Building Python Streaming Data Pipeline

jaehyeuk.oh@hpcnt.com

배경

Hyperconnect 에서는 Spark Streaming 을

Fraud detection
Rule-based Alerting
Realtime User Segmentation and A/B Testing
Matching

등에 활용해 왔습니다만,

Low Latency 구현
최적화
Failover 이후 State 유실
Python 으로 구현된 Model 의 Scala 로의 재구현

등에 어려움이 있었습니다.

배경

Low Latency 등 성능 이슈에 대응 하기 위한 목적으로 Native Streaming Framework 인 Flink 를 도입했고

Python Model 을 효과적으로 Streaming 에 적용하기 위해 Native Python 을 지원하는 방법을

Beam on Flink Runner 에서 찾았습니다.

여전히 Experimental 한 부분이 있지만, 빠르게 쓸만한 모습이 될 것으로 기대됩니다.

Tutorial 에서는 Direct Runner 를 사용하여 진행합니다.

목적

이번 튜토리얼을 통해

Transform, Time, Window, Trigger, State 등

Streaming Data Application 의 주요 개념을 익히고,

Python 환경에서 실습하여 익숙해지는 계기가 되기를 기대합니다.

1. Qwiklab 환경에서 Runner 실행
2. Streaming Data Application 개념
3. Local 환경에서 Lab 실행

Lab 0. 환경 설정

1. DataProc Cluster 생성
2. Master 접속 후 기본 설정
3. Direct / Dataflow / Flink Runner 실행 및 확인

Lab 0. 환경 설정

1. Project ID 확인 : qwiklabs-gcp-…

2. Cloud Storage Bucket 생성 : pycon##-bucket

3. DataProc Cluster 생성

!!! Component gateway, Project access 에 check 확인

--initialization-actions
gs://da-beam-flink/docker/docker.sh
gs://da-beam-flink/flink/flink.sh
gs://da-beam-flink/beam/beam.sh

--metadata
beam-job-service-snapshot = gs://da-beam-flink/beam-runners-flink-1.7-job-server-2.14.0-SNAPSHOT.jar
beam-image-enable-pull = true
beam-image-repository=gcr.io/azar-production-de-main/beam-flink
beam-image-version=v2.14.0
flink-start-yarn-session=true
flink-snapshot-url=https://archive.apache.org/dist/flink/flink-1.7.2/flink-1.7.2-bin-hadoop28-scala_2.11.tgz

Lab 0. 환경 설정

1. Master 접속 후 기본 설정

```
sudo apt install python-pip
sudo pip install apache-beam==2.14.0
sudo pip install six==1.12.0
sudo pip install google-cloud
sudo pip install google-cloud-storage
sudo pip install google-cloud-dataflow
sudo pip install “apache-beam[gcp]"
sudo pip install apache-beam==2.14.0
sudo pip install six==1.12.0

git clone https://github.com/JaehyeukOO/pypipeline-tutorial.git
```

