Kafka 운영

1 브로커 및 파티션 추가하기

Partition 추가

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
bin/kafka-topics.sh --describe --topic <topic-name> --bootstrap-server localhost:9092
bin/kafka-topics.sh --alter --topic <topic-name> --partitions 4 --bootstrap-server
localhost:9092
```

운영중인 Kafka Topic 이라면 매우 신중하게 결정해야함.

- 서비스 운영중인 Topic에 Partition 추가는 새로운 partition 으로 메세지 rebalance 가 되는 과정에서 시스템 성능에 영향을 끼칠 수 있음. 꼭 필요하다면, 서비스 임팩트가 상대적으로 작은 시간을 선택해야 함.
- 실제 해당 Topic 의 사용 사례를 고려해서, 필요시 테스트 서버 에서 테스트를 해보고 실행해야함.
- 모든 메세지를 RoundRobin 방식으로 처리하고 있다면, 데이터 규모에 따른 지연시간 이후 곧 정상처리가 시작될 수 있지만, 특정 Key 에 기반한 커스텀 Patitioner 에 기반한 Consumer 를 운영중이라면 메세지의 유실 가능성도 있으므로, 차라리 신규 Topic 을 생성하여 Migration 전략을 짜는 것이 더 나은 선택인 경우가 많음.
- 따라서, topic 의 최초 생성시, 데이터 확장 규모를 고려해서 partition 개수를 여유있게 설정.

Broker 추가

신규 Broker 의 server3.properties 파일 수정

```
broker.id=3
listeners=PLAINTEXT://localhost:9095
log.dirs=/tmp/kafka-logs3
```

신규 Broker 실행

```
bin/kafka-server-start.sh config/server3.properties &
```

Partition 재배치를 할 Topic 에 대한 json 파일 생성 ex) reassign-topic.json 파일 작성

```
vi reassign-topic.json
----------------------------------------------------------------------
{"topics":[{"topic": "topic5"}],"version":1}
```

위에서 생성한 reassign-topic.json 파일을 이용하여 최종 Target 구성 json 구조 확인

```
bin/kafka-reassign-partitions.sh --generate --topics-to-move-json-file
reassign-topics.json --broker-list "1,2,3" --bootstrap-server localhost:9092
```

추가된 Broker 를 고려해서 균등한 재배치 제안 json 데이터가 생성됨. Proposed~~ 파일을

```
Current partition replica assignment
{"version":1,"partitions":[{"topic":"topic5","partition":0,"replicas":[0],"log_
dirs":["any"]},{"topic":"topic5","partition":1,"replicas":[0],"log_dirs":["any"
]},{"topic":"topic5","partition":2,"replicas":[0],"log_dirs":["any"]},{"topic":
"topic5","partition":3,"replicas":[3],"log_dirs":["any"]}]}

Proposed partition reassignment configuration
{"version":1,"partitions":[{"topic":"topic5","partition":0,"replicas":[2],"log_
dirs":["any"]},{"topic":"topic5","partition":1,"replicas":[3],"log_dirs":["any"
]},{"topic":"topic5","partition":2,"replicas":[1],"log_dirs":["any"]},{"topic":
"topic5","partition":3,"replicas":[2],"log_dirs":["any"]}]}
```

Partition 재배치 실행

```
bin/kafka-reassign-partitions.sh --execute --reassignment-json-file
new_partition.json --bootstrap-server localhost:9092
```

Topic 의 Partition 재배치 상태 확인

```
bin/kafka-topics.sh --describe --topic topic5 --bootstrap-server localhost:9092

