https://www.michael-noll.com/blog/2013/03/13/running-a-multi-broker-apache-kafka-cluster-on-a-single-node/

서버별 jmx 포트 설정

https://www.confluent.io/blog/monitor-kafka-clusters-with-prometheus-grafana-and-confluent/

Prometheus와 Grafana를 사용한 Kafka 플랫폼 관리 및 모니터링

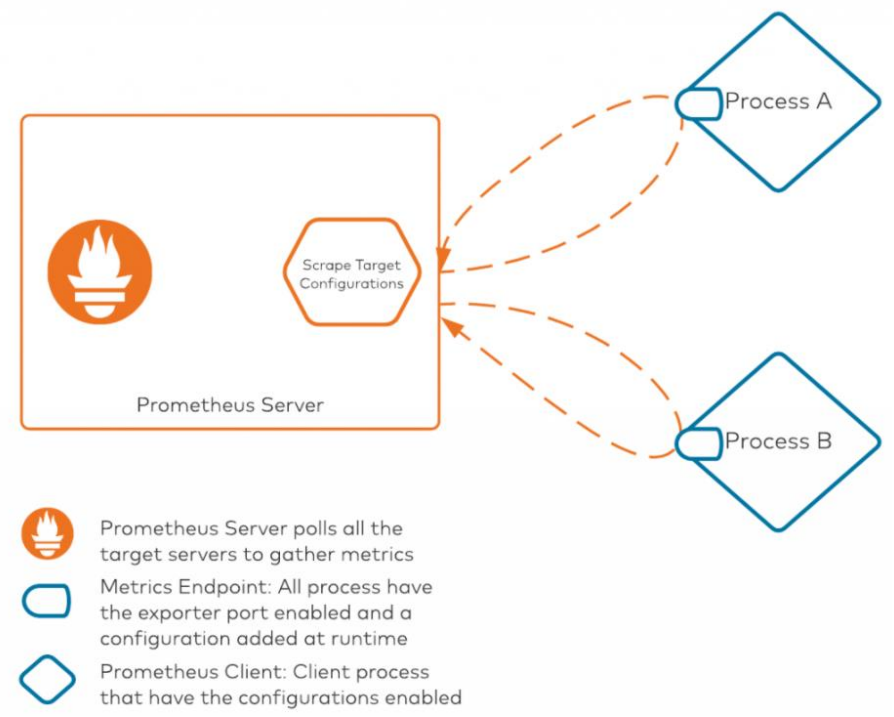
서비스 상관관계 분석, 통합 대시보드, 근본 원인 분석 또는 보다 세분화된 경보를 위해 모니터링 데이터를 메트릭 (Metrics) 집계 플랫폼으로 가져올 필요가 있음

- Kafka 클러스터의 JMX 데이터를 모니터링 시스템으로 전달
- 단일 UI에서 Kafka 서비스 데이터 스파이크를 메트릭과 상호연관
- Grafana 대시보드 설정

Prometheus는 수백 개 엔드 포인트를 스크래핑하면서 여러 플랫폼 메트릭을 집계하는데 사용되는 툴이다. Prometheus는 스크래핑과 집계 목적으로 특별히 개발된 시스템으로, 내부적으로 최적화된 방식으로 시간 분할된 (time-sliced) 데이터를 저장 및 검색할 수 있도록 시계열 데이터 저장소를 갖고 있으며, OpenMetrics 포맷을 사용한다. Grafana는 Prometheus와 통신하는 오픈 소스 차팅 및 대시보딩 툴이다. 코드 예는 Github의 jmx-monitoring-stacks 레포지터리 참조.

Prometheus 작동 원리

Prometheus는 서버측 컴포넌트와 클라이언트측 구성을 갖는 시스템이다. 서버측 컴포넌트는 모든 메트릭을 저장하고 모든 클라이언트를 스크래핑하는 역할을 한다. Prometheus는 일반적으로 클라이언트에서 데이터를 스크래핑하여 이를 서버에 전달하는 중간 컴포넌트를 사용하는 Elasticsearch와 Splunk와 같은 서비스와는 다르다. Prometheus 메트릭을 스크래핑하는 중간 컴포넌트가 없기 때문에 모든 폴 관련 구성은 서버 자체에 존재한다.



