AiFrenz

BeeAI 프레임워크로 인공지능 서비스 시작하기

2019-04-17

ETRI KSB융합연구단

이연희, 안후영

목차

- Part 1. BeeAI 프레임워크는 무엇인가요?
- Part 2. BeeAI 웹툴킷 알아보기
- Part 3. BeeAI 를 활용한 인공지능 서비스 사례 살펴보기

Part 2. BeeAI 웹툴킷 알아보기

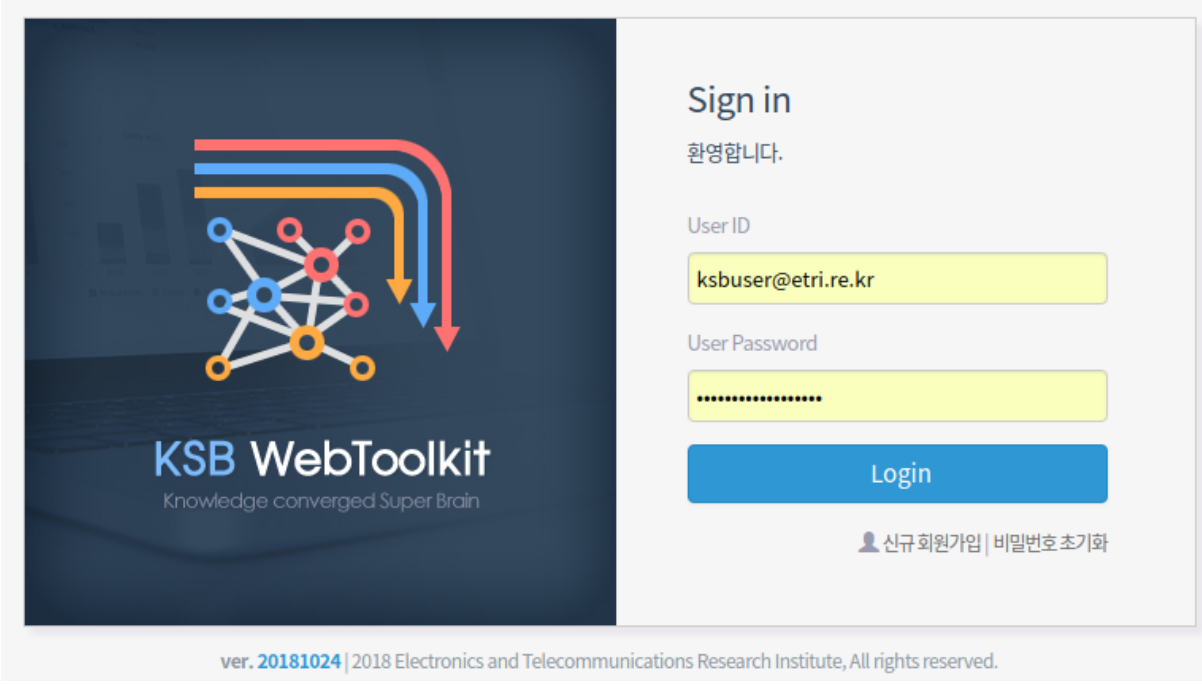
BeeAI 웹툴킷

- 용도
  - BeeAI 를 쉽고 효과적으로 사용하기 위한 웹 기반 저작 도구
- 공식 홈페이지
  - 주소 https://csleoss.etri.re.kr/kor.do
  - BeeAI 소개
  - 다운로드/ 설치 방법 및 매뉴얼 제공
  - 게시판 운영 중

접속 및 로그인

BeeAI

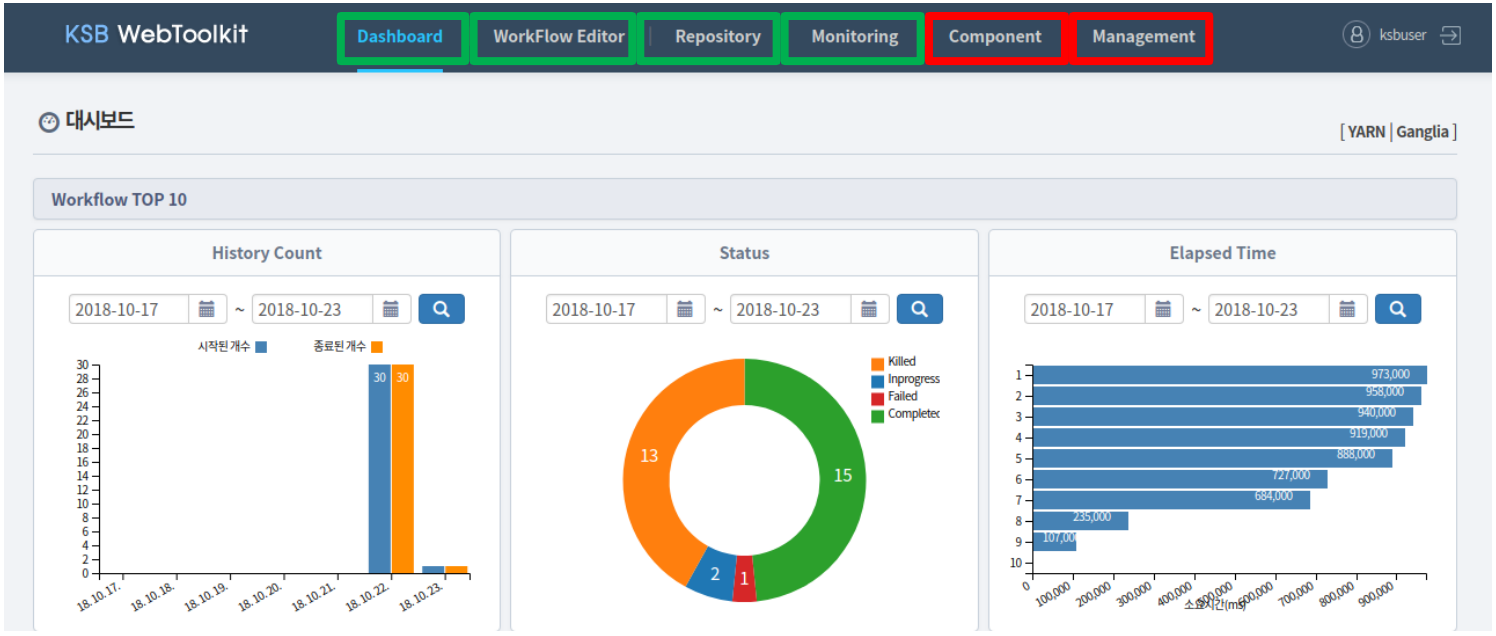
- 크롬 브라우저(또는 HTML5 표준 호환 브라우저) 활용
  - http://localhost:8080 에 접속
  - 기본 제공 되는 사용자 정보 (슈퍼관리자)
    - ID : ksbuser@etri.re.kr, PW : ksbuser@etri.re.kr



주요 기능

BeeAI

- 일반 사용자 기능(▬)
  - Dashboard, Workflow Editor, Monitoring, Repository
- 관리자 기능(▬)
  - Component, Management



쉬운 예제로 웹툴킷 살펴보기 BeeAI

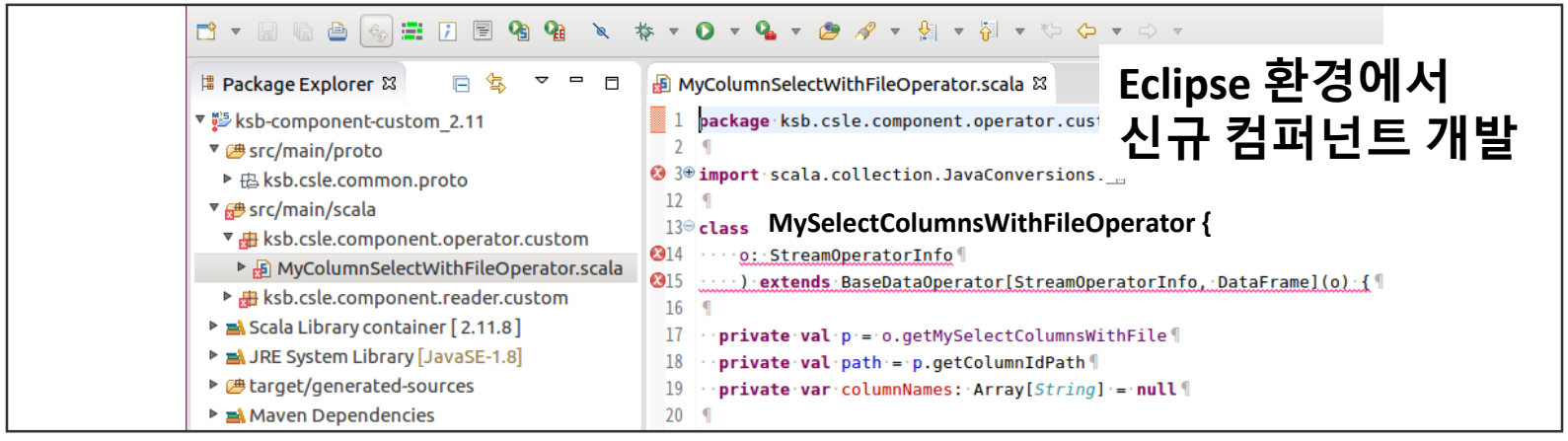
- 데이터 배치 처리
  - 데이터를 KMeans 클러스터링 처리해보기
    - 매뉴얼 참고
      https://csleoss.etri.re.kr/images/contents/manual_1.0/2.5.2.KMeansExample.html
- 딥러닝 모델 서빙하기
  - 텐서플로우 Inception 모델을 서빙하고, 이미지 분류 결과 받아보기
    - 매뉴얼 참고

      https://csleoss.etri.re.kr/images/contents/manual_1.0/2.5.11.InceptionServingExample.html

