Monitoring Kafka Cluster /w Prometheus

by swoogi@areum.kr

JW Song / Aug. 30, 2022

Something worth

What to talk about

- Why important?
- Various Solutions
- Why Prometheus?
- Examples
- Need more metrics
- I can't watch 24/7
- Our implementations

About Speaker

CCAAK & CCDAK since 2020

- 카프카 클러스터 관리 운영 도구 개발 중
  - 2020년 ~
  - 2021년 ~ 다수 고객사 v1 구축 및 운영 중
  - 연내 시즌2 및 Community Edition 출시 목표
- My Role
  - 기술 탐색 및 아키텍처
  - Kafka 관련 SW 패키징
    - zookeeper, kafka, kafka connect, kafka mirror maker2, etc.
    - Config file generator
    - Installation Packages
    - Dockerize
    - Helm Charts

Cluster Health

Why monitoring is so important?

- 카프카 사용 범위 확대
  - 예전에는 로그 수집 파이프라인 등의 2차 시스템으로 도입
    - 카프카가 멈추더라도 애플리케이션에 영향이 거의 없음
    - 카프카 클러스터 복구에 충분한 시간을 확보할 수 있음
  - 중요한 이벤트 메시지를 중계하는 1차 시스템으로 도입이 증가
    - 카프카가 멈추면 실시간으로 이루어지는 애플리케이션 서비스에서 척추 신경에 장애가 생기는 것과 같은 상황 발생할 수 있음
    - 최대한 빠른 시간안에 카프카 클러스터를 복구해야 함
    - 능동적인 모니터링을 통해 장애를 방지하는 것이 최선

Monitoring Kafka

https://docs.confluent.io/platform/current/kafka/monitoring.html

- 카프카는 내부 상태를 확인할 수 있는 여러 가지 측정 지표(metrics) 제공
  - broker metrics
    - kafka.server:type=ReplicaManager,name=UnderMinIsrPartitionCount
    - kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions
    - kafka.controller:type=KafkaController,name=OfflinePartitionsCount
    - kafka.controller:type=KafkaController,name=ActiveControllerCount
    - kafka.server:type=FetcherLagMetrics,name=ConsumerLag,clientId=([-.\w]+),topic=([-.\w]+),partition=([0-9]+)
  - zookeeper metrics
    - kafka.server:type=SessionExpireListener,name=SessionState
  - producer metrics
    - Starting with 0.8.2
  - consumer metrics
    - Starting with 0.9.0.0

Critical Cases

Broker or cluster stopped

- 데이터 디스크 full에 의한 브로커 중지
  - log.dirs로 지정된 디스크 볼륨에 남은 공간이 '0'이 되면서 브로커 프로세스는 살아 있지만 기능이 멈춤
  - log size per partition 등의 정보를 기본으로 제공하고 있지만, 브로커 서버 데이터 볼륨의 전체 크기 정보가 없어 부족함
- 주키퍼 네트워크 단절에 의한 클러스터 중단
  - 주키퍼 앙상블이 다른 네트워크 존에 구성됨
  - 네트워크 작업에 의해 카프카 브로커와 주키퍼 사이 네트워크가 단절됨
  - 계속된 메타데이터 업데이트 실패로 인해 브로커 프로세스 기능이 멈춤

Various Solutions

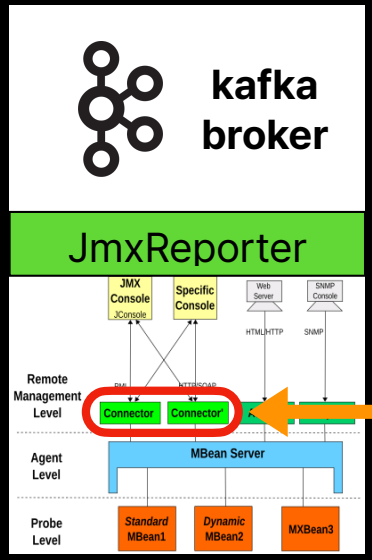
Monitoring kafka cluster

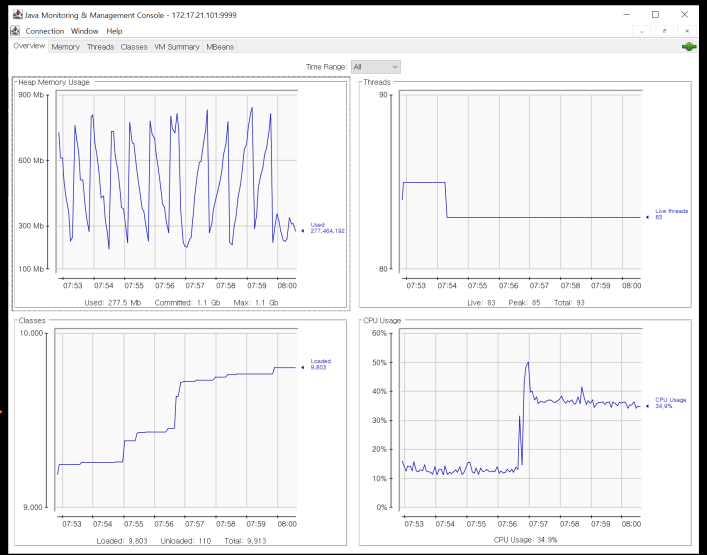
1. metric.reporters
   - 브로커 설정에 metric reporter를 등록
     - org.apache.kafka.common.metrics.MetricsReporter 인터페이스 구현
     - 대표적으로 org.apache.kafka.common.metrics.JmxReporter 클래스가 있음
     - JmxReporter는 metrics.reporters 설정에 항상 자동으로 포함됨
     - confluent는 io.confluent.metrics.reporter.ConfluentMetricsReporter를 통해 카프카 토픽으로 메트릭을 수집
2. JConsole via JMX
   - JConsole은 JMX를 통해 JVM의 상태를 바로 확인할 수 있는 도구
3. **Prometheus /w JMX exporter**
4. Elastic Stack /w Jolokia + Metricbeat

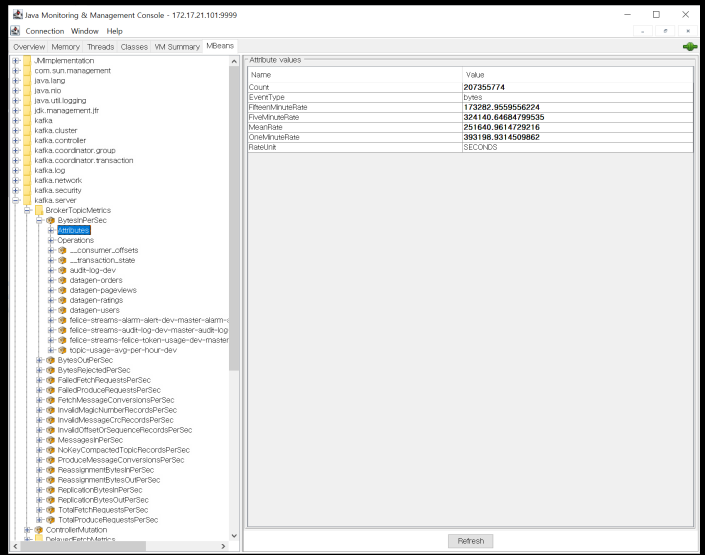
JMX?

Java Management eXtensions

- 자바 애플리케이션 관리와 모니터링을 위한 기술
  - 애플리케이션, 시스템 객체, 장치 등을 모니터링 및 관리
  - MBean(Managed Bean)에 의해 리소스가 제공됨
  - JDK와 함께 배포되는 JConsole을 통해 JVM에 접속 가능



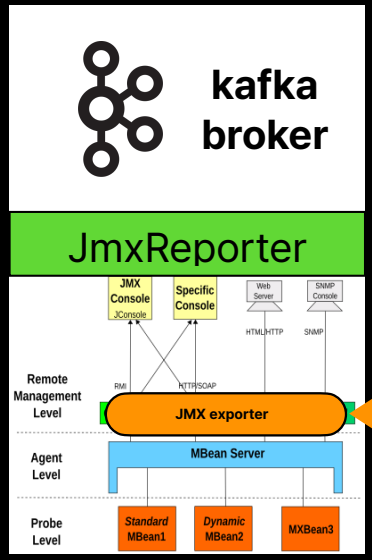


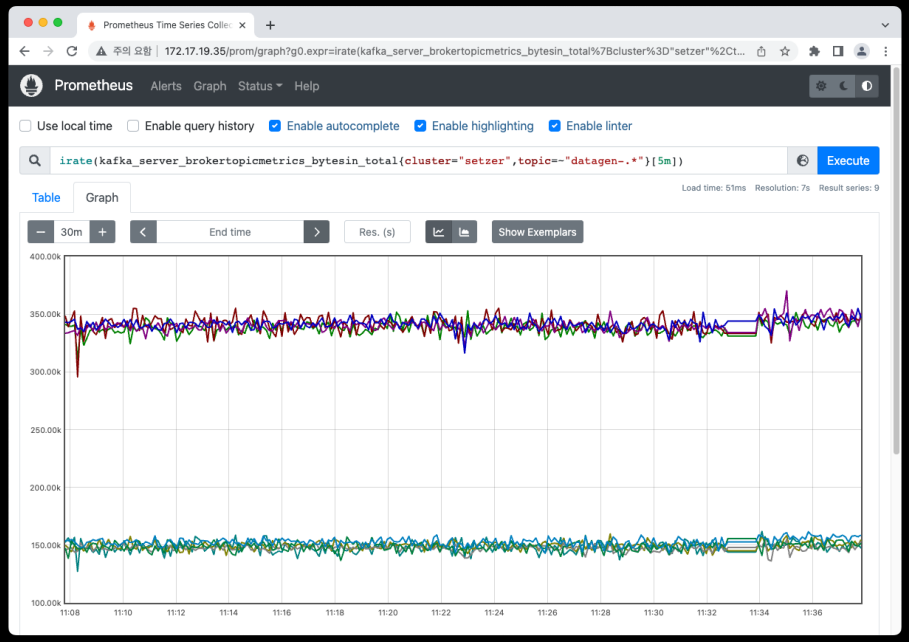


Scrapping Metrics

Prometheus /w JMX exporter

- JMX exporter
  - JMX 메트릭을 HTTP(S)로 노출해주는 javaagent
    - javaagent는 주 애플리케이션과 동일한 JVM에서 실행되며 최초 진입점 main()보다 먼저 실행될 수 있는 특별한 애플리케이션
    - Jolokia + Metricbeat도 비슷한 형태





