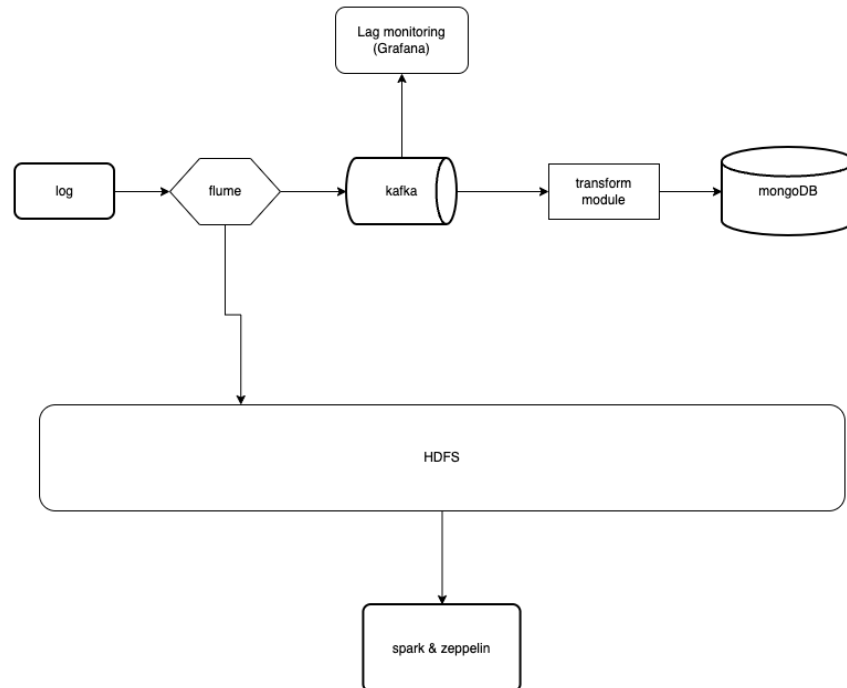


Data Pipeline Project(확장)



사용환경

HDP 2.6.5 virtual box(Red Hat 4.8.5-28)

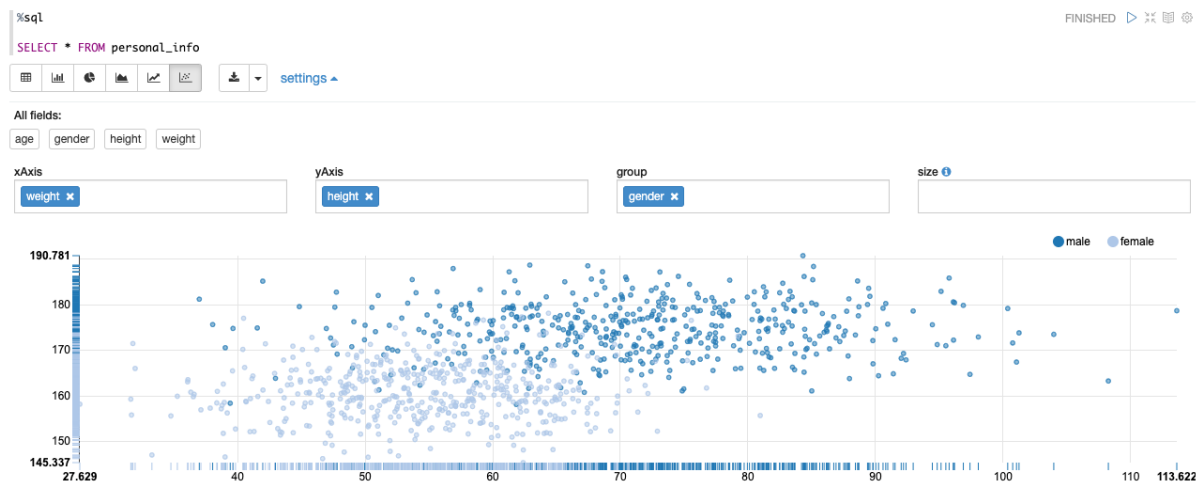
프로젝트 개요

- data pipeline을 이해해보자.
- kafka, flume, mongoDB, hdfs, spark, zeppelin를 이론으로 이해하고 실습해보자.
- 대시보드를 만들어보자.
- kafka monitoring을 해보자.
- 카카오 파이프라인 경험을 기반으로 하였습니다.
- 라인 기술블로그의 라인 쇼핑플랫폼 구축하기를 참고하였습니다.