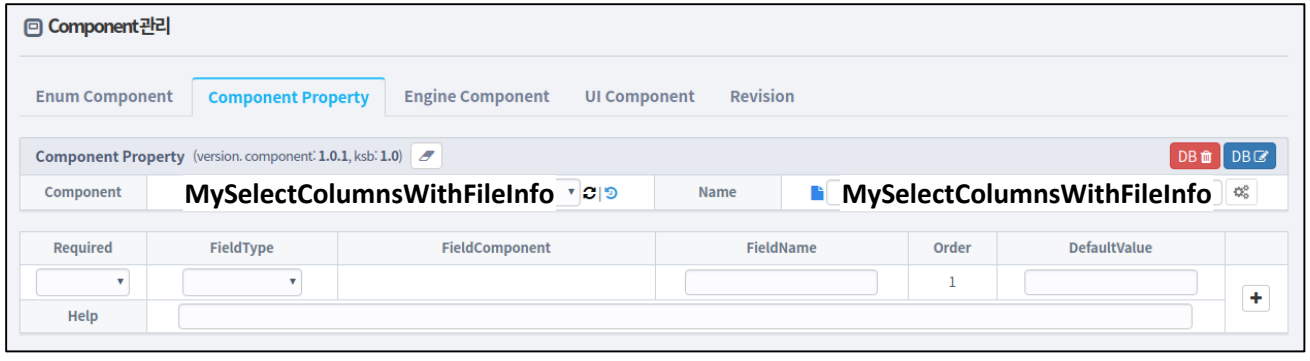
Component

BeeAI

- 개발자가 만든 신규 컴퍼넌트를 BeeAI 웹툴킷에 등록할 수 있도록 하는 Component 관리 기능

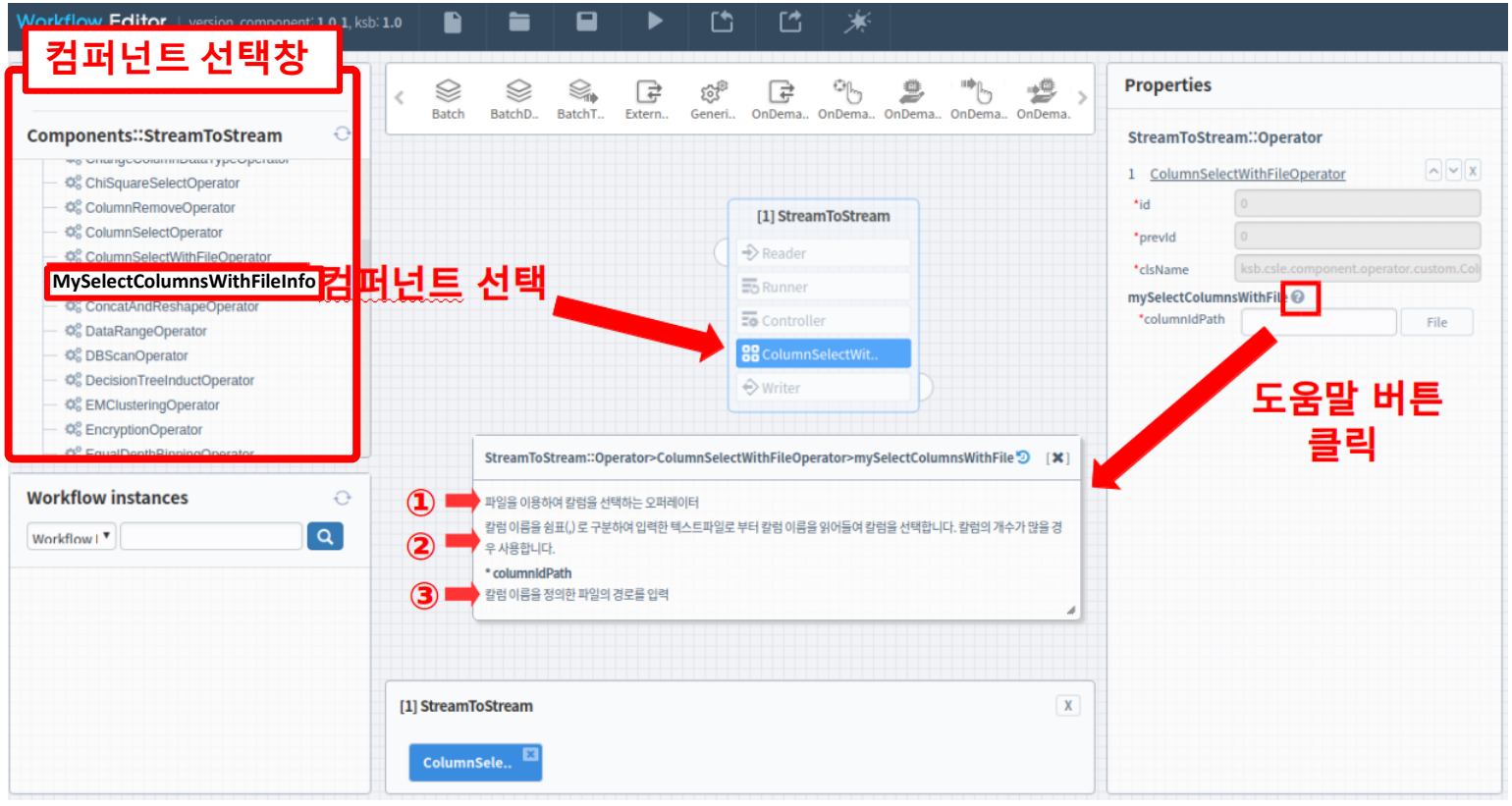


웹툴킷에 신규 컴퍼넌트 등록



Component

- 등록한 신규 컴퍼넌트는 컴퍼넌트 선택창에 보이게 됨
- 프레임워크 확장 효과



Part 2. BeeAI 를 활용한 인공지능 서비스 사례 살펴보기

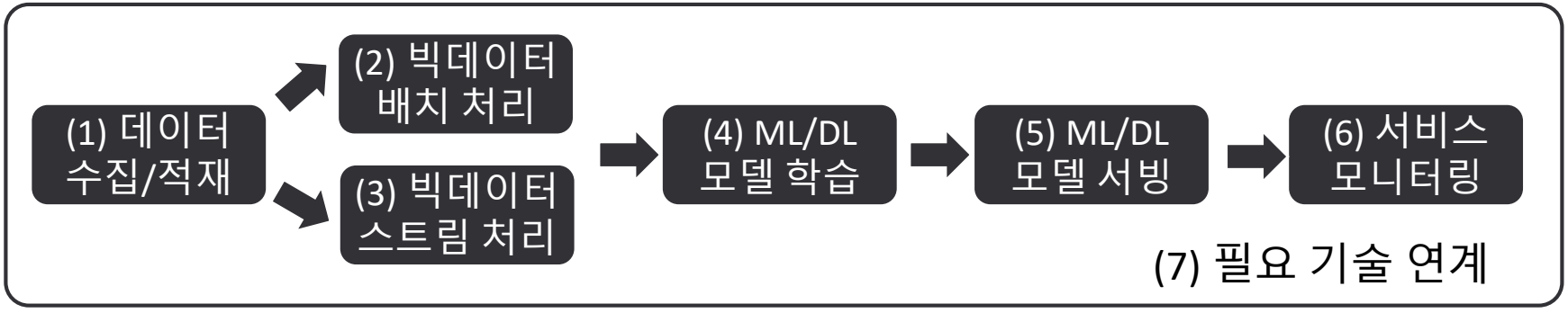
인공지능 서비스 제작

BeeAI

■ 제작 서비스 예

– 교통 속도 예측 AI 서비스

■ 필요 기술



인공지능 서비스 제작

BeeAI

■ 필요 기술 별 어려운 점

	필요 기술	어려운 점
1	데이터 수집/적재	데이터 유실, 효율적인 데이터 배치
2	빅데이터 배치 처리	분산 처리, 병렬 처리, 다양한 데이터 처리
3	빅데이터 스트림 처리	결함 복구
4	ML/DL 모델 학습	신규 학습 데이터가 반영된 최신 모델 유지
5	ML/DL 모델 서빙	서빙 요청 부하 분산
6	서비스 모니터링	서비스 구성 시스템들의 상태 정보 관리
7	필요 기술 연계	Seamless한 연계

인공지능 서비스 제작

BeeAI

■ BeeAI 의 장점

	필요 기술	어려운 점
1	데이터 수집/적재	데이터 유실, 효율적인 데이터 배치
2	빅데이터 배치 처리	분산 처리, 병렬 처리, 다양한 데이터
3	빅데이터 스트림 처리	결함 복구
4	ML/DL 모델 학습	신규 학습 데이터가 반영된 최신 모델
5	ML/DL 모델 서빙	서빙 요청 부하 분산
6	서비스 모니터링	서비스 구성 시스템들의 상태 정보 관
7	필요 기술 연계	Seamless한 연계

- 인공지능 개발자는 본인의 인공지능 로직 개발에만 집중 가능
- 인공지능 서비스 제작에 높은 생산성 유지 가능
  - 시스템 연계의 많은 부분을 BeeAI 가 해결

튜토리얼: 교통 속도 예측 AI 서비스 워크플로우 만들기

교통 속도 예측 AI 서비스

BeeAI

- 문제
  - 교통 속도 예측을 통한 강남 지역 교통 효율 향상
- 입력 데이터
  - 택시에 부착된 센서에서 수집되는 실시간 속도 스트림 데이터
- 결과
  - 최신 버전의 교통 속도 예측 모델
  - 도로 링크들의 15분 후 교통 속도 예측 결과

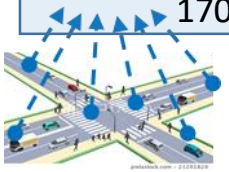
교통 속도 예측 AI 서비스

BeeAI

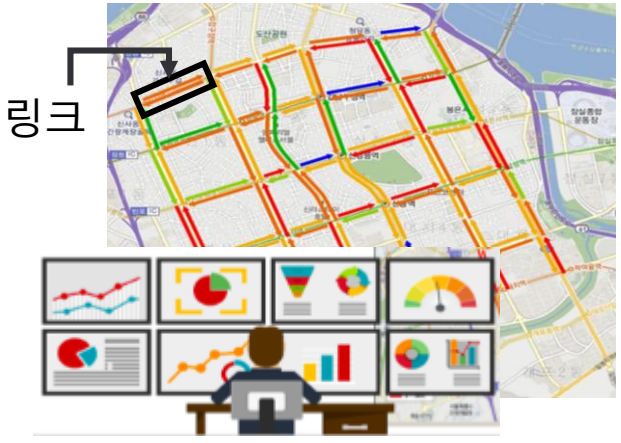
실시간 교통 센서 데이터

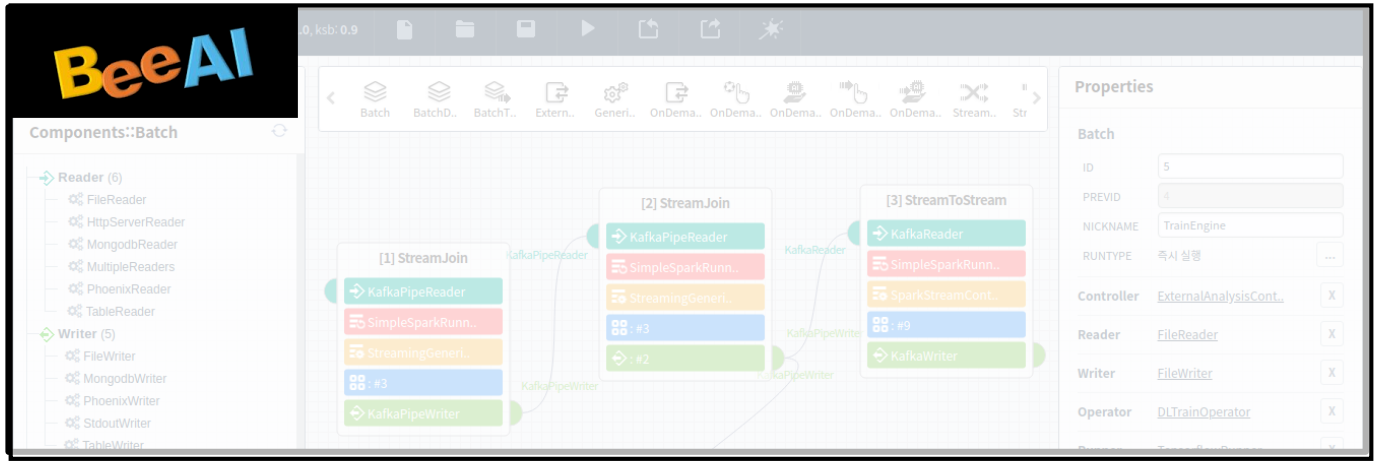
측정 시간	링크아이디	속도
2016-01-01 00:00:27	1220034700	32.8
2016-01-01 10:06:10	1220019901	72.0
2016-01-01 10:05:40	1220049900	40.5
⋮	⋮	⋮
2016-01-31 23:55:57	1220036600	69.8

170개 링크의 초단위 속도



지능형 교통 예측 서비스





교통 속도 예측 AI 서비스

BeeAI

■ BeeAI 워크플로우 편집 및 실행

Kafka stream File

BeeAI

엔진 A
실시간
스트림
데이터 수집/
전처리(1분)

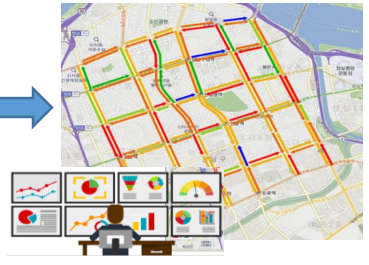
엔진 B
실시간
스트림
데이터 수집/
전처리(5분)