Expose Metrics

Starting kafka brokers /w JMX exporter

- 카프카 시작 시 특별한 환경 변수
  - JMX_PORT, KAFKA_JMX_OPTS
  - LOG_DIR, KAFKA_LOG4J_OPTS
  - KAFKA_OPTS
  - KAFKA_HEAP_OPTS, KAFKA_JVM_PERFORMANCE_OPTS
- javaagent : JMX exporter
  - https://github.com/prometheus/jmx_exporter
  - https://github.com/prometheus/jmx_exporter/tree/master/example_configs

```
# START KAFKA BROKER
KAFKA_OPTS=-javaagent:/path/to/jmx_prometheus_javaagent-0.17.0.jar=9404:/path/to/kafka-2_0_0.yml \
bin/kafka-server-start.sh -daemon ./config/server.properties

# TEST JMX exporter
curl localhost:9404/metrics
```

Exporter Rules

Configuring JMX exporter

Extra Rules

```
lowercaseOutputName: true

rules:
# very specific rule for cluster id
- pattern: 'kafka.server<type=KafkaServer, name=ClusterId><>Value: (.+)'
  name: kafka_server_kafkaserver_clusterid
  value: 1
  labels:
    clusterId: "$1"
  help: "kafka server JMX metric cluster id"
  type: GAUGE
## very specific rule for app-info
- pattern: kafka.(\w+)<type=app-info><>start-time-ms
  name: kafka_$1_start_time_seconds
  help: "Kafka $1 JMX metric start time seconds"
  type: GAUGE
  valueFactor: 0.001
- pattern: 'kafka.(\w+)<type=app-info><>(commit-id|version): (.+)'
  name: kafka_$1_info
  value: 1
  labels:
    $2: "$3"
  help: "Kafka $1 JMX metric info version and commit-id"
  type: GAUGE
# Special cases and very specific rules
- pattern : kafka.server<type=(.+), name=(.+), clientId=(.+), topic=(.+), partition=(.*)><>Value
  name: kafka_server_$1_$2
  type: GAUGE
  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.coordinator.(\w+)<type=(.+), name=(.+)><>Value
  name: kafka_coordinator_$1_$2_$3
  type: GAUGE

# Generic per-second counters with 0-2 key/value pairs
- pattern: kafka.(\w+)<type=(.+), name=(.+)PerSec\w*, (.+)=(.+), (.+)=(.+)><>Count
  name: kafka_$1_$2_$3_total
```

Example Config

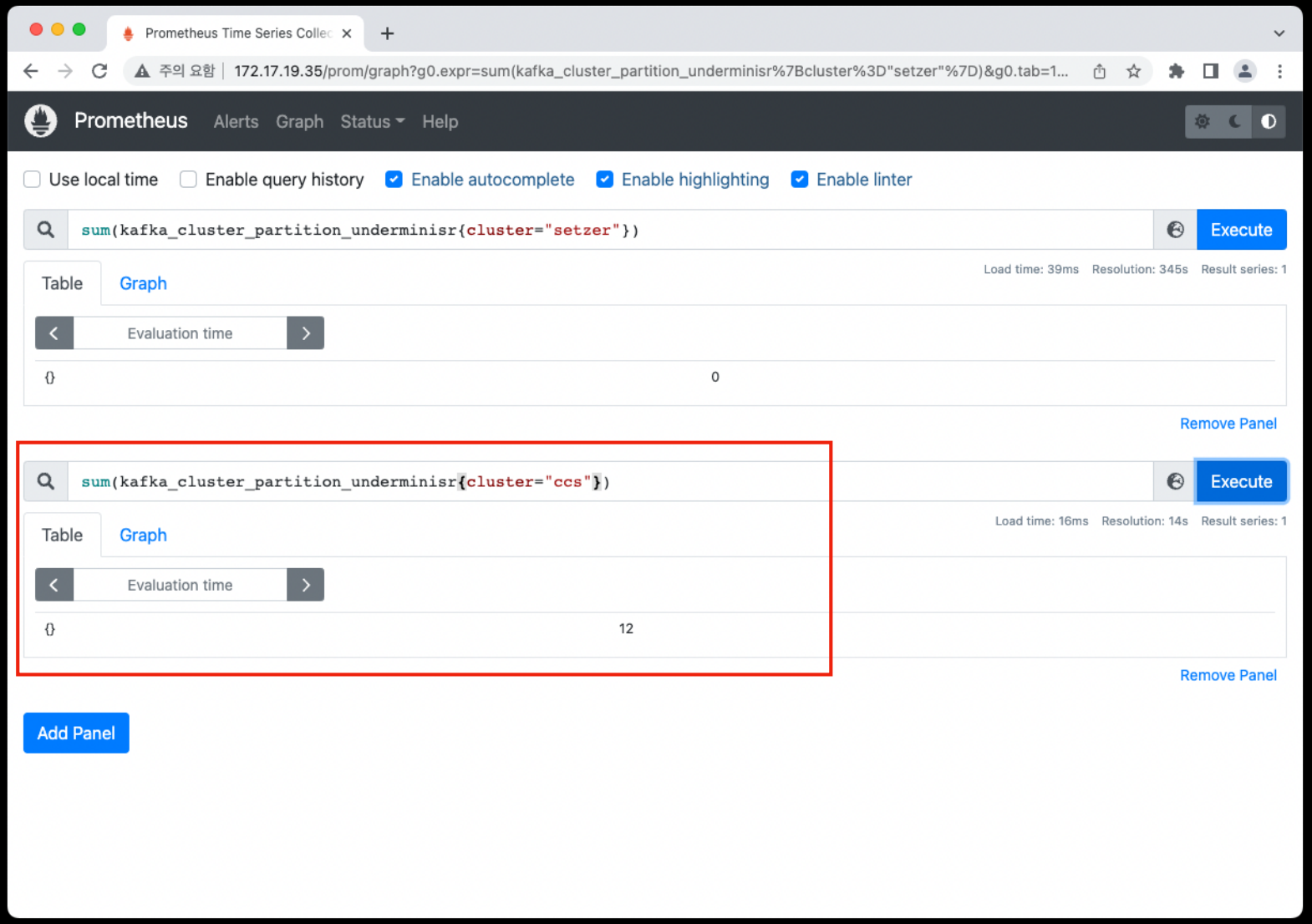
Scrapping kafka metrics from brokers

```yaml
## prometheus.yml ##

#### job : kafka ####
- job_name: kafka
  scrape_interval: 5s
  file_sd_configs:
    - files:
      - targets/kafka.yaml
      refresh_interval: 5m

#### job : kafka_node ####
- job_name: kafka_node
  scrape_interval: 5s
  file_sd_configs:
    - files:
      - targets/kafka_node.yaml
      refresh_interval: 5m
```

```yaml
## kafka.yaml ##

- targets:
  - "172.17.25.20:9404"
  labels:
    cluster: "ccs"
    brokerId: "0"
- targets:
  - "172.17.25.21:9404"
  labels:
    cluster: "ccs"
    brokerId: "1"
- targets:
  - "172.17.25.22:9404"
  labels:
    cluster: "ccs"
    brokerId: "2"
- targets:
  - "172.17.25.40:9404"
  labels:
    cluster: "setzer"
    brokerId: "0"
- targets:
  - "172.17.25.41:9404"
  labels:
    cluster: "setzer"
    brokerId: "1"
- targets:
  - "172.17.25.42:9404"
  labels:
    cluster: "setzer"
    brokerId: "2"
```