프로젝트 가정 및 설명

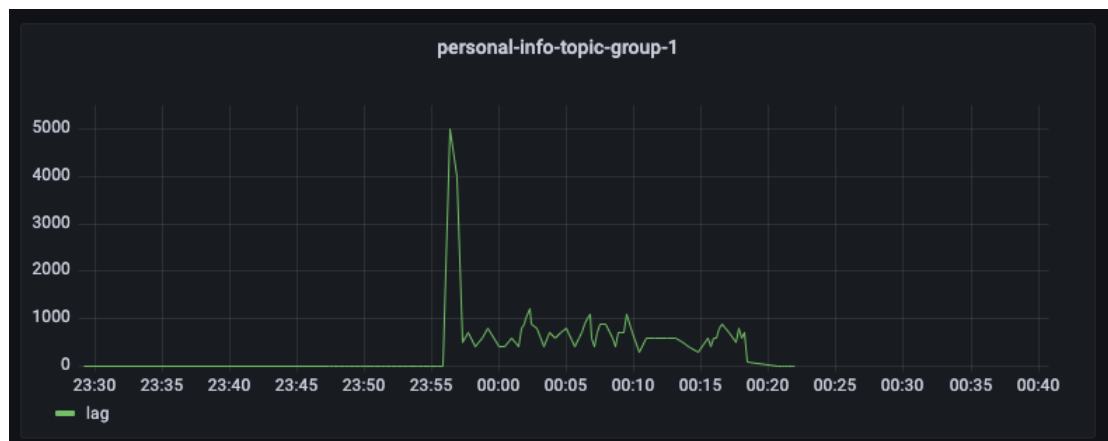
- 나이, 성별, 키, 체중에 대한 데이터가 로그 형태로 수집됩니다.
- 로그는 플럼이 수집하여 카프카와 HDFS로 보내주게 됩니다.
- kafka
 - kafka로 보내진 데이터는 transform하여 mongoDB에 적재합니다.
 - kafka의 데이터는 lag을 수집하여 transform이 제대로 되고 있는지 모니터링 합니다.
 - mongoDB는 서비스에 사용됩니다.(가정)
- HDFS
 - hdfs의 데이터를 zeppelin환경에서 spark를 이용하여 dash board를 만듭니다.

dash board



키와 몸무게에 상관관계 없이 log를 발생시켰다. 그래서 키와 몸무게가 관계없이 산개해있다.

kafka lag



프로젝트 상세

코드를 vim으로 작성하는데 formatter를 설정하지 않아서 코드가 지저분합니다.

Flume 명령어

```
# flume run
/usr/hdp/current/flume-server/bin/flume-ng agent --conf conf --conf-file /home/maria_dev/toy_project/myflumelogs.conf --name
a1 -Dflume.root.logger=INFO

# flume log path
/var/log/flume/flume.log
```

kafka 명령어

```
# kafka make topic
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --create --zookeeper sandbox-hdp.hortonworks.com:2181 --replication-factor
1 --partitions 1 --topic personal-info-topic

# kafka topic list
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --list --zookeeper sandbox-hdp.hortonworks.com:2181

# kafka consumer console
/usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic personal-info-topic --from-beg
inning

# check kafka lag
/usr/hdp/current/kafka-broker/bin/kafka-consumer-groups.sh --bootstrap-server sandbox-hdp.hortonworks.com:6667 --group group-
1 --describe
```