엔진 D
딥러닝 모델
학습용
전처리

엔진 E
딥러닝 모델
주기적
재학습

모델 갱신

엔진 C
딥러닝 모델
예측용
전처리/
스트림 서빙

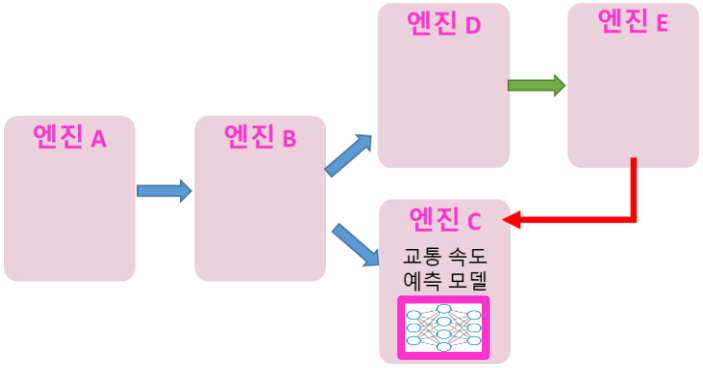


교통 속도 예측 AI 서비스

BeeAI

■ 교통 속도 예측 모델

- 15분 후 교통 속도를 예측하는 텐서플로우 LSTM* 모델
  - 교통 속도의 "시간에 따른 패턴"을 학습하여 교통 상황 예측 가능



- 입력 데이터
  - 지난 40분 동안 교통 속도로 만들어진 170개 링크의 입력 시퀀스

링크아이디 170개

PRCS_DATE	1210006200	1210006300	1210011200	1210011300	...
2016-01-01 04:40:00	0.1330555555555556	0.1965555555555548	-0.06025000000000008	-0.2623333333333333	
2016-01-01 04:45:00	0.14731250000000007	0.1982857142857144	0.054888888888888876	-0.26949999999999996	
2016-01-01 04:50:00	0.1397777777777779	0.2013124999999999	0.054777777777777814	-0.23966666666666664	
2016-01-01 04:55:00	0.138	0.1992222222222222	0.04656249999999995	-0.25293750000000004	
2016-01-01 05:00:00	0.12724999999999997	0.18427777777777765	0.08184999999999998	-0.2682857142857143	
2016-01-01 05:05:00	0.1517222222222222	0.2014999999999999	0.09960000000000002	-0.22312500000000002	
2016-01-01 05:10:00	0.13856250000000003	0.18700000000000006	0.03761111111111115	-0.20350000000000001	
2016-01-01 05:15:00	0.14087500000000008	0.16856250000000006	0.0842857142857143	-0.13961111111111107	

04시 40분 ~ 05시 15분

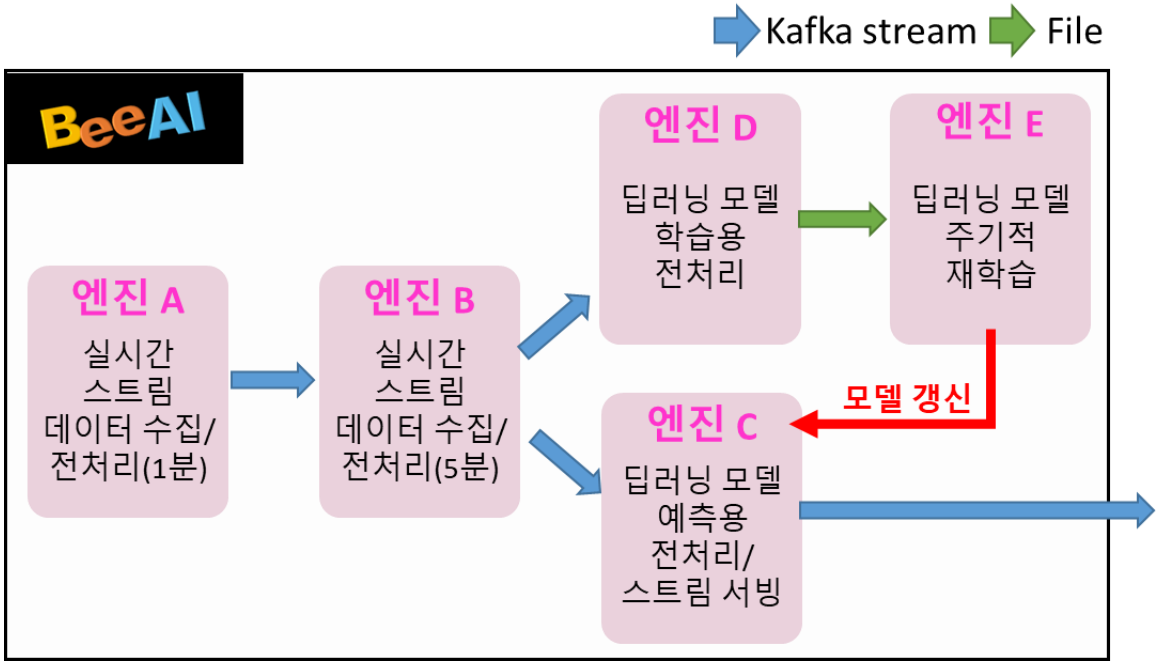
초 단위 입력 데이터에 대한 5분 단위 평균 속도

*RNN(Recurrent Neural Networks)의 종류 중 하나로, 시계열 데이터에서 패턴을 인식하는 인공 신경망

교통 속도 예측 AI 서비스

BeeAI

- 튜토리얼 구성
  1. 엔진 D 제작
  2. 엔진 E 제작
  3. 엔진 C 제작
  4. 엔진 A, B 제작
  5. 엔진 A, B, C 조합
  6. 엔진 A, B, C, D, E 조합
  7. 딥러닝 모델 온디맨드 서빙



1. 학습용 데이터 전처리

BeeAI

■ 학습용 데이터 전처리 엔진 (엔진 D)

- 입력
  - 각 링크의 5 분 단위 평균속도 스트림 데이터

PRCS_DATE	LINK_ID	PRCS_SPD
2016-01-01 00:00:00	1210006200	61
⋮	⋮	⋮
2016-01-01 00:05:00	1210011200	49.8
⋮	⋮	⋮
2016-01-31 23:55:00	1220037300	85.5

201601_kangnam.csv

- 프로세싱
  - 교통 모델의 학습 데이터 형태로 입력 시퀀스 만들기
    - ✓ 10 초 마다 윈도우크기로 입력 데이터를 자르고
      - 170개 링크의 9주기 데이터, 1530 개
    - ✓ 정제된 형태의 8주기 데이터로 변형
      - OrderBy(PRC_DATE), MinMaxScaling, Pivot
- 출력
  - traffic_processing.csv 파일

PRC_DATE 링크아이디 170개

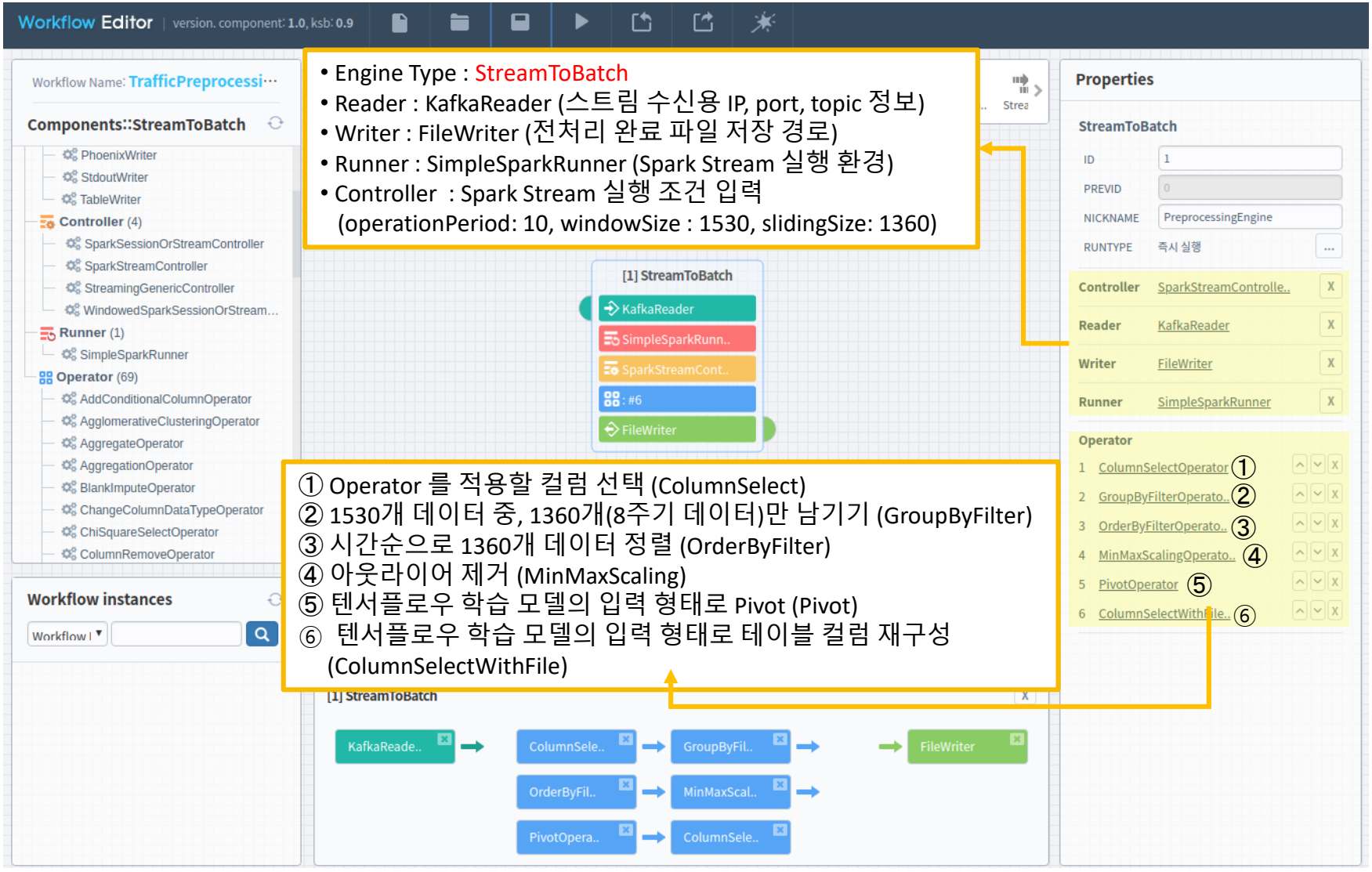
	A	B	C	D	E	F	G	H	I	J	K	L
1	2016-01-01 0:10	-0.275	-0.18288	-0.2536	-0.20422	0.289944	-0.35728	-0.37989	-0.35	-0.21106	-0.23319	-0.25028
2	2016-01-01 0:15	-0.25833	-0.229	-0.27014	-0.26094	0.27875	-0.35386	-0.36844	-0.3351	-0.21915	-0.29628	-0.25257
3	2016-01-01 0:20	-0.24094	-0.23356	-0.257						-0.20217	-0.29819	-0.27525
4	2016-01-01 0:25	-0.28583	-0.25044	-0.289						-0.21144	-0.29872	-0.25994
5	2016-01-01 0:30	-0.23314	-0.22906	-0.28						-0.21886	-0.27969	-0.24814
6	2016-01-01 0:35	-0.14879	-0.16319	-0.28156	-0.23036	0.286444	-0.3338	-0.34657	-0.2335	-0.19194		
7	2016-01-01 0:40	-0.13308	-0.29344	-0.25372	-0.24513	0.31085	32607	-0.31583	-0.214	-0.19288		
8	2016-01-01 0:45	-0.05183	-0.17939	-0.21929	-0.019	0.33	28371	-0.185	-0.12471	-0.06679		
9	2016-01-01 0:50	-0.094	-0.25879	-0.1505	-0.09562	0.3318	-0.15244	-0.20528	-0.18494	-0.11625	-0.13422	-0.10821
10	2016-01-01 0:55	-0.1255	-0.2098	-0.14171	-0.16169	0.335563	-0.16217	-0.125	-0.0622	-0.12875	-0.11867	-0.13333