Topic: topic5 TopicId: V3BjGO7YS8KIe5xuSyGIdg PartitionCount: 4
ReplicationFactor: 1  Configs: segment.bytes=1073741824
  Topic: topic5 Partition: 0  Leader: 2 Replicas: 2 Isr: 2
  Topic: topic5 Partition: 1  Leader: 3 Replicas: 3 Isr: 3
  Topic: topic5 Partition: 2  Leader: 1 Replicas: 1 Isr: 1
  Topic: topic5 Partition: 3  Leader: 2 Replicas: 2 Isr: 2
```

운영중인 Kafka Cluster 라면

- 처리중인 데이터 규모에 따라 Partition 재 배치에 따른 네트워크 사용량과 CPU 사용량 증가에 따른 임팩트가 있을 수 있음.
- 따라서, 상대적으로 사용량이 작은 시간을 이용하는 것이 바람직.
- 상황에 따라 임시로 retention 을 작게 설정하거나, topic 을 나눠서 실행해서 부하를 감소시키는 방안을 고려할 수 있음.

Kafka 운영

2 인증 추가하기

Kafka SASL(Simple Authentication and Security Layer) 인증 종류

- SASL/PLAIN: 간단하게 사용자 이름과 암호를 사용하여 인증
- SASL/SCRAM: SCRAM(Salted Challenge Response Authentication Mechanism) 메커니즘을 사용하는 SASL - PLAIN 보다 개선된 보안을 제공
- SASL/GSSAPI : 커버로스 인증서버를 이용하여 인증
- SASL/OAUTHBEARER: OAUTH BEARER 메커니즘을 사용하는 JWT(JSON 웹 토큰)를 사용하여 인증 - Non-production 용

SASL/SCRAM (Salted Challenge Response Authentication Mechanism)

1. 주키퍼를 실행시킨 후, 주키퍼에 Broker 간 통신에 사용할 Credential(인증정보) 생성

```
bin/kafka-configs.sh --zookeeper localhost:2181 --alter --add-config
'SCRAM-SHA-256=[iterations=8192,password=admin-password]' --entity-type users --entity-name admin
```

2. 주키퍼에 Producer/Consumer 에서 사용할 Credential(인증정보) 생성

```
bin/kafka-configs.sh --zookeeper localhost:2181 --alter --add-config
'SCRAM-SHA-256=[iterations=8192,password=password]' --entity-type users --entity-name username
```

3. JAAS(Java Authentication and Authorization Service) config(kafka_server_jaas.conf) 에 Broker 용 인증정보 설정

```
KafkaServer {
    org.apache.kafka.common.security.scram.ScramLoginModule required
    username="admin"
    password="admin-password";
};
```

4. Kafka Broker config(server.properties) 에 인증정보 설정

```
listeners=SASL_PLAINTEXT://localhost:9092
security.inter.broker.protocol=SASL_PLAINTEXT
sasl.mechanism.inter.broker.protocol=SCRAM-SHA-256
sasl.enabled.mechanisms=SCRAM-SHA-256
```

SASL/SCRAM (Salted Challenge Response Authentication Mechanism)

5. Kafka Broker 실행시 JAAS config 를 사용하도록 kafka_server_jaas.conf 파일 경로를 KAFKA_OPTS 옵션에 추가한 후 Kafka Broker 를 실행

```
export KAFKA_OPTS="-Djava.security.auth.login.config=/Users/ocg/Downloads/kafka_2.13-2.8.2/config/kafka_server_jaas.conf"
```

6-1. Java Producer 의 Properties 에 SASL/SCRAM 인증정보를 추가하여 실행하여 확인

```
...
Properties configs = new Properties();
...
configs.put("security.protocol", "SASL_PLAINTEXT");
configs.put("sasl.mechanism", "SCRAM-SHA-256");
configs.put("sasl.jaas.config", "org.apache.kafka.common.security.scram.ScramLoginModule required
username='alice' password='alice-password';");

KafkaProducer<String, String> producer = new KafkaProducer<>(configs);
...
```

SASL/SCRAM (Salted Challenge Response Authentication Mechanism)

6-2. 또는, Producer 쪽에 SASL/SCRAM 인증정보를 별도의 파일로 만들어 놓고 실행할 수도 있음(producer.properties)

```
security.protocol=SASL_PLAINTEXT
sasl.mechanism=SCRAM-SHA-256
sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required username='alice'
password='alice-password';
```

6-2. 현재 Kafka Broker 에서는 인증을 요구하고 있으므로 아래와 같이 CLI 호출시 인증정보를 포함하여 호출한다.