Flume conf

```
# Name the components on this agent
a1.sources = r1
a1.sinks = k1 k2
a1.channels = c1 c2

# Describe/configure the source
a1.sources.r1.type = spooldir
a1.sources.r1.spoolDir = /home/maria_dev/toy_project/flume_log
a1.sources.r1.deletePolicy = immediate
a1.sources.r1.fileHeader = true
a1.sources.r1.interceptors = timestampInterceptor
a1.sources.r1.interceptors.timestampInterceptor.type = timestamp

# Describe the sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = personal-info-topic
a1.sinks.k1.kafka.bootstrap.servers = sandbox-hdp.hortonworks.com:6667
a1.sinks.k1.kafka.flumeBatchSize = 20
a1.sinks.k1.kafka.producer.acks = 1
a1.sinks.k1.kafka.producer.linger.ms = 1

# Describe the sink2
a1.sinks.k2.type = hdfs
a1.sinks.k2.hdfs.path = /user/maria_dev/flume/%y-%m-%d/%H%M/%S
a1.sinks.k2.hdfs.filePrefix = events-
a1.sinks.k2.hdfs.round = true
a1.sinks.k2.hdfs.roundValue = 10
a1.sinks.k2.hdfs.roundUnit = minute
a1.sinks.k2.hdfs.fileType = DataStream

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 100

# Use a channel which buffers events in memory
a1.channels.c2.type = memory
a1.channels.c2.capacity = 10000
a1.channels.c2.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1 c2
a1.sinks.k1.channel = c1
a1.sinks.k2.channel = c2
a1.sources.r1.selector.type = replicating
```

log producer

```
# log producer
```

```

import os
import argparse
import random
import shutil
from time import sleep

class LogProducer:
    def __init__(self, path):
        self.path = path
        pass

    def _make_file_name(self):
        file_list = os.listdir(self.path)
        file_list = set(file_list)

        file_num = 0
        file_format = "person_info_%d.log"

        while True:
            if file_format % file_num in file_list:
                file_num += 1
            else:
                file_name = file_format % file_num
                break

        return file_name

    def _make_log(self):
        file_name = self._make_file_name()

        # num_sample = random.randrange(900, 1100)
        num_sample = 1000

        genders = random.choices(['male', 'female'], k=num_sample)
        ages = random.choices(range(20, 24), k=num_sample)
        heights = []
        weights = []

        for gender in genders:
            if gender=='male':
                heights.append(random.normalvariate(174.2, 5.89))
                weights.append(random.normalvariate(71.5, 11.63))
            else:
                heights.append(random.normalvariate(160.9, 5.89))
                weights.append(random.normalvariate(55.1, 8.58))

        rows = ''
        for i in range(num_sample):
            rows += f'{ages[i]} {genders[i]} {heights[i]} {weights[i]} \n'

        with open(os.path.join(self.path, file_name), 'w') as f:
            f.write(rows)

        shutil.copyfile(os.path.join(self.path, file_name), os.path.join('flume_log', file_name))

    def make_log_ntime(self, n):
        for _ in range(n):
            print("start log making")
            self._make_log()
            print("finish log making")
            sleep(5)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--filepath', default='./log')
    parser.add_argument('-n', '--ntime', default='10')
    args = parser.parse_args()

    log_producer = LogProducer(args.filepath)
    log_producer.make_log_ntime(int(args.ntime))

