kakaocloud

Kubeflow 생태계에서 완성하는 ML 워크플로우 : 데이터 분석부터 자동화까지

2025.08.26

카카오엔터프라이즈

정지성

Contents

1. Kubeflow 소개

2. ML Task : 로드 밸런서 트래픽 예측
   - 2.1. 시나리오 소개
   - 2.2. (Step 1) 데이터 탐색 및 모델 개발
   - 2.3. (Step 2) 하이퍼 파라미터 튜닝
   - 2.4. (Step 3) 모델 서빙
   - 2.5. (Step 4) 파이프라인 구축

3. 카카오클라우드 서비스
   - 3.1. Kubeflow
   - 3.2. MLOps on Kakaocloud

kakaocloud

1. Kubeflow 소개

Kubeflow?

Kubeflow

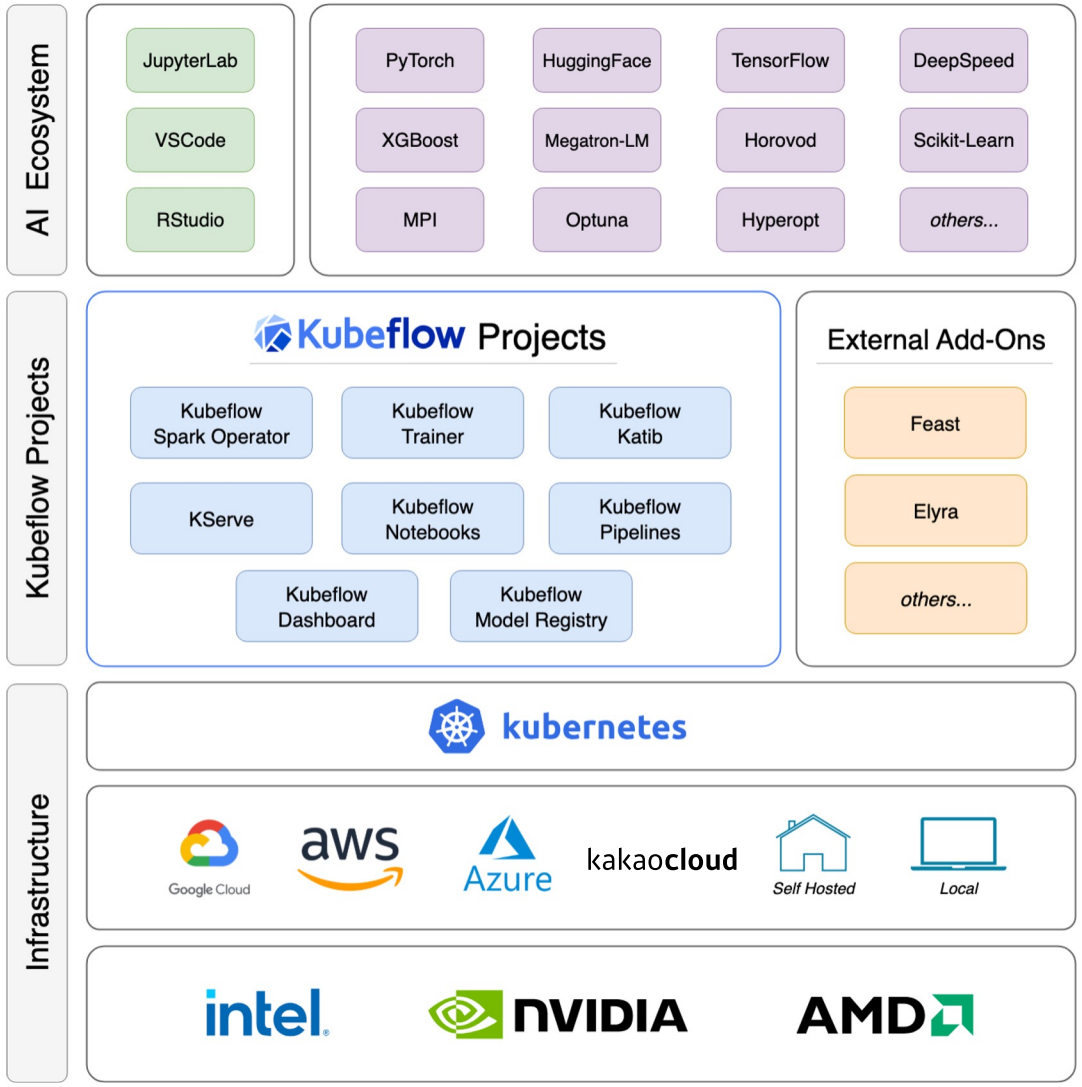
Kubeflow란?

- 머신러닝 (ML) 라이프 사이클의 각 단계를 해결하기 위한 **오픈소스 프로젝트 커뮤니티**이자 **생태계**
- 독립형 컴포넌트(프로젝트)들과 통합, 관리 툴을 통해 **AI 플랫폼**으로 사용

목표

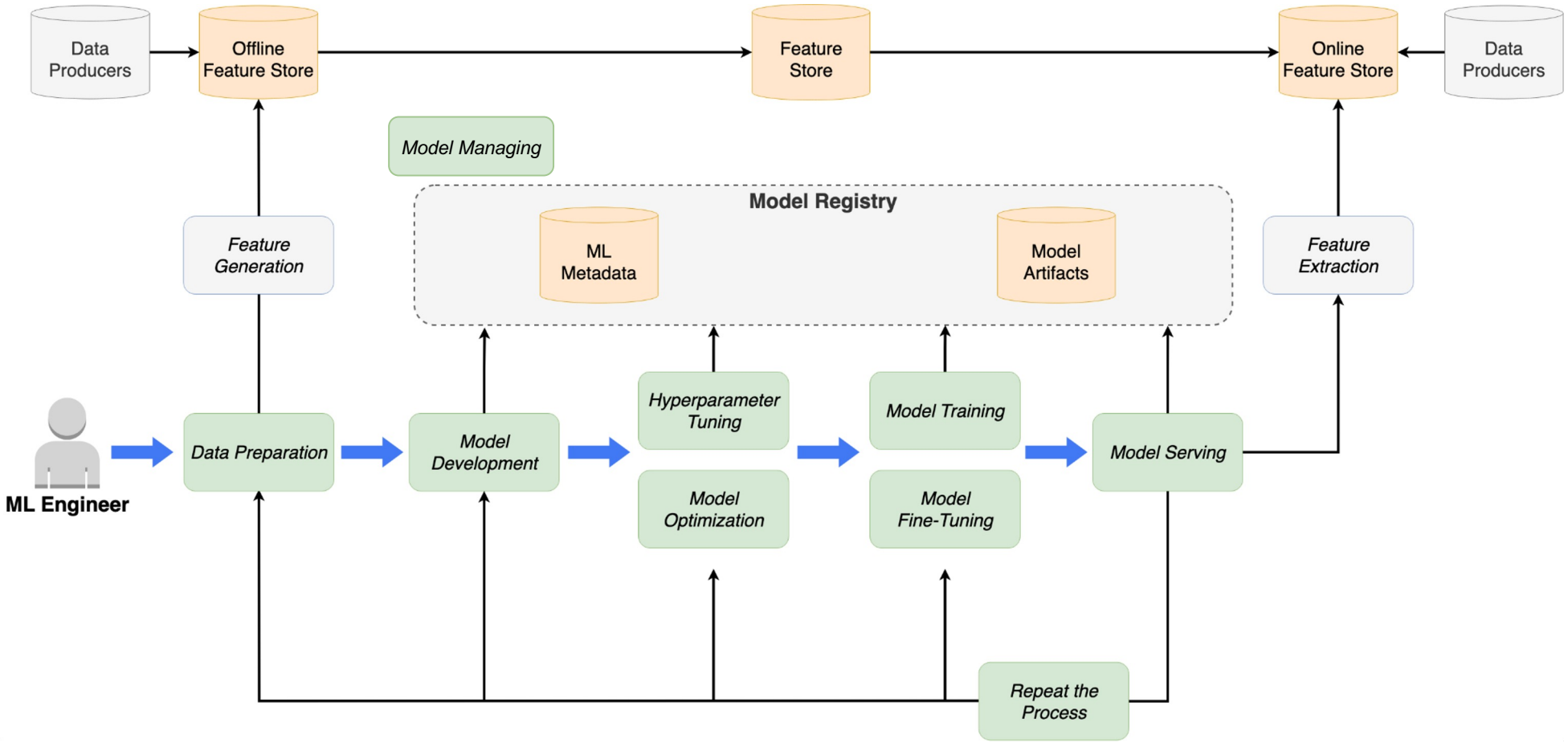
- Kubernetes 에서 AI/ML을 쉽게 이식, 확장 가능
- 새로운 ML 서비스를 만드는 것이 아닌, 검증된 오픈소스 ML 툴들을 k8s 위에 쉽게 배포하고 운영하는 것

Kubeflow Ecosystem



이미지 출처 : http://www.kubeflow.org/

ML Lifecycle



이미지 출처 : http://www.kubeflow.org/

Kubeflow 프로젝트

Spark Operator

분산 ETL, 대규모 데이터 전처리/집계

Data Preparation

Notebook

EDA/시각화, 피쳐 엔지니어링
모델 개발, 실험 코드 작성

Data Preparation *Model Development*

Katib

하이퍼 파리미터, 모델 최적화
(HPO, NAS, Early Stopping)