Lab 0. 환경 설정

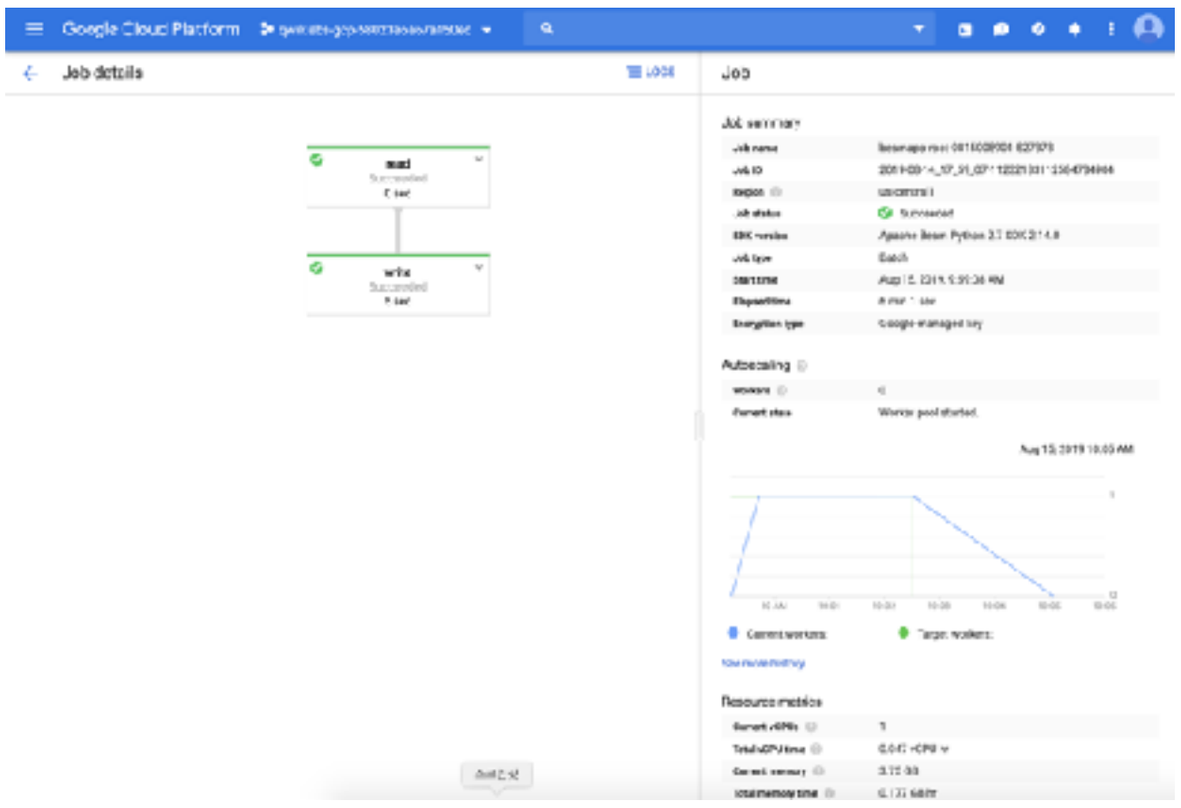
1. Direct / Dataflow / Flink Runner 실행 및 확인

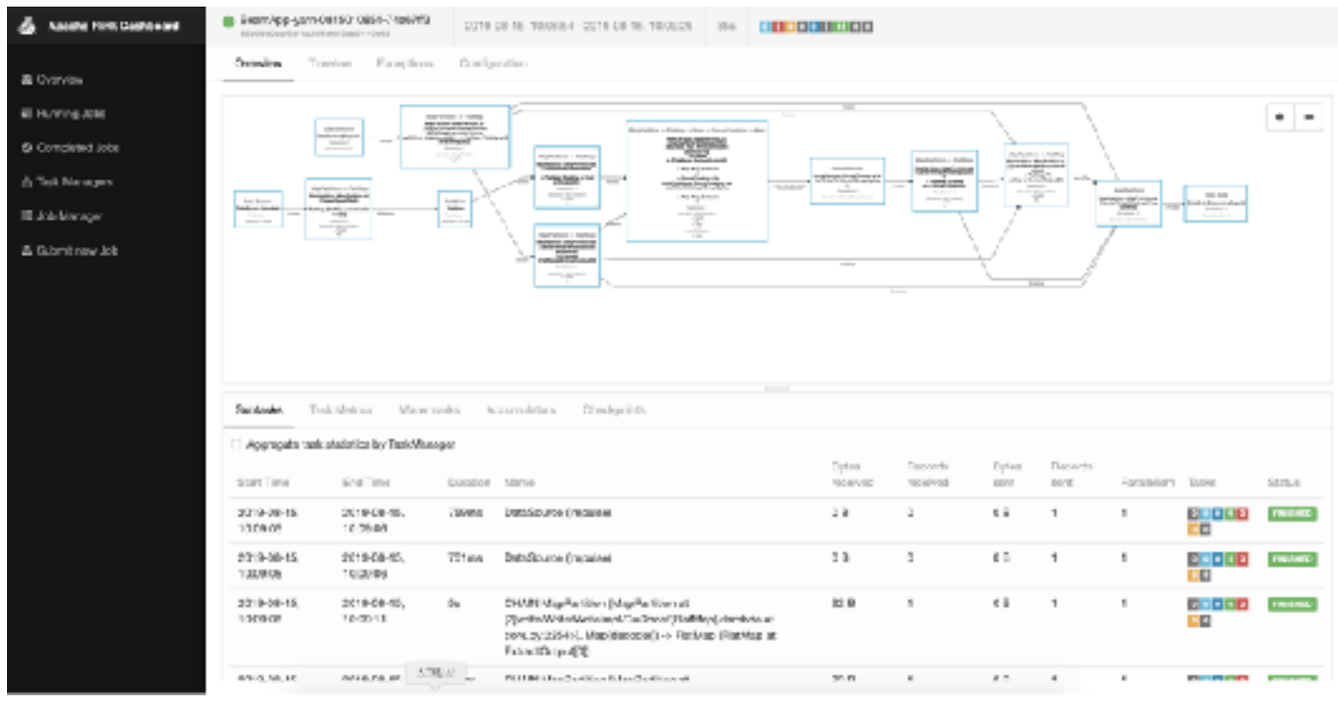
```
gsutil cp gs://da-beam-flink/1_direct_run.py .
gsutil cp gs://da-beam-flink/2_dataflow_run.py .
gsutil cp gs://da-beam-flink/3_flink_run.py .

sudo python 1_direct_run.py
sudo python 2_dataflow_run.py
sudo python 3_flink_run.py
```

