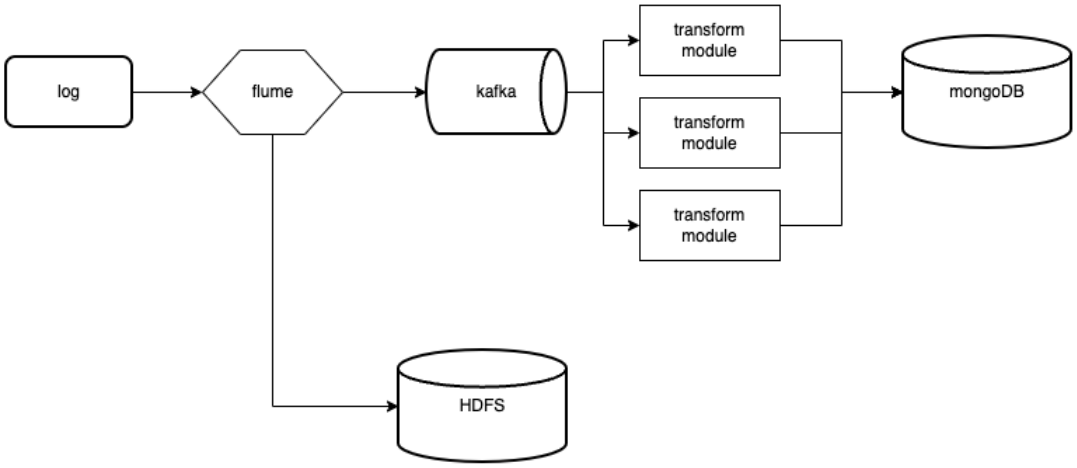
Data pipeline project(Toy project)



사용환경

- HDP 2.6.5 virtual box(Red Hat 4.8.5-28)

프로젝트 개요

- data pipeline을 이해해보자.
- kafka, flume, mongoDB, hdfs를 이론으로 이해하고 실습해보자.
- 라인 기술블로그의 라인 쇼핑플랫폼 구축하기를 참고하였습니다.

프로젝트 가정 및 설명

- web의 로그들을 임의의 폴더에 쌓아둡니다.(가정)
  - 쌓는 방식은 이렇습니다.
  - 특정 시간 단위로 새로운 로그파일을 만듭니다.
- 새로운 로그파일을 만들게 되면, 플럼으로 작성이 완료된 로그파일을 읽어옵니다.
- 플럼의 싱크는 카프카의 토픽에 메시지를 넣어주고, data transform을 담당하는 module은 메시지를 가져와서 원하는 형태로 변형 후 mongoDB에 적재됩니다.
- 플럼으로 수집된 로그는 raw data 형태로 hdfs에 적재됩니다.
- hdfs는 data lake로 활용될 수 있습니다.(가정)
- mongoDB는 서비스에 사용됩니다.(가정)

장점

- 확장성이 좋습니다. 데이터를 더 많이 처리해야한다면, transform을 담당하는 프로세스를 하나 더 띄워주는 것으로 해결할 수 있습니다.
- 이를 통해서 transform이 오래 걸리는 연산이라고 할지라도 부담이 적어집니다.

단점

- 로그를 수집하는 플럼 쪽에서 병목이 있을 수 있습니다.

단점에 대한 해결방안

- 로그를 여러 폴더에 나누어 저장을 하고, 플럼 여러개로 수집할 수 있도록 구축(?)

후기

- kafka는 굉장합니다.

code

```
import time

from kafka import KafkaConsumer
from pymongo import MongoClient


class Transformer:
    def __init__(self):
        self.consumer = KafkaConsumer('mytopic',
                                      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
        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()
            time.sleep(10)


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