Model Optimization *Hyperparameter Tuning*

Kubeflow TRAINER

데이터 및 모델 병렬, 분산 학습

Model Training *Model Fine-Tuning*

KServe

온라인, 배치 추론, 오토 스케일,
Canary 배포

Model Serving

Kubeflow MODEL REGISTRY

모델 버저닝, 스테이징
메타데이터 관리

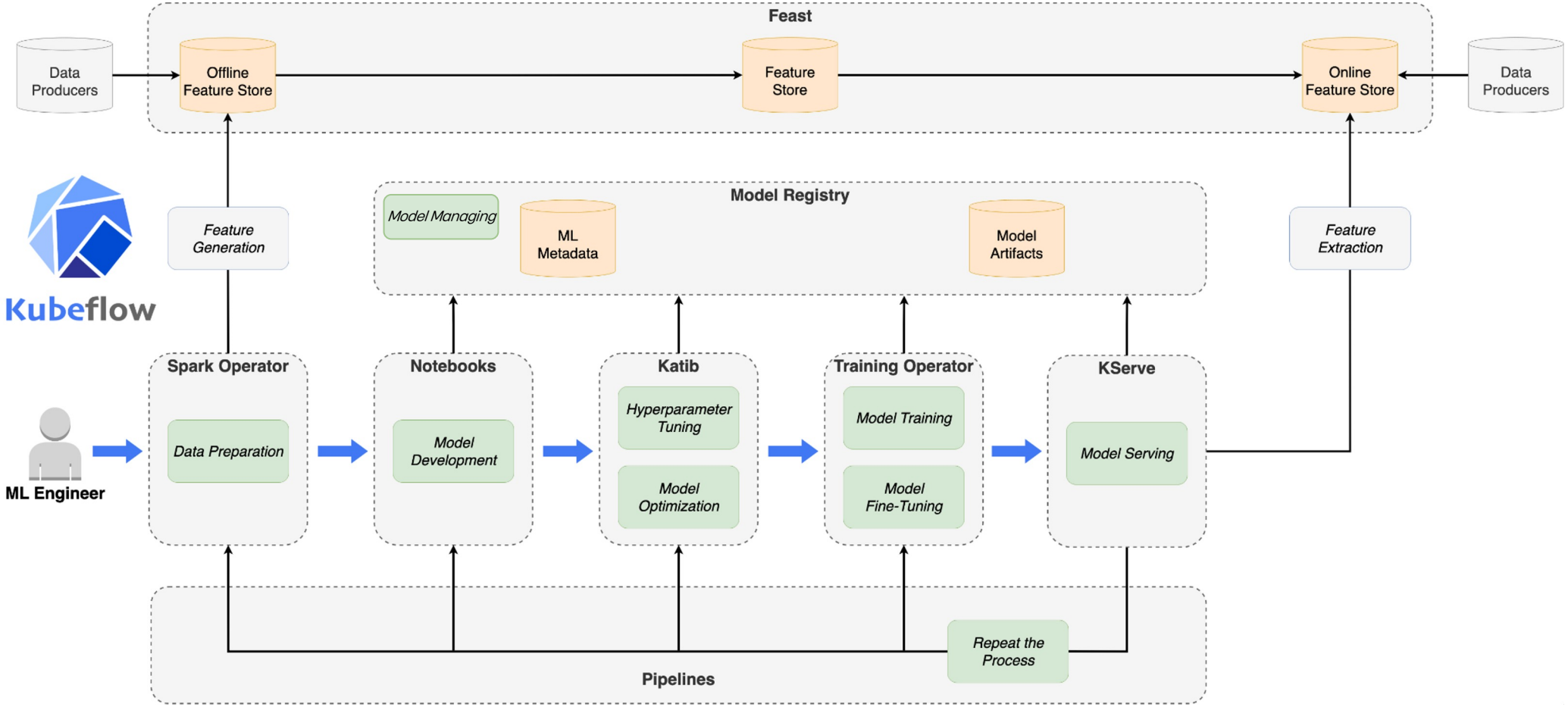
Model Managing

PIPELINE

파이프라이닝, DAG 오케스트레이션,
스케줄링

Repeat the Process

Kubeflow Projects in the ML Lifecycle



이미지 출처 : http://www.kubeflow.org/

Kubeflow Platform

AI Reference Platform

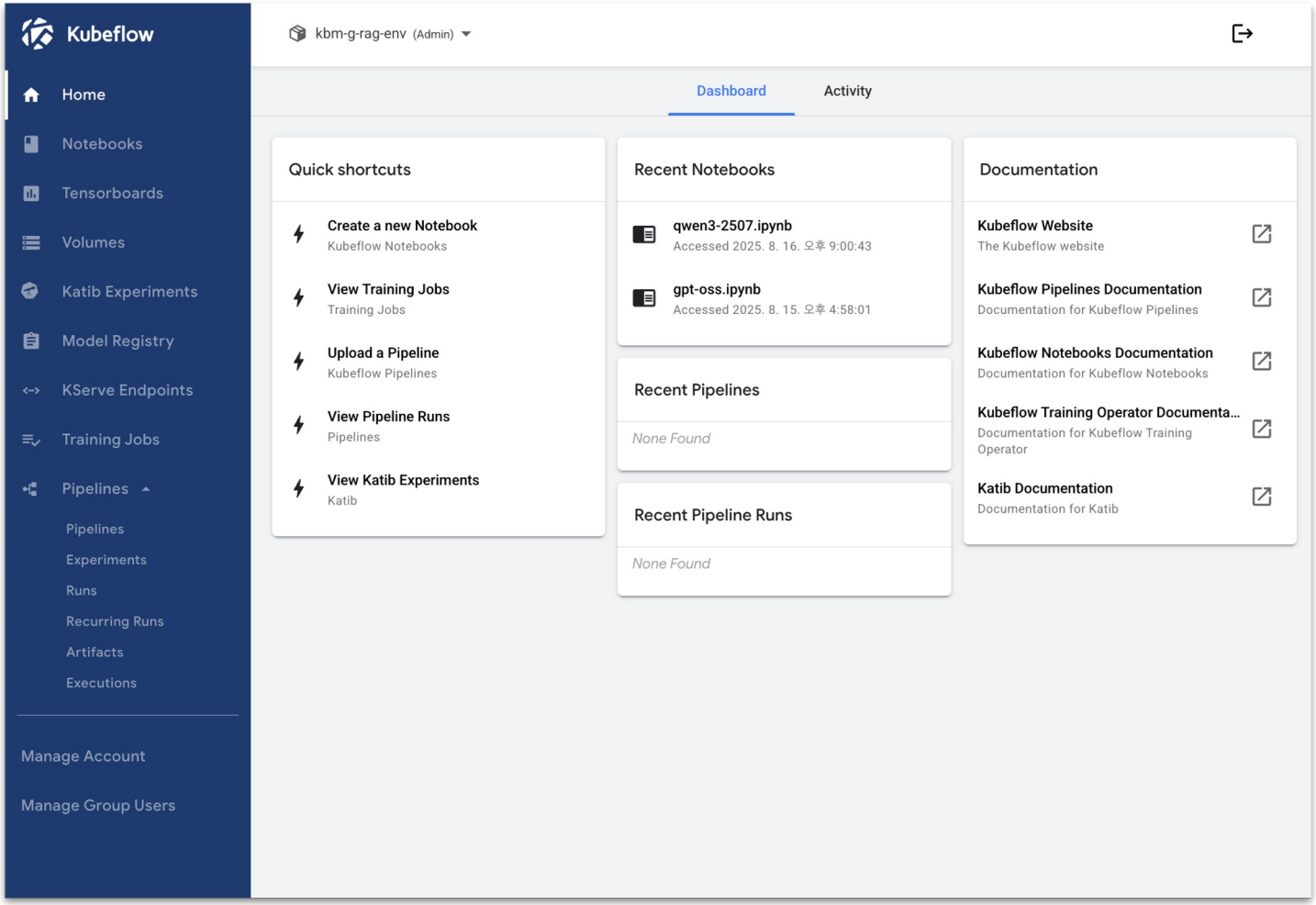
Central Dashboard

Profile Controller

+

Kubeflow Projects

Kubeflow Platform Dashboard



kakaocloud

2. ML Task : 로드 밸런서 트래픽 예측

시나리오 소개

Step 1. 데이터 탐색 및 모델 개발

Step 2. 하이퍼 파라미터 튜닝

Step 3. 모델 서빙

Step 4. 파이프라인 구축

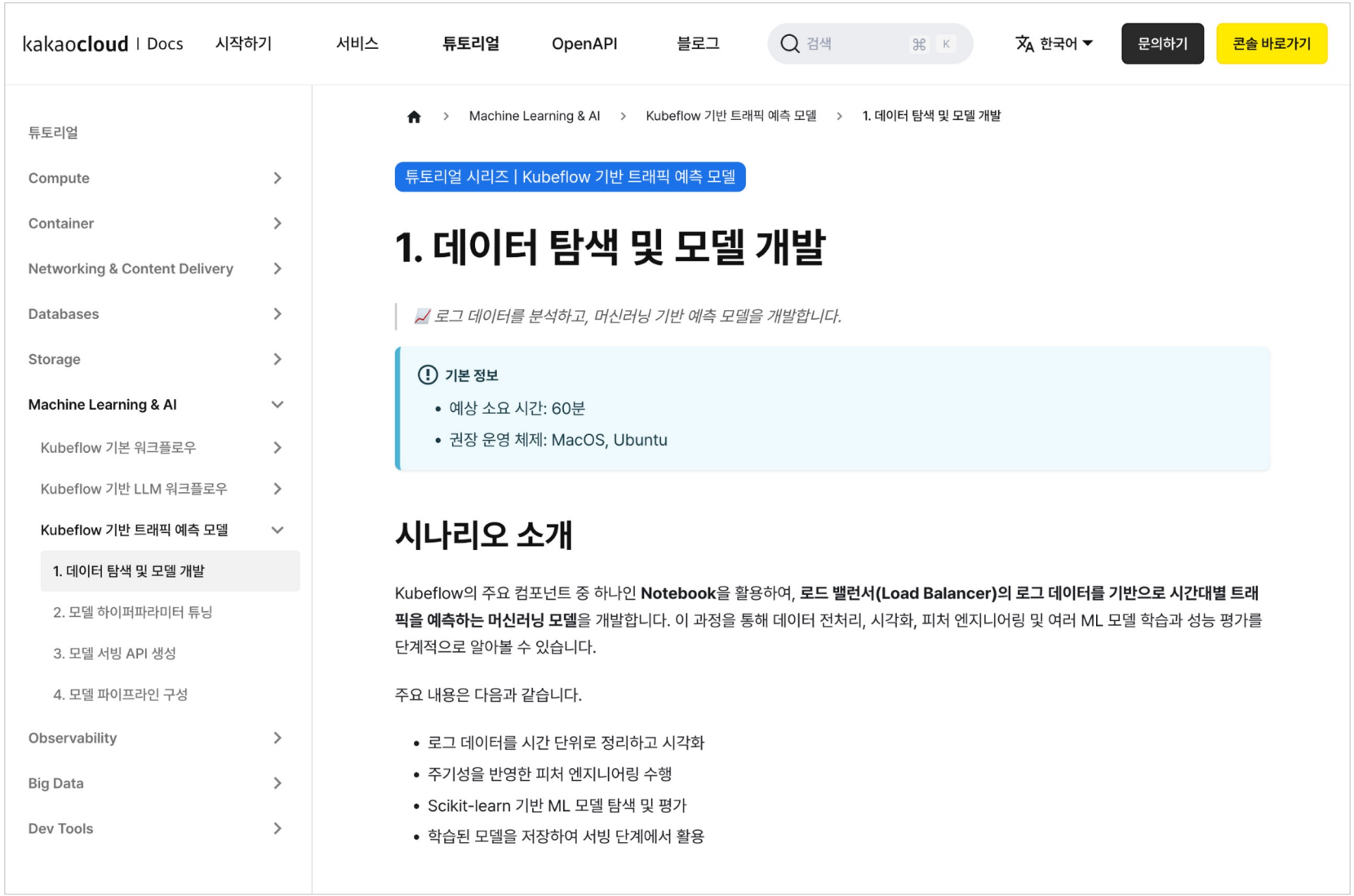
시나리오 소개

트래픽 예측

- 로드 밸런서는 30분 단위로 엑세스 로그를 생성
- 로그 데이터를 활용, 시간대 별 트래픽을 예측 하는 모델 개발

ML 워크플로우

시나리오 소개



튜토리얼 문서 : https://docs.kakaocloud.com/tutorial

kakaocloud

ML Task
로드 밸런서 트래픽 예측

시나리오 소개

Step 1. 데이터 탐색 및 모델 개발 Notebook

Step 2. 하이퍼 파라미터 튜닝

Step 3. 모델 서빙

Step 4. 파이프라인 구축

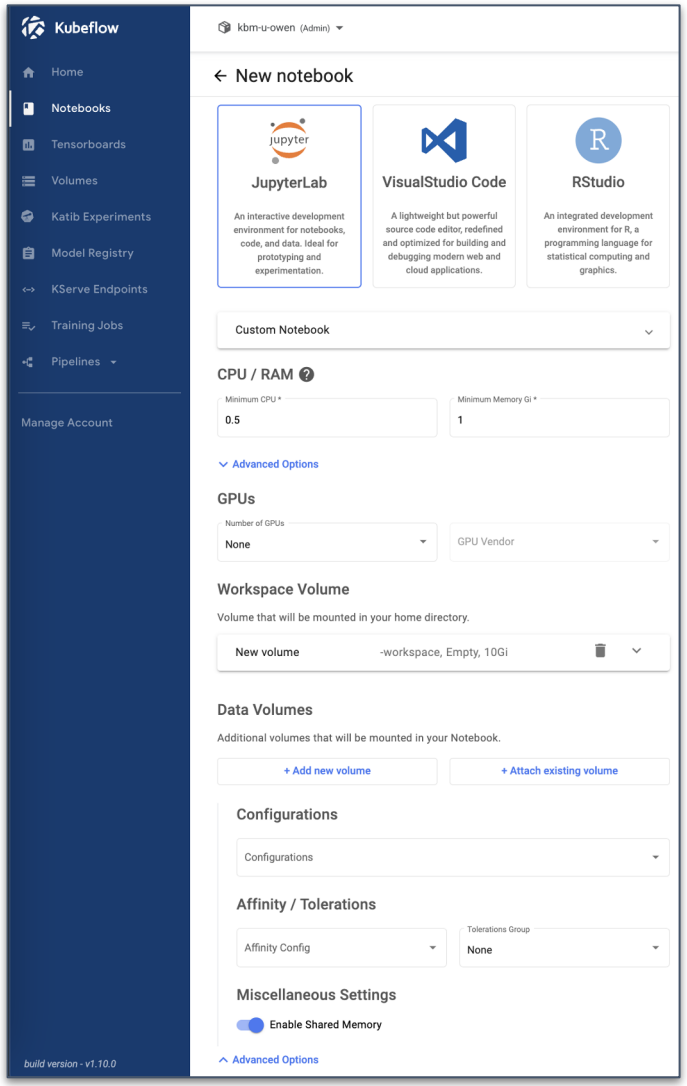
Notebook

쿠버네티스 클러스터 내에서 웹 기반 개발 환경을 제공하는 프로젝트

주요 기능

- JupyterLab, Rstudio, VSCode 지원
- 사용자가 직접 노트북 컨테이너 생성
- 필수 패키지가 설치된 이미지 제공
- Profile, RBAC 권한 제어로 관리, 쉽게 격리, 공유