https://cloud.google.com/sdk/docs/quickstart-macos?hl=ko

```
 ./google-cloud-sdk/install.sh
. ~/.bash_profile
gcloud init
```

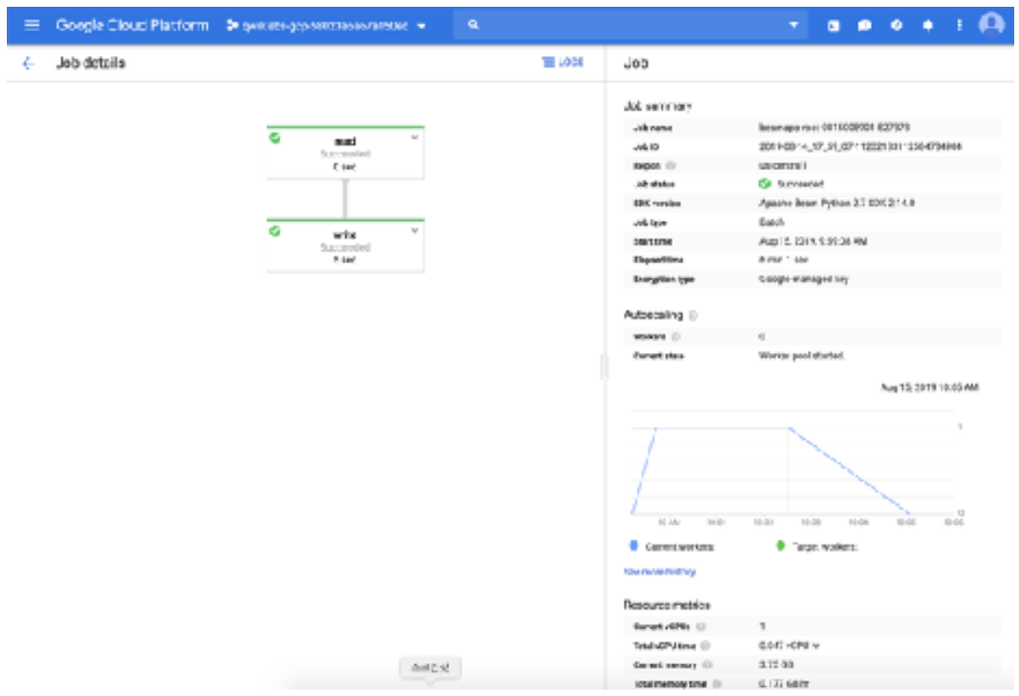


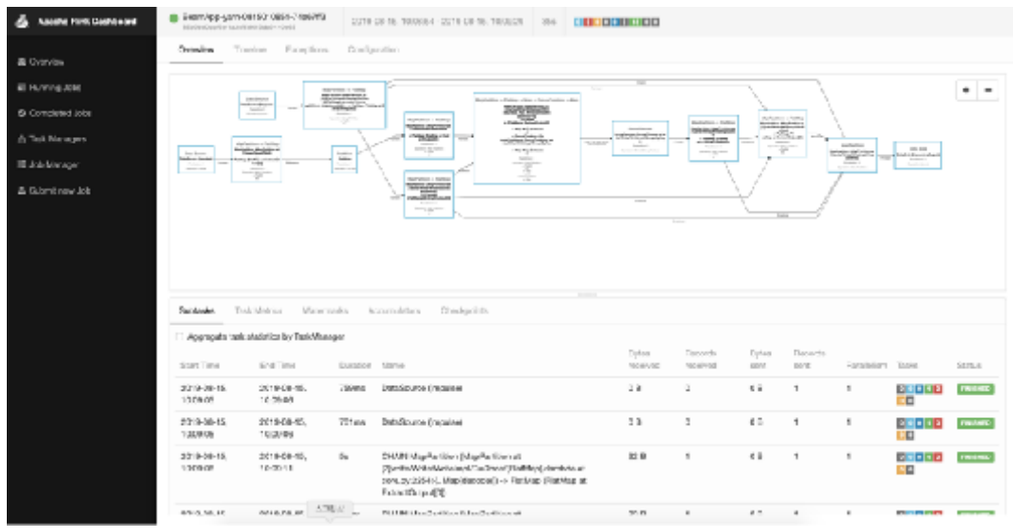


Pipeline

먼저 주요 개념들에 대해 짚고 넘어갑시다.

. Pipeline, Workflow graph, DAG





Pipeline

먼저 주요 개념들에 대해 짚고 넘어갑시다.

. PCollection / PTransform

```
pipeline
 | 'read' >> ReadFromText(“gs://dataflow-samples/shakespeare/kinglear.txt”)
 | 'format' >> beam.Map(format_message)
 | 'write' >> WriteToText("gs://da-beam-flink/out1.txt")
```