```

transformer

```
# transformer

import time

from kafka import KafkaConsumer
from pymongo import MongoClient

class Transformer:
    def __init__(self):
        self.consumer = KafkaConsumer('personal-info-topic', bootstrap_servers='sandbox-hdp.hortonworks.com:6667', group_id='group-1', enable_auto_commit=True, auto_offset_reset='earliest', consumer_timeout_ms=1000)

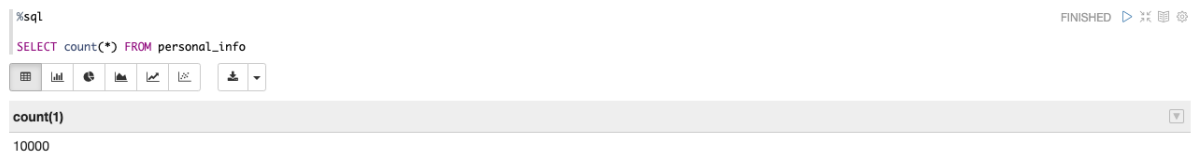
    def transform(self):
        msgs = []
        for message in self.consumer:
            print("Topic: {}, Partition: {}, Offset: {}, Key: {}, Value: {}".format(message.topic, message.partition, message.offset, message.key, message.value.decode('utf-8')))
            msgs.append({'msg' : message.value.decode('utf-8')+' processed'})
        # msgs to hdfs, mg
        if msgs:
            self._mg_put(msgs)

    def _mg_put(self, msgs):
        client = MongoClient()
        posts = client.mydb.posts
        posts.insert_many(msgs)

    def run(self):
        print('start')
        while True:
            self.transform()
            print('sleep')
            time.sleep(10)

if __name__ == '__main__':
    trm = Transformer()
    trm.run()
```

hdfs와 mongo에 들어간 row확인



count(1)
10000

```
> db.posts.count()
10000
```

1000 rows 파일을 10개 넘겨주었고, hdfs와 mongoDB 양쪽에 제대로 들어갔음을 확인했습니다.

Zeppelin code

```

final case class PersonalInfo(age: Int, gender: String, height: Float, weight: Float)
val lines = sc.textFile("hdfs://user/maria_dev/flume/*/*/*/*").map(x => {val fields = x.split(" "); PersonalInfo(fields(0).toInt, fields(1).toString, fields(2).toFloat, fields(3).toFloat)})
warning: there was one unchecked warning; re-run with -unchecked for details
defined class PersonalInfo
lines: org.apache.spark.rdd.RDD[PersonalInfo] = MapPartitionsRDD[79] at map at <console>:45
Took 1 sec. Last updated by anonymous at May 12 2022, 2:18:57 PM.

import sqlContext.implicits._
val personalInfoDF = lines.toDF()
personalInfoDF.printSchema()

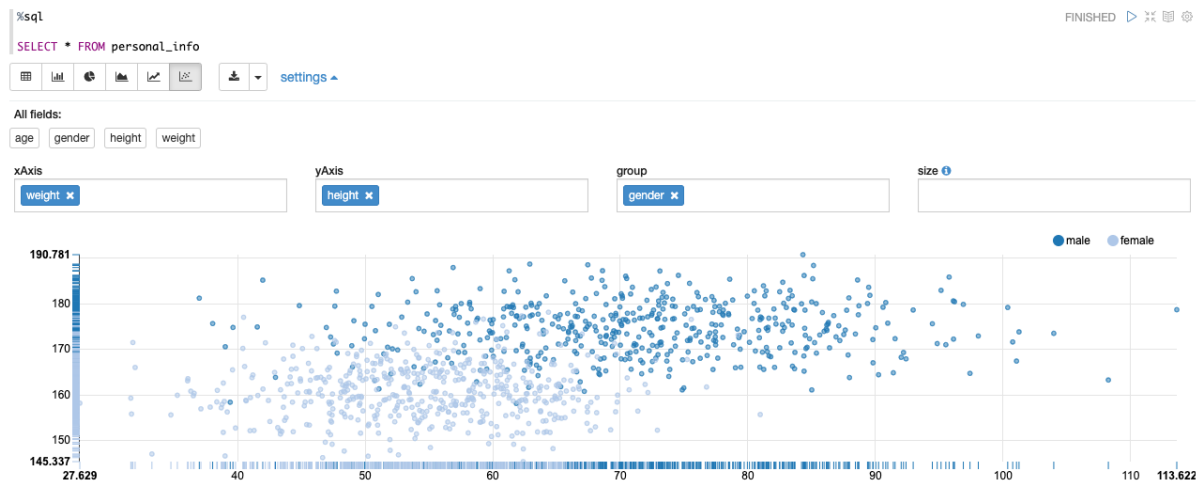
import sqlContext.implicits._
personalInfoDF: org.apache.spark.sql.DataFrame = [age: int, gender: string ... 2 more fields]
root
 |-- age: integer (nullable = false)
 |-- gender: string (nullable = true)
 |-- height: float (nullable = false)
 |-- weight: float (nullable = false)
Took 11 sec. Last updated by anonymous at May 12 2022, 2:19:12 PM.

personalInfoDF.registerTempTable("personal_info")
warning: there was one deprecation warning; re-run with -deprecation for details
Took 10 sec. Last updated by anonymous at May 12 2022, 2:19:12 PM.

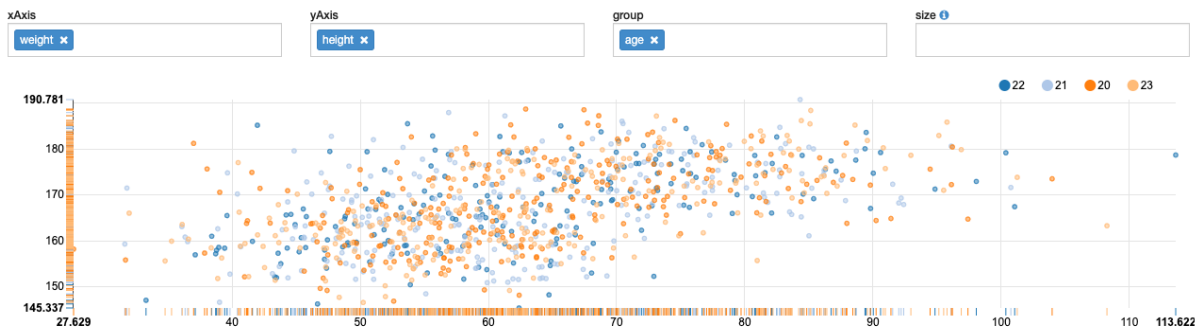
%sql
SELECT * FROM personal_info