Notebook 생성

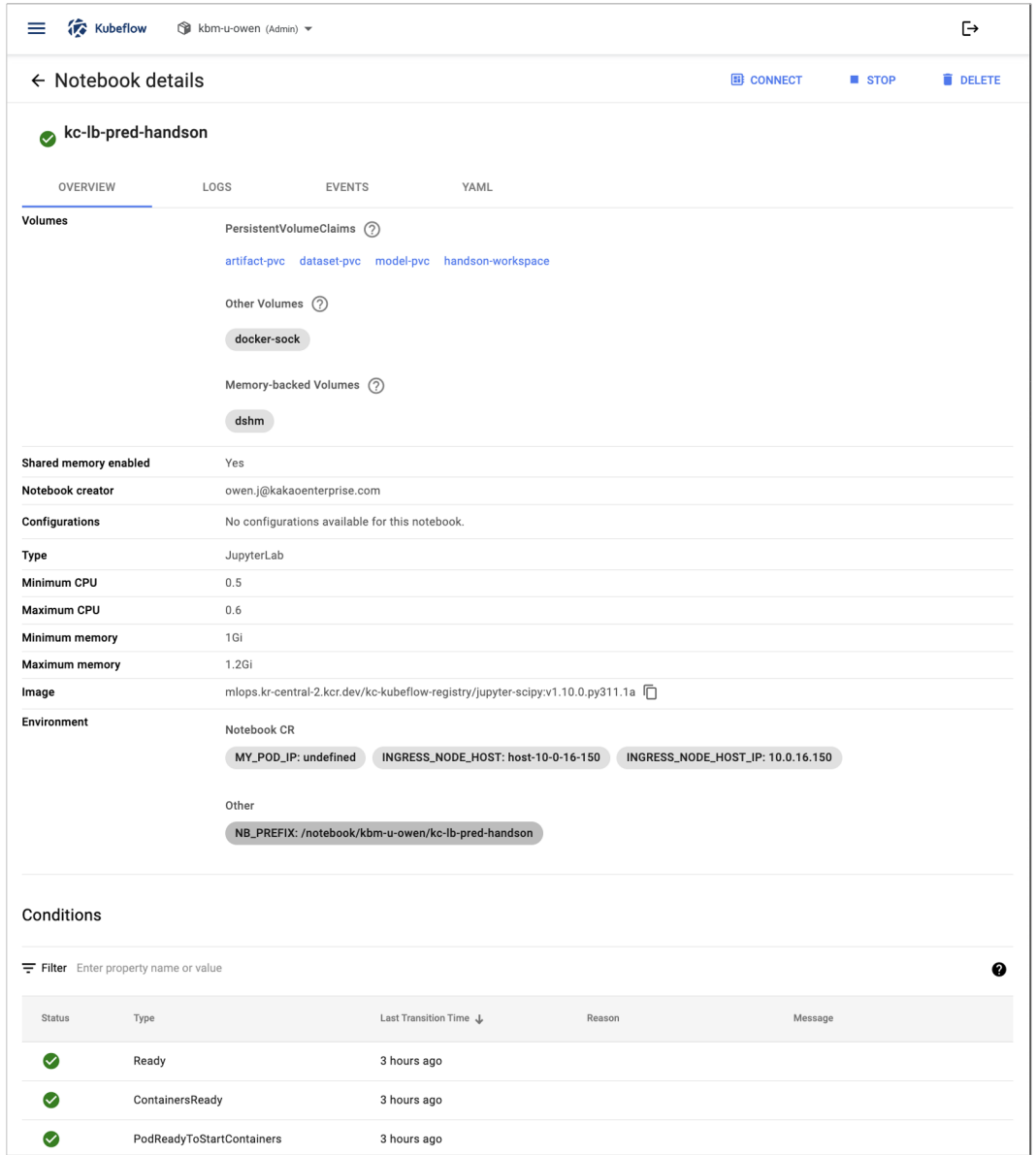


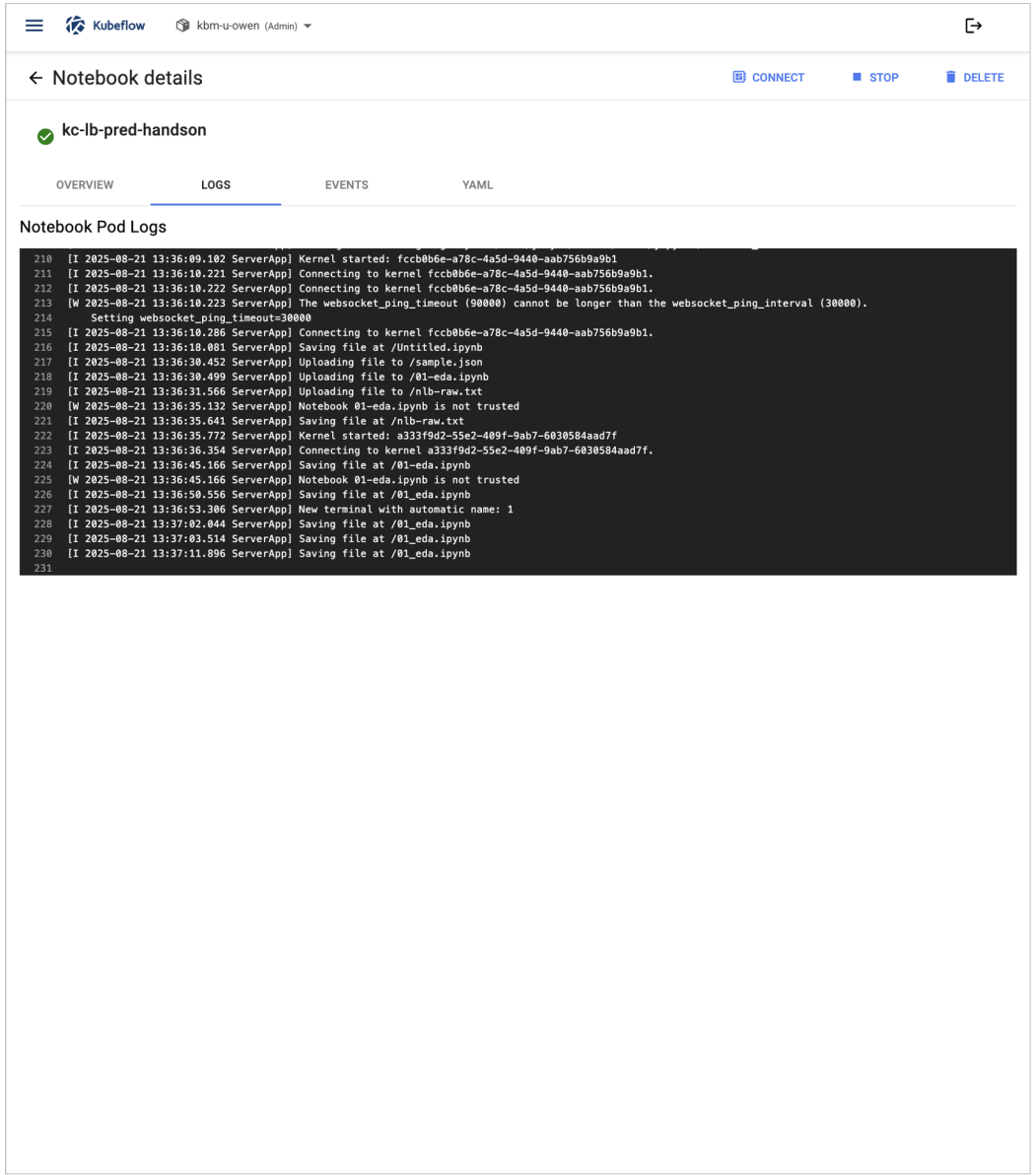
환경 & 이미지 설정

리소스 설정

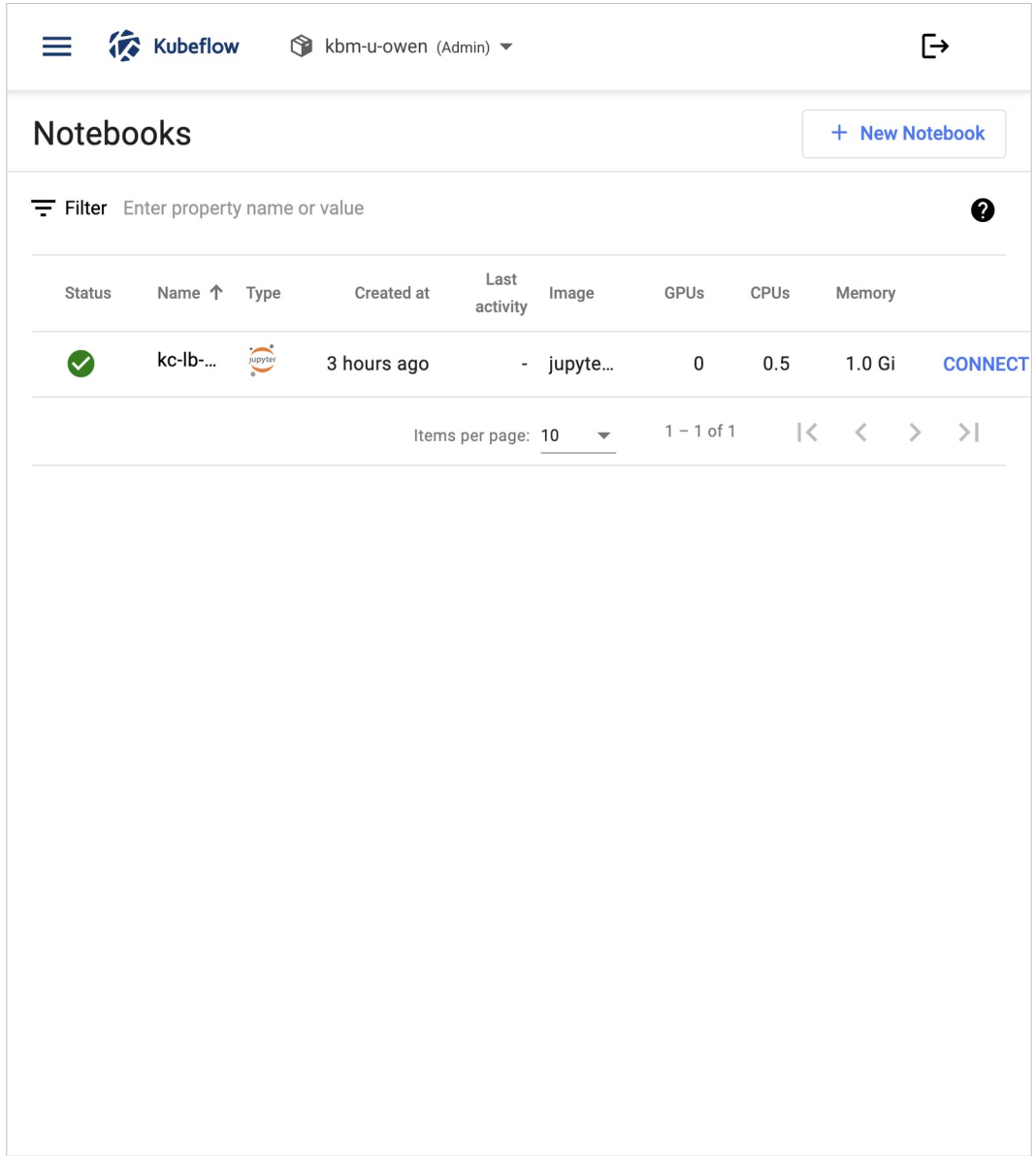
Pod, Node 관련 설정

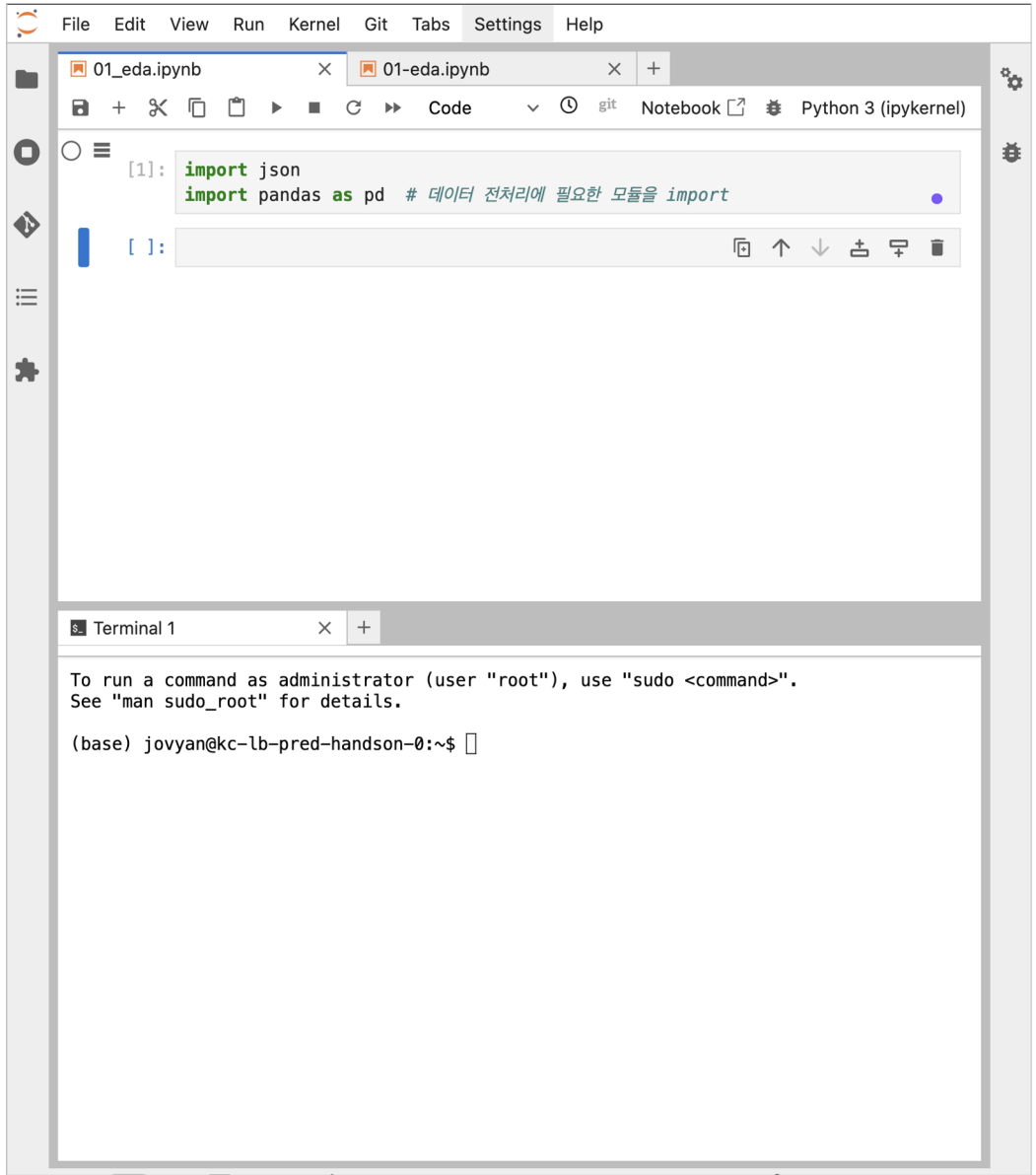
Notebook 관리


Kubeflow
kbm-u-owen (Admin)
Notebook details
CONNECT
STOP
DELETE
kc-lb-pred-handson
OVERVIEW
LOGS
EVENTS
YAML
Volumes
PersistentVolumeClaims
artifact-pvc
dataset-pvc
model-pvc
handson-workspace
Other Volumes
docker-sock
Memory-backed Volumes
dshm
Shared memory enabled
Yes
Notebook creator
owen.j@kakaoenterprise.com
Configurations
No configurations available for this notebook.
Type
JupyterLab
Minimum CPU
0.5
Maximum CPU
0.6
Minimum memory
1Gi
Maximum memory
1.2Gi
Image
mlops.kr-central-2.kcr.dev/kc-kubeflow-registry/jupyter-scipy:v1.10.0.py311.1a
Environment
Notebook CR
MY_POD_IP: undefined
INGRESS_NODE_HOST: host-10-0-16-150
INGRESS_NODE_HOST_IP: 10.0.16.150
Other
NB_PREFIX: /notebook/kbm-u-owen/kc-lb-pred-handson
Conditions
Filter
Enter property name or value
Status
Type
Last Transition Time
Reason
Message
Ready
3 hours ago
ContainersReady
3 hours ago
PodReadyToStartContainers
3 hours ago


Kubeflow
kbm-u-owen (Admin)
Notebook details
CONNECT
STOP
DELETE
kc-lb-pred-handson
OVERVIEW
LOGS
EVENTS
YAML
Notebook Pod Logs
[I 2025-08-21 13:36:09.102 ServerApp] Kernel started: fccb0b6e-a78c-4a5d-9440-aab756b9a9b1
[I 2025-08-21 13:36:10.221 ServerApp] Connecting to kernel fccb0b6e-a78c-4a5d-9440-aab756b9a9b1.
[I 2025-08-21 13:36:10.222 ServerApp] Connecting to kernel fccb0b6e-a78c-4a5d-9440-aab756b9a9b1.
[W 2025-08-21 13:36:10.223 ServerApp] The websocket_ping_timeout (90000) cannot be longer than the websocket_ping_interval (30000).
Setting websocket_ping_timeout=30000
[I 2025-08-21 13:36:10.286 ServerApp] Connecting to kernel fccb0b6e-a78c-4a5d-9440-aab756b9a9b1.
[I 2025-08-21 13:36:18.081 ServerApp] Saving file at /Untitled.ipynb
[I 2025-08-21 13:36:30.452 ServerApp] Uploading file to /sample.json
[I 2025-08-21 13:36:30.499 ServerApp] Uploading file to /01-eda.ipynb
[I 2025-08-21 13:36:31.566 ServerApp] Uploading file to /nlb-raw.txt
[W 2025-08-21 13:36:35.132 ServerApp] Notebook 01-eda.ipynb is not trusted
[I 2025-08-21 13:36:35.641 ServerApp] Saving file at /nlb-raw.txt
[I 2025-08-21 13:36:35.772 ServerApp] Kernel started: a333f9d2-55e2-409f-9ab7-6030584aad7f
[I 2025-08-21 13:36:36.354 ServerApp] Connecting to kernel a333f9d2-55e2-409f-9ab7-6030584aad7f.
[I 2025-08-21 13:36:45.166 ServerApp] Saving file at /01-eda.ipynb
[W 2025-08-21 13:36:45.166 ServerApp] Notebook 01-eda.ipynb is not trusted
[I 2025-08-21 13:36:50.556 ServerApp] Saving file at /01_eda.ipynb
[I 2025-08-21 13:36:53.306 ServerApp] New terminal with automatic name: 1
[I 2025-08-21 13:37:02.044 ServerApp] Saving file at /01_eda.ipynb
[I 2025-08-21 13:37:03.514 ServerApp] Saving file at /01_eda.ipynb
[I 2025-08-21 13:37:11.896 ServerApp] Saving file at /01_eda.ipynb

Notebook 접속


Kubeflow
kbm-u-owen (Admin)
Notebooks
New Notebook
Filter
Enter property name or value
Status
Name
Type
Created at
Last activity
Image
GPUs
CPUs
Memory
kc-lb-...
3 hours ago
-
jupyte...
0
0.5
1.0 Gi
CONNECT
Items per page: 10
1 – 1 of 1


File
Edit
View
Run
Kernel
Git
Tabs
Settings
Help
01_eda.ipynb
01-eda.ipynb
Code
Notebook
Python 3 (ipykernel)
[1]:
import json
import pandas as pd # 데이터 전처리에 필요한 모듈을 import
[ ]:
Terminal 1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
(base) jovyan@kc-lb-pred-handson-0:~$

데이터 전처리


```python
import json
import pandas as pd  # 데이터 전처리에 필요한 모듈을 import

file_path = '/home/jovyan/nlb-raw.txt'  # NLB 파일 경로
raw_data = []
# load nlb dataset
with open(file_path) as f:
    for line in f:  # 각 line 에 있는 JSON형식 텍스트를 읽음
        # load json data
        data = json.loads(line)  # JSON 형식 문자열을 python dict 타입으로 변환
        # append to raw_data
        raw_data.append(data)

raw_df = pd.json_normalize(raw_data)  # key:value 형식의 tabular 데이터로 변환, pandas의 Dataframe 객체 형식
```

```python
# 시간을 30분 간격으로 변경
log_time_sr = pd.to_datetime(raw_df['time'],format='%Y/%m/%d %H:%M:%S:%f').dt.floor('30min')

# 30분 간격으로 로그의 수를 세고, 각 시간 별 로그 수를 dictionary 타입으로 저장
log_count_dict = log_time_sr.dt.floor('30min').value_counts(dropna=False).to_dict()

# 데이터셋 시간 범위를 생성 (30분 간격)
time_range = pd.date_range(start='2024-04-01', end='2024-05-01', freq='30min')

# 시간과 로그 수를 칼럼으로 갖는 데이터프레임 생성
df = pd.DataFrame({'datetime': time_range})
df['count'] = df['datetime'].apply(lambda x: log_count_dict.get(x, 0))
```