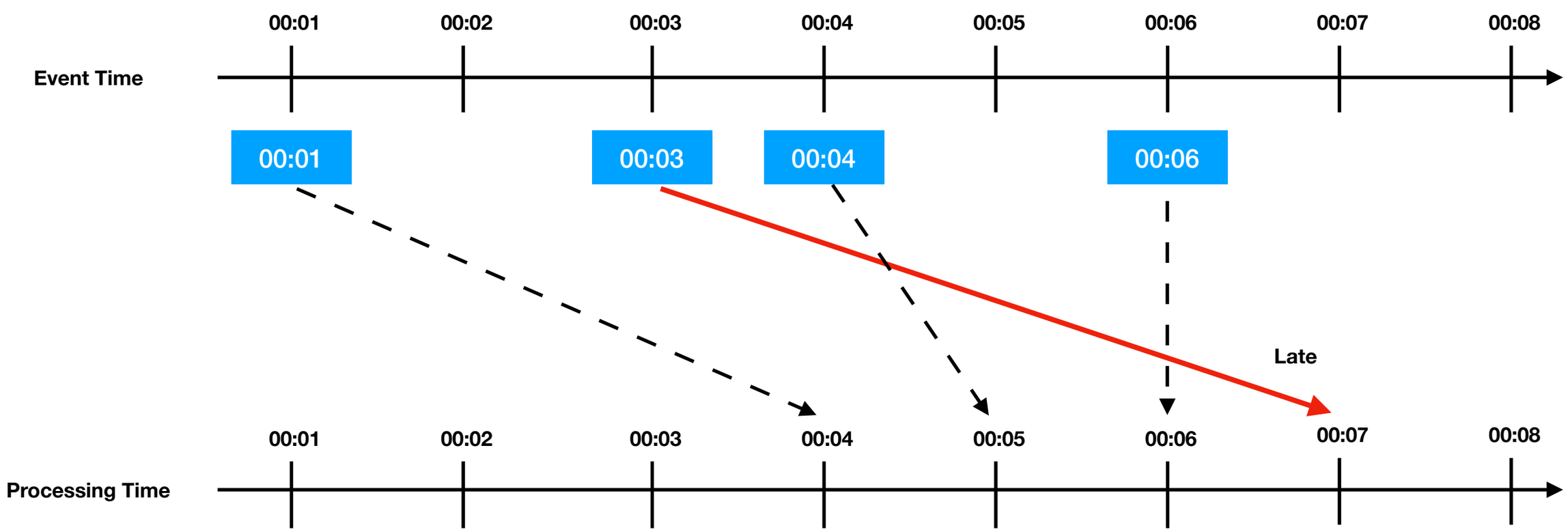
Source

. Bounded (ex. File)
. Unbounded (ex. Kafka, PubSub)