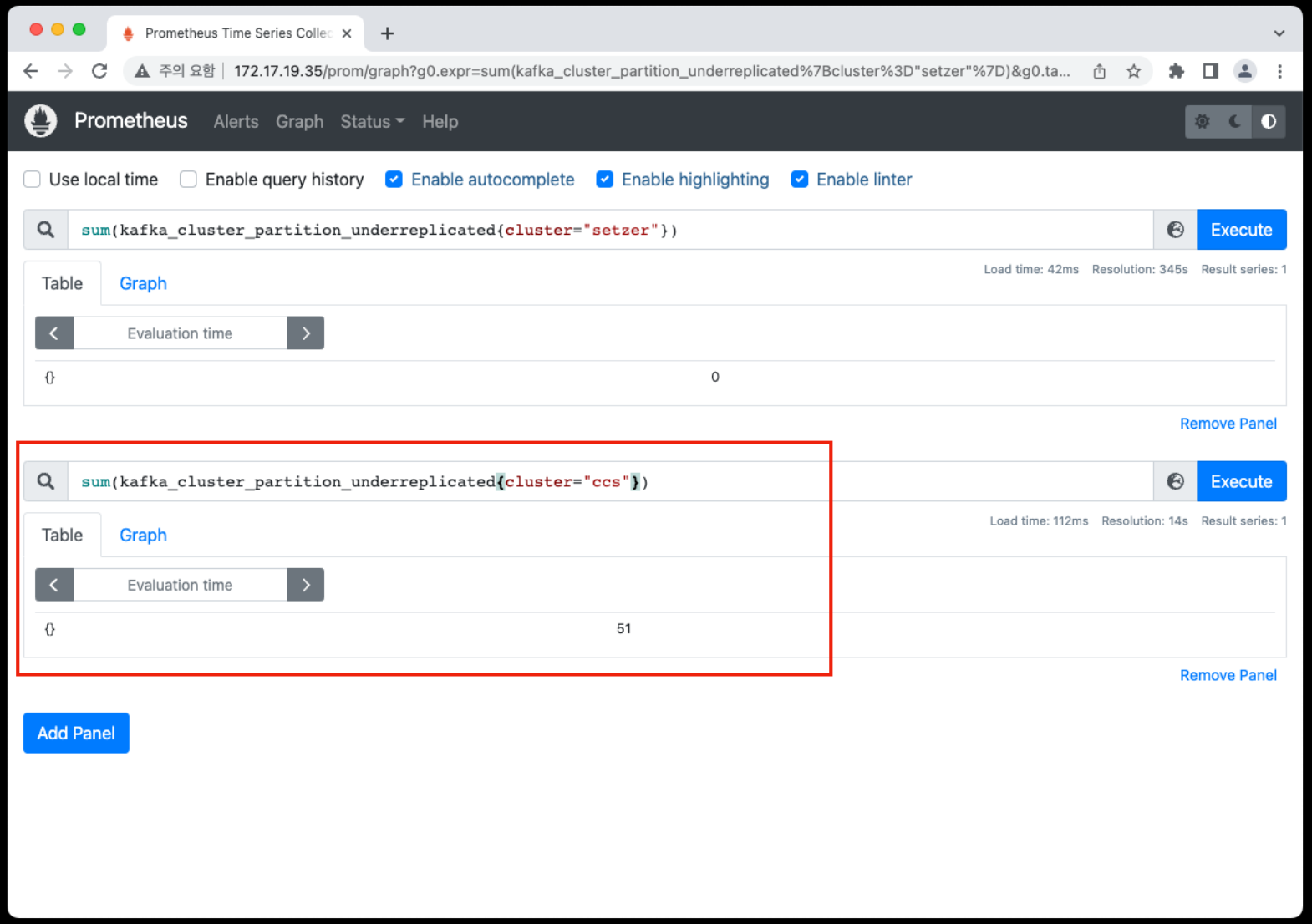
Metrics Example

kafka.server:type=ReplicaManager,name=UnderMinIsrPartitionCount



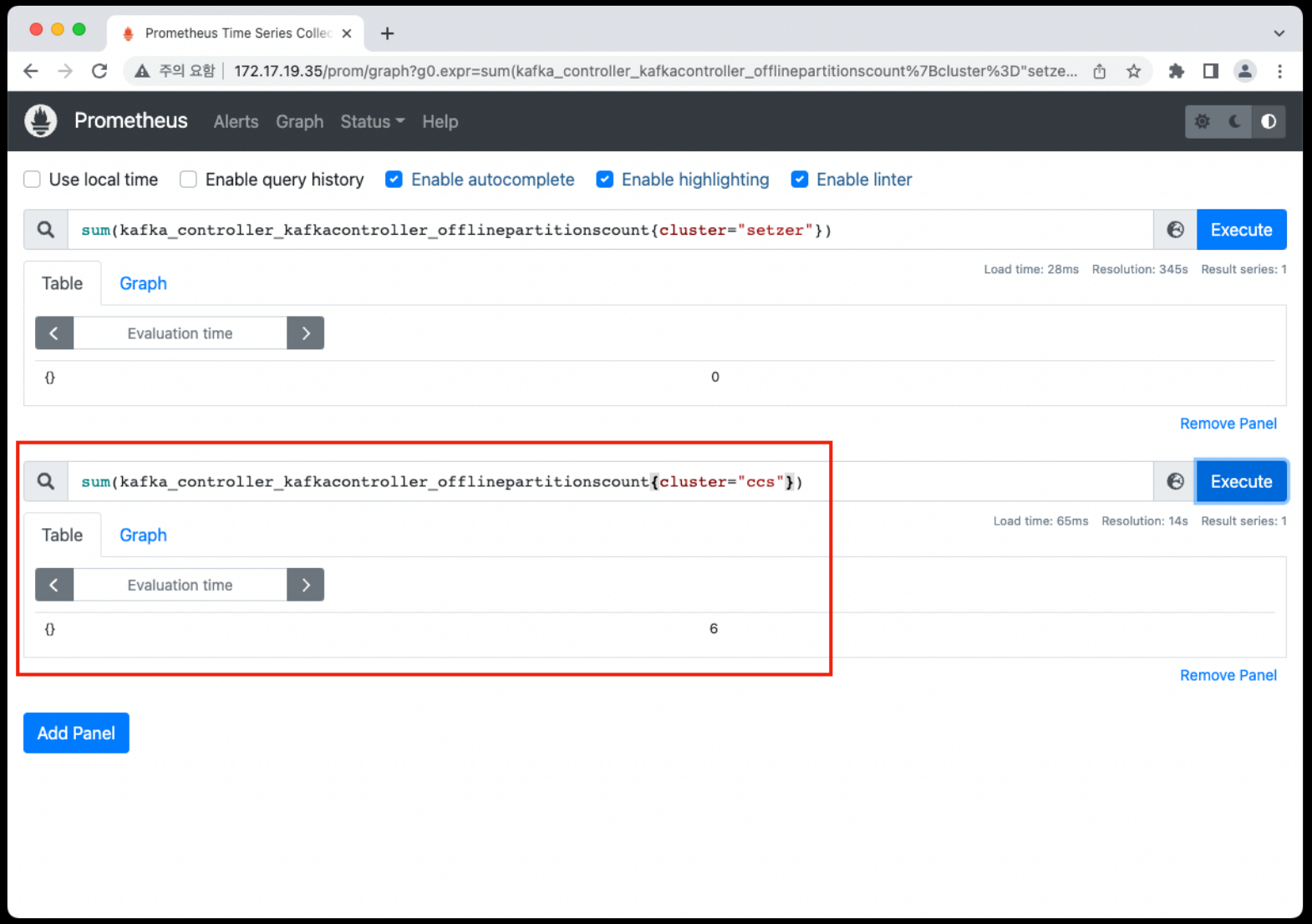
Metrics Example

kafka.server:type=ReplicaManager,name=UnderReplicatedPartitions



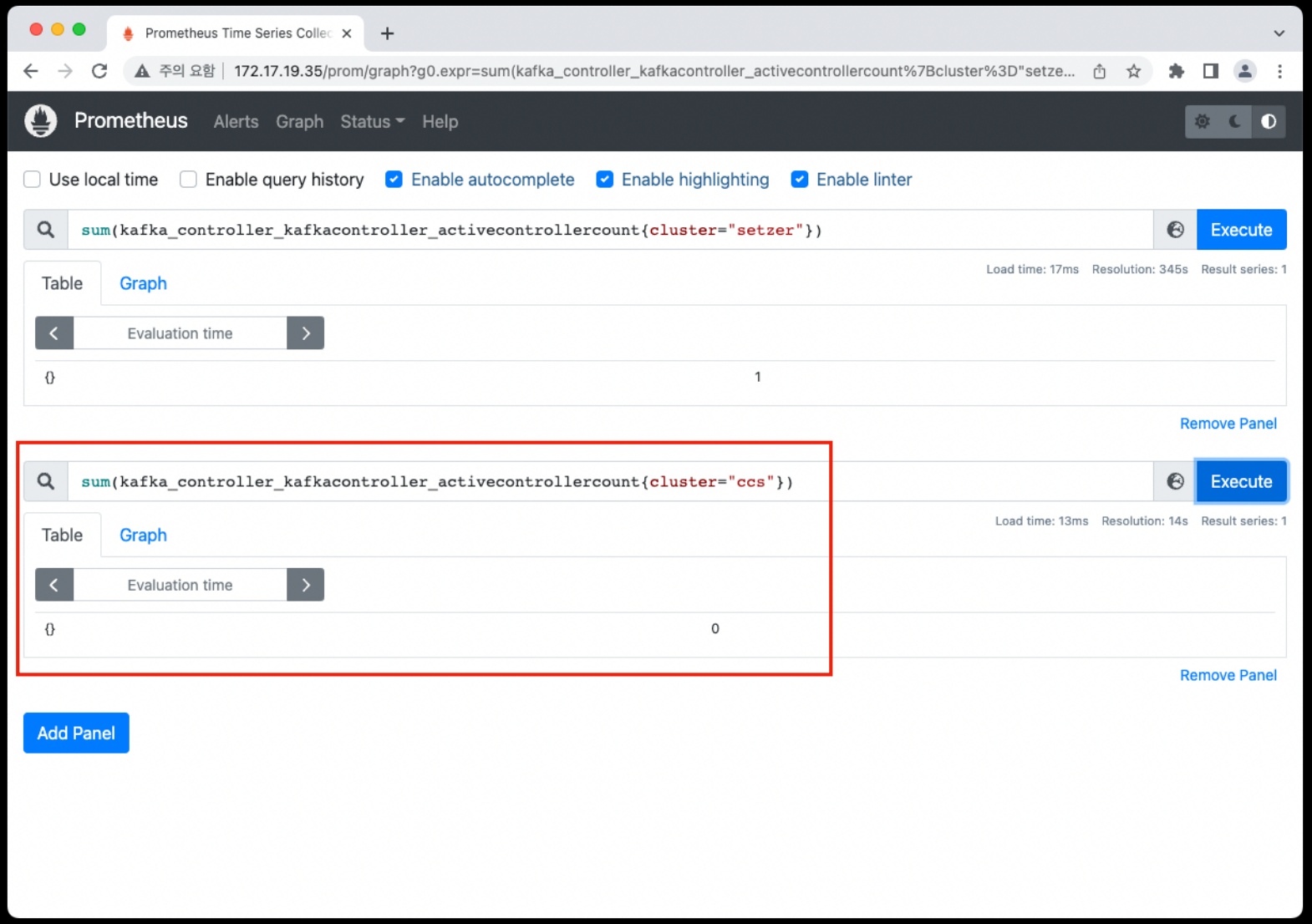
Metrics Example

kafka.controller:type=KafkaController,name=OfflinePartitionsCount



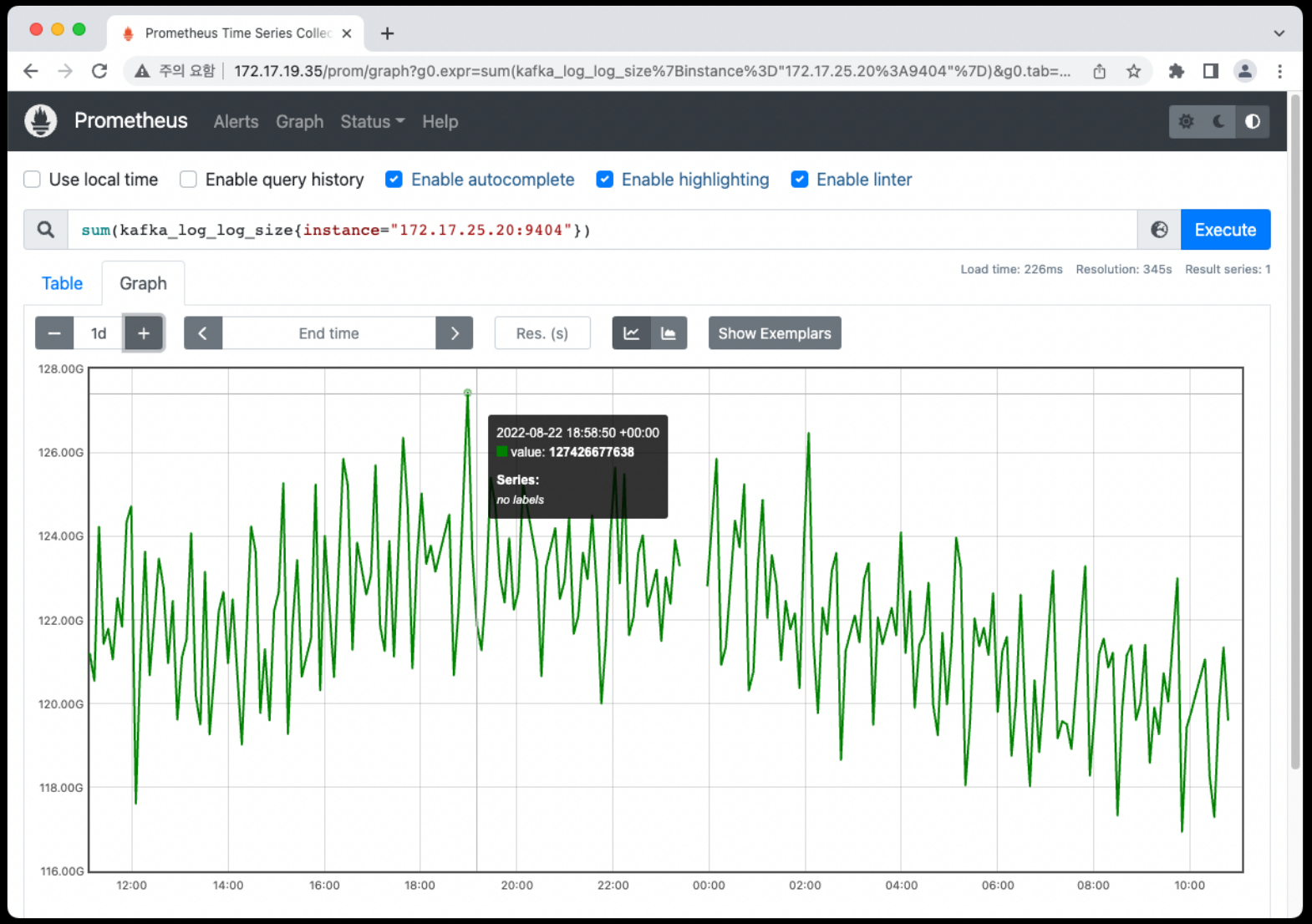
Metrics Example

kafka.controller:type=KafkaController,name=ActiveControllerCount



Metrics Example

kafka.log:type=Log,name=Size



Advantages

Why prometheus is better

- Apache ZooKeeper
  - 3.6.0 버전부터 Metrics Provider로 Prometheus를 번들로 포함
  - 간단한 설정만으로 주키퍼 및 JVM 메트릭을 HTTP를 통해 노출
  - 한 가지 도구를 이용하여 카프카 클러스터 모니터링 시스템 구축

```
# ENABLE METRICS PROVIDER @zoo.cfg
metricsProvider.className=org.apache.zookeeper.metrics.prometheus.PrometheusMetricsProvide
metricsProvider.httpPort=7000
metricsProvider.exportJvmInfo=true
```

What about Server

More metrics are needed

- Node Exporter
  - 노드의 각종 메트릭을 HTTP로 노출해 주는 서버 애플리케이션