JSON Line

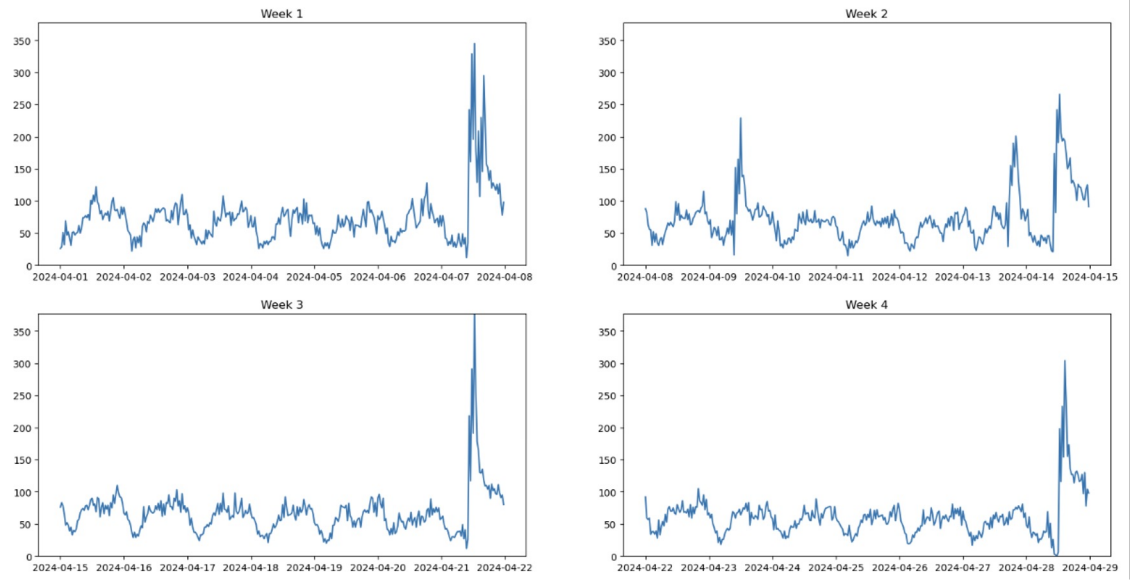

```
{"project_id":"ZWRwLXRlYW0K","time":"2024\/04\/01
00:03:19:000000","lb_id":"ZGF0YS1jb3JlLXBhcnQK","listener_id":"b3dlbi5qCg==","client_port":"172.1.2.3:4321","de
stination_port":"10.1.2.3:1234","tls_cipher":"-","tls_protocol_version":"-"}
{"project_id":"ZWRwLXRlYW0K","time":"2024\/04\/01
00:04:25:000000","lb_id":"ZGF0YS1jb3JlLXBhcnQK","listener_id":"b3dlbi5qCg==","client_port":"172.1.2.3:4321","de
stination_port":"10.1.2.3:1234","tls_cipher":"-","tls_protocol_version":"-"}
{"project_id":"ZWRwLXRlYW0K","time":"2024\/04\/01
00:05:07:000000","lb_id":"ZGF0YS1jb3JlLXBhcnQK","listener_id":"b3dlbi5qCg==","client_port":"172.1.2.3:4321","de
stination_port":"10.1.2.3:1234","tls_cipher":"-","tls_protocol_version":"-"}
{"project_id":"ZWRwLXRlYW0K","time":"2024\/04\/01
00:06:39:000000","lb_id":"ZGF0YS1jb3JlLXBhcnQK","listener_id":"b3dlbi5qCg==","client_port":"172.1.2.3:4321","de
stination_port":"10.1.2.3:1234","tls_cipher":"-","tls_protocol_version":"-"}
{"project_id":"ZWRwLXRlYW0K","time":"2024\/04\/01
00:07:15:000000","lb_id":"ZGF0YS1jb3JlLXBhcnQK","listener_id":"b3dlbi5qCg==","client_port":"172.1.2.3:4321","de
stination_port":"10.1.2.3:1234","tls_cipher":"-","tls_protocol_version":"-"}
{"project_id":"ZWRwLXRlYW0K","time":"2024\/04\/01
```


DataFrame

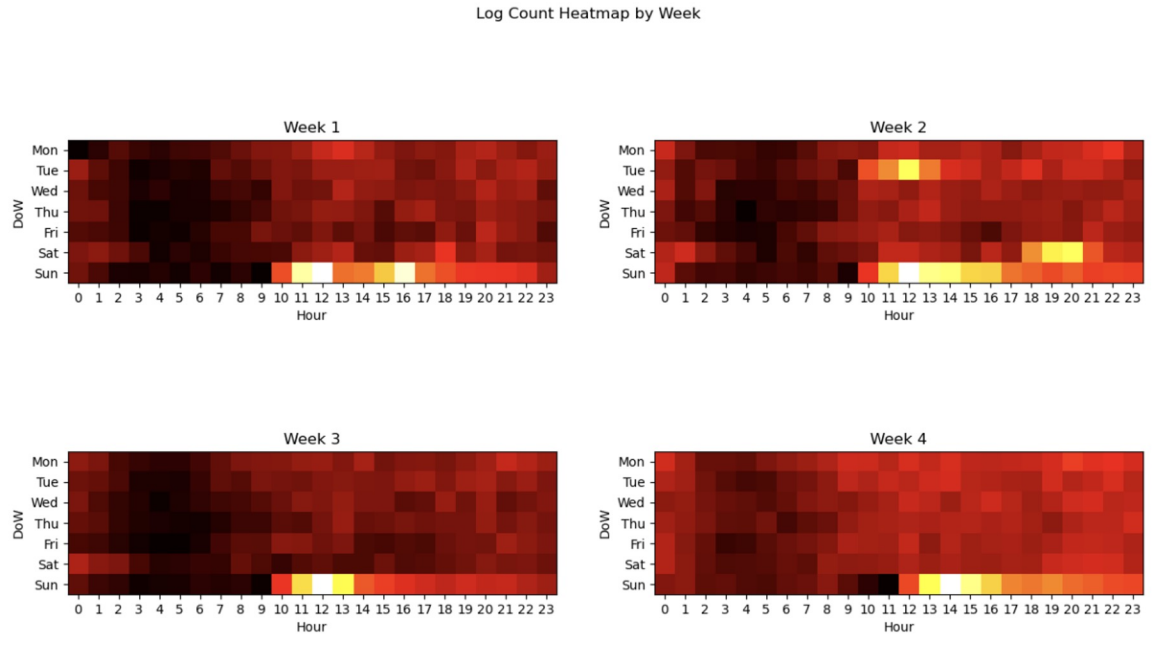
	datetime	count
0	2024-04-01 00:00:00	26
1	2024-04-01 00:30:00	29
2	2024-04-01 01:00:00	51
3	2024-04-01 01:30:00	32
4	2024-04-01 02:00:00	69

데이터 분석


```
fig, axs = plt.subplots(2, 2, figsize=(20, 10))
fig.suptitle('Log Count by Week - 4 weeks')

for i in range(4):
    axs[i//2][i%2].set_ylim(0, max(df['count']))
    axs[i//2][i%2].plot(df[df['week'] == 14+i]['datetime'], df[df['week'] == 14+i]['count'])
    axs[i//2][i%2].set_title(f'Week {1+i}')

plt.show()
```




```
fig, axs = plt.subplots(2, 2, figsize=(15, 8))
fig.suptitle('Log Count Heatmap by Week')

for i in range(4):
    week = i + 14
    df_grouped = df[df['week'] == week].groupby(["hour", "dow"])["count"].sum().reset_index()
    df_heatmap = df_grouped.pivot(index="dow", columns="hour", values="count")
    axs[i//2][i%2].set_title(f'Week {i+1}')
    axs[i//2][i%2].imshow(df_heatmap, cmap='hot', interpolation='nearest')
    axs[i//2][i%2].set_xticks(range(24))
    axs[i//2][i%2].set_xticklabels(range(24))
    axs[i//2][i%2].set_yticks(range(7))
    axs[i//2][i%2].set_yticklabels(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])
    axs[i//2][i%2].set_xlabel('Hour')
    axs[i//2][i%2].set_ylabel('DoW')

plt.show()
```



피처 엔지니어링

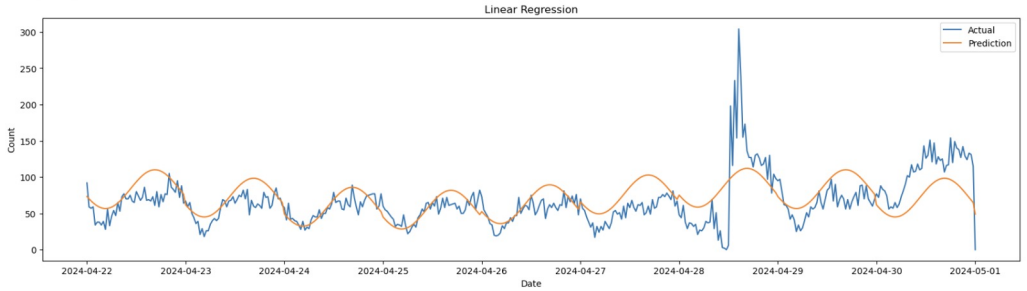

```python
import numpy as np

time_sr = df['datetime'].apply(lambda x: x.hour * 2 + x.minute // 30)
dow_sr = df['datetime'].dt.dayofweek

dataset = pd.DataFrame()
dataset['datetime'] = df['datetime']  # 이후 작업 편의를 위해 설정

# 시간 관련 feature x1, x2
dataset['x1'] = np.sin(2*np.pi*time_sr/48)
dataset['x2'] = np.cos(2*np.pi*time_sr/48)
# 요일 관련 feature x3, x4
dataset['x3'] = np.sin(2*np.pi*dow_sr/7)
dataset['x4'] = np.cos(2*np.pi*dow_sr/7)

# 예측하고자 하는 대상, label
dataset['y'] = df['count']

# 데이터셋 저장 (이후 실습에서 사용)
dataset.to_csv('/home/jovyan/dataset/nlb-sample.csv', index=False)

dataset.head(5)
```

	datetime	x1	x2	x3	x4	y
0	2024-04-01 00:00:00	0.000000	1.000000	0.0	1.0	26
1	2024-04-01 00:30:00	0.130526	0.991445	0.0	1.0	29
2	2024-04-01 01:00:00	0.258819	0.965926	0.0	1.0	51
3	2024-04-01 01:30:00	0.382683	0.923880	0.0	1.0	32
4	2024-04-01 02:00:00	0.500000	0.866025	0.0	1.0	69

모델 개발

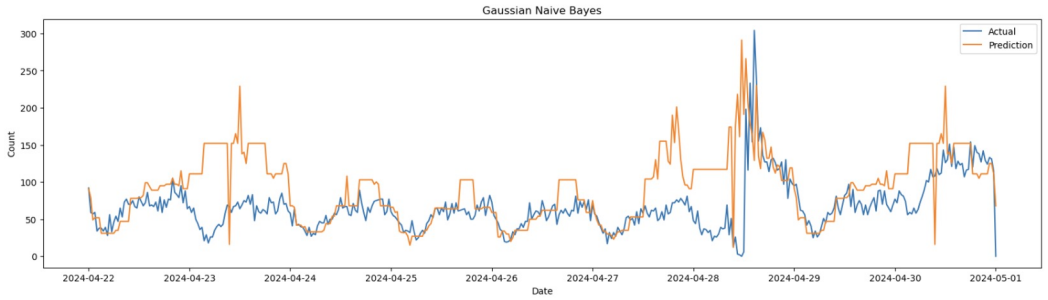

```python
from sklearn.linear_model import LinearRegression

params = {
    # 모델 문서를 참고하여 파라미터 설정
}
model = LinearRegression(**params)
model.fit(train_df[['x1', 'x2', 'x3', 'x4']], train_df['y'])
predictions = model.predict(test_df[['x1', 'x2', 'x3', 'x4']])
print_metrics(test_df['y'], predictions)
draw_predictions(test_df=test_df,
                 predictions=predictions,
                 model_name='Linear Regression')
```

```
MSE: 859.1780
MAE: 21.4167
R2: 0.2425
```



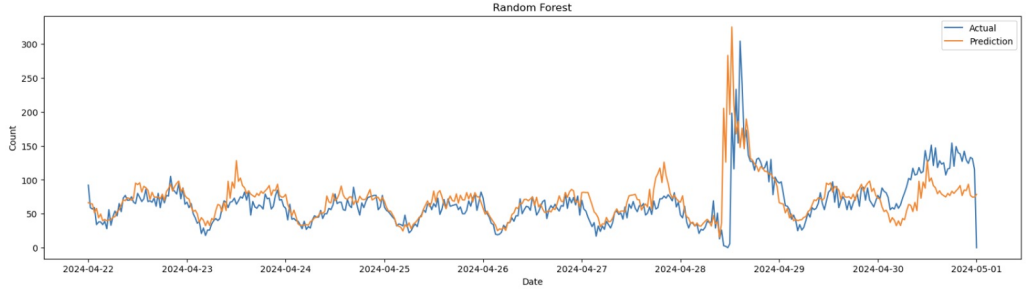

```python
from sklearn.naive_bayes import GaussianNB
params = {
    # 모델 문서를 참고하여 파라미터 설정
}
model = GaussianNB(**params)
model.fit(train_df[['x1', 'x2', 'x3', 'x4']], train_df['y'])
predictions = model.predict(test_df[['x1', 'x2', 'x3', 'x4']])
print_metrics(test_df['y'], predictions)
draw_predictions(test_df=test_df,
                 predictions=predictions,
                 model_name='Gaussian Naive Bayes')
```

```
MSE: 2506.4180
MAE: 32.8060
R2: -1.2099
```



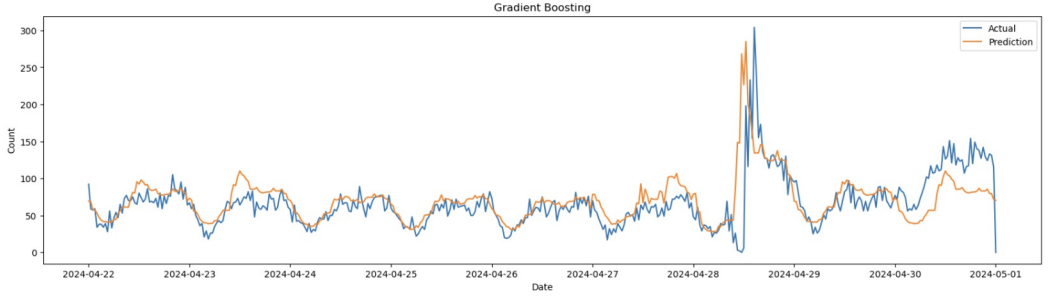

```python
from sklearn.ensemble import RandomForestRegressor

params = {
    # 모델 문서를 참고하여 파라미터 설정
}
model = RandomForestRegressor(**params)
model.fit(train_df[['x1', 'x2', 'x3', 'x4']], train_df['y'])
predictions = model.predict(test_df[['x1', 'x2', 'x3', 'x4']])
print_metrics(test_df['y'], predictions)
draw_predictions(test_df=test_df,
                 predictions=predictions,
                 model_name='Random Forest')
```

```
MSE: 914.3245
MAE: 17.0766
R2: 0.1938
```

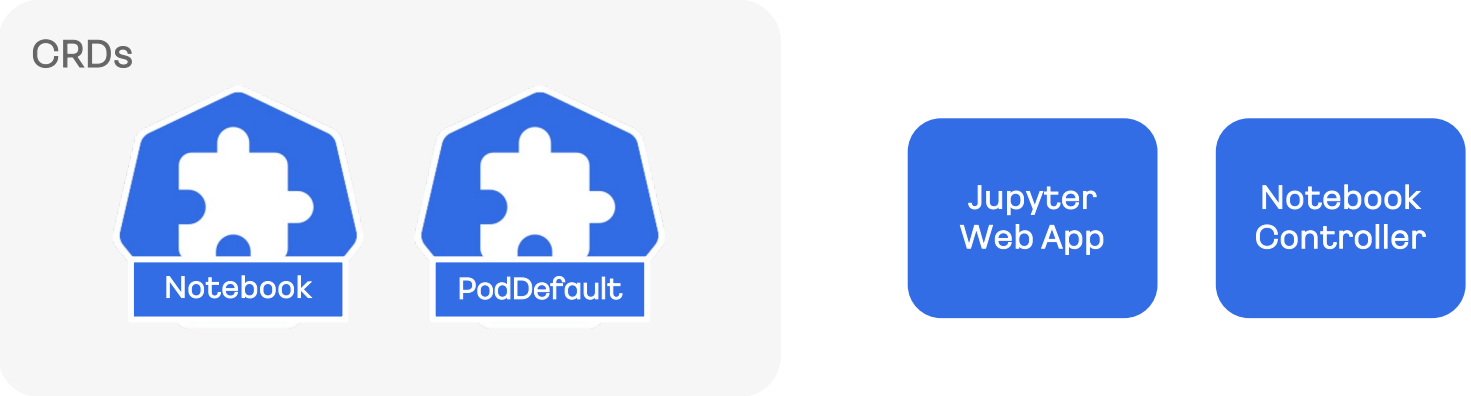



```python
from sklearn.ensemble import GradientBoostingRegressor

params = {
    # 모델 문서를 참고하여 파라미터 설정
}
model = GradientBoostingRegressor(**params)
model.fit(train_df[['x1', 'x2', 'x3', 'x4']], train_df['y'])
predictions = model.predict(test_df[['x1', 'x2', 'x3', 'x4']])
print_metrics(test_df['y'], predictions)
draw_predictions(test_df=test_df,
                 predictions=predictions,
                 model_name='Gradient Boosting')
```

```
MSE: 876.9636
MAE: 16.9527
R2: 0.2268
```