```
bin/kafka-console-producer.sh --topic topic5 --bootstrap-server localhost:9092 --producer.config ./producer.properties
```

Kafka 운영

3 클러스터 마이그레이션

Cluster Migration

1. 서비스 임팩트를 최소화 하고 데이터 유실에 대한 문제가 발생하지 않도록 사전에 Publisher/Consumer 측과 인터뷰를 통해 세부적인 마이그레이션 작업 계획 및 롤백 계획 작성
2. 새로운 Zookeeper, Kafka Cluster 생성 및 방화벽 작업 등 네트워크, 필요한 보안프로세스 실행
3. 데이터 마이그레이션 실행 - 1번의 계획에 따라, 아래의 방법중 선택하거나 여러방법을 조합할수도 있음.
   a. 신규 Cluster 용 Broker 를 기존 Cluster 추가하고, partition 을 reassign 한 후, 기존 Cluster 를 Shutdown 하는 방법
   b. MirrorMaker2 를 세팅하여 기존 Cluster 에서 새로운 Cluster 로 실시간으로 데이터를 동기화 하는 방법
   c. Kafka Connect 등을 이용해서 기존 Cluster 에서 새로운 Cluster 로 재전송 하는 방법
   d. Application 레벨에서 두개의 Kafka Cluster 에 Dual Wriwte/Dual Read 등으로 처리하는 방법
4. Producer/Consumer Application 의 Endpoint 변경
5. 최종 검증 및 마이그레이션 종료, 면밀히 모니터링 하여 롤백에 대해 대비

MirrorMaker2 실행 설정

```
vi config/connect-mirror-maker.properties
-----------------------------------------------------------------------------
# specify any number of cluster aliases
clusters = A, B

# connection information for each cluster
# This is a comma separated host:port pairs for each cluster
# for e.g. "A_host1:9092, A_host2:9092, A_host3:9092"
A.bootstrap.servers = A_host1:9092, A_host2:9092, A_host3:9092
B.bootstrap.servers = B_host1:9092, B_host2:9092, B_host3:9092