  - CPU, Memory, Disk, 네트워크 및 다수의 메트릭 제공
  - https://github.com/prometheus/node_exporter

Example Config

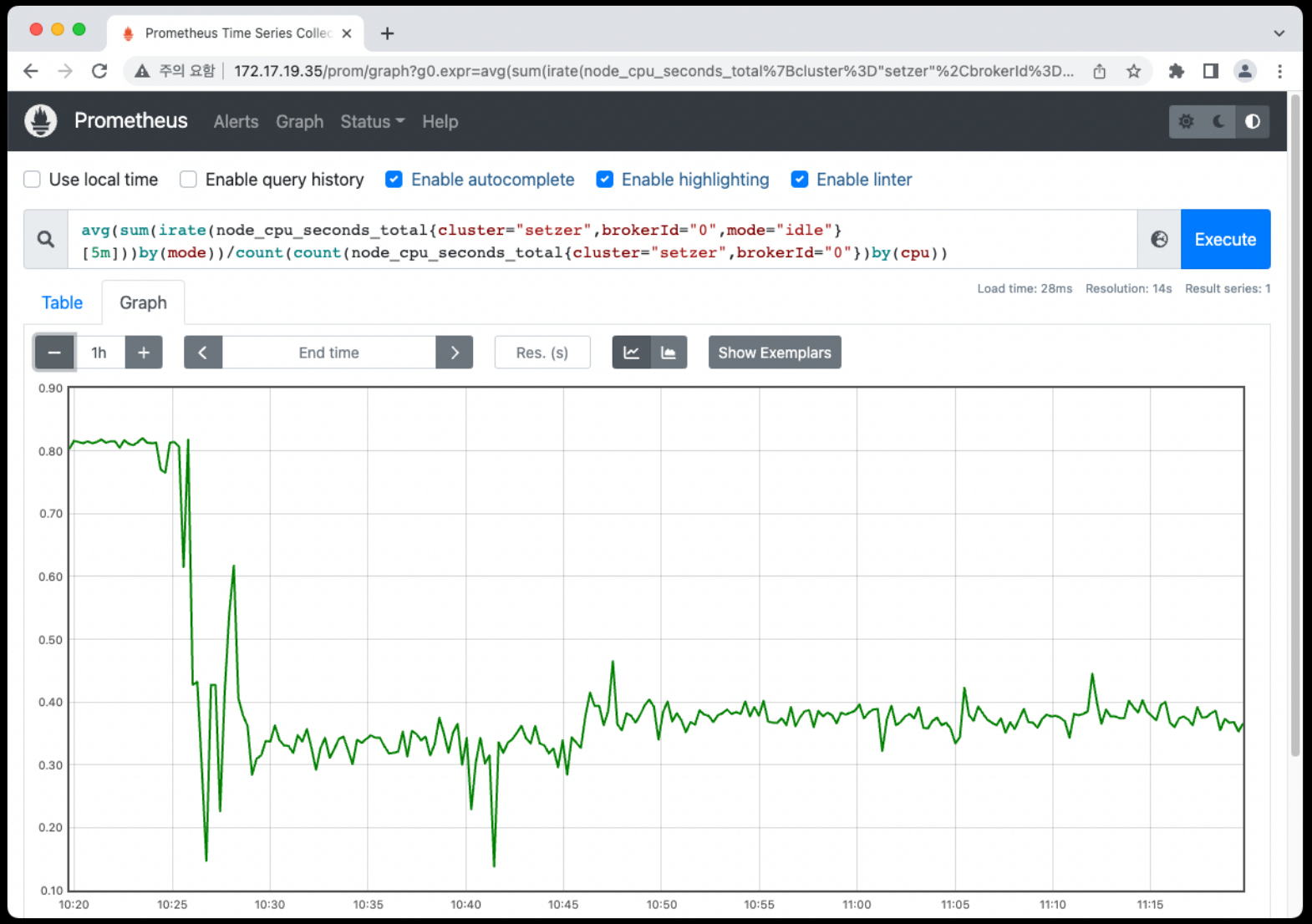
Scrapping node metrics from brokers

```yaml
## prometheus.yml ##

#### job : kafka ####
- job_name: kafka
  scrape_interval: 5s
  file_sd_configs:
    - files:
      - targets/kafka.yaml
      refresh_interval: 5m

#### job : kafka_node ####
- job_name: kafka_node
  scrape_interval: 5s
  file_sd_configs:
    - files:
      - targets/kafka_node.yaml
      refresh_interval: 5m
```

```yaml
## kafka_node.yaml ##

- targets:
  - "172.17.25.20:9100"
  labels:
    cluster: "ccs"
    brokerId: "0"
- targets:
  - "172.17.25.21:9100"
  labels:
    cluster: "ccs"
    brokerId: "1"
- targets:
  - "172.17.25.22:9100"
  labels:
    cluster: "ccs"
    brokerId: "2"
- targets:
  - "172.17.25.40:9100"
  labels:
    cluster: "setzer"
    brokerId: "0"
- targets:
  - "172.17.25.41:9100"
  labels:
    cluster: "setzer"
    brokerId: "1"
- targets:
  - "172.17.25.42:9100"
  labels:
    cluster: "setzer"
    brokerId: "2"
```