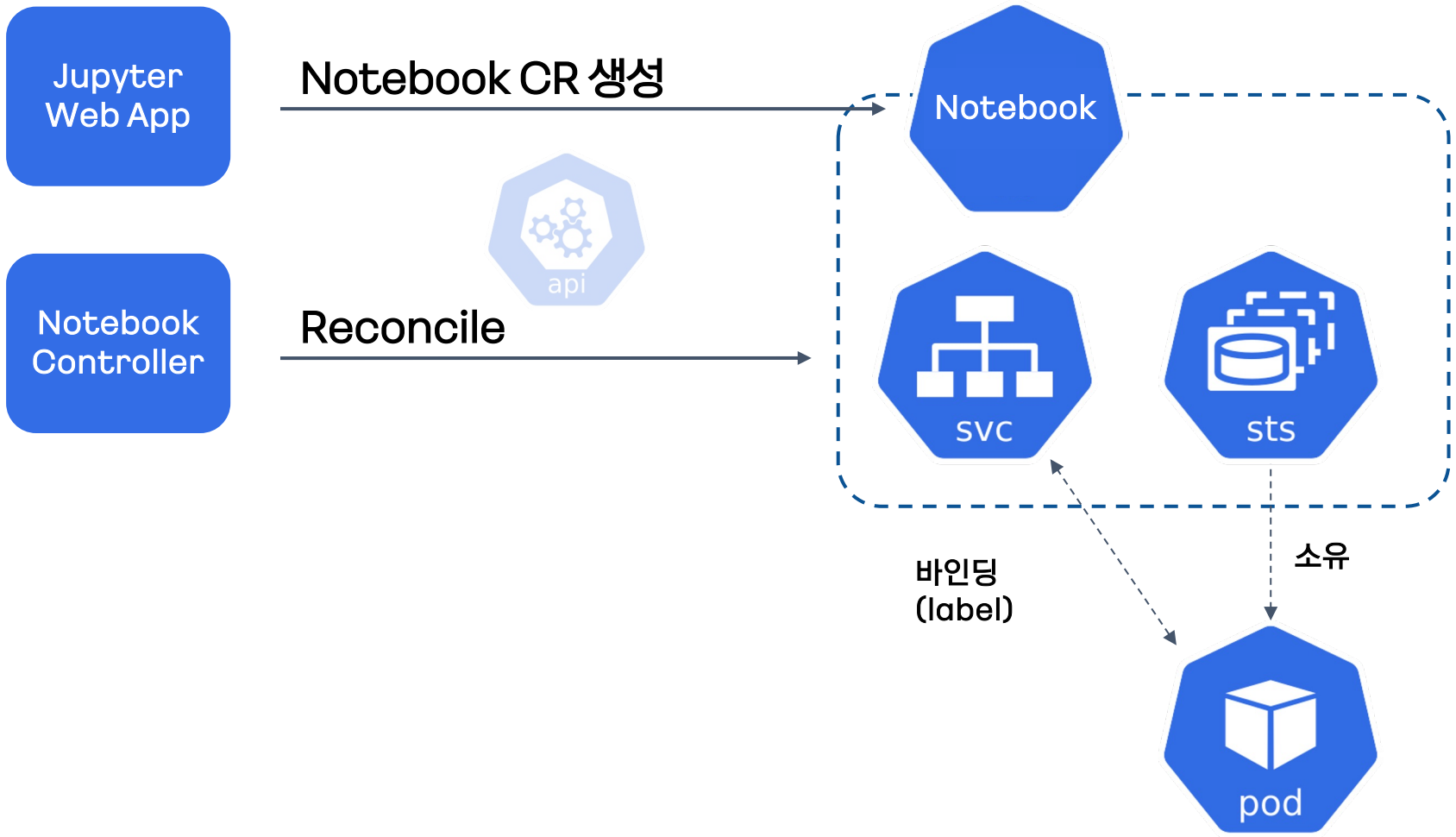


구성 요소

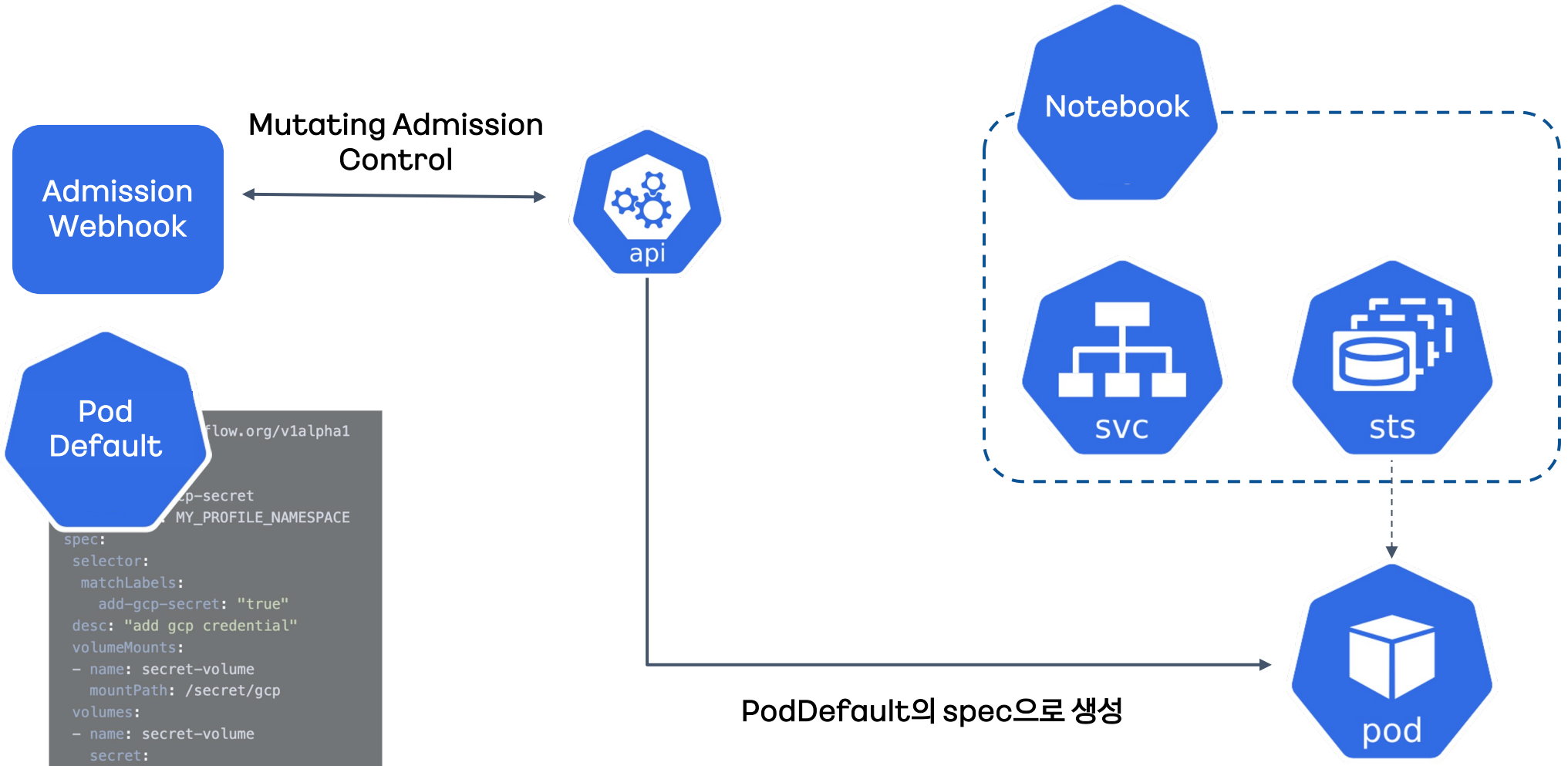


이름	설명
Notebook	노트북 서버의 원하는 spec과 status를 정의하는 CRD
PodDefault	Pod 생성 시 볼륨/환경 변수/사이드 카 등을 mutation으로 주입하기 위한 CRD
Jupyter Web App	대시보드에서 Notebook CR을 생성/수정/삭제 하는 프론트엔드 앱
Notebook Controller	Notebook CR을 감시하고 실제 리소스로 리컨실

동작방식 : Notebook CR



동작방식: Pod Default CR



kakaocloud

ML Task
로드 밸런서 트래픽 예측

시나리오 소개

Step 1. 데이터 탐색 및 모델 개발

Step 2. 하이퍼 파라미터 튜닝 Katib

Step 3. 모델 서빙

Step 4. 파이프라인 구축

Katib

자동화된 기계학습 (AutoML)을 위한 프로젝트

주요 기능

- AutoML : Hyperparameter Tuning, Early Stopping, Neural Architecture Search
- 다양한 최적화 알고리즘 지원
- ML 학습 프레임워크 종류에 상관 없이 사용 가능
- Multi-node & GPU 오케스트레이션

Katib

하이퍼파라미터 튜닝 ?

모델의 예측 정확도를 최대화하기 위해 하이퍼파라미터* 값들을 최적화하는 과정

- 하이퍼파라미터란, 모델의 학습 과정을 제어하는 설정 값. 모델 구조 관련 값(신경망 깊이, 너비)이나 학습률, 반복 횟수 등에 해당
- Katib는 다양한 최적화 알고리즘을 사용하여 모델의 하이퍼파라미터를 자동으로 튜닝

Katib에서 지원하는 하이퍼파라미터 튜닝 알고리즘

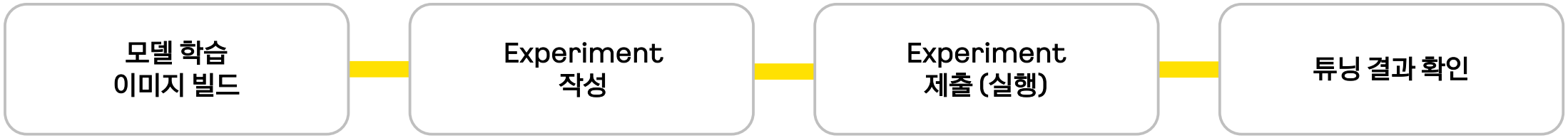
알고리즘	설명
Grid Search	가능한 모든 하이퍼파라미터 조합을 체계적으로 탐색하여 최적의 조합을 찾습니다.
Random Search	하이퍼파라미터의 범위 내에서 무작위로 조합을 선택하여 탐색 효율성을 높입니다.
Bayesian Optimization	이전의 결과를 바탕으로 하이퍼파라미터의 가장 유망한 영역을 예측하고 탐색합니다.
Hyperband	리소스 할당을 조절하며 다양한 하이퍼파라미터 설정을 빠르게 평가합니다.
Tree of Parzen Estimators (TPE)	확률 모델을 사용하여 더 좋은 하이퍼파라미터 값을 예측합니다.
Multivariate TPE	변수 간의 상호작용을 고려하여 TPE 알고리즘을 확장한 형태입니다.
CMA-ES (Covariance Matrix Adaptation Evolution Strategy)	대규모 문제에 적합한 진화적 전략을 사용하여 하이퍼파라미터를 최적화합니다.
Sobol Quasirandom Sequence	저차원의 균등 분포를 가지는 시퀀스를 사용하여 효율적으로 공간을 탐색합니다.
Population Based Training	모델의 세대를 거듭하면서 하이퍼파라미터를 지속적으로 조정합니다.

Katib

컴포넌트

컴포넌트 명	설명
Experiment	AutoML을 위한 실험의 설정과 목표를 정의
Suggestion	Experiment에서 정의된 정보를 바탕으로 하이퍼파라미터의 새로운 세트를 제안함
Trial	Suggestion으로부터 받은 하이퍼파라미터 세트를 실제 ML 학습 작업에 적용하여 평가하는 과정, 각 Trial은 하나의 학습 작업을 의미
Worker	각 Trial을 실행하는 물리적 또는 가상의 실행 단위

하이퍼파라미터 튜닝 프로세스



모델 학습 이미지 빌드

모델 학습 코드 작성

tune.py

```python
import argparse
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score

parser = argparse.ArgumentParser()
parser.add_argument('--n_estimators', type=int, default=100)
parser.add_argument('--learning_rate', type=float, default=0.1)
parser.add_argument('--max_depth', type=int, default=3)
parser.add_argument('--subsample', type=float, default=1.0)
parser.add_argument('--data_version', type=str, default='sample')
parser.add_argument('--random_state', type=int, default=1234)

if __name__ == "__main__":
    args = parser.parse_args()
    df = pd.read_csv(f'/dataset/nlb-{args.data_version}.csv')  # 전처리된 데이터셋 경로

    model = GradientBoostingRegressor(
        n_estimators=args.n_estimators,
        learning_rate=args.learning_rate,
        max_depth=args.max_depth,
        subsample=args.subsample,
        random_state=args.random_state
    )

    X = df[['x1', 'x2', 'x3', 'x4']]
    y = df['y']

    scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_absolute_err
    mae = -scores.mean()
    print("MAE=%f" % mae)
```

하이퍼파라미터 파싱

모델 설정

Metric 출력

도커 및 종속성 파일 작성

Dockerfile

```
FROM python:3.10.0-slim
WORKDIR /app
COPY requirements.txt /app/requirements.txt
COPY tune.py /app/tune.py
RUN pip install -r requirements.txt
RUN rm requirements.txt
CMD ["python", "/app/tune.py"]
```

requirements.txt

```
pandas==2.2.2
numpy==2.1.1
scikit-learn==1.5.1
scipy==1.14.1
```

Experiment 작성


```yaml
apiVersion: kubeflow.org/v1beta1
kind: Experiment
metadata:
  name: lb-gbr-tune
spec:
  objective:
    type: minimize
    goal: 0.0
    objectiveMetricName: MAE
  metricCollectorSpec:
    collector:
      kind: StdOut
  parallelTrialCount: 2
  maxTrialCount: 10
  maxFailedTrialCount: 3
  algorithm:
    algorithmName: random
  parameters:
    - name: n_estimators
      parameterType: int
      feasibleSpace:
        min: "50"
        max: "200"
    - name: learning_rate
      parameterType: double
      feasibleSpace:
        min: "0.01"
        max: "0.2"
    - name: max_depth
      parameterType: int
      feasibleSpace:
        min: "2"
        max: "10"
    - name: subsample
      parameterType: double
      feasibleSpace:
        min: "0.5"
        max: "1.0"
```

```yaml
trialTemplate:
  primaryContainerName: training-container
  trialParameters:
    - name: n_estimators
      description: "Number of estimators"
      reference: "n_estimators"
    - name: learning_rate
      description: "Learning rate"
      reference: "learning_rate"
    - name: max_depth
      description: "Maximum depth of the trees"
      reference: "max_depth"
    - name: subsample
      description: "Subsample ratio"
      reference: "subsample"
  trialSpec:
    apiVersion: batch/v1
    kind: Job
    spec:
      template:
        metadata:
          annotations:
            sidecar.istio.io/inject: 'false'
        spec:
          containers:
            - name: training-container
              image: {{IMAGE}}
              command:
                - "python"
                - "/app/tune.py"
              args:
                - "--n_estimators"
                - "${trialParameters.n_estimators}"
                - "--learning_rate"
                - "${trialParameters.learning_rate}"
                - "--max_depth"
                - "${trialParameters.max_depth}"
                - "--subsample"
                - "${trialParameters.subsample}"
              resources:
                requests:
                  cpu: '0.5'
                  memory: 1Gi
                limits:
                  cpu: '1'
                  memory: 2Gi
```