<Prometheus 프로세스>

Prometheus 서버 – Prometheus 서버는 자신들의 메트릭을 특정 포트에 노출한 모든 프로세스/클라이언트를 폴링하고 있으며, Prometheus 메트릭이 노출되는 모든 서버 IP 주소/호스트명과 포트를 열거하는 구성 파일을 내부적으로 갖고 있다. Scrape 타겟 구성은 Prometheus 내에서 모든 타겟을 계속 매핑하는 파일로 자동화없이 모든 것을 수작업으로 배치할 때 Scrape 타겟이 필요하다. Prometheus는 또한 서비스 발견 모듈도 지원하는데 메트릭을 노출하고 있는 사용가능 서비스를 발견하는데 활용할 수 있다. 이러한 자동 발견은 포드명이 일시적인 쿠버네티스 기반 배치와 사용될 때 훌륭한 툴이다. Prometheus 서비스 발견

클라이언트 프로세스 – Prometheus 활용을 원하는 모든 클라이언트는 2 가지 구성이 필요하다. 우선 Prometheus 호환 포맷 (OpenMetrics)으로 메트릭을 노출하기 위해 Prometheus 클라이언트 라이브러리를 사용해야 한다. 다음에 JMX 메트릭 추출을 위해 YAML 구성 파일을 사용해야 한다. 이 구성 파일은 소비를 위해 속성 일부를 변환, 재명명 및 필터링하는데 사용된다. JVM MBeans이 이 구성 파일을 사용하여 소비를 위한 특정 포맷으로 노출, 변환 및/또는 재명명되기 때문에 JVM 클라이언트에 대해 YAML 구성 파일이 필요하다.

구성 파일

Prometheus 클라이언트 엑스포터(Exporter) 설정

JMX exporter JAR 파일 : 이 파일은 모든 JVM 메트릭을 Prometheus 호환 포맷으로 노출시키는 역할을 담당한다. JAR 파일의 경우 Prometheus 클라이언트가 있는 모든 서버에 복사해야 하며 커맨드 라인에서 Java 에이전트 스위치를 사용해 활성화될 것이다. JAR file

Exporter 구성 파일 : configuration files. https://github.com/prometheus/jmx_exporter

이들 구성 파일은 Prometheus 호환 포맷으로의 변환을 위해 MBeans을 읽는데 모든 서버와 JVM에서 필요하다. 구성 파일은 일부 속성명을 재명명하고 기본 Bean명에 이들을 추가하며 노출된 MBean에 대한 값으로 특정 필드를 사용한다.

카프카 브로커 메트릭 엔드 포인트 설정

구성 파일 예

lowercaseOutputName: true

rules:

Special cases and very specific rules

- pattern : kafka.server<type=(.+), name=(.+), clientId=(.+), topic=(.+), partition=(.*)><>Value

  name: kafka_server_$1_$2

  type: GAUGE大

  labels:

    clientId: "$3"

    topic: "$4"

    partition: "$5"

- pattern : kafka.server<type=(.+), name=(.+), clientId=(.+), brokerHost=(.+), brokerPort=(.+)><>Value

  name: kafka_server_$1_$2

  type: GAUGE

  labels:

    clientId: "$3"

    broker: "$4:$5"

- pattern : kafka.server<type=KafkaRequestHandlerPool, name=RequestHandlerAvgIdlePercent><>OneMinuteRate

  name: kafka_server_kafkarequesthandlerpool_requesthandleravgidlepercent_total

  type: GAUGE

1. ../prometheus-jmx 디렉토리 생성

mkdir ../Prometheus-jmx

2. 디렉토리 접근권한

chmod +rx ../Prometheus-jmx

3. 디렉토리 변경

cd ../Prometheus-jmx

4. 구성 파일 다운로드

wget https://github.com/confluentinc/jmx-monitoring-stacks/blob/6.1.0-post/shared-assets/jmx-

exporter/kafka_broker.yml

wget https://raw.githubusercontent.com/prometheus/jmx_exporter/master/example_configs/kafka-2_0_0.yml

https://github.com/prometheus/jmx_exporter/tree/master/example_configs

5. JAR 파일 다운로드

wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.15.0/jmx_prometheus_javaagent-0.15.0.jar