Time

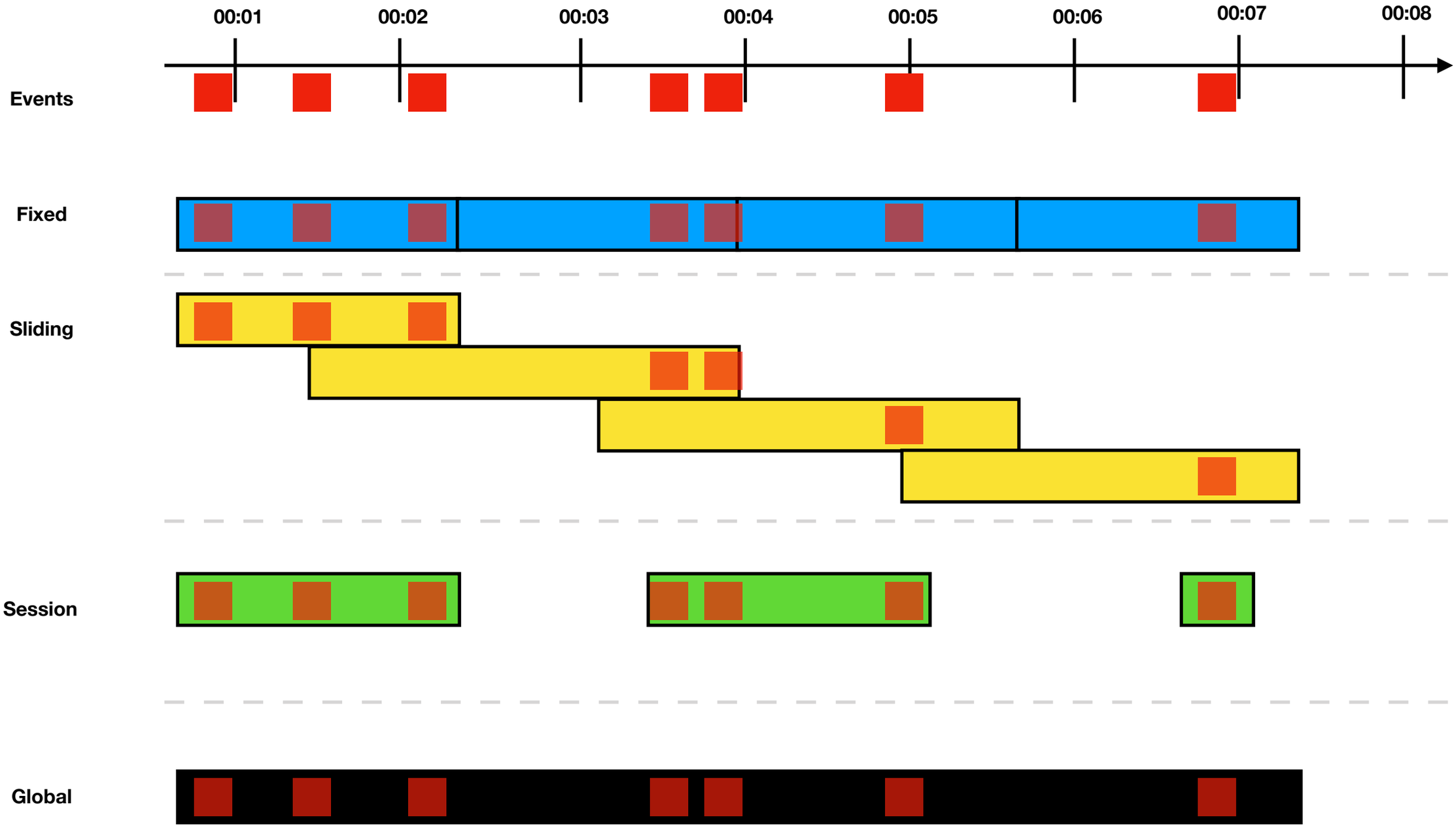
Time

. Event Time
. Processing Time
. Ingestion Time



Window