```

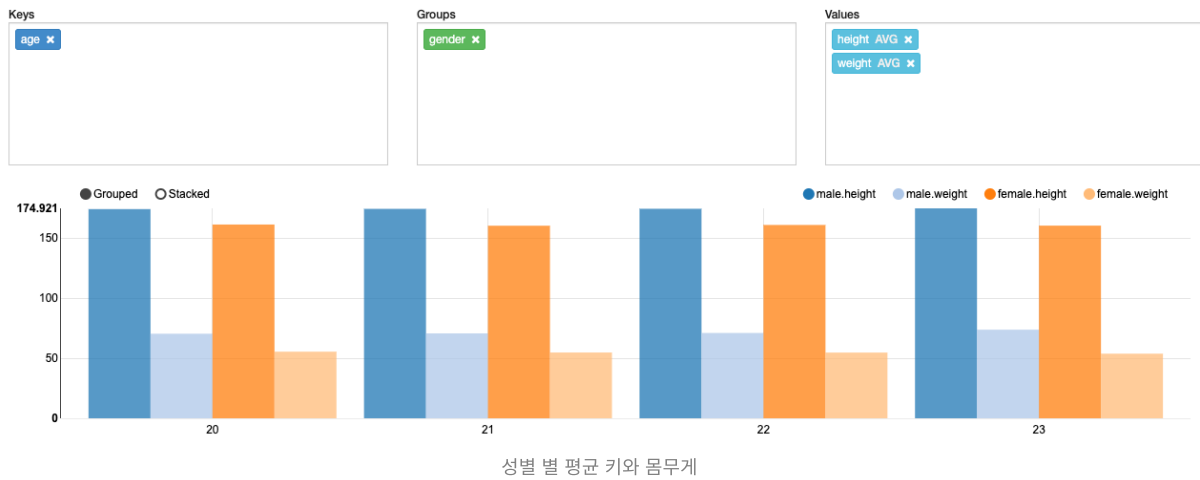
spark와 zeppelin을 통한 대시보드



키와 몸무게에 상관관계 없이 log를 발생시켰다. 그래서 키와 몸무게가 관계없이 산개해있다.