6. kafka-server-start.sh 파일 수정 (kafka-console-consumer, kafka-console-producer 등)

export KAFKA_OPTS="-javaagent:/usr/local/confluent-6.2.0/Prometheus-jmx/jmx_prometheus_javaagent-0.15.0.jar=7071: /usr/local/confluent-6.2.0/Prometheus-jmx/kafka-broker.yml" (포트 번호)

7. 카프카 서버 재시작 후 엔드포인트 확인

http://<kafka-broker-host-name>:7071/metrics

* Kafka exporter https://github.com/danielqsj/kafka_exporter

https://knowjea.github.io/2021/03/01/.Kafka,-Prometheus,-Grafana.html

토픽 및 파티션에 대해 보다 자세한 정보를 생성, Kafka 재기동 불필요

브로커 서버 HW 리소스 모니터링, 노드 익스포터 (Node Exporter)

$ wget https://github.com/prometheus/node_exporter/releases/download/v1.0.1/node_exporter-1.0.1.linux-386.tar.gz

$ sudo tart -xvzf node_exporter-1.0.1.linux-386.tar.gz /path/to/install

Bin/Node_exporter 실행

각각의 브로커에서 node_exporter 실행하고 Prometheus 설정 파일에 kakfa jmx외에 포함

Prometheus.yml에 jmx 외 node exporter 추가

config.

```
- job_name: 'kafka-nodes'

  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.
```

```
    static_configs:
      - targets:
        - 'localhost:9100' # node_exporter
```

Kafka exporter (Lag 모니터링)

https://github.com/danielqsj/kafka_exporter

다운로드 후 kafka exporter 실행

Prometheus.yml에 kafka exporter 추가

Grafana 대시보드에 추가 (via import kafka exporter)

config.

```
  - job_name: 'kafka-exporter'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets:
        - 'localhost:9308' # kafka_exporter
```

Prometheus 설치

1. 바이너리 파일 다운로드

wget https://github.com/prometheus/prometheus/releases/download/v2.31.1/prometheus-2.31.1.linux-amd64.tar.gz

2. 압축 해제

tar -xvzf prometheus-2.31.1.linux-amd64.tar.gz

3. 설정, prometheus.yml

Scrape_configs 태그 밑에

```
- job_name: "kafka"
    static_configs:
      - targets:
```

```
      - "kafka1:1234"
      - "kafka2:1234"
    labels:
      env: "dev"
```

kafka_exporter 스크래핑 시 디폴트 타임아웃 시간인 10초를 넘어가 **context deadline exceeded** 에러가 발생하여 30초로 변경

```
# my global config
  global:
    scrape_interval:     30s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
    evaluation_interval: 30s # Evaluate rules every 15 seconds. The default is every 1 minute.
    scrape_timeout: 30s

  # Alertmanager configuration
  alerting:
    alertmanagers:
    - static_configs:
      - targets:
        # - alertmanager:9093

  # Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
  rule_files:
    # - "first_rules.yml"
    # - "second_rules.yml"

  # A scrape configuration containing exactly one endpoint to scrape:
  # Here it's Prometheus itself.
  scrape_configs:
    # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
    - job_name: 'kafka'

      # metrics_path defaults to '/metrics'
      # scheme defaults to 'http'.

      static_configs:
        - targets:
          - 'localhost:9308' # kafka_exporter 별도 설치 필요
```