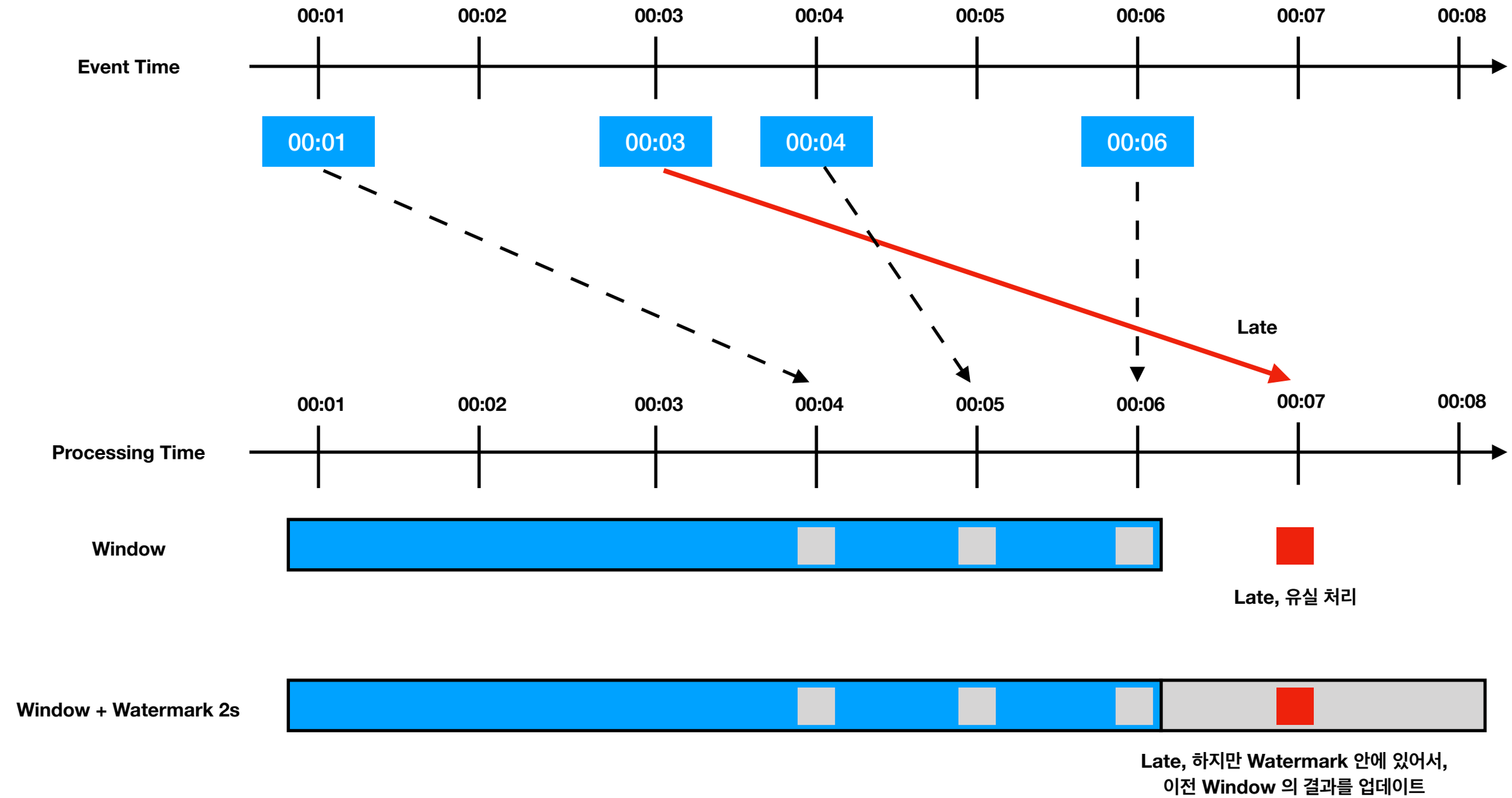
. Tumbling (Fixed) Window
. Sliding Window
. Session Window
. Global Window