Experiment 작성


```yaml
apiVersion: kubeflow.org/v1beta1
kind: Experiment
metadata:
  name: lb-gbr-tune
spec:
  objective:
    type: minimize
    goal: 0.0
    objectiveMetricName: MAE
  metricCollectorSpec:
    collector:
      kind: StdOut
  parallelTrialCount: 2
  maxTrialCount: 10
  maxFailedTrialCount: 3
  algorithm:
    algorithmName: random
  parameters:
    - name: n_estimators
      parameterType: int
      feasibleSpace:
        min: "50"
        max: "200"
    - name: learning_rate
      parameterType: double
      feasibleSpace:
        min: "0.01"
        max: "0.2"
    - name: max_depth
      parameterType: int
      feasibleSpace:
        min: "2"
        max: "10"
    - name: subsample
      parameterType: double
      feasibleSpace:
        min: "0.5"
        max: "1.0"
```

튜닝 목표 관련 설정

```
print("MAE=%f" % mae)
```

결과 수집 관련 설정

Trial 실행 관련 설정

서치 알고리즘 설정

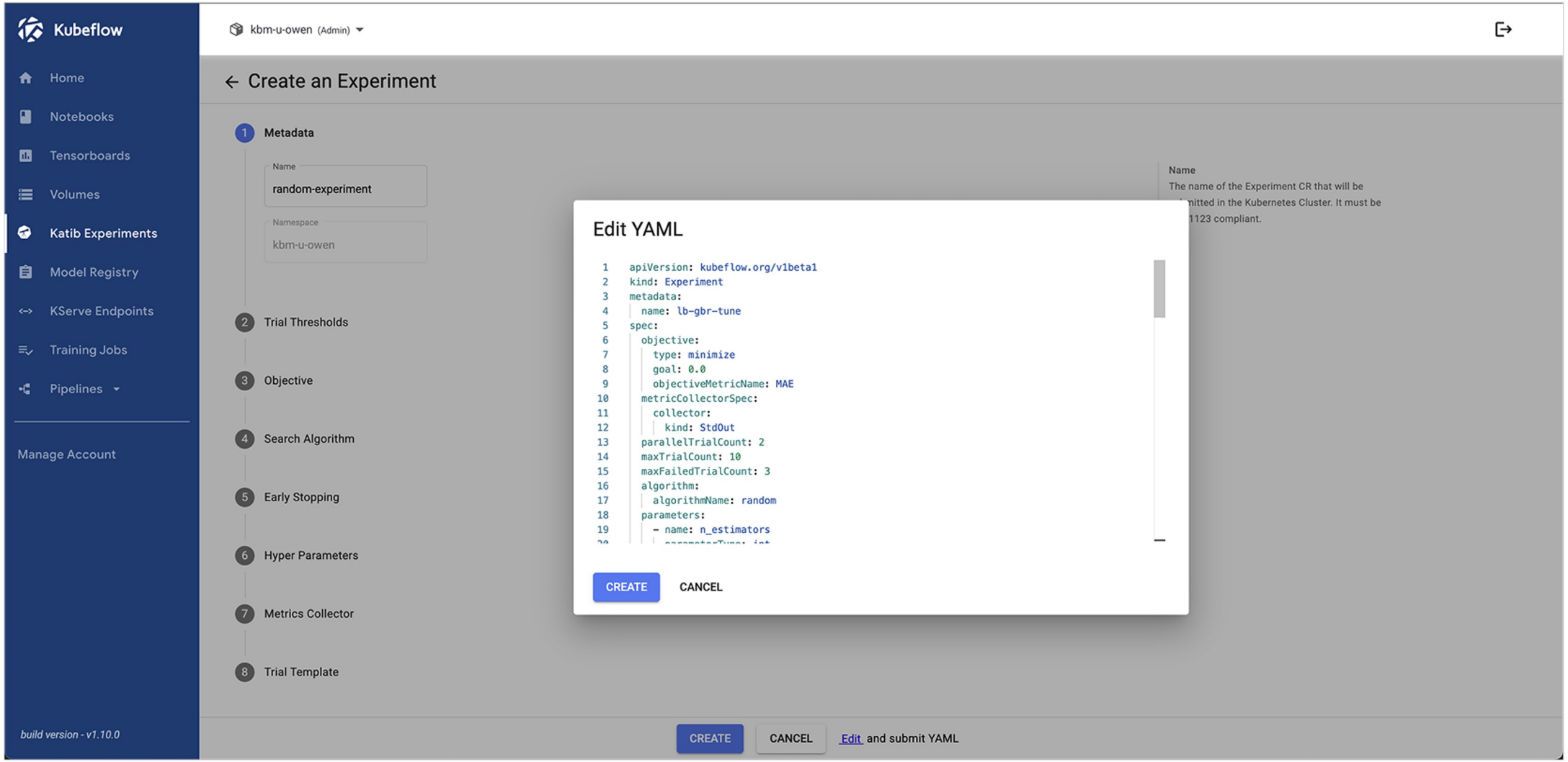
하이퍼파라미터 범위 설정

```yaml
  trialTemplate:
    primaryContainerName: training-container
    trialParameters:
      - name: n_estimators
        description: "Number of estimators"
        reference: "n_estimators"
      - name: learning_rate
        description: "Learning rate"
        reference: "learning_rate"
      - name: max_depth
        description: "Maximum depth of the trees"
        reference: "max_depth"
      - name: subsample
        description: "Subsample ratio"
        reference: "subsample"
    trialSpec:
      apiVersion: batch/v1
      kind: Job
      spec:
        template:
          metadata:
            annotations:
              sidecar.istio.io/inject: 'false'
          spec:
            containers:
              - name: training-container
                image: {{IMAGE}}
                command:
                  - "python"
                  - "/app/tune.py"
                args:
                  - "--n_estimators"
                  - "${trialParameters.n_estimators}"
                  - "--learning_rate"
                  - "${trialParameters.learning_rate}"
                  - "--max_depth"
                  - "${trialParameters.max_depth}"
                  - "--subsample"
                  - "${trialParameters.subsample}"
                resources:
                  requests:
                    cpu: '0.5'
                    memory: 1Gi
                  limits:
                    cpu: '1'
                    memory: 2Gi
```

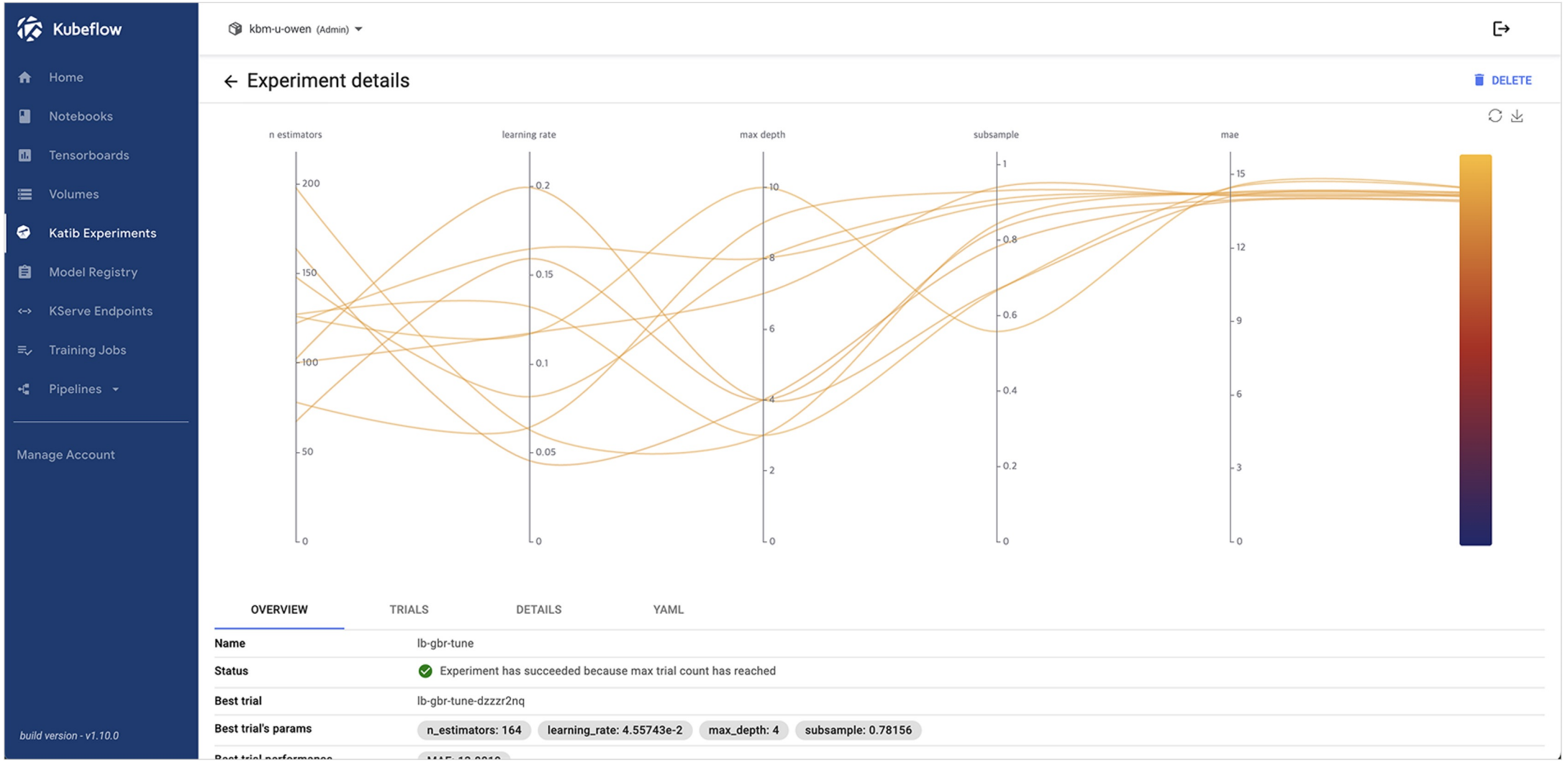
하이퍼파라미터 변수 설정

실행 리소스 (Worker) 관련 설정

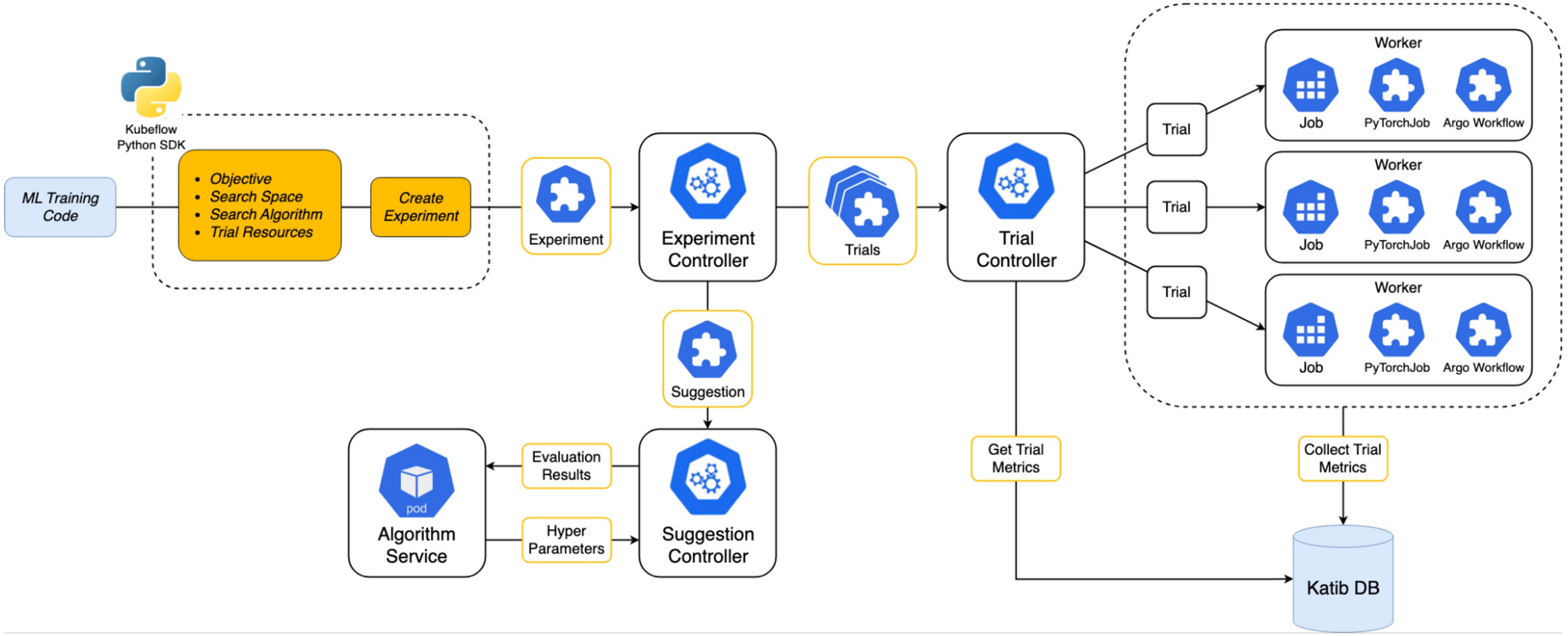
Experiment 제출



튜닝 결과 확인



동작 방식



이미지 출처 : http://www.kubeflow.org/

kakaocloud

ML Task
로드 밸런서 트래픽 예측

시나리오 소개

Step 1. 데이터 탐색 및 모델 개발

Step 2. 하이퍼 파라미터 튜닝

Step 3. 모델 서빙 KServe

Step 4. 파이프라인 구축

KServe

예측, 생성 추론을 위해 만들어진 표준 모델 추론 플랫폼

주요 기능

- 표준화된 추론 프로토콜 제공
- Scale to Zero 를 포함, 서버리스 추론 워크로드 지원
- 확장성 높은 지능형 라우팅 지원
- 고급 배포 기능 지원
  (e.g. canary rollout, experiment, ensembles, transformers)

KServe

주요 개념

- Inference Service

  : 모델 서빙 워크로드를 배포하고, 관리하기 위한 기본 리소스

- Serving Runtime

  : 서빙 모델을 위한 런타임 환경을 정의하는 리소스

지원 포맷

- LightGBM
- SKLearn
- MLFlow
- Paddle
- TensorFlow
- PyTorch
- XGBoost
- Hugging Face
- ONNX
- TensorRT
- PMML

Inference Service 작성


```yaml
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
    name: lb-predictor
spec:
    predictor:
        model:
            modelFormat:
                name: sklearn
            runtime: kserve-sklearnserver
            args:
                - '--model_name=lb-predictor'
            storageUri: "pvc://model-pvc/lb-predictor"
```