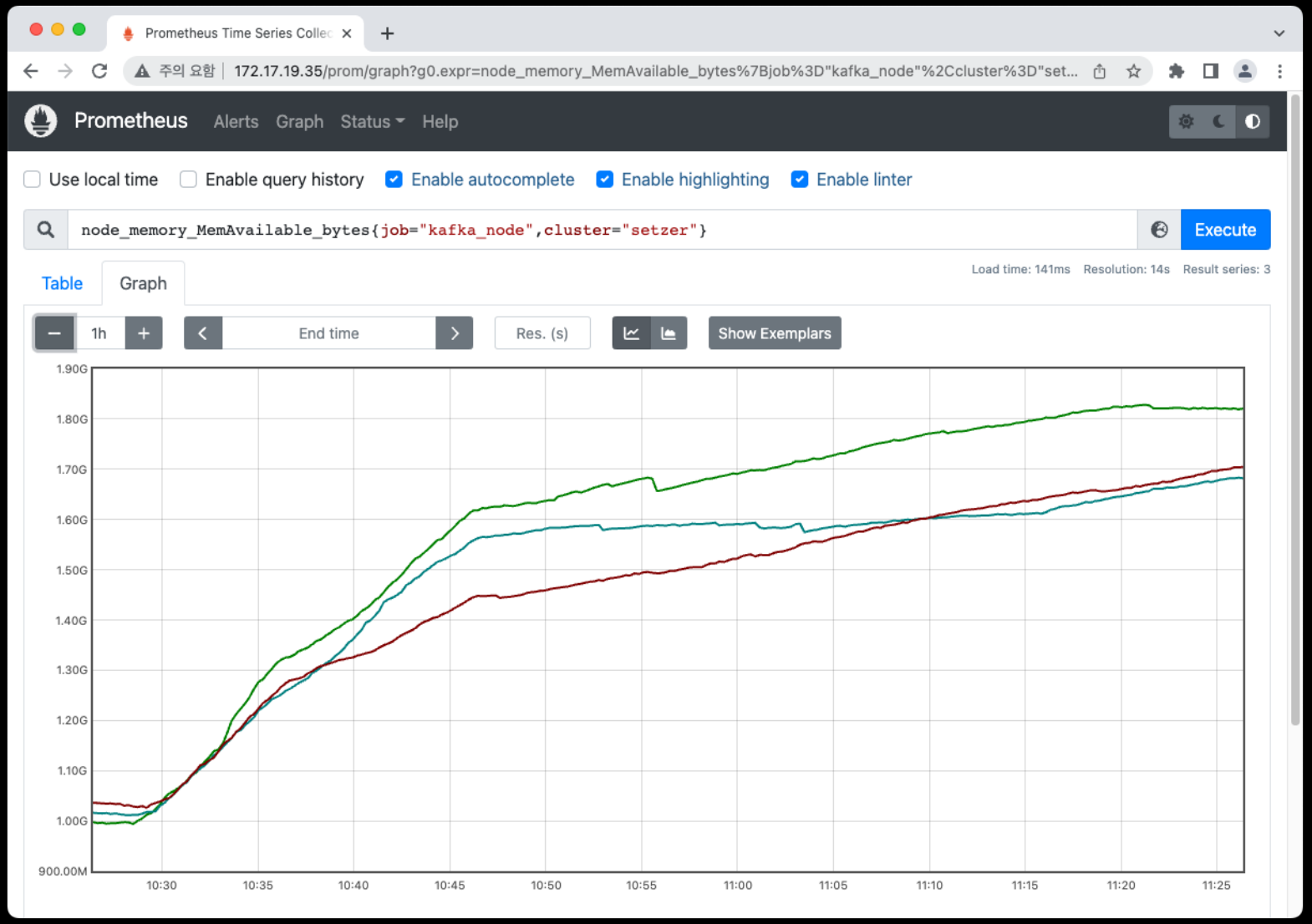
Metrics Example

Average CPU idle



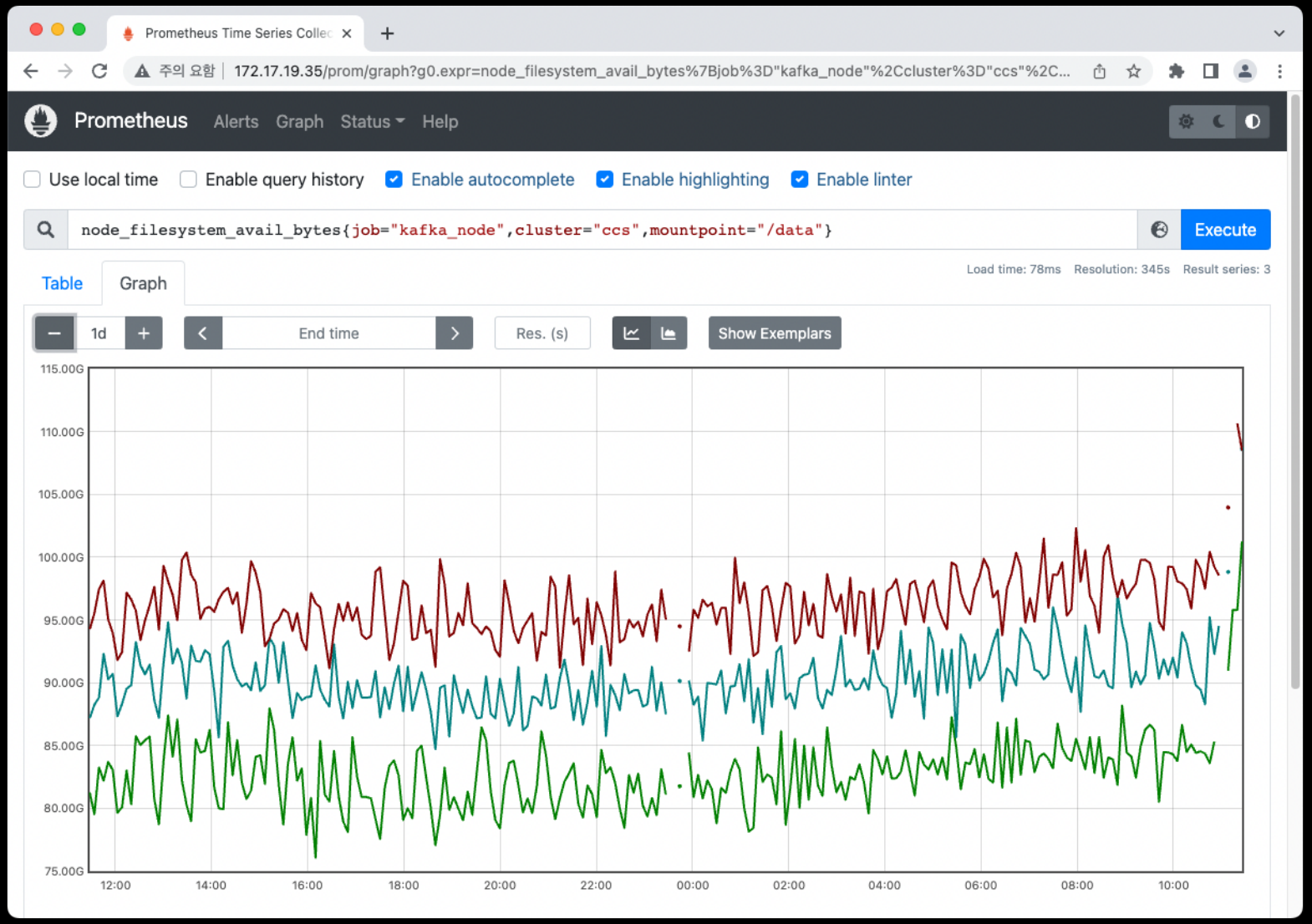
Metrics Example

Memory available bytes



Metrics Example

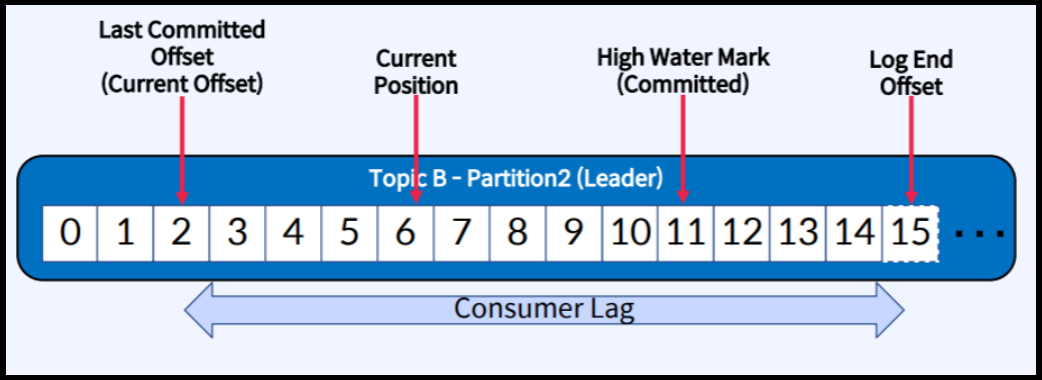
Filesystem available bytes



Offsets

Everything is working fine?