Irregular Time

하지만 Event Data 는 항상 순서대로 들어오지 않고, 유실도 있습니다.

Watermark (for drop)
Late elements



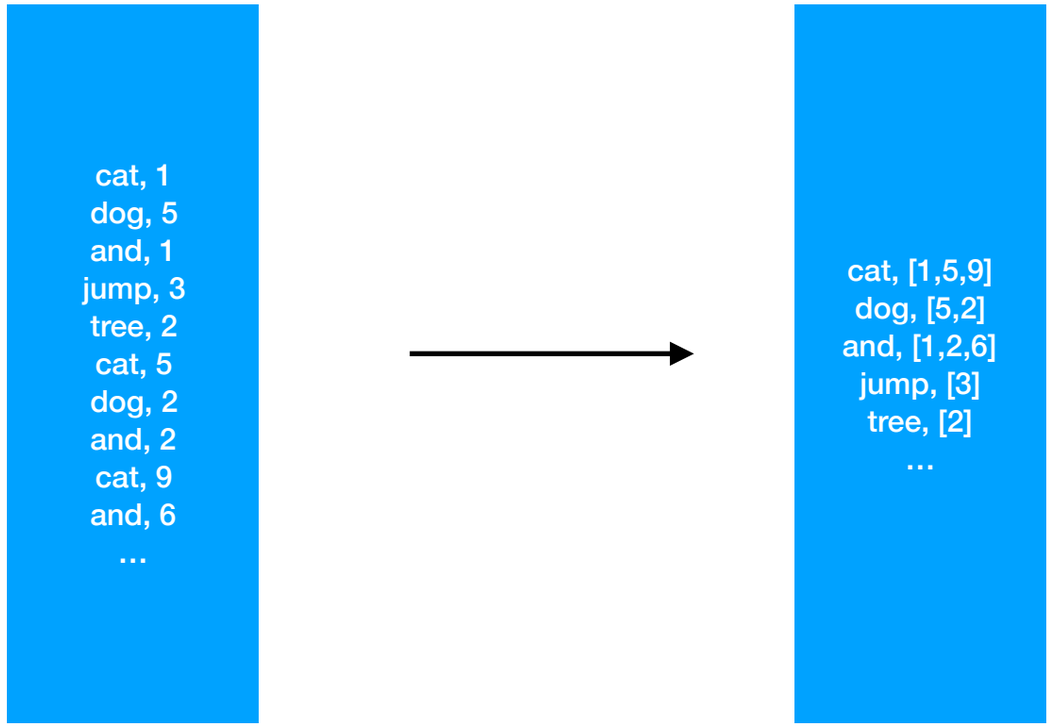
Trigger

. Event time trigger
. Processing time trigger
. Data-driven trigger (횟수)
. Composite trigger

Transform

. ParDo / Map / FlatMap (N to N or M)

