2.2.1 kafka
2. Create a directory for jmx exporter inside kafka folder
    a. mkdir prometheus
    b. cd prometheus
    c. wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.3.1/jmx_prometheus_javaagent-0.3.1.jar
    d. wget https://raw.githubusercontent.com/prometheus/jmx_exporter/master/example_configs/kafka-2_0_0.yml
    e. Edit the config file of step d by adding the following text at the end (be careful with indentation)

```
- pattern: kafka.producer<type=producer-metrics, client-id=(.+)><>(.+):\w*
name: kafka_producer_$2
- pattern: kafka.consumer<type=consumer-metrics, client-id=(.+)><>(.+):\w*
name: kafka_consumer_$2
- pattern: kafka.consumer<type=consumer-fetch-manager-metrics, client-id=(.+)><>(.+):\w*
 name: kafka_consumer_$2
```

    f. The config and the exporter java agent will be later on passed to kafka server, kafka producer and kafka consumer to export their metrics (specified above) to prometheus

Jmx Exporter Configuration Model

- The config file consists of a set of rules to collect attributes of java beans
- Each rule has a pattern to match against bean attribute
  - domain<beanpropertyName1=beanPropertyValue1, beanpropertyName2=beanPropertyValue2, ...><key1, key2, ...>attrName: value
- As soon as an attribute matched against a rule, rule processing will stop and attribute is exported to prometheus.

Start Kafka (1/2)

1. Start zookeeper by running this command from the kafka folder
    a. bin/zookeeper-server-start.sh config/zookeeper.properties
2. Start the Kafka server (replace USER_NAME with your username)
    a. Open another terminal and run this command from the kafka folder

KAFKA_HEAP_OPTS="-Xmx1000M -Xms1000M" KAFKA_OPTS="-javaagent:/home/USER_NAME/summerSchool/kafka/prometheus/jmx_prometheus_javaagent-0.3.1.jar=7071:/home/USER_NAME/summerSchool/kafka/prometheus/kafka-2_0_0.yml" bin/kafka-server-start.sh config/server.properties

Environment variable for memory requirements of the command
Environment variable for jmx exporter java agent and its config file
Command to start the kafka server
Configuration of the kafka server

Start Kafka (2/2)

3. Start Kafka consumer
    a. Open another terminal and run this command from kafka folder

KAFKA_OPTS="-javaagent:/home/USER_NAME/summerSchool/kafka/prometheus/jmx_prometheus_javaagent-0.3.1.jar=7072:/home/USER_NAME/summerSchool/kafka/prometheus/kafka-2_0_0.yml" /home/USER_NAME/summerSchool/kafka/bin/kafka-console-consumer.sh --bootstrap-server 0.0.0.0:9092 --topic test --from-beginning

3. Start Kafka producer
    a. Open another terminal and run this command from kafka folder

KAFKA_OPTS="-javaagent:/home/USER_NAME/summerSchool/kafka/prometheus/jmx_prometheus_javaagent-0.3.1.jar=7073:/home/USER_NAME/summerSchool/kafka/prometheus/kafka-2_0_0.yml" /home/USER_NAME/summerSchool/kafka/bin/kafka-console-producer.sh --broker-list 0.0.0.0:9092 --topic test

Install Prometheus

1. From the summerSchool folder run
   a. wget https://github.com/prometheus/prometheus/releases/download/v2.12.0-rc.0/prometheus-2.12.0-rc.0.linux-amd64.tar.gz
   b. tar xf prometheus-2.12.0-rc.0.linux-amd64.tar.gz
   c. mv prometheus-2.12.0-rc.0.linux-amd64 prometheus
   d. cd prometheus/
   e. Edit prometheus.yml by adding this text to the end (specifying the targets that prometheus will get the metrics from)
      (be careful with indentation)

```
- job_name: 'kafka-server'
                static_configs:
                - targets: ['127.0.0.1:7071']
- job_name: 'kafka-consumer'
                static_configs:
                - targets: ['127.0.0.1:7072']
- job_name: 'kafka-producer'
                static_configs:
                - targets: ['127.0.0.1:7073']
```

Run the Prometheus server

1. Now run prometheus server with
    a. ./prometheus --config.file=prometheus.yml
2. At this point communication of kafka producer and consumer can be tracked from http://localhost:9090/graph by choosing metrics with the kafka prefix, for example: kafka_consumer_incoming_byte_rate
3. However, a better way is to use grafana for monitoring multiple metrics in a real-time dashboard

Prometheus Alerts Graph Status Help

Enable query history

kafka_consumer_bytes_consumed_total

Load time: 182ms
Resolution: 14s
Total time series: 1

Execute - insert metric at cursor -

Graph Console

1h Until Res. (s) stacked

Remove Graph

Add Graph

Install & run Grafana

In the previous steps we were able to monitor *individual* metrics of kafka from Prometheus manually. However, Grafana can perform same operation by reading from a config file.

1. Create a folder in the summerSchool a folder named Grafana
    a. mkdir grafana
    b. wget https://dl.grafana.com/oss/release/grafana-6.3.3.linux-amd64.tar.gz
    c. tar xf grafana-6.3.3.linux-amd64.tar.gz
    d. mv grafana-6.3.3 grafana
    e. cd grafana/
2. Run Grafana server
    a. bin/grafana-server start
    b. Go to http://localhost:3000 and see the grafana dashboard (user=admin , password=admin)
    c. Add prometheus as a datastore for grafana ("datasource" -> prometheus)
    d. Download the predesigned garfana dashboard for kafka from here and unzip to get the json file https://activewizards.com/content/blog/Kafka_Monitoring_with_Prometheus,_Telegraf,_and_Grafana/grafana-system-and-kafka.zip
    e. Import dashboard json in grafana dashboard by going to Dashboard > home > + > import json

Monitor Kafka message exchange metrics from Grafana

By exchanging messages on kafka between producer and consumer, kafka metrics will change and can be observed from the Grafana dashboard in real-time.

Prometheus Rules

- You can extend Prometheus with some rules for monitoring and alerting
- This extension is applied to prometheus server after adding following text to the prometheus config file (prometheus/prometheus.yml)

```
rule_files:
  - 'prometheus.rules.yml'
```

- Create a new file in this directory named Prometheus.rules.yml
- Add the following alert definition to Prometheus.rules.yml file

```
groups:
- name: example
  rules:

  # Alert when kafka consumer consumes more than 100 bytes
  - alert: ConsumerOverload
    expr: kafka_consumer_bytes_consumed_total > 100
    for: 2s
    labels:
      severity: page
    annotations:
      summary: "consumer instance {{ $labels.instance }} overloaded"
```

Homework

- Try to train machine learning models for various metrics
- Currently Prometheus expr only accepts mathematical expression
- Extension to prometheus for accepting ML models in the expr field

```
groups:
- name: example
  rules:

  # Alert when kafka consumer consumes more than 100 bytes
  - alert: ConsumerOverload
    expr: kafka_consumer_bytes_consumed_total > 100
    for: 2s
    labels:
      severity: page
    annotations:
      summary: "consumer instance {{ $labels.instance }} overloaded"
```