1360개 속도 데이터

8주기

1. 학습용 데이터 전처리

BeeAI

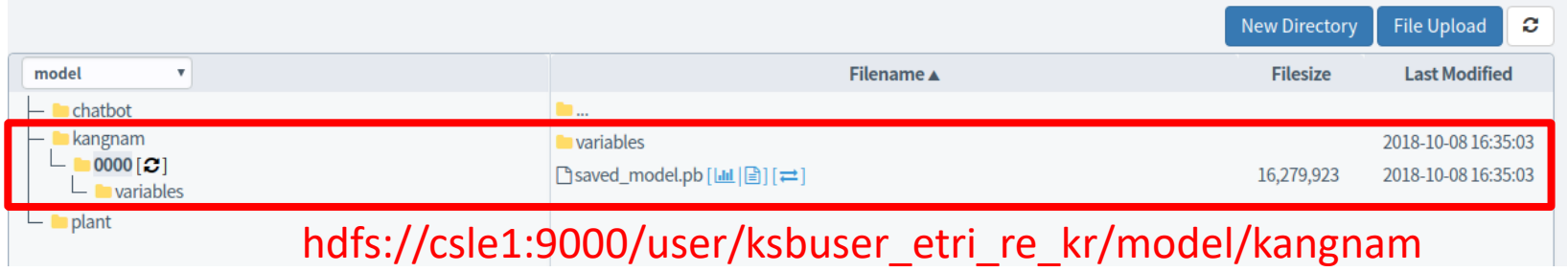


2. 교통 모델 학습

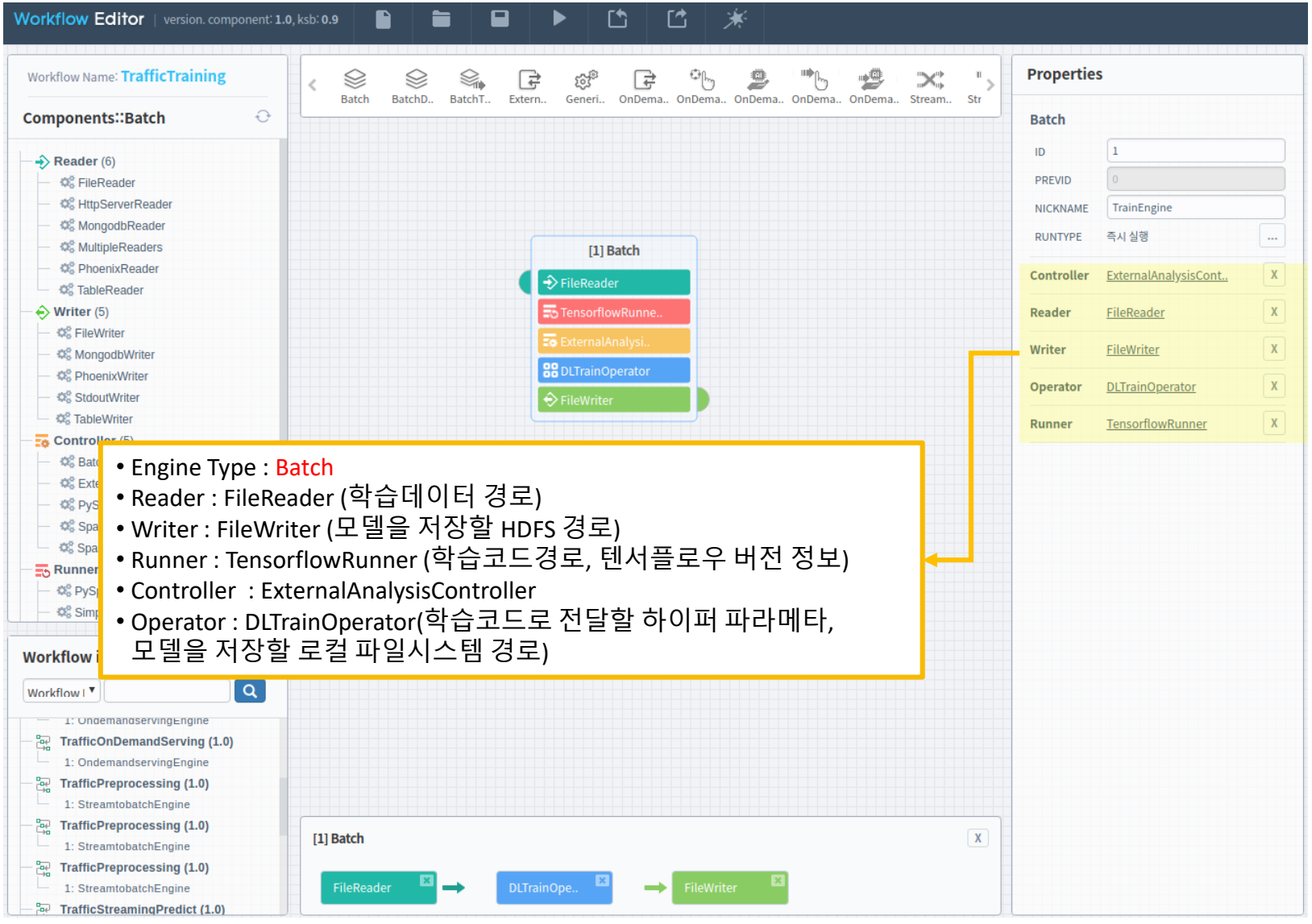
BeeAI

■ 교통 모델 학습 엔진 (엔진 E)

- 입력
  - 전처리 완료된 학습데이터(traffic_processing.csv)
  - 15분 후 교통 속도를 예측하는 텐서플로우 LSTM 학습 코드
- 프로세싱
  - traffic_processing.csv 파일을 입력으로 텐서플로우 학습 수행
- 출력
  - 15분 후 교통 속도를 예측하는 텐서플로우 모델



2. 교통 모델 학습


Workflow Editor | version. component: 1.0, ksb: 0.9
Workflow Name: TrafficTraining
Components::Batch
Reader (6)
FileReader
HttpServerReader
MongodbReader
MultipleReaders
PhoenixReader
TableReader
Writer (5)
FileWriter
MongodbWriter
PhoenixWriter
StdoutWriter
TableWriter
Batch
BatchD..
BatchT..
Extern..
Generi..
OnDema..
Stream..
[1] Batch
FileReader
TensorflowRunne..
ExternalAnalysi..
DLTrainOperator
FileWriter
Properties
Batch
ID
1
PREVID
NICKNAME
TrainEngine
RUNTYPE
즉시 실행
Controller
ExternalAnalysisCont..
Reader
FileReader
Writer
FileWriter
Operator
DLTrainOperator
Runner
TensorflowRunner
• Engine Type : Batch
• Reader : FileReader (학습데이터 경로)
• Writer : FileWriter (모델을 저장할 HDFS 경로)
• Runner : TensorflowRunner (학습코드경로, 텐서플로우 버전 정보)
• Controller : ExternalAnalysisController
• Operator : DLTrainOperator(학습코드로 전달할 하이퍼 파라메타,
모델을 저장할 로컬 파일시스템 경로)
TrafficOnDemandServing (1.0)
1: OndemandservingEngine
TrafficPreprocessing (1.0)
1: StreamtobatchEngine
TrafficStreamingPredict (1.0)
[1] Batch
FileReader
DLTrainOpe..
FileWriter

3. 서빙용 데이터 전처리 및 스트림 서빙 BeeAI

■ 서빙용 데이터 전처리/ 스트림 서빙 엔진 (엔진 C)

- 입력
  - 각 링크의 5 분 단위 평균속도 스트림 데이터
  - 15분 후 교통 속도를 예측하는 텐서플로우 모델
    - ✓ hdfs://csle1:9000/user/ksbuser_etri_re_kr/model/kangnam/
- 프로세싱
  - 서빙용 데이터 전처리
    - ✓ 텐서플로우 모델의 서빙 데이터 형태로 입력 시퀀스 만들기
      - 10 초 마다 윈도우크기로 입력 데이터를 자르고
        - 170개 링크의 9주기 데이터, 1530 개
      - 정제된 형태의 8주기 데이터로 변형
        - OrderBy(PRC_DATE), MinMaxScaling, Pivot
      - 8주기 데이터를 벡터로 변환
        - VectorAssembleColumn, Flatten

학습용 데이터 전처리와 동일

  - 예측 수행
    - ✓ TensorFlowPredictOperator

3. 서빙용 데이터 전처리 및 스트림 서빙 BeeAI

■ 서빙용 데이터 전처리/ 스트림 서빙 엔진 (엔진 C)

– 서빙 결과

Workflow | Workflow History

필터링 ▾ [전체:44714건]

Workflow Name	Engine ID	Engine Type	Engine Nickname	Start Time ▾	End Time	Status
TrafficStreamServing	1	Streamtostream	PredictEngine	2018-10-08 18:19:07	2018-10-08 18:19:07	Inprogress

[Workflow History 탭의 서빙 상태]

csle@csle1: ~

{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 61.0, "LINK_ID": "1210006200"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 66.8, "LINK_ID": "1210006300"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 49.8, "LINK_ID": "1210011200"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 33.4, "LINK_ID": "1210011300"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 45.0, "LINK_ID": "1210013600"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 27.4, "LINK_ID": "1210013700"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 26.5, "LINK_ID": "1210030000"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 48.3, "LINK_ID": "1210030100"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 86.0, "LINK_ID": "1210030600"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 85.6, "LINK_ID": "1210030700"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 43.8, "LINK_ID": "1220001500"}
{"PRCS_DATE": "2016-01-01 00:00:00", "PRCS_SPD": 59.3, "LINK_ID": "1220001600"}

[스트림 형태로 질의]

csle@csle1: ~/kafka

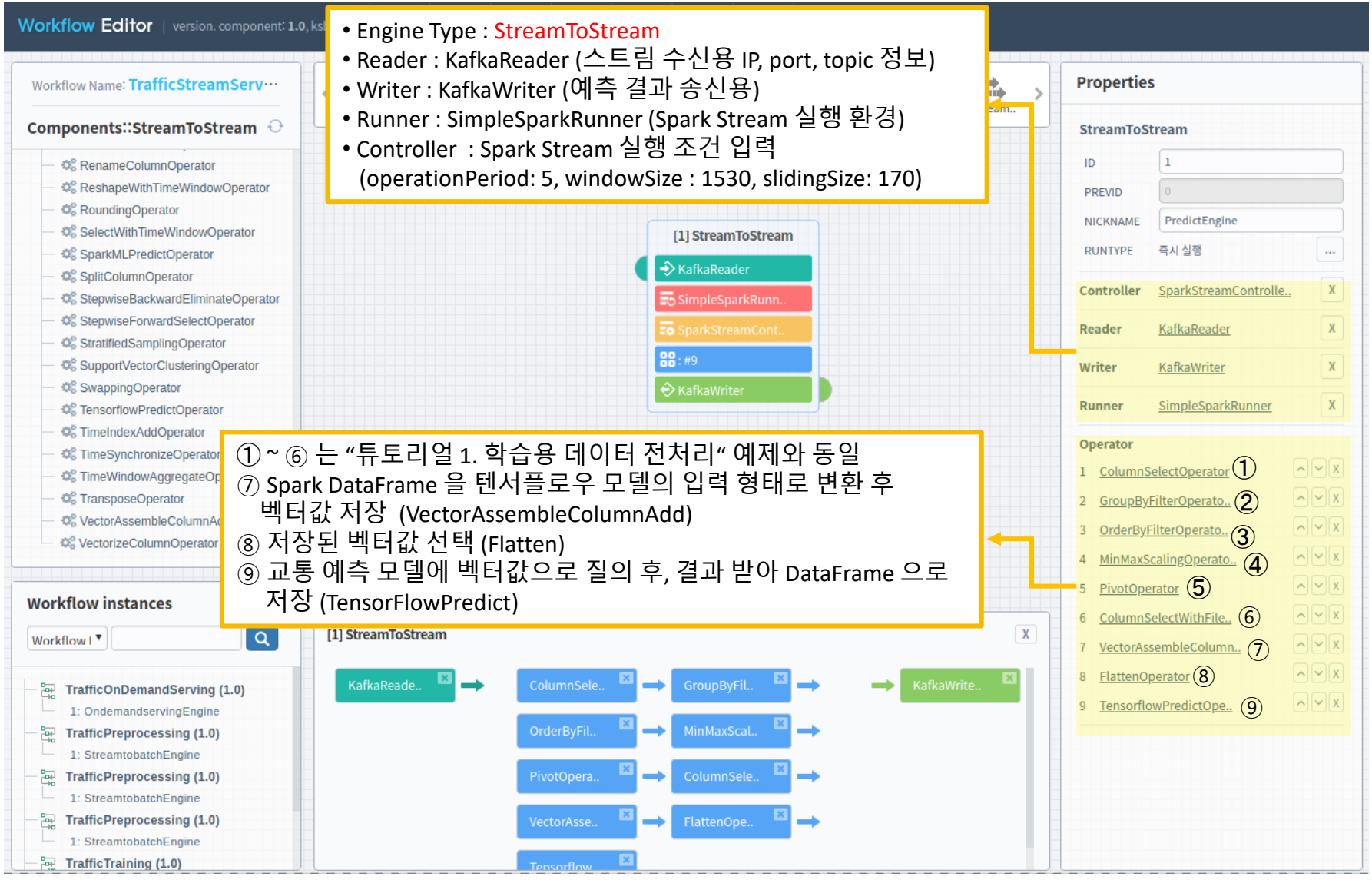
```
csle@csle1:~/kafka$ ./bin/kafka-console-consumer.sh --zookeeper localhost:2181 -
-bootstrap-server localhost:9092 --topic kangnam_output
{"in1":[-0.24843749999999998,-0.30625,-0.20970000000000005,-0.2124444444444445,0
.31522222222222207,-0.25231250000000005,-0.34678571428571425,-0.1828888888888889
3,-0.15200000000000002,-0.16155555555555556,-0.12485714285714278,0.0128888888888
8895,-0.28088888888888885,-0.11121428571428571,-0.18768749999999995,-0.151428571
42857147,-0.23662500000000003,-0
0.0656666666666667,-0.3190000000
,-0.2644444444444444,-0.03005555
4447,-0.22514285714285714,-0.148
```