- 'localhost:7071' *# JMX exporter*
- 'localhost:9100' *# node exporter*

4. Prometheus 실행 후 확인

http://localhost:9090/graph

Grafana 설치

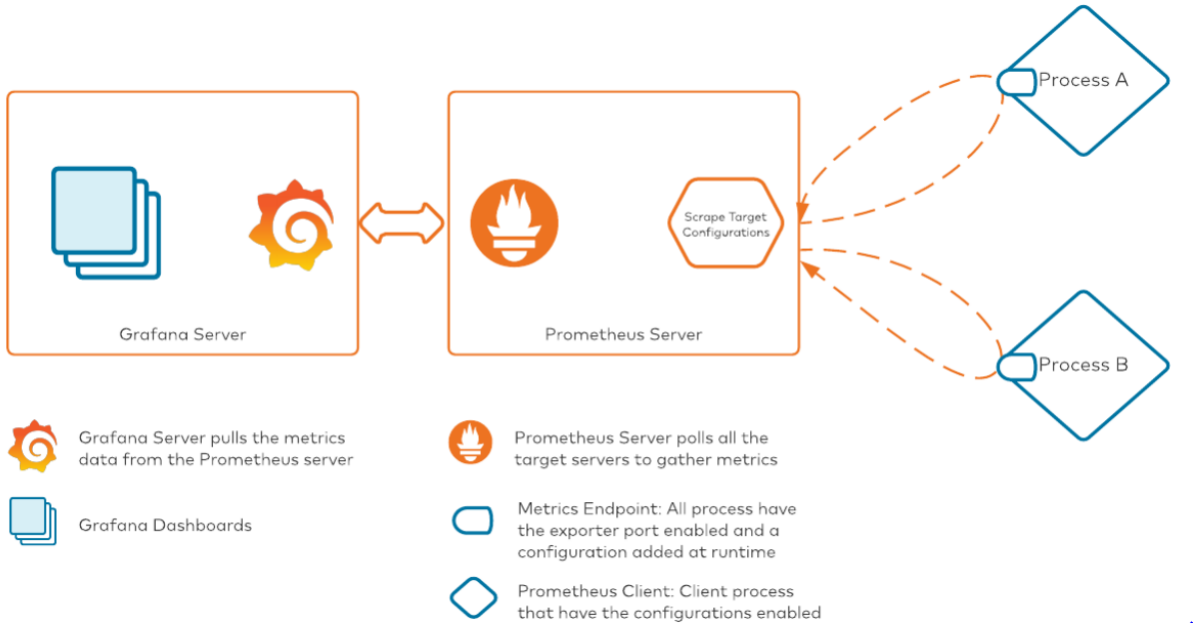
https://grafana.com/grafana/download

https://knowjea.github.io/grafana/2021/03/01/Grafana-%EC%84%A4%EC%B9%98-%EB%B0%8F-%EC%8B%A4%ED%96%89.html

Grafana 실행 후 http://localhost:3000 접속

Prometheus와 그라파나 연결

Prometheus 서버로 스트리밍되는 메트릭 데이터가 있으며 그라파나를 통해 대시보드를 만들 수 있음



그라파나 대시보드 설정

1. JSON 파일을 이용하여 그라파나로 대시보드 import. jmx-monitoring-stacks repository에서 다운로드

2. 대시보드 > Manage 에서 Import

* 스크래핑 데이터를 사용하여 직접 대시보드를 만들 수도 있지만 기존 작성된 대시보드를 import하여 활용

https://grafana.com/grafana/dashboards 에서 kafka로 검색한 후 원하는 대시보드 import

Kafka CMAK Manager

이전 버전 jdk8

신규 버전 jdk11, openlogic openjdk 11 다운로드 후 경로 설정

https://www.openlogic.com/openjdk-downloads 에서 jdk 다운로드

sbt 설치

https://www.scala-sbt.org/1.x/docs/Installing-sbt-on-Linux.html

sudo rm -f /etc/yum.repos.d/bintray-rpm.repo

curl -L https://www.scala-sbt.org/sbt-rpm.repo > sbt-rpm.repo

sudo mv sbt-rpm.repo /etc/yum.repos.d/

sudo yum install sbt

https://github.com/yahoo/CMAK/releases 에서 소스 다운로드

$ PATH=/usr/local/ openlogic-openjdk-11.0.8+10-linux-x64/bin:$PATH ₩

JAVA_HOME=/usr/local/ openlogic-openjdk-11.0.8+10-linux-x64 ₩

./sbt -java-home /usr/local/ openlogic-openjdk-11.0.8+10-linux-x64 dist

카프카 매니저 CMAK 실행, 기본 포트 9000, 여기서는 8080 등으로 변경가능
$ bin/cmak -Dconfig.file=/usr/local/cmak-3.0.0.5/conf/application.conf -Dhttp.port=9000 -java-home /usr/local/openlogic-openjdk-11.0.8+10-linux-x64

Kafka-server-start 파일에 export JMX_PORT=9999 추가