- "컨슈머 랙"?
  - LEO(Log End Offset) - LCO(Last Committed Offset)
    - LEO = 프로듀서가 다음에 보내온 메시지를 쓸 위치
    - LCO = 컨슈머가 마지막으로 커밋한 위치
  - 값이 큰 것 보다 전체적인 경향이 더 중요함
  - 값이 계속 커지는 것은 메시지 유입에 비해 처리가 늦다는 것을 의미



Consumer Healthy

Evaluate consumer lag /w burrow

The Burrow (Franz Kafka Stop-Motion Short Film)

- burrow
  - https://github.com/linkedin/Burrow
  - "컨슈머 랙"을 지속해서 모니터링하고 평가할 수 있는 오픈소스 도구
  - 평가된 값을 HTTP로 노출하여 Prometheus를 통해 수집할 수 있음
  - 일정 시간 간격으로 수집한 메트릭 시리즈를 윈도로 설정하여 평가
    - https://github.com/linkedin/Burrow/wiki/http-request-consumer-group-status
    - https://github.com/linkedin/Burrow/blob/master/core/protocol/evaluator.go

Code	Name	Description
0	NOTFOUND	컨슈머 그룹 없음
1	OK	컨슈머 그룹 / 파티션 상태 GOOD
2	WARN	그룹 혹은 파티션 상태 WARNING. (예) 오프셋은 이동하지만 랙도 증가
3	ERR	그룹 상태 ERROR. (예) 오프셋은 멈췄지만, 랙은 0이 아님
4	STOP	파티션 상태 STOPPED. (예) 장시간 오프셋 커밋 없음
5	STALL	파티션 상태 STALLED. (예) 오프셋은 커밋되지만 랙에 변화 없고 0이 아님
6	REWIND	파티션 상태 REWIND. (예) 오프셋이 이전 값보다 작은 값으로 커밋 됨

Example Config

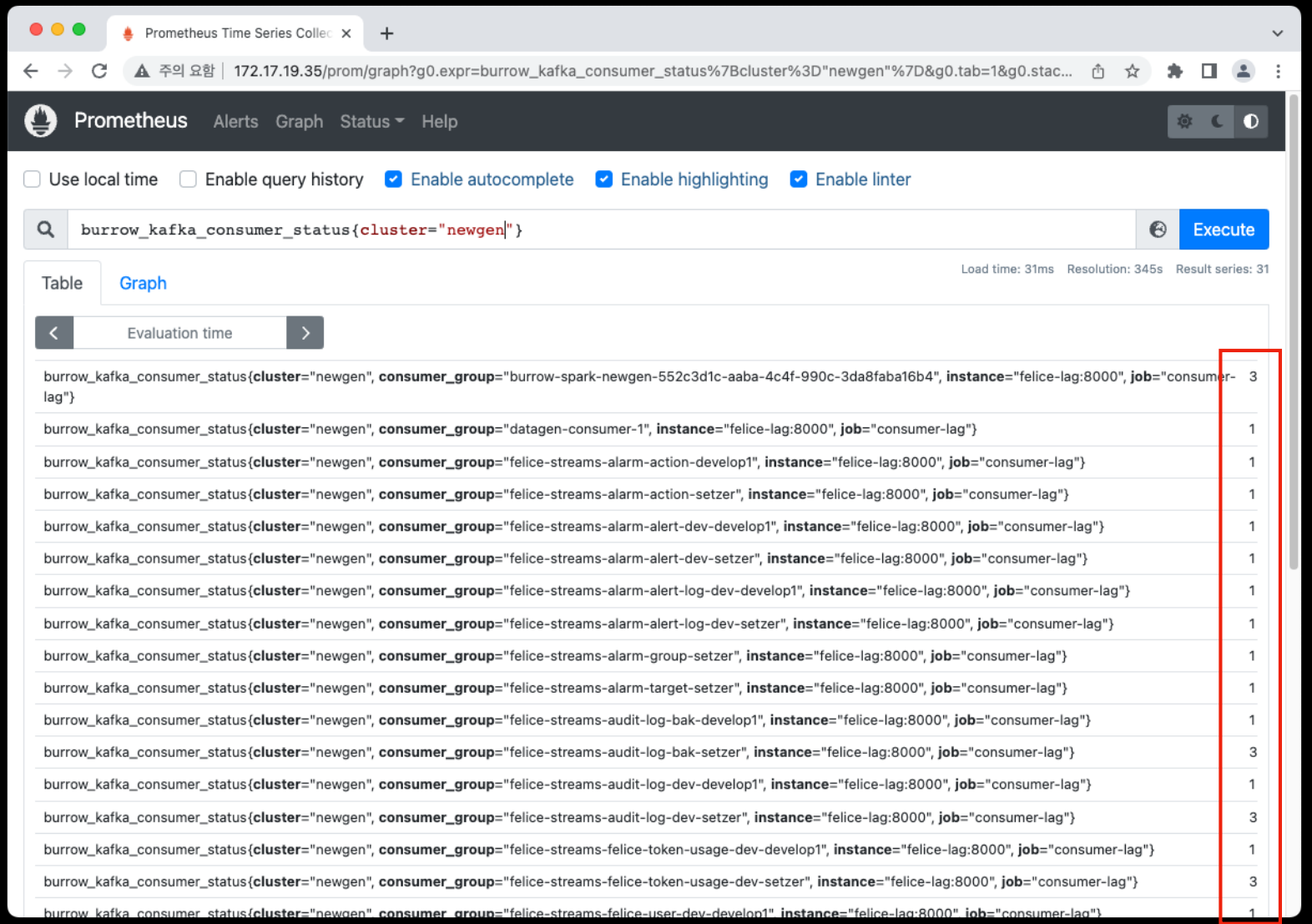
Scrapping consumer healthy from the burrow

```yaml
## prometheus.yml ##

  - job_name: 'consumer-lag'
    scrape_interval: 5s
    static_configs:
      - targets: [ 'burrow:8000' ]
```

```yaml
## burrow.yaml ##

logging:
  level: "info"
httpserver:
  default:
    address: ":8000"
storage:
  felice:
    intervals: 5
    class-name: "inmemory"
    expire-group: 86400
sasl:
  ccs:
    mechanism: "SCRAM-SHA-512"
    username: "admin"
    password: "admin-secret"
cluster:
  ccs:
    servers:
    - "172.17.25.20:9093"
    - "172.17.25.21:9093"
    - "172.17.25.22:9093"
    class-name: "kafka"
    client-profile: "ccs"
    topic-refresh: 30
    offset-refresh: 5
    groups-reaper-refresh: 300
  setzer:
    servers:
    - "172.17.25.40:9092"
    - "172.17.25.41:9092"
    - "172.17.25.42:9092"
    class-name: "kafka"
    client-profile: "setzer"
client-profile:
  ccs:
    sasl: "ccs"
    read-timeout: 5
    dial-timeout: 10
    kafka-version: "3.0.0"
  setzer:
    read-timeout: 5
    dial-timeout: 10
    kafka-version: "3.0.0"
```

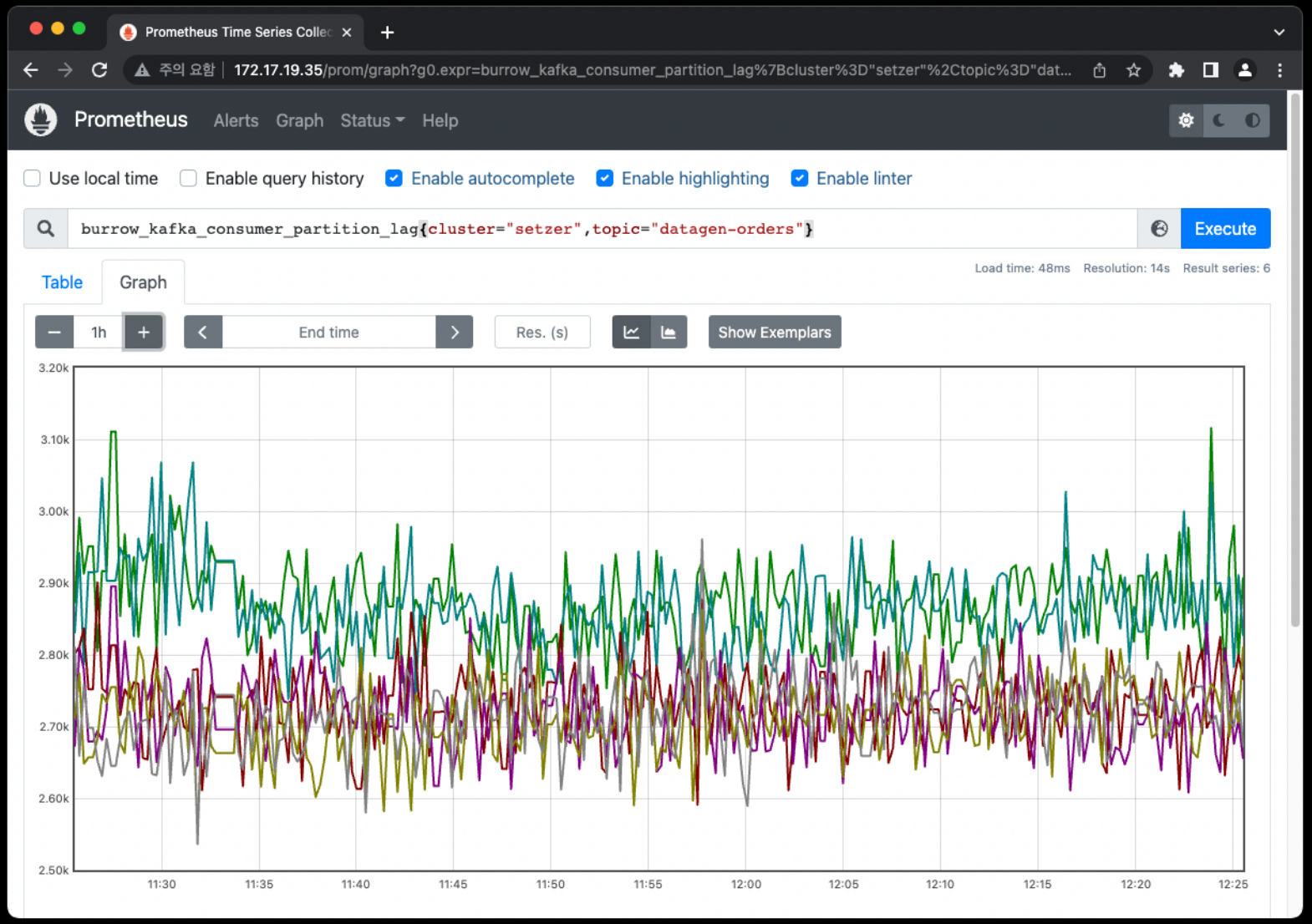
Metrics Example

Consumer status



Metrics Example

Partition lag



Constraint

Weak point of the burrow

- 카프카 인증 지원 부족
  - 카프카 SASL은 다양한 메커니즘 지원
    - PLAIN, SCRAM-SHA-256/SCRAM-SHA-512, GSSAPI, OAUTHBEARER
  - Burrow는?
- 개발이 느린 편
  - sarama는 업데이트되었지만...
  - 코드는 업데이트되었지만 매뉴얼 업데이트는...
  - 쌓여 가는 "Pull requests"
    - SASL/PLAIN, Config Auto Reload, SASL/GSSAPI, ...