연속적인 스트림 형태로 170개 링크의 15분 후 속도 예측 값 수신

[Kafka-consumer 활용]

[질의 후 예측 결과 받기]

3. 서빙용 데이터 전처리 및 스트림 서빙



4. 실시간 데이터 전처리

BeeAI

■ 실시간 데이터 전처리 엔진 (엔진 A, B)

[missing value/ 잡음 존재, 정렬되지 않은 상태]

측정 시간	링크아이디	속도
2016-01-01 00:00:27	1220034700	32.8
2016-01-01 10:06:10	1220019901	72.0
2016-01-01 10:05:40	1220049900	40.5
⋮	⋮	⋮
2016-01-31 23:55:57	1220036600	69.8

201601_kangnam_orgarnized_new.csv

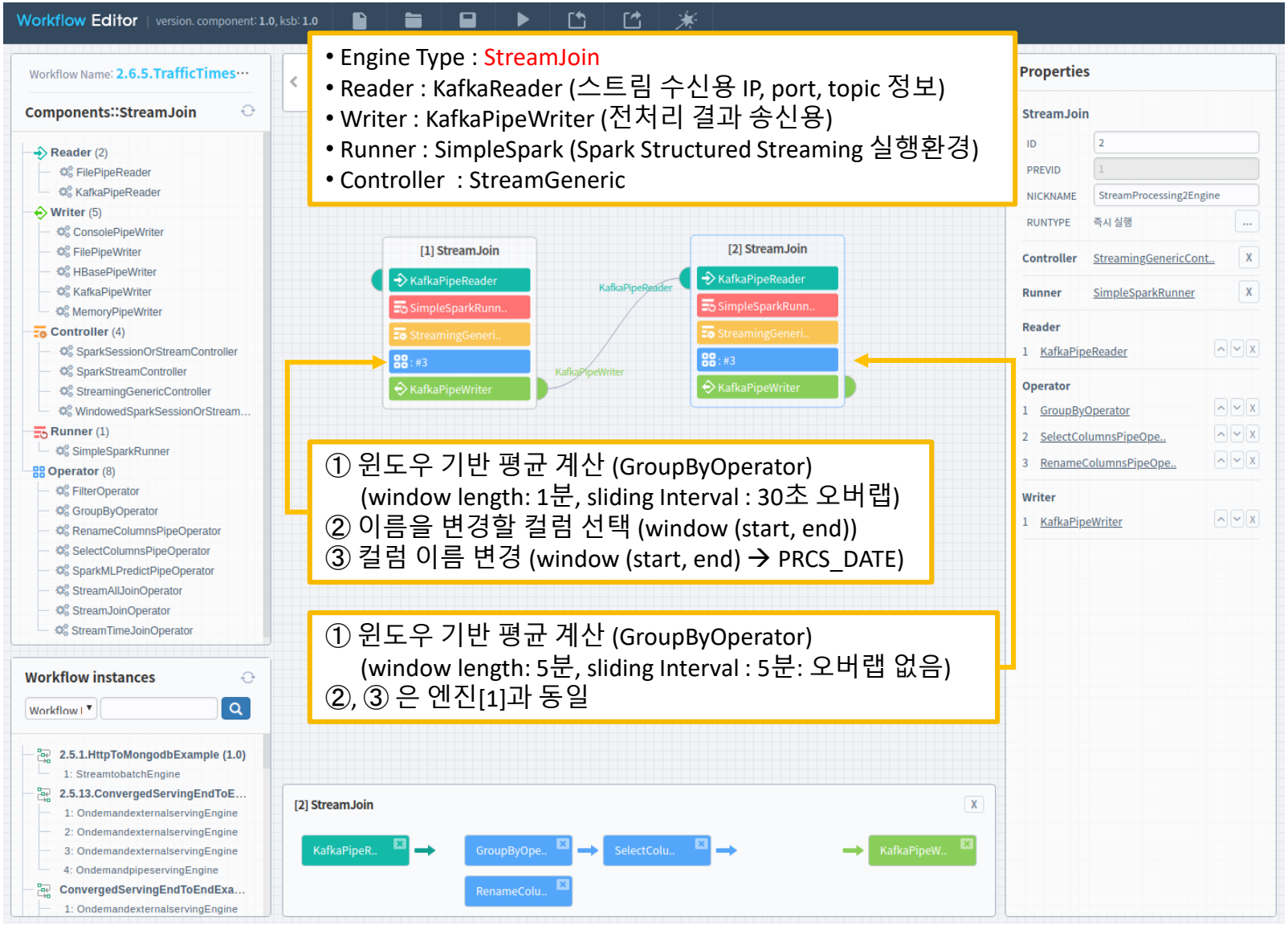
- 입력
  - 170개 링크에서 무작위로 입력되는 초 단위 속도 데이터
- 프로세싱
  - Spark Structured Streaming 이용
    - ✓ 늦게 도착하는 데이터 고려 watermark
    - ✓ 윈도우 크기(1분/5분) 단위 속도 aggregation
      - Missing value 채우기, 잡음 smoothing
- 출력
  - 각 링크의 5 분 단위 평균속도 스트림 데이터

PRCS_DATE	LINK_ID	PRCS_SPD
2016-01-01 00:00:00	1210006200	61
⋮	⋮	⋮
2016-01-01 00:05:00	1210011200	49.8
⋮	⋮	⋮
2016-01-31 23:55:00	1220037300	85.5

201601_kangnam.csv

4. 실시간 데이터 전처리

BeeAI



5. 실시간 데이터 전처리 및 모델 서빙 BeeAI

- 실시간 데이터 전처리/ 모델 서빙 엔진 (엔진 A, B, C 조합)
  - 입력
    - 170개 링크에서 무작위로 입력되는 초 단위 속도 데이터
    - 15분 후 교통 속도를 예측하는 텐서플로우 모델
  - 프로세싱
    - 실시간 데이터 전처리 (튜토리얼4)
      - ✓ 늦게 도착하는 데이터 고려 watermark
      - ✓ 윈도우 크기(1분/5분) 단위 속도 aggregation
        - Missing value 채우기, 잡음 smoothing
    - 서빙용 데이터 전처리 (튜토리얼3)
      - ✓ 텐서플로우 모델의 서빙 데이터 형태로 입력 시퀀스 만들기
        - 10 초 마다 윈도우크기로 입력 데이터를 자르고
          - 170개 링크의 9주기 데이터, 1530 개
        - 정제된 형태의 8주기 데이터로 변형
          - OrderBy(PRC_DATE), MinMaxScaling, Pivot
        - 8주기 데이터를 벡터로 변환
          - VectorAssembleColumn, Flatten
    - 예측 수행
      - ✓ TensorFlowPredictOperator

5. 실시간 데이터 전처리 및 모델 서빙 BeeAI

■ 실시간 데이터 전처리/ 모델 서빙 엔진 (엔진 A, B, C 조합)

– 서빙 결과

Workflow Name	Description	Batch	SubmitTime	KSB Version	Component Version	Status	-
EngineType	NickName	RunType	StartTime	EndTime	Period		-
TrafficStreamingPredict	강남 교통 예측 시나리오	false	2018-10-10 11:33:48	0.9	1.0	Inprogress	
Streamjoin	*StreamProcessingEngine*	*즉시 실행*			*ONCE*	*Inprogress*	[■]
Streamjoin	*StreamProcessing2Engine*	*즉시 실행*			*ONCE*	*Inprogress*	[■]
Streamtostream	*PredictEngine*	*즉시 실행*			*ONCE*	*Inprogress*	[■]

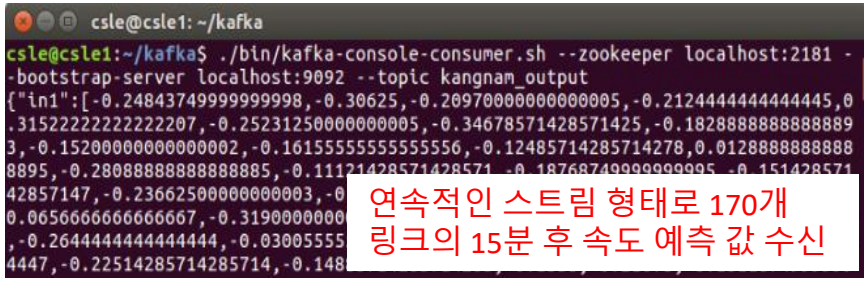
[Workflow History 탭의 서빙 상태]

[missing value/ 잡음 존재, 정렬되지 않은 상태]

측정 시간	링크아이디	속도
2016-01-01 00:00:27	1220034700	32.8
2016-01-01 10:06:10	1220019901	72.0
2016-01-01 10:05:40	1220049900	40.5
⋮	⋮	⋮
2016-01-31 23:55:57	1220036600	69.8