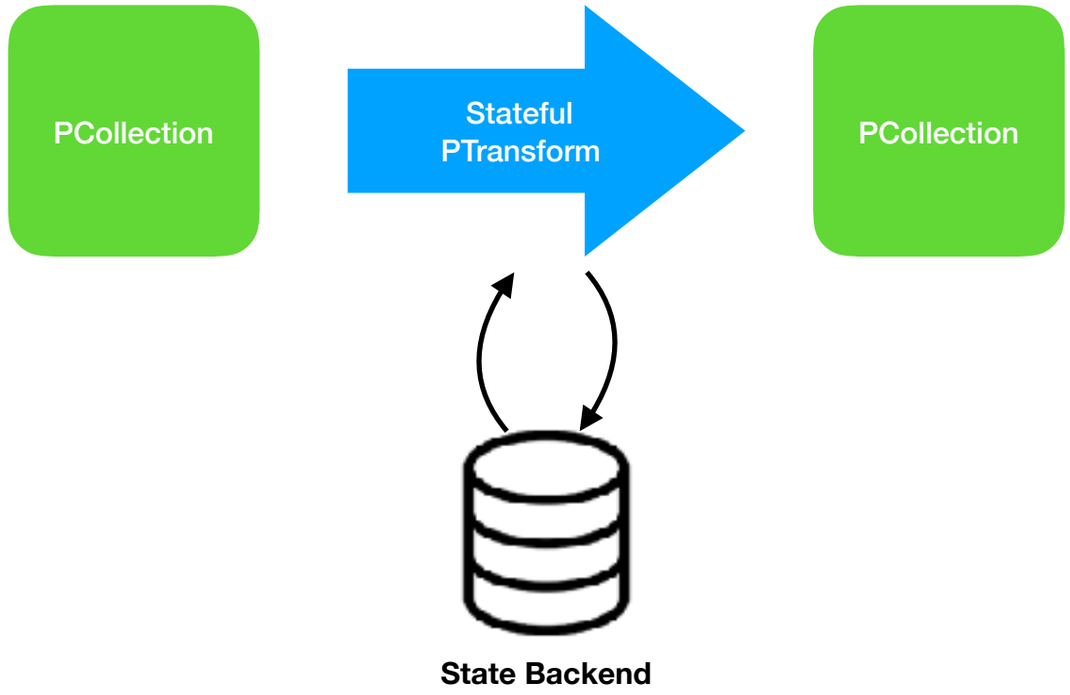
. Combine, GroupByKey (aggregation)



State

Operator State (ex kafka connector offset)

Keyed State



Lab 1. Batch Read And Count

File 내용을 읽고, 중복 단어를 세어봅니다

Goal : FlatMap, Map, GroupByKey, CombineGlobally 의 동작 확인

Lab 2. Python Built-in I/O

File / Log 외에

GCP Pub/Sub, BigQuery 만 지원하고 있습니다.

다음 Lab 을 위해 Pub/Sub Topic 을 만들어봅시다.

1. service_account 생성
2. Ingest to PubSub

Lab 2. Streaming Read And Count

. Timestamp 를 설정해봅시다.

. Windowing / Trigger 조건을 변경하며 결과를 확인합시다.

Fixed / Sliding / AfterCount (ACCUMULATING / DISCARDING)

Lab 3. Tweet Counter

. twitter api 사용을 위한 키를 생성합시다.

. tweepy to pubsub generator 를 셋업합니다.

. processing time trigger 를 활용해서 일정 시간 마다 갯수를 집계합니다.

. key words 를 포함하는 tweet 만 Filtering 해봅시다.

Lab 4. Top Trending Hashtag

. Hashtag 를 분리하는 DoFn 을 작성합니다.

. Hashtag 로 GroupBy 하여 상위 tag 를 주기적으로 출력합니다.

. State 를 활용하여 처음 발견된 tag 들의 변화를 추적합니다.

Lab 5. Sentimental Analysis

. sentimental package 를 활용하여, sentiment value 를 계산합니다.

. 상위 hashtag 의 sentiment 평균을 정기적으로 집계합니다.