나이 역시 상관관계를 고려하지 않았다. 따라서 상관관계가 드러나지 않는다.



Grafana command

```
# start
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl status grafana-server

# restart
sudo systemctl restart grafana-server
sudo systemctl status grafana-server
```

grafana의 csv data source를 이용하였습니다.

kafka-consumer-groups.sh를 이용하여 원하는 토픽의 lag를 csv형태로 grafana에 공급하였습니다.

```
# kafka lag to csv
import subprocess
import csv
import argparse
from datetime import datetime
import time

def make_lag_log(topic):
    outputs = subprocess.getoutput('/usr/hdp/current/kafka-broker/bin/kafka-consumer-groups.sh --bootstrap-server sandbox-hd
p.hortonworks.com:6667 --group group-1 --describe')
    outputs = outputs.split('\n')
    info_dict = dict()

    for output in outputs[:-1]:
        row = output.split()
        if row[0]=='TOPIC':
            break

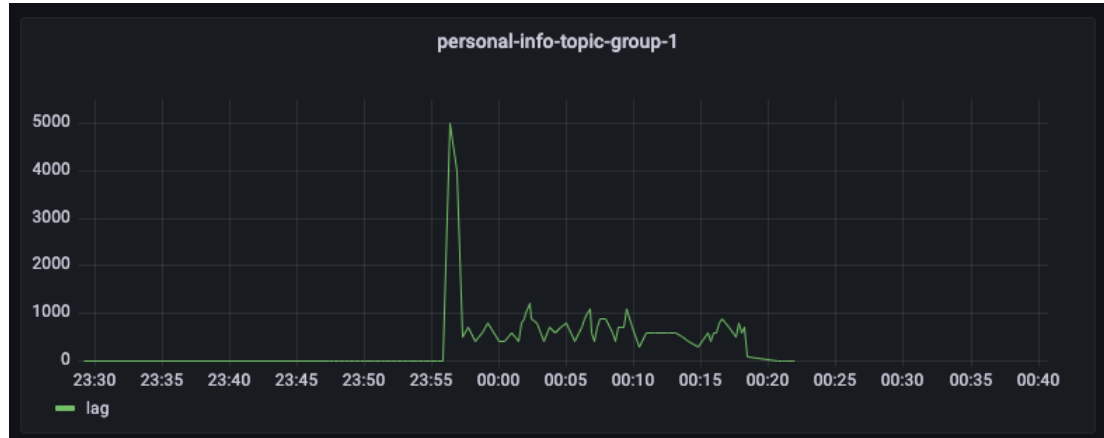
        info_dict[row[0]] = (row[4], datetime.now().strftime('%Y-%m-%d %H:%M:%S'))

    if topic in info_dict:
        with open(f'/grafana_log/{topic}.lag', 'a') as f:
            f.write(f'{info_dict[topic][0]}, {info_dict[topic][1]}\n')
    else:
        print(f'There is no {topic}')

if __name__=='__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--topic', default='personal-info-topic')
    args = parser.parse_args()
    while True:
```

```
make_lag_log(args.topic)
time.sleep(10)
```

kafka lag monitoring



이슈

- 적은 데이터를 넣을 때는 HDFS와 mongoDB 양쪽에 잘들어가는 것을 확인했으나, 많은 데이터를 넣게 되면 문제가 생기는 것을 확인했습니다. 아직 원인을 파악 중입니다.

```
> db.posts.count()
56400
```

%sql

```
SELECT count(*) FROM personal_info
```



count(1)

30101