201601_kangnam_orgarnized_new.csv

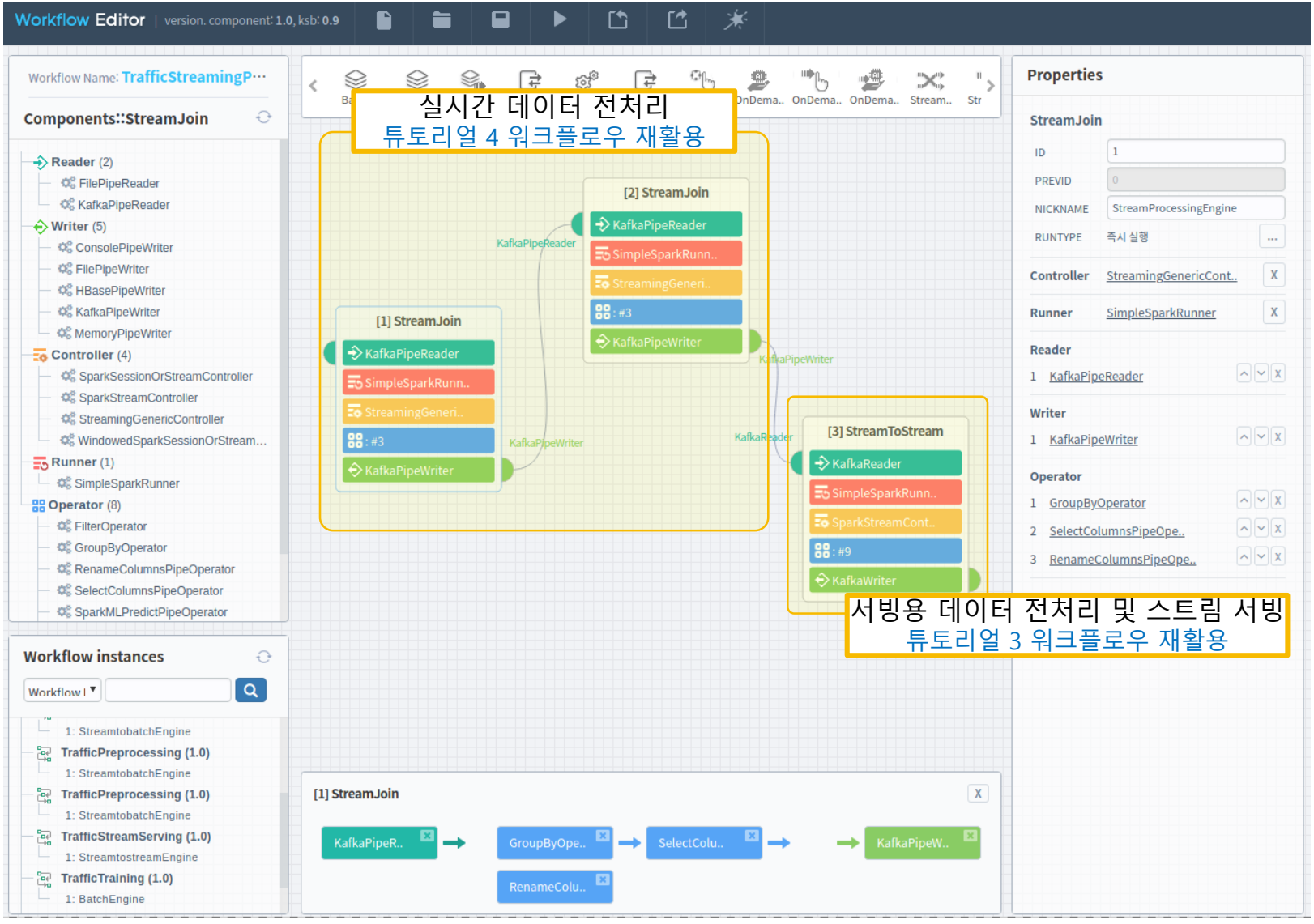
[스트림 형태로 질의]



[Kafka-consumer 활용]

[질의 후 예측 결과 받기]

5. 실시간 데이터 전처리 및 모델 서빙 BeeAI

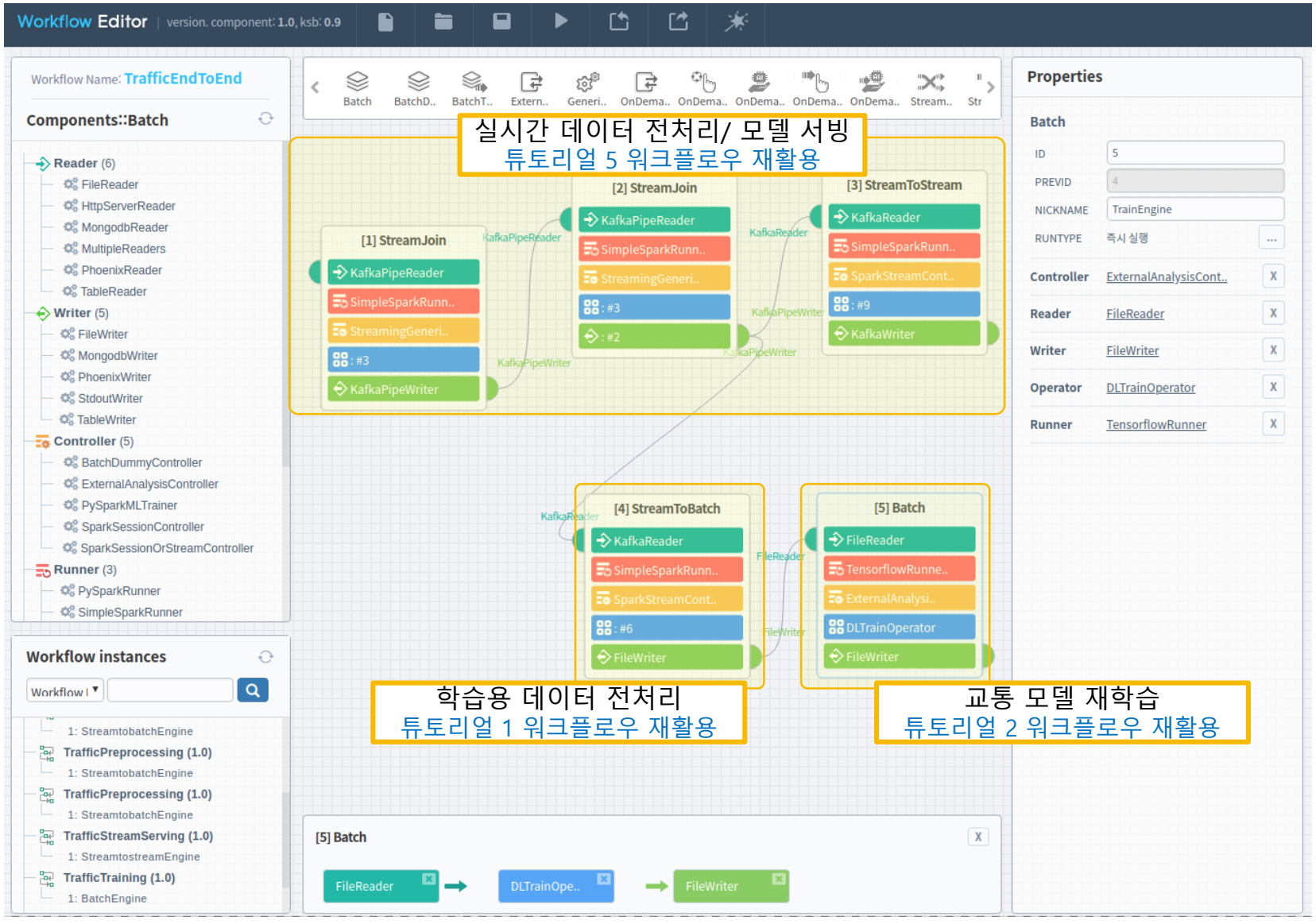


6. 실시간 데이터 전처리/모델 서빙/ 재학습 BeeAI

■ 전처리/ 모델 서빙/ 갱신 엔진 (엔진 A, B, C, D, E 조합)

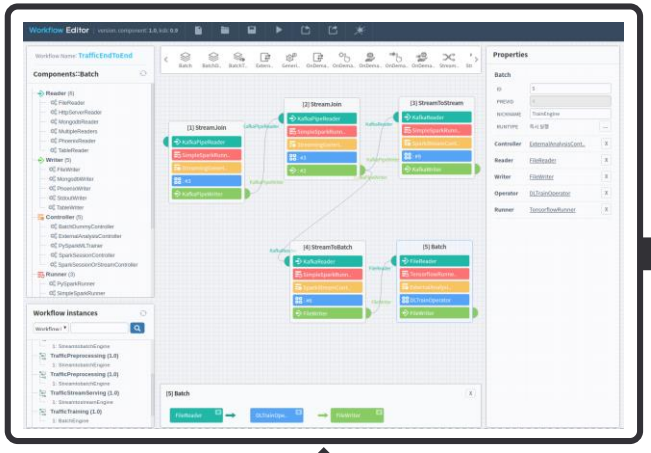
- 입력
  - 170개 링크에서 무작위로 입력되는 초 단위 속도 데이터
  - 15분 후 교통 속도를 예측하는 텐서플로우 모델
- 프로세싱
  - 실시간 데이터 전처리
  - 서빙용 데이터 전처리
  - 학습용 데이터 전처리
  - 교통 모델 주기적 재학습
  - 교통 모델 스트림 서빙
- 결과
  - 출력 : 주기적으로 재 학습 된 최신 텐서플로우 모델
  - 교통 모델 스트림 서빙

6. 실시간 데이터 전처리/모델 서빙/ 재학습

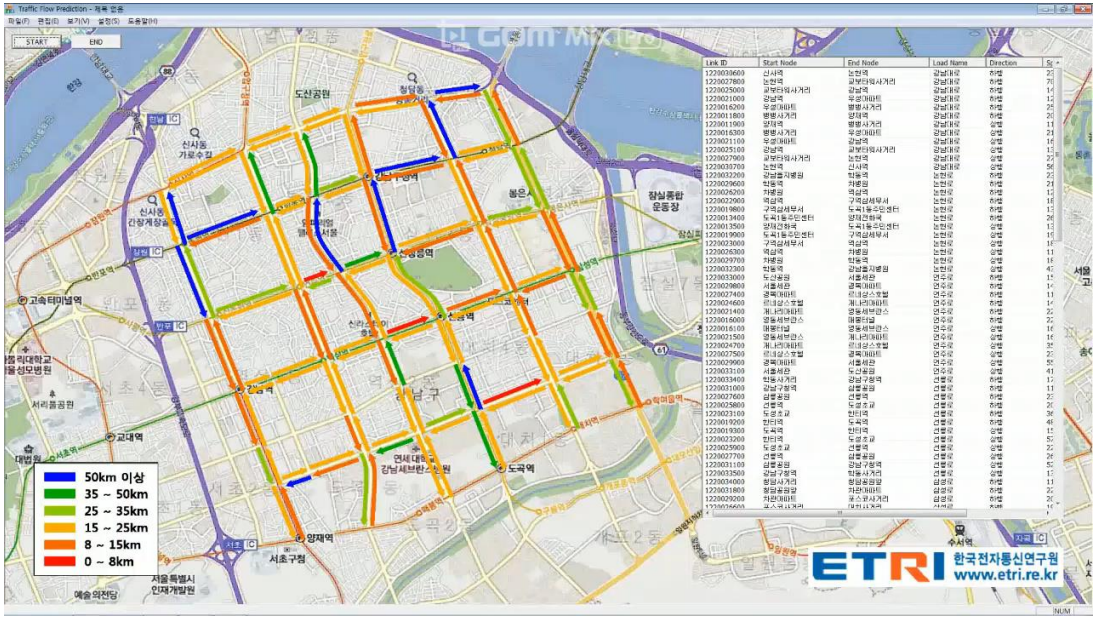


6. 실시간 데이터 전처리/모델 서빙/ 재학습 BeeAI

BeeAI 워크플로우 편집/ 실행



지능형 교통 예측 서비스



측정 시간	링크아이디	속도
2016-01-01 00:00:27	1220034700	32.8
2016-01-01 10:06:10	1220019901	72.0
2016-01-01 10:05:40	1220049900	40.5
⋮	⋮	⋮
2016-01-31 23:55:57	1220036600	69.8