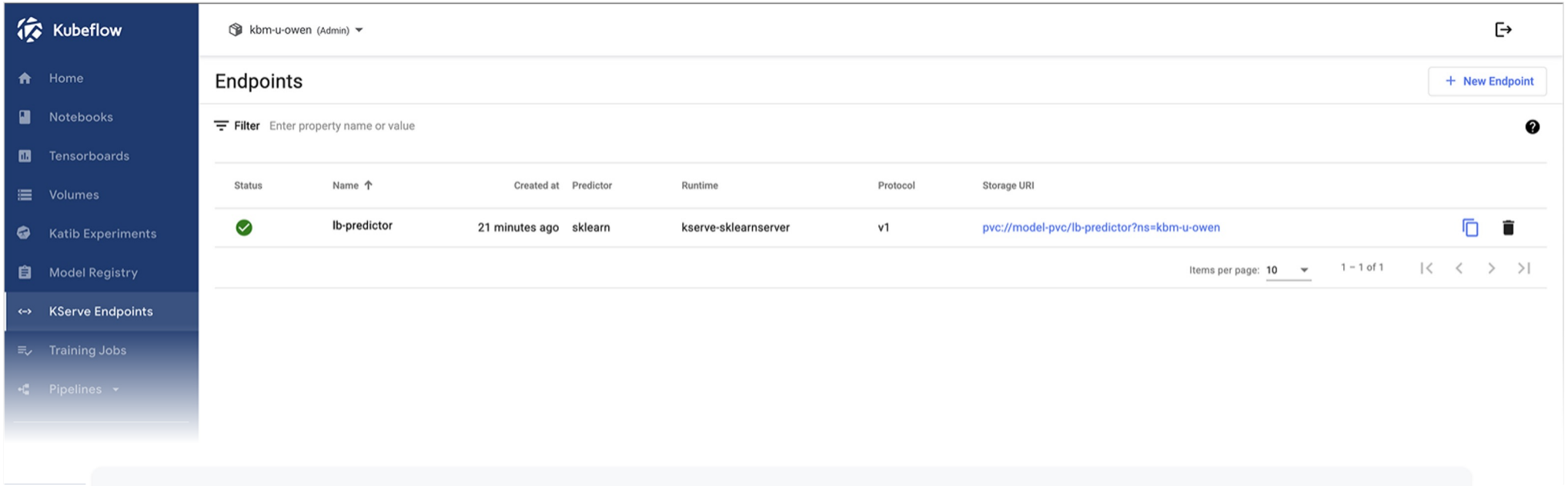
모델 포맷 지정

서빙 런타임 지정

아규먼트 설정

저장된 모델 위치

Inference Service 생성 및 테스트



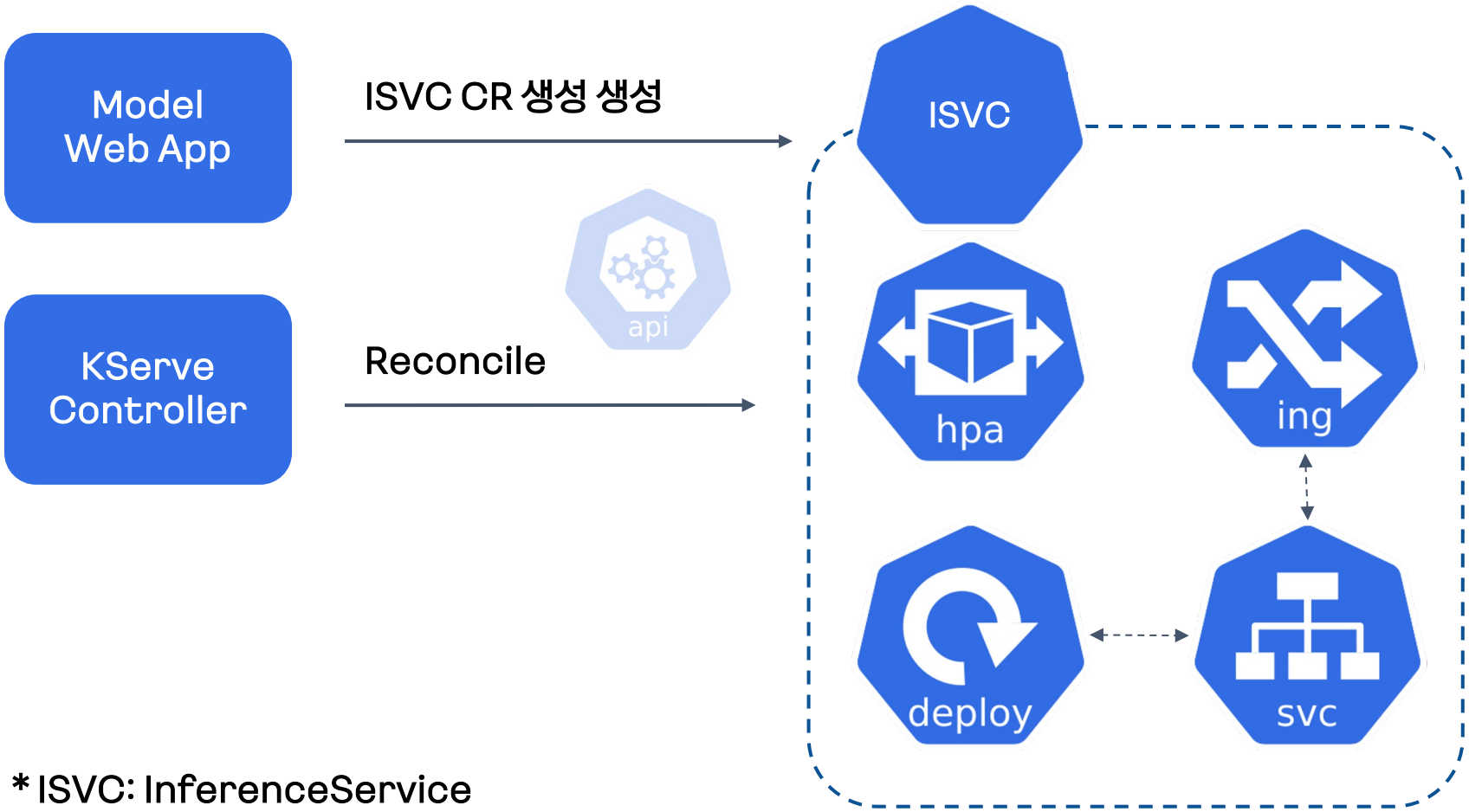
추론 명령어

```
curl -X POST http://{ISVC_NAME}.{NAMESPACE}.svc.cluster.local/v1/models/{MODEL_NAME}:predict  \
-H "Content-Type: application/json" \
-d '{"instances": [[0.1, 0.1, 0.1, 0.1]]}' # 입력 데이터
```

실행 결과

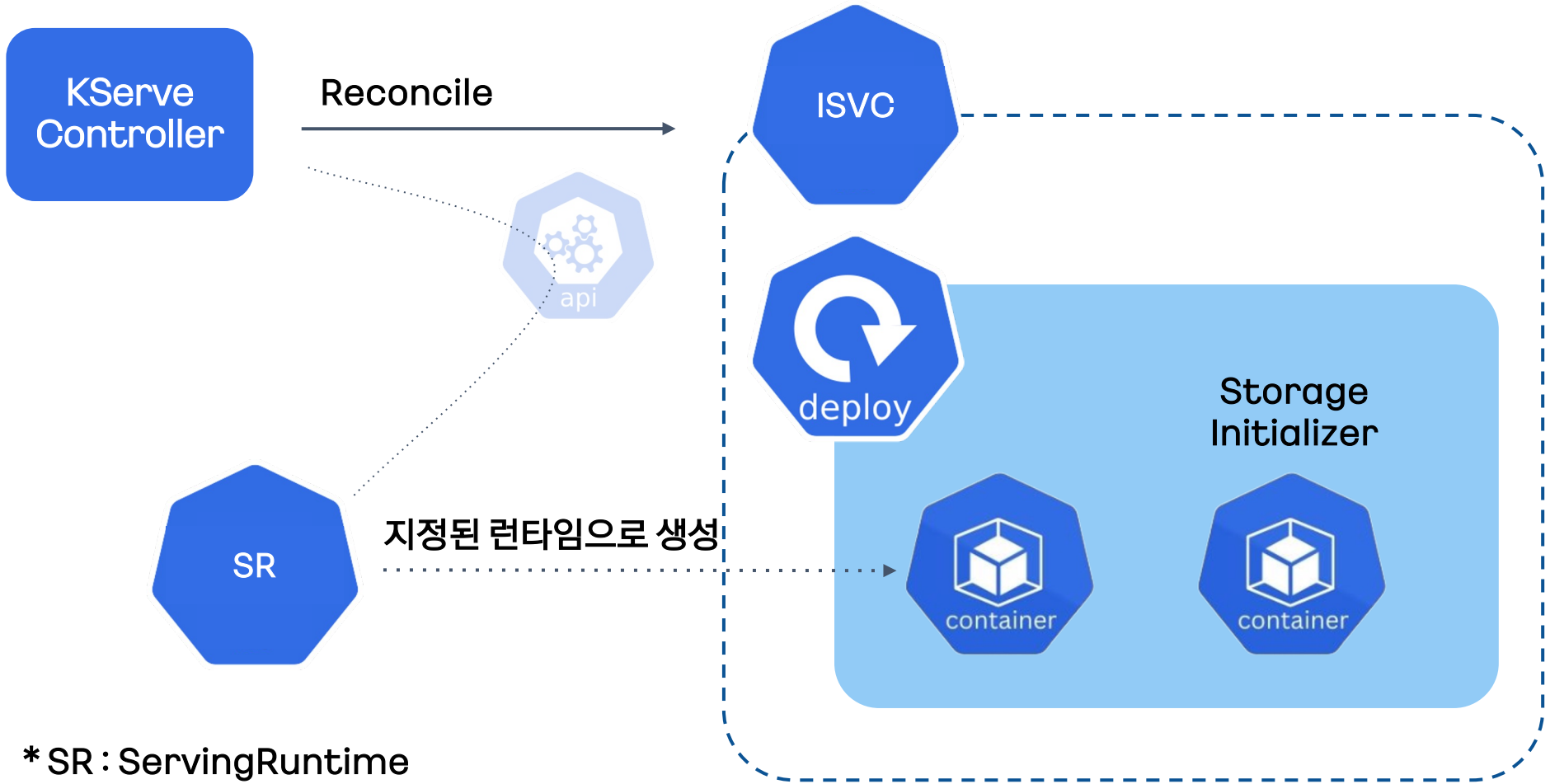
```
{"predictions":[46.52174265571983]}
```

동작 방식 : Inference Service



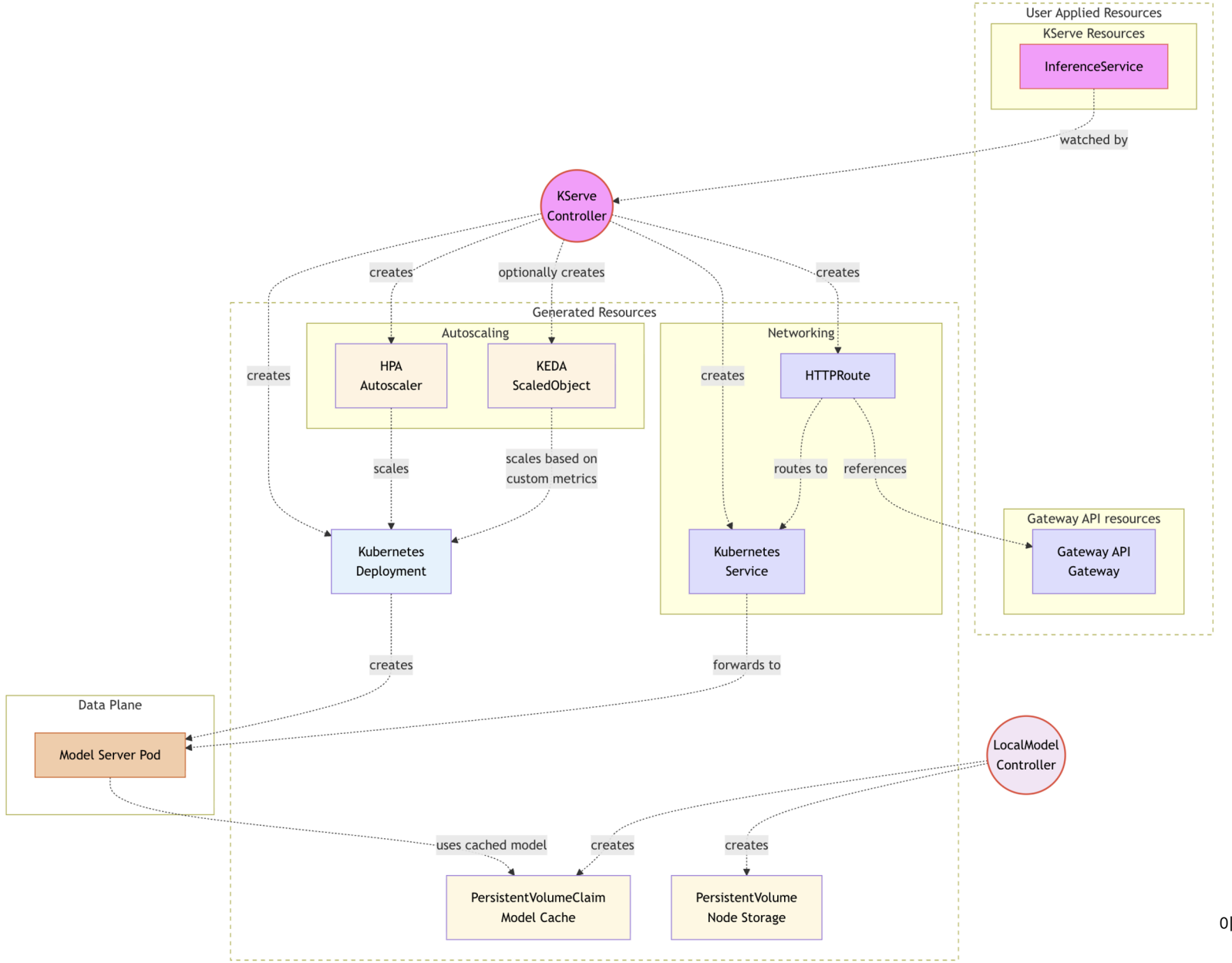
* ISVC: InferenceService

동작 방식 : Serving Runtime



* SR : ServingRuntime

Control Plane



이미지 출처 : https://kserve.github.io

kakaocloud

2. ML Task 로드 밸런서 트래픽 예측

시나리오 소개

Step 1. 데이터 탐색 및 모델 개발

Step 2. 하이퍼 파라미터 튜닝

Step 3. 모델 서빙

Step 4. 파이프라인 구축 Pipeline

Pipeline

ML 워크플로우를 구축하고 배포하기 위한 플랫폼

주요 기능

- 파이프라인 작성 SDK 제공
- UI를 제공하여 파이프라인 관련 아티팩트를 쉽게 관리, 추적 및 시각화
- 병렬 실행 및 캐싱을 통한 효율적인 리소스 사용

Pipeline

주요 개념

용어	설명
파이프라인 (Pipeline)	머신러닝 전체 워크플로우를 정의하는 단위, 여러개의 컴포넌트로 구성
컴포넌트 (Component)	워크플로우의 한 단계를 수행하는 실행 단위
실험 (Experiment)	작성한 파이프라인을 다양한 설정으로 반복 실행하는 논리적 워크스페이스
런 (Run)	실험 내에서 수행되는 단일 파이프라인 인스턴스

파이프라인 구축 프로세스

컴포넌트 작성 → 파이프라인 작성 및 컴파일 → 파이프라인 등록 및 실행

컴포넌트 작성

컴포넌트 정의 데코레이터

```python
@dsl.component(base_image=LB_BASE_IMAGE)
def preprocessing(
    source_data: str,
    data_mnt_path: str,
    start_date: str,
    eval_date: str,
    raw_data: Output[Dataset]
):
    import json
    import pandas as pd
    import os

    # file path
    file_path = os.path.join(data_mnt_path, source_data)

    json_data = []
    # load nlb dataset
    with open(file_path) as f:
        for line in f:  # 각 line 에 있는 JSON형식 텍스트를 읽음
            # load json data
            data = json.loads(line)  # JSON 형식 문자열을 python dict 타입으로 변환
            # append to raw_data
            json_data.append(data)

    raw_df = pd.json_normalize(json_data)  # key:value 형식의 tabular 데이터로 변환, pandas의 Dataframe 객체 형식
    # 시간을 30분 간격으로 변경
    log_time_sr = pd.to_datetime(raw_df['time'],format='%Y/%m/%d %H:%M:%S:%f').dt.floor('30min')

    # 30분 간격으로 로그의 수를 세고, 각 시간 별 로그 수를 dictionary 타입으로 저장
    log_count_dict = log_time_sr.dt.floor('30min').value_counts(dropna=False).to_dict()

    # 데이터셋 시간 범위를 생성 (30분 간격)
    # collect_date = pd.to_datetime(collect_date)
    time_range = pd.date_range(start=start_date, end=eval_date , freq='30min')

    # 시간과 로그 수를 칼럼으로 갖는 데이터프레임 생성
    df = pd.DataFrame({'datetime': time_range})
    df['count'] = df['datetime'].apply(lambda x: log_count_dict.get(x, 0))

    df.to_csv(raw_data.path, index=False)
```