# enable and configure individual replication flows
A->B.enabled = true

# regex which defines which topics gets replicated. For eg "foo-.*"
A->B.topics = .*

B->A.enabled = true
B->A.topics = .*
…
```

MirrorMaker2 실행

```
bin/connect-mirror-maker.sh config/connect-mirror-maker.properties
```

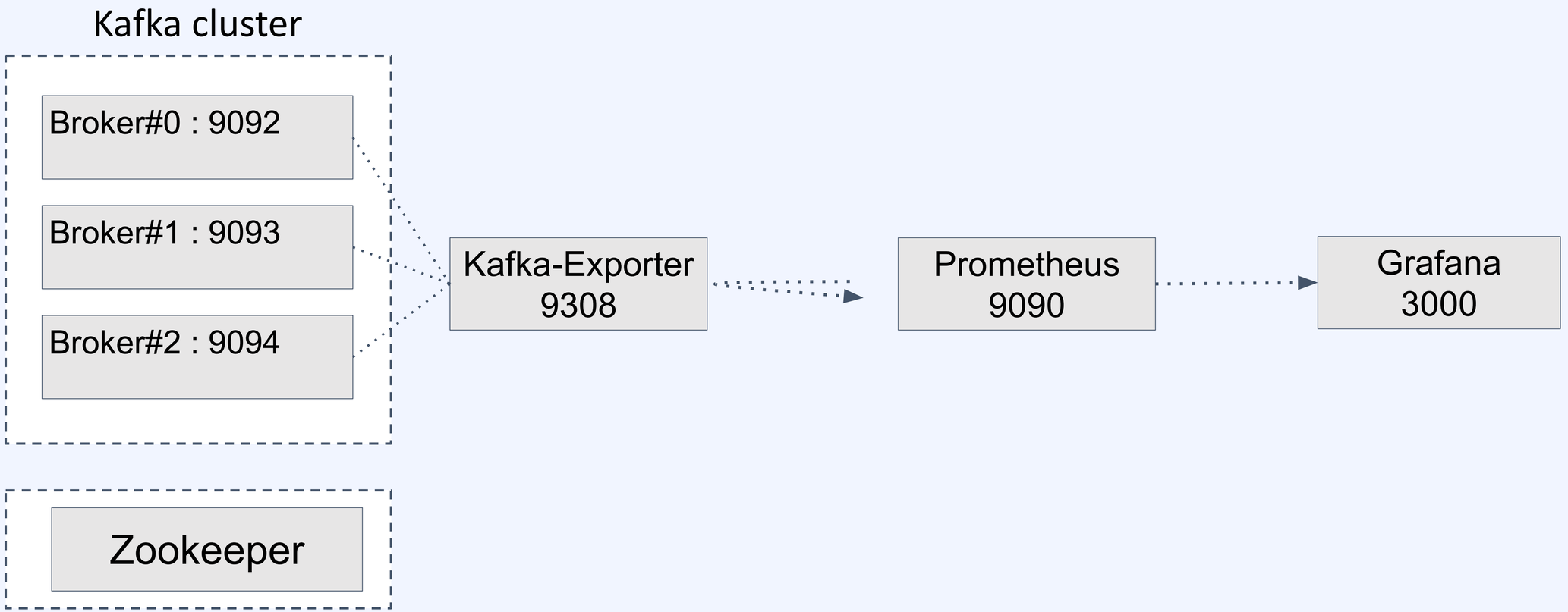
Kafka 운영

4 모니터링

Kafka Monitoring Tool

1. CMAK(Kafka Manager) - by yahoo. Managing Cluster, Topic, Offset
   a. https://github.com/yahoo/CMAK
2. Burrow - by linkedin. focusing lag of offset
   a. https://github.com/linkedin/Burrow
3. Xinfra Monitor(Kafka Monitor) - by linkedin
   a. https://github.com/linkedin/kafka-monitor
4. Cruise Control - by linkedin
   a. https://github.com/linkedin/cruise-control
5. Exporter + Prometheus + Grafana
   a. https://github.com/prometheus/jmx_exporter
   b. https://github.com/danielqsj/kafka_exporter
   c. https://github.com/prometheus/node_exporter
   d. https://prometheus.io/
   e. https://grafana.com/

Kafka exporter + Prometheus + Grafana



Kafka exporter + Prometheus + Grafana

1. Kafka Exporter 를 다운로드 받아 압축을 풀고 실행한다. https://github.com/danielqsj/kafka_exporter/releases

```
# MacBook 에서 kafka exporter 실행시 “개발자를 확인할 수 없기 때문에….” 메세지가 나오는 경우
# xattr ./kafka_exporter
# xattr -d com.apple.quarantine ./kafka_exporter
./kafka_exporter --kafka.server=localhost:9092 --kafka.server=localhost:9093 --kafka.server=localhost:9094
```

2. Prometheus 를 다운로드 받아 압축을 풀고 실행한다. https://prometheus.io/download/

```
# 파일 끝에 추가
vi ./prometheus.yml
  - job_name: "kafka-exporter"
    static_configs:
      - targets:
        - localhost:9308
```

```
# MacBook 에서 prometheus 실행시 “개발자를 확인할 수 없기 때문에…” 메세지가 나오는 경우
# xattr ./prometheus
# xattr -d com.apple.quarantine ./prometheus
./prometheus
```

Kafka exporter + Prometheus + Grafana

3. Grafana 를 다운로드 받아 압축을 풀고 실행한다. https://grafana.com/grafana/download?pg=get&plcmt=selfmanaged-box1-cta1

```
bin/grafana-server web
```

4. Configuration → Add data source 에서 Prometheus 선택하고, URL 에 http://localhost:9090 입력후 Save

5. Import Dashboard 에서 grafana.com 에 이미 등록되어 있는 kafka exporter Overview Dashboard (id=7589) 를 import