170개 링크의 초단위 속도

실시간 교통 센서 데이터

7. 교통 모델 온디맨드 서빙 BeeAI

■ 교통 모델 온디맨드 서빙 엔진

- 입력

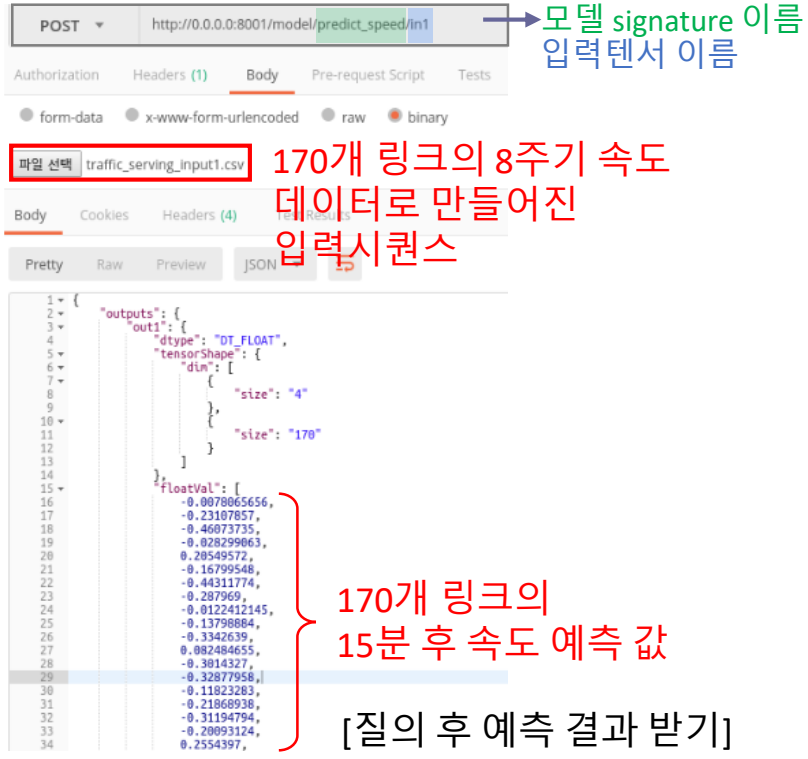
- 15분 후 교통 속도를 예측하는 텐서플로우 모델
  - ✓ hdfs://csle1:9000/user/ksbuser_etri_re_kr/model/kangnam/0000

- 서빙 결과

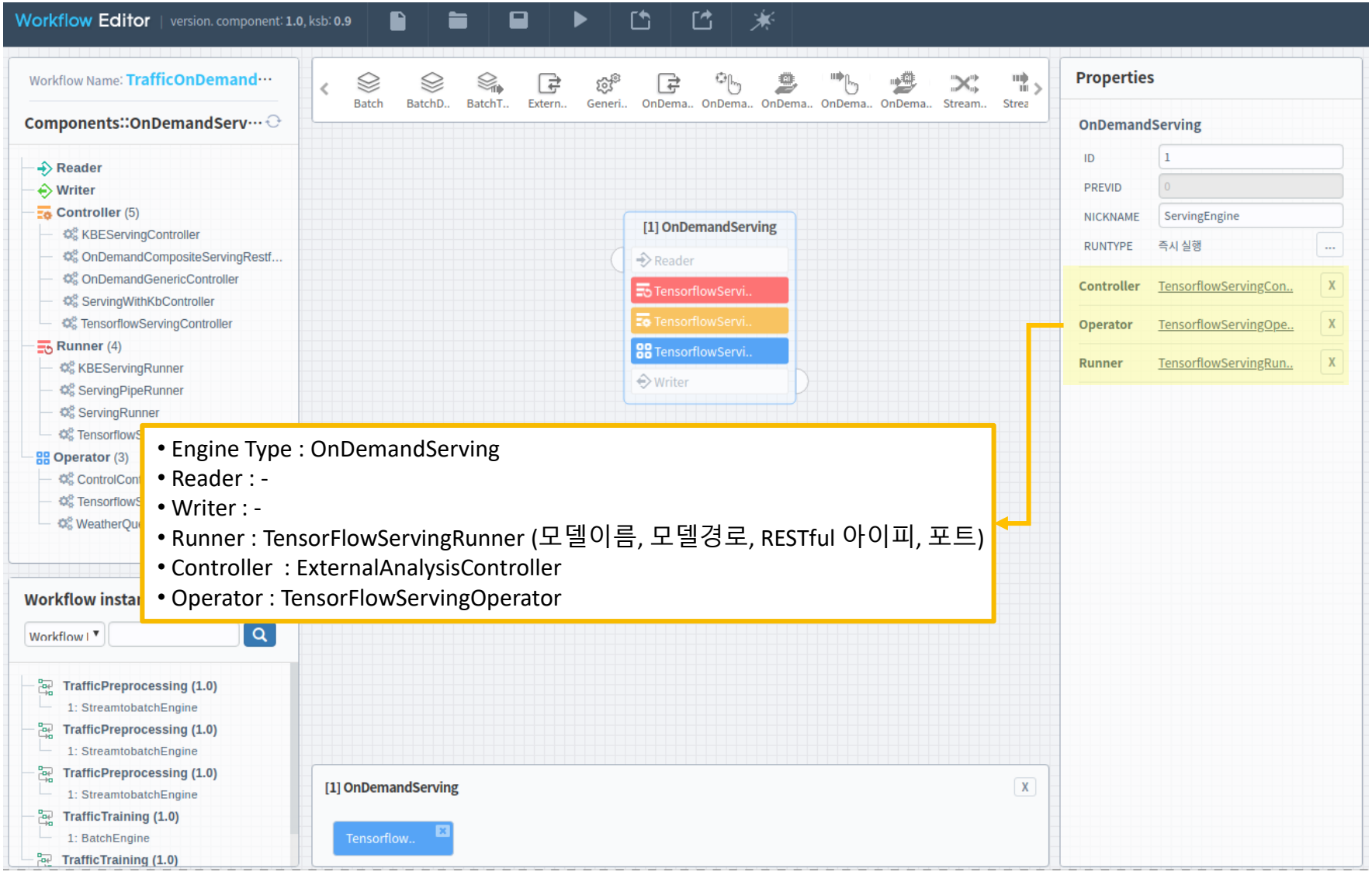
Services

Engine			
Type	Name	URL	Path
Ondemandserving	ServingEngine	http://10.217.10.72:8001	model/kangnam

[웹툴킷 대쉬보드의 서빙 상태]



7. 교통 모델 온디맨드 서빙 BeeAI

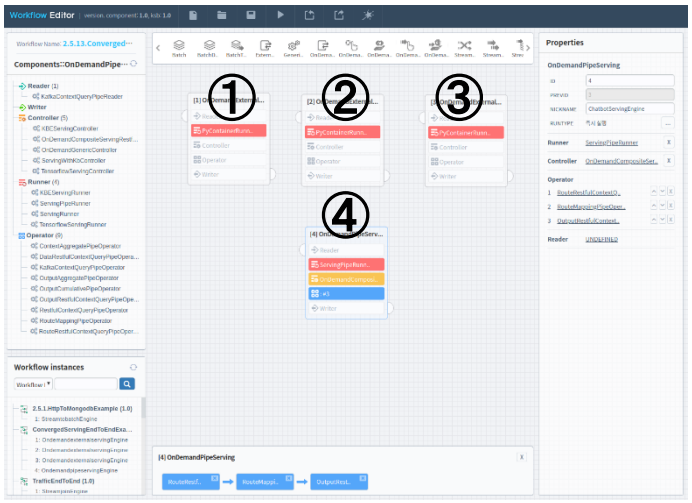


융합 서빙을 이용한 챗봇

BeeAI

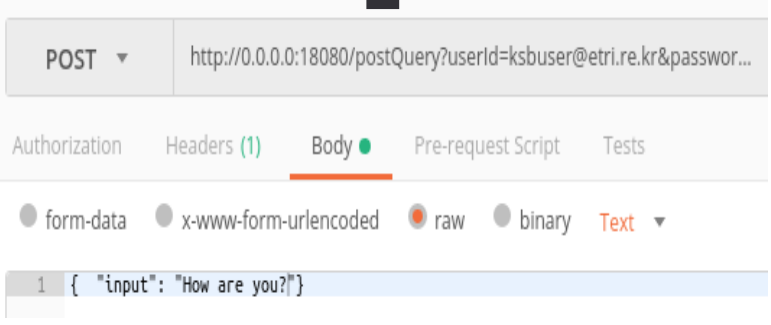
- 문제
  - 다양한 주제로 학습된 대화 모델을 융합하여 범용적인 챗봇 만들기
- 입력 데이터
  - 사용자의 질의 문장
- 모델
  - Classification TensorFlow 모델 (대화 주제 분류용)
  - Chitchat TensorFlow 모델 (일상 대화용)
  - Travel Agency TensorFlow 모델 (여행 대화용)
- 결과
  - 챗봇의 응답 문장

융합 서빙을 이용한 챗봇

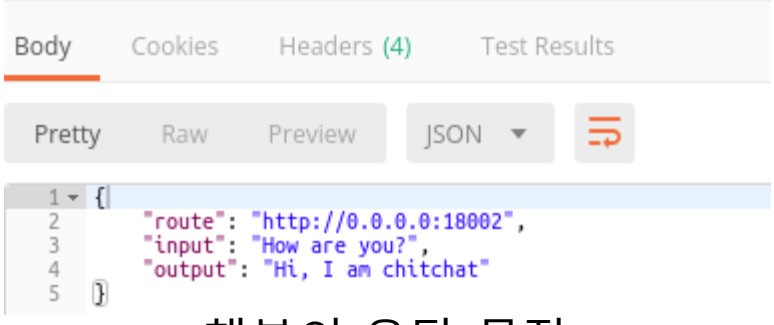


BeeAI 워크플로우 편집/ 실행

① Classification 모델 Dockerize
② Chitchat 모델 Dockerize
③ Travel Agency 모델 Dockerize
④ RESTful API들을 연결하여 챗봇 서빙



사용자 질의 문장



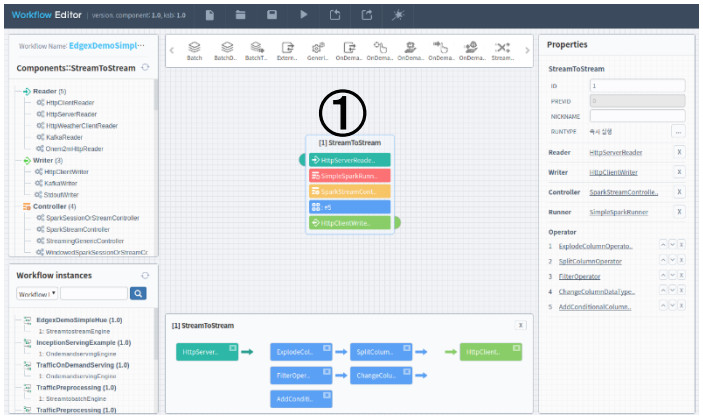
챗봇의 응답 문장

EdgeX 연동을 통한 장치 제어

BeeAI

■ 문제

– 방안의 온도에 따라 색상이 변화하는 감성 전등 개발

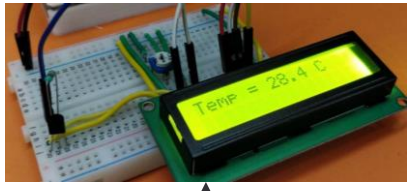


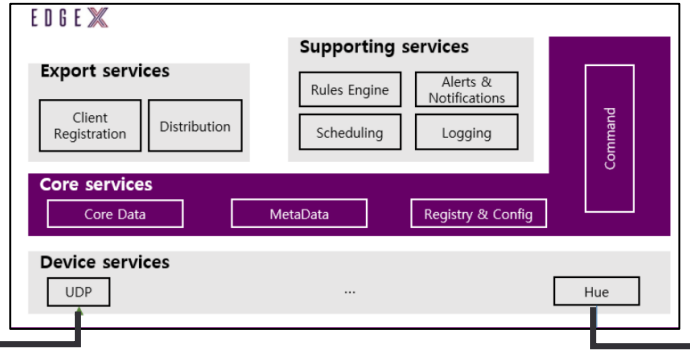
BeeAI 워크플로우 편집/ 실행

- Edge-X로부터 온도 데이터 수집
- 스트림데이터 전처리
- 제어 규칙 설정
  (AddConditionalColumnOperator)
- 제어 메세지 스트림 전달