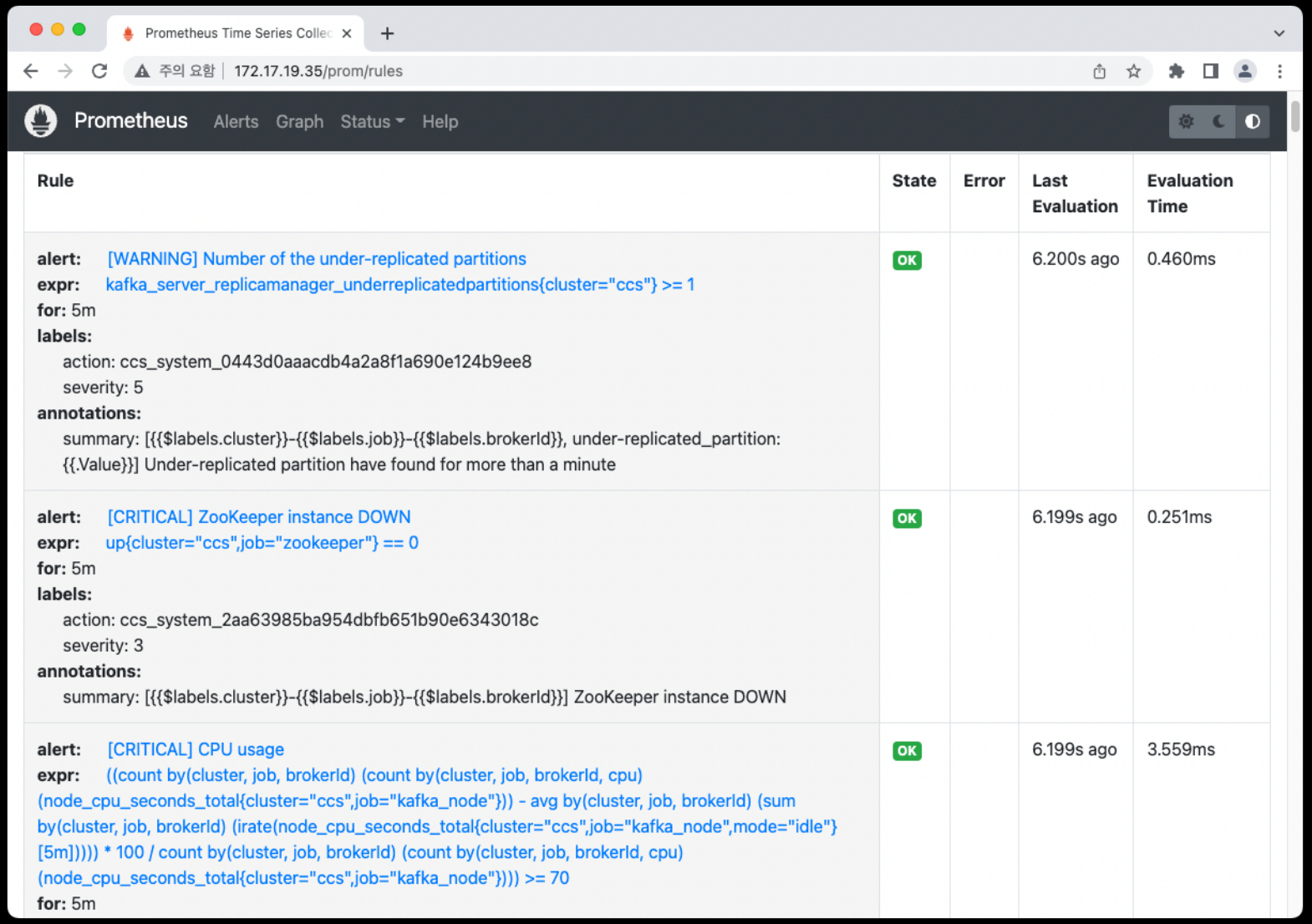
Can't watch 24/7

Need a way to know events

- Prometheus
  - targets = 메트릭 스크랩 대상
  - rules = 스크랩된 메트릭 평가 규칙
    - 평가 규칙에 부합하는 경우 Alertmanager로 alert 전달
- Alertmanager
  - alert를 email, slack, pageduty 등으로 전송
- Burrow
  - notifier 모듈 = 컨슈머 그룹 상태를 외부 HTTP 서버로 전송
  - 프로메테우스 rule을 통해 더 상세하게 평가 가능

Rule Example

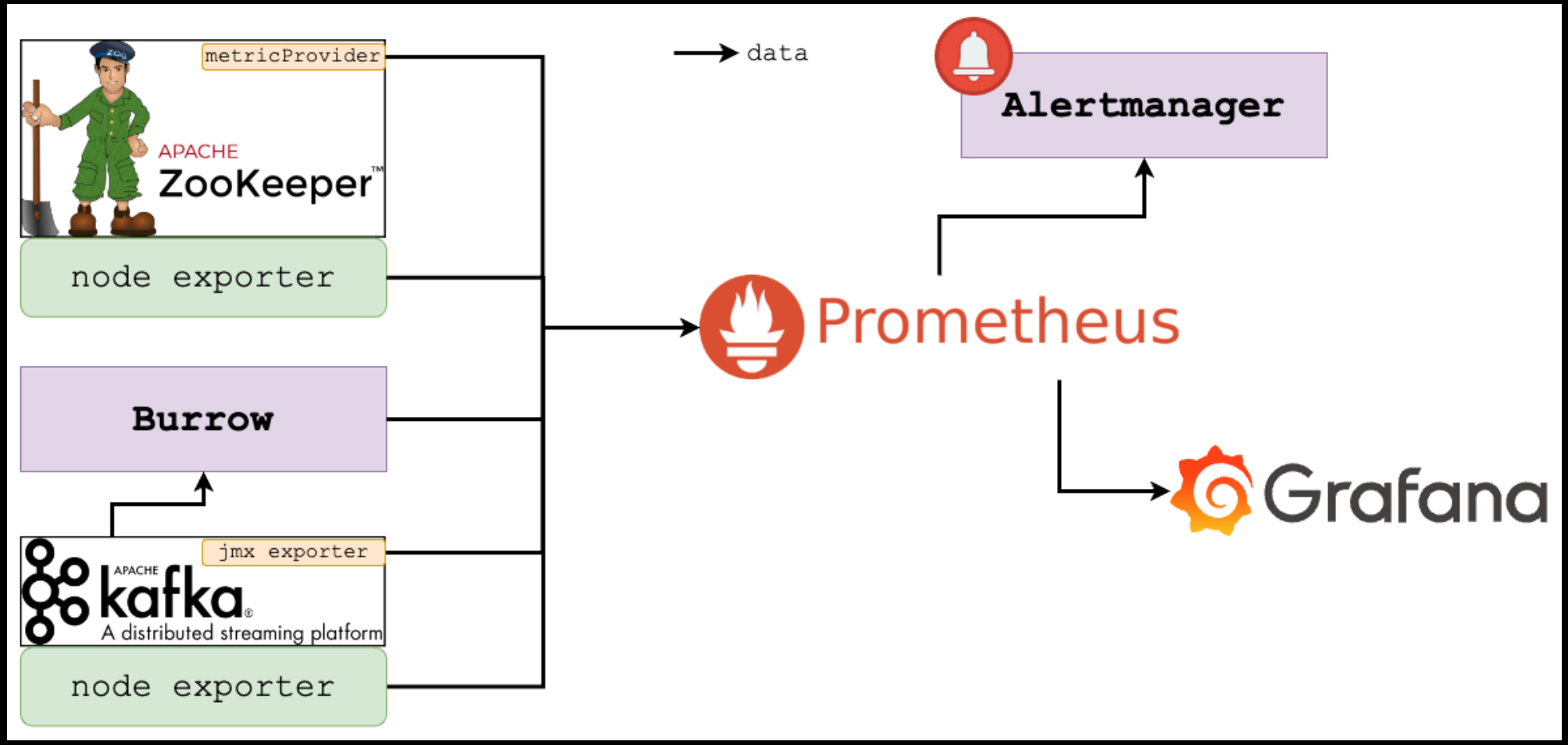
Alerting issues



Rule	State	Error	Last Evaluation	Evaluation Time
alert: [WARNING] Number of the under-replicated partitions **expr:** kafka_server_replicamanager_underreplicatedpartitions{cluster="ccs"} >= 1 **for:** 5m **labels:** action: ccs_system_0443d0aaacdb4a2a8f1a690e124b9ee8 severity: 5 **annotations:** summary: [{{$labels.cluster}}-{{$labels.job}}-{{$labels.brokerId}}, under-replicated_partition: {{.Value}}] Under-replicated partition have found for more than a minute	OK		6.200s ago	0.460ms
alert: [CRITICAL] ZooKeeper instance DOWN **expr:** up{cluster="ccs",job="zookeeper"} == 0 **for:** 5m **labels:** action: ccs_system_2aa63985ba954dbfb651b90e6343018c severity: 3 **annotations:** summary: [{{$labels.cluster}}-{{$labels.job}}-{{$labels.brokerId}}] ZooKeeper instance DOWN	OK		6.199s ago	0.251ms
alert: [CRITICAL] CPU usage **expr:** ((count by(cluster, job, brokerId) (count by(cluster, job, brokerId, cpu) (node_cpu_seconds_total{cluster="ccs",job="kafka_node"})) - avg by(cluster, job, brokerId) (sum by(cluster, job, brokerId) (irate(node_cpu_seconds_total{cluster="ccs",job="kafka_node",mode="idle"}[5m])))) * 100 / count by(cluster, job, brokerId) (count by(cluster, job, brokerId, cpu) (node_cpu_seconds_total{cluster="ccs",job="kafka_node"}))) >= 70 **for:** 5m	OK		6.199s ago	3.559ms