온도 스트림 데이터 ↑

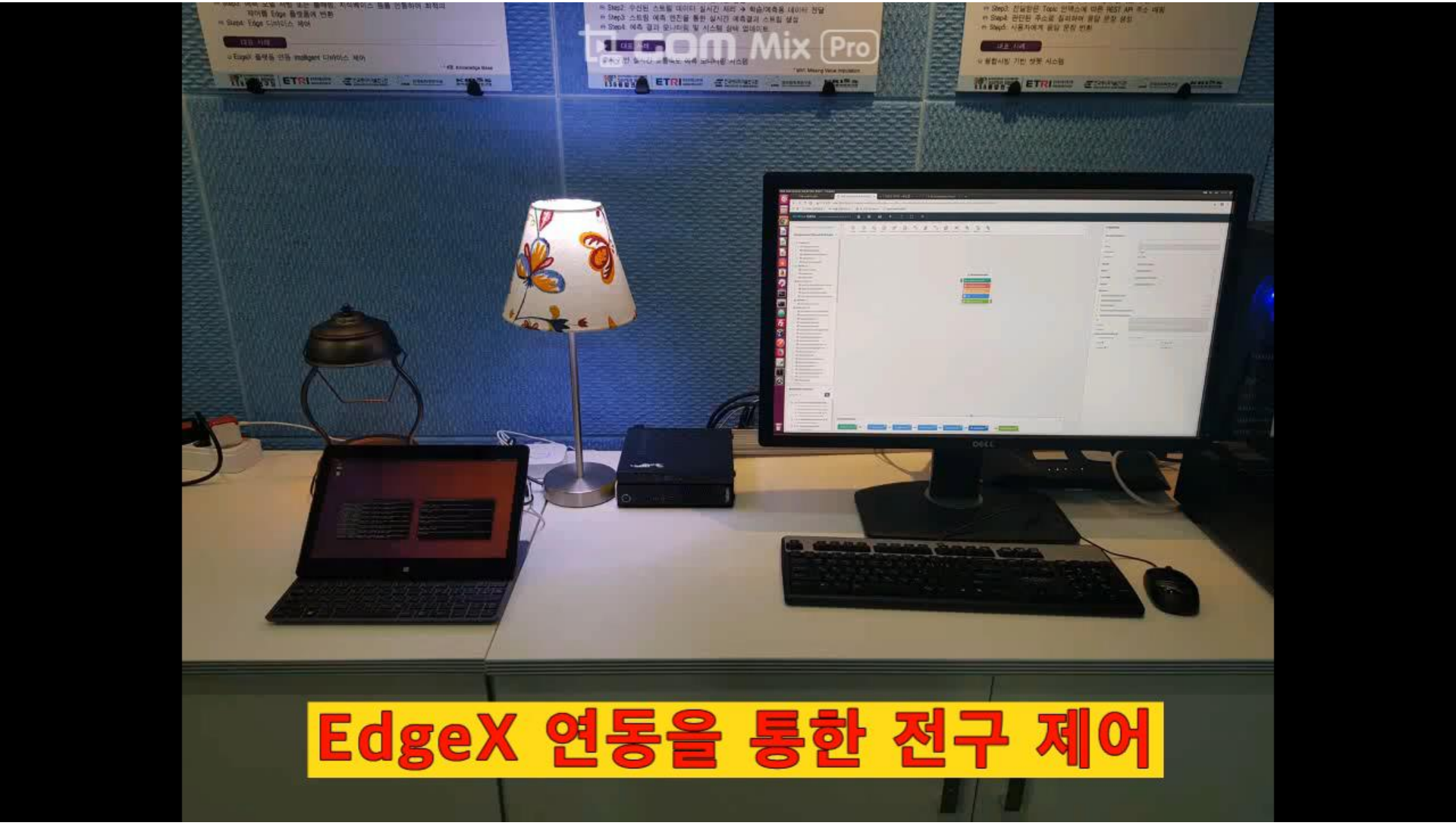
↓ 규칙 기반 제어 메세지





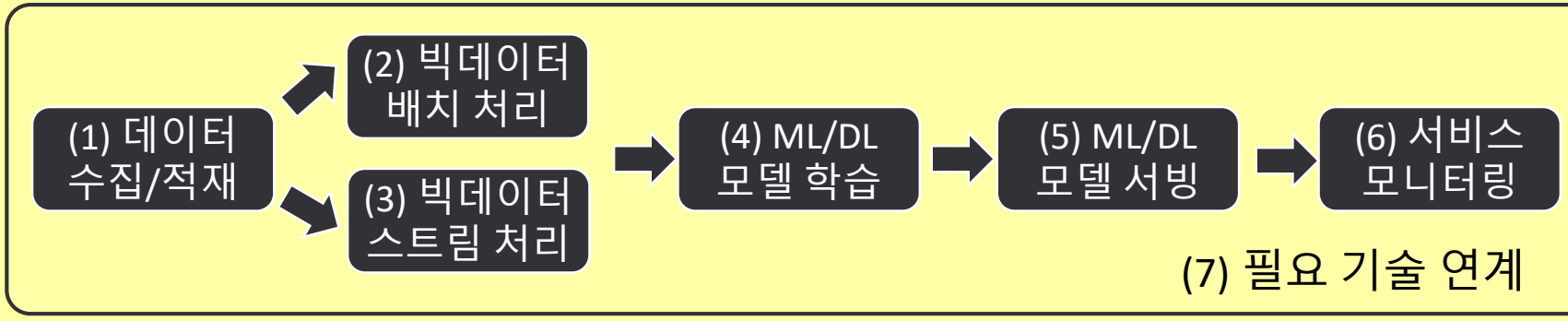
방안 온도에 따른
전구 색상 제어

EdgeX 연동을 통한 장치 제어



BeeAI 의 비전

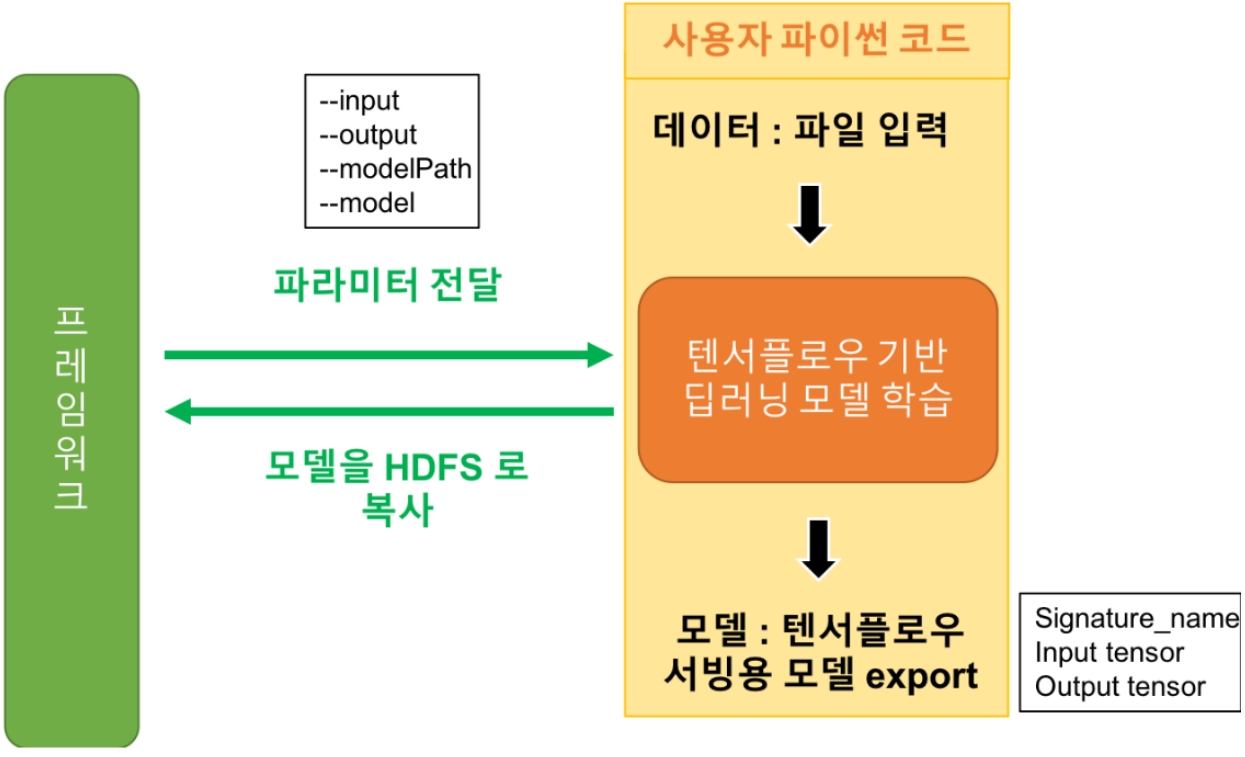
- 오픈 소스화를 통해 인공지능 서비스 제작에 필요한 기술들로 풍부하게 채워진 BeeAI



BeeAI supporting the End-to-End AI workflow

TensorFlow 학습코드 작성 BeeAI

- BeeAI-TensorFlow 연동 구조
  - 사용자의 TF python 코드에 파라미터를 전달하여 학습 실행
  - TF Serving 용으로 export된 모델을 HDFS 로 복사해 오는 방식



TensorFlow 학습코드 작성 BeeAI

- BeeAI-TensorFlow 연동을 위해 구현할 부분
  1) BeeAI 에서 전달하는 파라메타를 받는 부분
     - 학습 데이터 경로, 모델 저장 경로, 학습 epoch 등
  2) 학습된 모델을 서빙용으로 export 하는 부분
     - 매뉴얼 참조
       - ✓ 학습 코드 작성 가이드라인
         https://csleoss.etri.re.kr/images/contents/manual_1.0/2.7.2.KSB_TfPyCoding_Guide.html

파이썬 모듈 사용하기

BeeAI

- Python 으로 구현된 사용자 정의 함수를 Docker 이미지로 변환하고, REST API 추가하기
  - KSB Dockerize 라이브러리 활용
    - OnDemandExternalServing엔진
    - 사용자 Python 코드를 Docker 이미지로 자동 변환
    - Docker 이미지에 REST API 기능을 자동으로 추가
      - ✓ 사용자 Python 코드로 자료 입력 및 출력 결과 수신 가능
  - 구현할 부분
    - __init.py__, base.py (main_func), requirements.txt
  - 매뉴얼 참조
    - 사용자 정의 Python 함수 작성 가이드라인 https://csleoss.etri.re.kr/images/contents/manual_1.0/2.7.1.KSB_Dockerize.html

파이썬 모듈 사용하기

BeeAI

- KSB Dockerize 라이브러리 활용 예
  - 챗봇 서빙에 활용할 chitchat 파이썬 코드를 Dockerize

```
~/ksb-csle/examples/pyModules/ChatbotServing/chitchat$ ls -al

2 ksbuser_etri_re_kr ksbuser_etri_re_kr 4096  4월  15 14:25 .
5 ksbuser_etri_re_kr ksbuser_etri_re_kr 4096 11월   6 15:56 ..
1 ksbuser_etri_re_kr ksbuser_etri_re_kr    0 11월   6 15:56 __init__.py
1 ksbuser_etri_re_kr ksbuser_etri_re_kr  958 11월   6 15:56 base.py
1 ksbuser_etri_re_kr ksbuser_etri_re_kr    5 11월   6 15:56 requirements.txt
~/ksb-csle/examples/pyModules/ChatbotServing/chitchat$
```

➡ ChatboatServing/chitchat/ 폴더 내부 구조
- __init.py, base.py, requirements.txt

```
def main_func(x):

    # get input
    input = x
    text = tune_text(input)

    # call your function
    output_text = predict(text)

    return str(output_text)
```

```
if __name__ == '__main__':
    input = getInput()
    # print(input)
    out = predict(input)
    print(out)
~
```

➡ base.py
- main_func() 구현 필수
  : REST API 로 입력 받고, 출력 결과 리턴하는 기능

```
nltk
```

➡ requirements.txt
- 사용자 코드 내에 필요한 파이썬 라이브러리 명시 (선택사항)

감사합니다