Avengers Assemble

Integrate all parts into one system



Client metrics

Scrapping metrics via pushgateway

1. client(producer, consumer)에서 메트릭을 추출
   - org.apache.kafka.clients.producer.KafkaProducer<K,V>

```
Map<MetricName,? extends Metric>        metrics()
                                        Get the full set of internal metrics maintained by the producer.
```

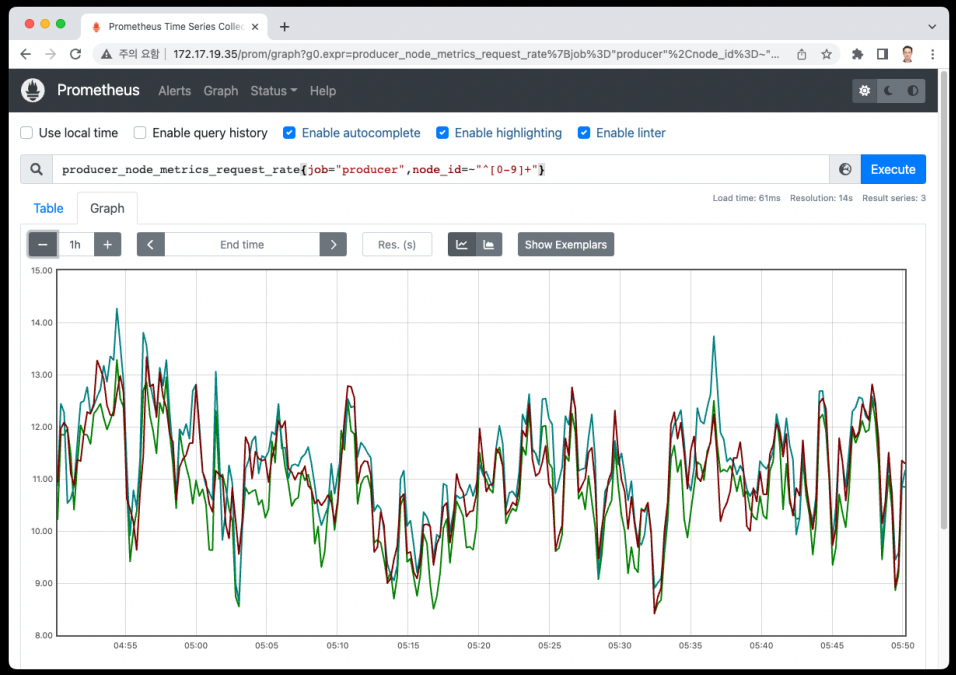
   - org.apache.kafka.clients.consumer.KafkaConsumer<K,V>

```
Map<MetricName,? extends Metric>        metrics()
                                        Get the metrics kept by the consumer
```

2. pushgateway에 전송
   - https://github.com/prometheus/pushgateway
   - io.prometheus.client.exporter.PushGateway

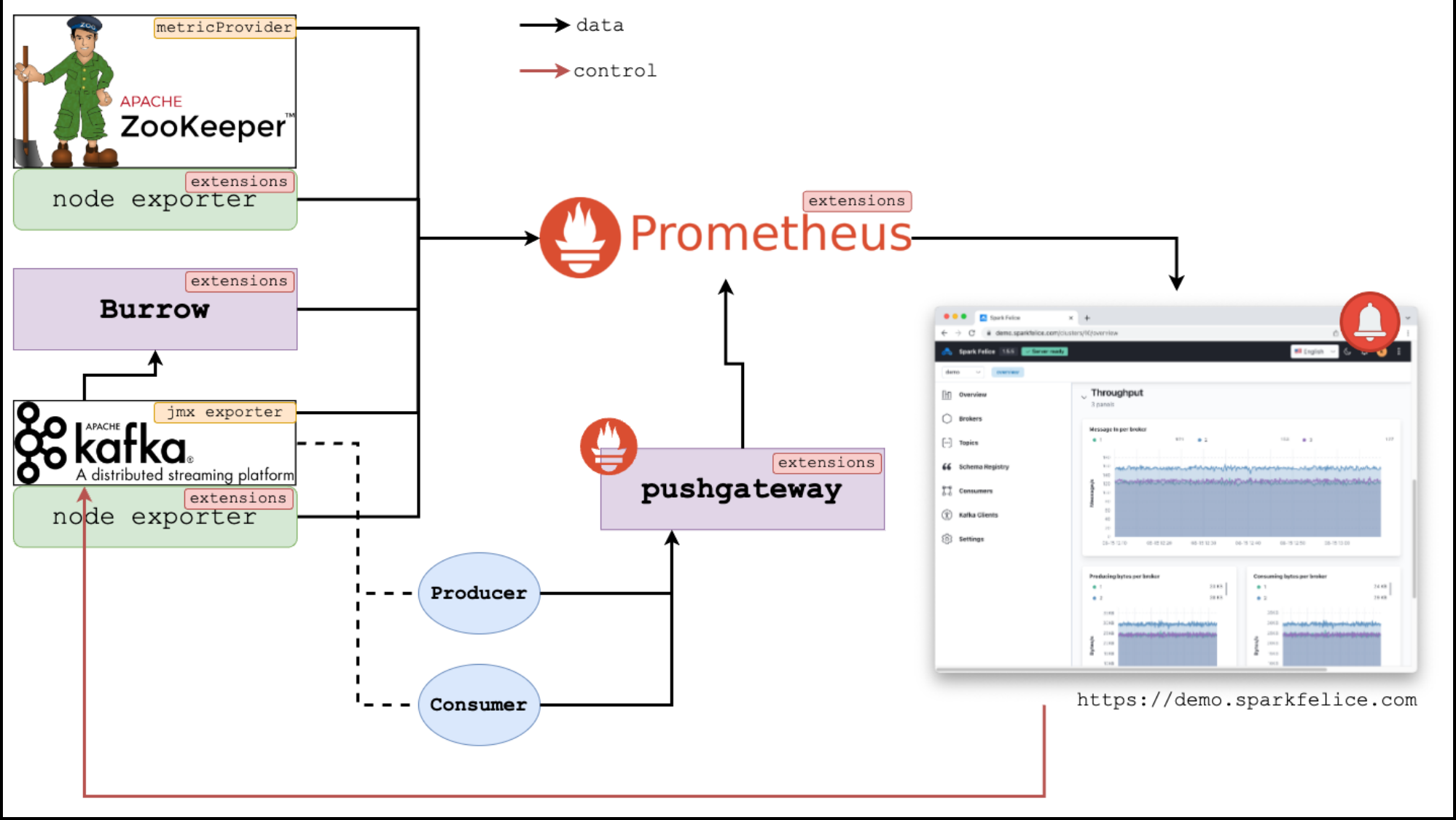
```
void executeBatchJob() throws Exception {
  CollectorRegistry registry = new CollectorRegistry();
  Gauge duration = Gauge.build()
      .name("my_batch_job_duration_seconds").help("Duration of my batch job in seconds.").register(registry);
  Gauge.Timer durationTimer = duration.startTimer();
  try {
    // Your code here.

    // This is only added to the registry after success,
    // so that a previous success in the Pushgateway isn't overwritten on failure.
    Gauge lastSuccess = Gauge.build()
        .name("my_batch_job_last_success").help("Last time my batch job succeeded, in unixtime.").register(registry);
    lastSuccess.setToCurrentTime();
  } finally {
    durationTimer.setDuration();
    PushGateway pg = new PushGateway("127.0.0.1:9091");
    pg.pushAdd(registry, "my_batch_job");
  }
}
```

3. prometheus에서 scrap



One more thing...

Our implementations



What's different?

Our custom modifications

- Burrow
  - Notifier, ZooKeeper 모듈 비활성화
  - 설정 파일 변경 시 자동 재시작
  - SASL/PLAIN, GSSAPI, OAUTHBEARER 인증 메커니즘
- Node Exporter
  - 네트워크 상세 연결 정보 추가
- Prometheus
  - 설정 파일 변경 시 자동 재시작
- Pushgateway
  - 메트릭 TTL 적용

Kafka's Fiance, Felice

Our Monitoring and Managing Tool for AK

- 카프카 클러스터 모니터링
  - 주키퍼, 브로커, 서버, 메시지 입출력, 컨슈머, etc.
- 카프카 클러스터 관리
  - 토픽, 파티션, 카프카 계정, ACL, 스키마, etc.
- Alerting
  - 알림 룰, 알림 타깃, 알림 조회, etc.
- Enterprise Features
  - Audit Log, RBAC, etc.
- Community Edition Coming Soon~

Questions?

Hope this helps