컴포넌트 작성


```python
@dsl.component(base_image=LB_BASE_IMAGE)
def featurization(
    data_version: str,
    eval_date: str,
    data_mnt_path: str,
    raw_data: Input[Dataset]
):
    import pandas as pd
    import numpy as np
    import os

    df = pd.read_csv(raw_data.path,parse_dates=['datetime'])

    time_sr = df['datetime'].apply(lambda x: x.hour * 2 + x.minute // 30)
    dow_sr = df['datetime'].dt.dayofweek

    dataset = pd.DataFrame()
    dataset['datetime'] = df['datetime']  # 이후 작업 편의를 위해 설정

    # 시간 관련 feature
    dataset['x1'] = np.sin(2*np.pi*time_sr/48)
    dataset['x2'] = np.cos(2*np.pi*time_sr/48)
    # 요일 관련 feature
    dataset['x3'] = np.sin(2*np.pi*dow_sr/7)
    dataset['x4'] = np.cos(2*np.pi*dow_sr/7)

    # 예측하고자 하는 대상, label
    dataset['y'] = df['count']

    train_df = dataset[dataset['datetime'] < eval_date]
    test_df = dataset[dataset['datetime'] >= eval_date]

    train_df.to_csv(os.path.join(data_mnt_path,f'nlb-{data_version}.csv'), index=False)
    test_df.to_csv(os.path.join(data_mnt_path,f'nlb-{data_version}-test.csv'), index=False)
```

컴포넌트 작성


```python
@dsl.component(base_image=LB_BASE_IMAGE, packages_to_install=['kubeflow-katib', 'scikit-learn==1.6.1'])
def tuning_with_katib(
    data_mnt_path: str,
    model_mnt_path: str,
    artifact_mnt_path: str,
    katib_exp_name: str,
    model_version: str,
    data_version: str
):
    import os
    import ast
    import joblib,json

    import pandas as pd
    from sklearn.ensemble import GradientBoostingRegressor

    import kubeflow.katib as katib

    katib_client = katib.KatibClient()

    katib_exp = katib_client.get_experiment(katib_exp_name)
    katib_exp.metadata.name += f'-d{data_version}'
    exp_name = katib_exp.metadata.name
    katib_exp.metadata.resource_version = None

    container_spec = katib_exp.spec.trial_template.trial_spec['spec']['template']['spec']['containers'][0]
    container_spec['args'].append('--data_version')
    container_spec['args'].append(data_version)

    katib_client.create_experiment(katib_exp)
    katib_client.wait_for_experiment_condition(name=exp_name)


    optim_params = katib_client.get_optimal_hyperparameters(exp_name)
    print(optim_params)

    params = { param.name: ast.literal_eval(param.value)   for param in optim_params.parameter_assignments}


    df = pd.read_csv(os.path.join(data_mnt_path,f'nlb-{data_version}.csv'))
    print("read df")
    X = df[['x1', 'x2', 'x3', 'x4']]
    y = df['y']


    model = GradientBoostingRegressor(**params)
    model.fit(X, y)
    print("Model train")


    model_dir = os.path.join(model_mnt_path, model_version)
    os.makedirs(model_dir, exist_ok=True)
    joblib.dump(model, os.path.join(model_dir,'model.joblib'))
    print("Dump to model")

    artifact_dir = os.path.join(artifact_mnt_path, model_version)
    os.makedirs(artifact_dir, exist_ok=True)
    with open(os.path.join(artifact_mnt_path, model_version,'param.json'),'w') as f:
        json.dump(params,f)
```

컴포넌트 작성


```python
@dsl.component(base_image=LB_BASE_IMAGE, packages_to_install=['scikit-learn==1.6.1'])
def evaluate_model(
    data_mnt_path: str,
    model_mnt_path: str,
    artifact_mnt_path: str,
    model_version: str,
    data_version: str
):
    import os
    import joblib
    import json

    import pandas as pd
    from sklearn.metrics import mean_absolute_error
    df = pd.read_csv(os.path.join(data_mnt_path,f'nlb-{data_version}-test.csv'))
    X = df[['x1', 'x2', 'x3', 'x4']]
    real = df['y']

    model_dir = os.path.join(model_mnt_path, model_version)
    model = joblib.load(f'/{model_dir}/model.joblib')

    pred = model.predict(X)


    mae = mean_absolute_error(real,pred)
    # you can add more metrics (e.g. mse, r2, etc.)
    metrics = {'mae': mae}
    print(f"MAE: {mae}")

    artifacts_path = os.path.join(artifact_mnt_path, model_version)
    with open(os.path.join(artifacts_path,'score.json'),'w') as f:
        json.dump(metrics,f)
```

컴포넌트 작성


```python
@dsl.component(base_image=LB_BASE_IMAGE, packages_to_install=['model-registry==0.2.21'])
def push_to_registry(
    model_registry_host: str,
    model_registry_port: int,
    mr_author: str,
    model_name: str,
    model_version: str,
    model_pvc: str,
    namespace: str
):
    from model_registry import ModelRegistry

    if not model_registry_host.startswith("http"):
        model_registry_host = f"http://{model_registry_host}"

    registry = ModelRegistry(
        server_address=model_registry_host, # default: http://model-registry-service.kubeflow.svc.cluster.l
        port=model_registry_port,
        author=mr_author,
        is_secure=False
    )
    registry.register_model(
        model_name,
        f"pvc://{model_pvc}/{model_version}/model.joblib",
        model_format_name="joblib",
        model_format_version="1",
        version=model_version,
        metadata={"namespace": namespace}
    )
```

컴포넌트 작성


```python
@dsl.component(base_image=LB_BASE_IMAGE, packages_to_install=['--user','model-registry==0.2.21', 'kserve==0
def update_kserve(
    model_registry_host: str,
    model_registry_port: int,
    mr_author: str,
    model_name: str,
    model_version: str
):
    from model_registry import ModelRegistry
    from kubernetes import client
    import kserve

    if not model_registry_host.startswith("http"):
        model_registry_host = f"http://{model_registry_host}"

    registry = ModelRegistry(
        server_address=model_registry_host, # default: http://model-registry-service.kubeflow.svc.cluster.l
        port=model_registry_port,
        author=mr_author,
        is_secure=False
    )

    model = registry.get_registered_model(model_name)
    print("Registered Model:", model, "with ID", model.id)

    version = registry.get_model_version(model_name, model_version)
    print("Model Version:", version, "with ID", version.id)

    art = registry.get_model_artifact(model_name, model_version)
    print("Model Artifact:", art, "with ID", art.id)

    isvc = kserve.V1beta1InferenceService(
        api_version=kserve.constants.KSERVE_GROUP + "/v1beta1",
        kind=kserve.constants.KSERVE_KIND_INFERENCESERVICE,
        metadata=client.V1ObjectMeta(
            name=model_name,
            labels={
                "modelregistry/registered-model-id": model.id,
                "modelregistry/model-version-id": version.id,
            },
        ),
        spec=kserve.V1beta1InferenceServiceSpec(
            predictor=kserve.V1beta1PredictorSpec(
                sklearn=kserve.V1beta1SKLearnSpec(
                    args=[f'--model_name={model_name}'],
                    storage_uri=art.uri
                )
            )
        ),
    )
    ks_client = kserve.KServeClient()
    try:
        ks_client.get(model_name)
        ks_client.replace(model_name,isvc)
    except: # assumption: not found err
        ks_client.create(isvc)
```

파이프라인 작성


```python
@dsl.pipeline(name="Load Prediction Model Pipeline")
def lb_model_pipeline(
    source_data: str,
    start_date: str,
    eval_date: str,
    mr_host: str,
    mr_author: str,
    namespace: str,
    dataset_pvc: str='dataset-pvc',
    model_pvc: str='model-pvc',
    artifact_pvc: str='artifact-pvc',
    katib_exp_name: str='lb-gbr-tune',
    mr_port: int=8080,
    model_name: str='lb-predictor'
):
    # step1: Load dataset
    load_dataset_task = preprocessing(
        source_data=source_data,
        data_mnt_path='/dataset',
        start_date=start_date,
        eval_date=eval_date
    )
    kubernetes.mount_pvc(
        load_dataset_task,
        pvc_name=dataset_pvc,
        mount_path='/dataset'
    )

    # step2: Featurization
    featurization_task = featurization(
        data_version=start_date,
        eval_date=eval_date,
        data_mnt_path='/dataset',
        raw_data=load_dataset_task.outputs['raw_data']
    )
    featurization_task.after(load_dataset_task)

    kubernetes.mount_pvc(
        featurization_task,
        pvc_name=dataset_pvc,
        mount_path='/dataset'
    )
```

파이프라인 정의 데코레이터

컴포넌트 사용

볼륨 마운트

아티팩트 전달

종속성 설정

파이프라인 작성


```python
@dsl.pipeline(name="Load Prediction Model Pipeline")
def lb_model_pipeline(
    source_data: str,
    start_date: str,
    eval_date: str,
    mr_host: str,
    mr_author: str,
    namespace: str,
    dataset_pvc: str='dataset-pvc',
    model_pvc: str='model-pvc',
    artifact_pvc: str='artifact-pvc',
    katib_exp_name: str='lb-gbr-tune',
    mr_port: int=8080,
    model_name: str='lb-predictor'
):
    # step1: Load dataset
    load_dataset_task = preprocessing(
        source_data=source_data,
        data_mnt_path='/dataset',
        start_date=start_date,
        eval_date=eval_date
    )
    kubernetes.mount_pvc(
        load_dataset_task,
        pvc_name=dataset_pvc,
        mount_path='/dataset'
    )

    # step2: Featurization
    featurization_task = featurization(
        data_version=start_date,
        eval_date=eval_date,
        data_mnt_path='/dataset',
        raw_data=load_dataset_task.outputs['raw_data']
    )
    featurization_task.after(load_dataset_task)

    kubernetes.mount_pvc(
        featurization_task,
        pvc_name=dataset_pvc,
        mount_path='/dataset'
    )
```

```python
    # step3: Tuning
    tuning_model_task = tuning_with_katib(
        katib_exp_name=katib_exp_name,
        model_version=start_date,
        data_version=start_date,
        data_mnt_path='/dataset',
        model_mnt_path='/model',
        artifact_mnt_path='/artifact'
    )
    tuning_model_task.after(featurization_task)

    kubernetes.mount_pvc(
        tuning_model_task,
        pvc_name=dataset_pvc,
        mount_path='/dataset'
    )
    kubernetes.mount_pvc(
        tuning_model_task,
        pvc_name=model_pvc,
        mount_path='/model'
    )
    kubernetes.mount_pvc(
        tuning_model_task,
        pvc_name=artifact_pvc,
        mount_path='/artifact'
    )

    # step4: Evaluation
    evaluate_model_task = evaluate_model(
        model_version=start_date,
        data_version=start_date,
        data_mnt_path='/dataset',
        model_mnt_path='/model',
        artifact_mnt_path='/artifact'
    )
    evaluate_model_task.after(tuning_model_task)

    kubernetes.mount_pvc(
        evaluate_model_task,
        pvc_name=dataset_pvc,
        mount_path='/dataset'
    )
    kubernetes.mount_pvc(
        evaluate_model_task,
        pvc_name=model_pvc,
        mount_path='/model'
    )
    kubernetes.mount_pvc(
        evaluate_model_task,
        pvc_name=artifact_pvc,
        mount_path='/artifact'
    )
```

```python
    # step5: Regist model
    regist_model_task = push_to_registry(
        model_name=model_name,
        model_version=start_date,
        model_pvc=model_pvc,
        model_registry_host=mr_host,
        model_registry_port=mr_port,
        mr_author=mr_author,
        namespace=namespace
    )
    regist_model_task.after(evaluate_model_task)

    kubernetes.mount_pvc(
        regist_model_task,
        pvc_name=artifact_pvc,
        mount_path='/artifact'
    )

    # step6: Update svc
    update_task = update_kserve(
        model_name=model_name,
        model_version=start_date,
        model_registry_host=mr_host,
        model_registry_port=mr_port,
        mr_author=mr_author,

    )
    update_task.after(regist_model_task)
```

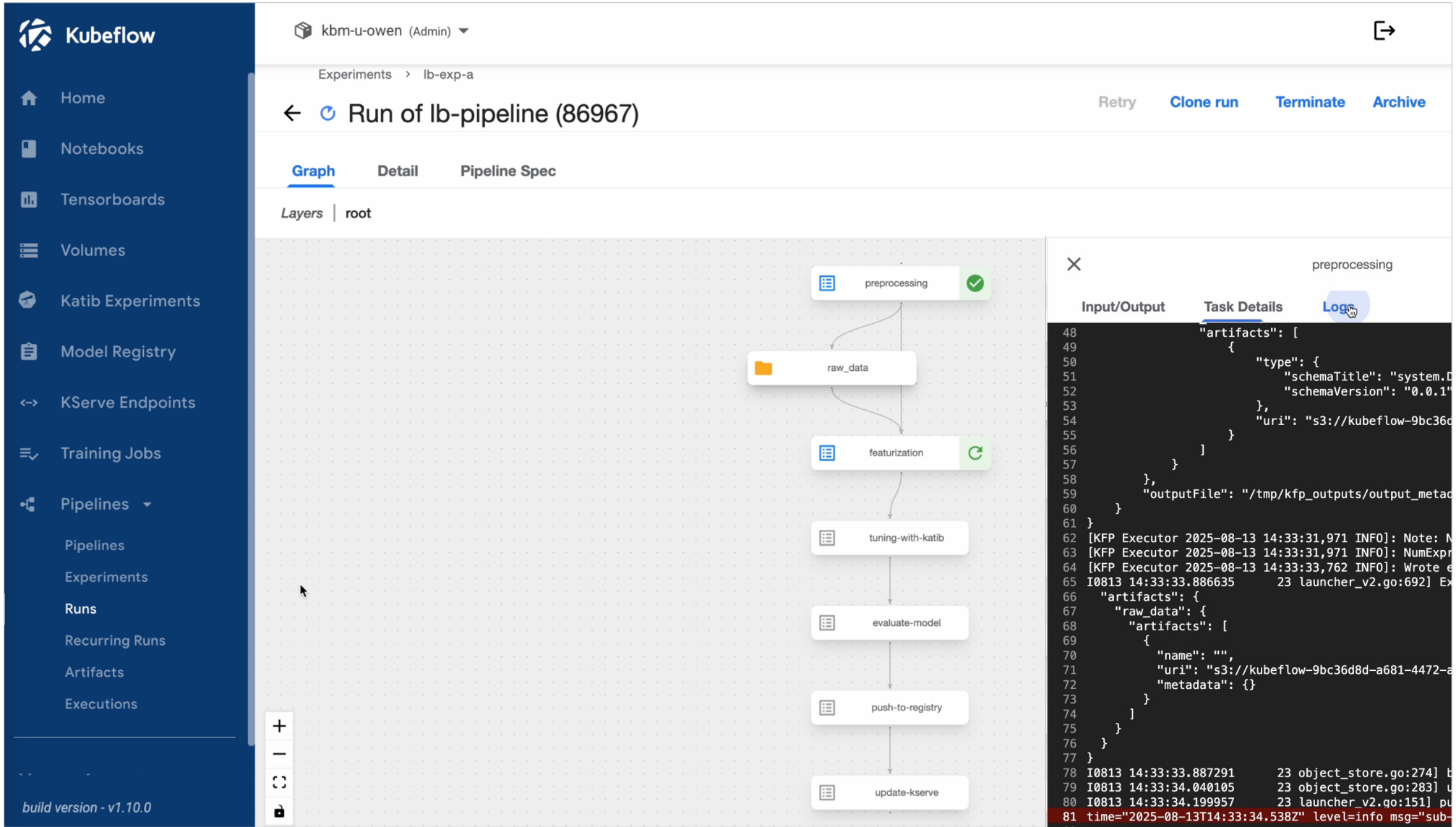
파이프라인 컴파일


```
import kfp
kfp.compiler.Compiler().compile(lb_model_pipeline, "lb-pipeline.yaml")
```

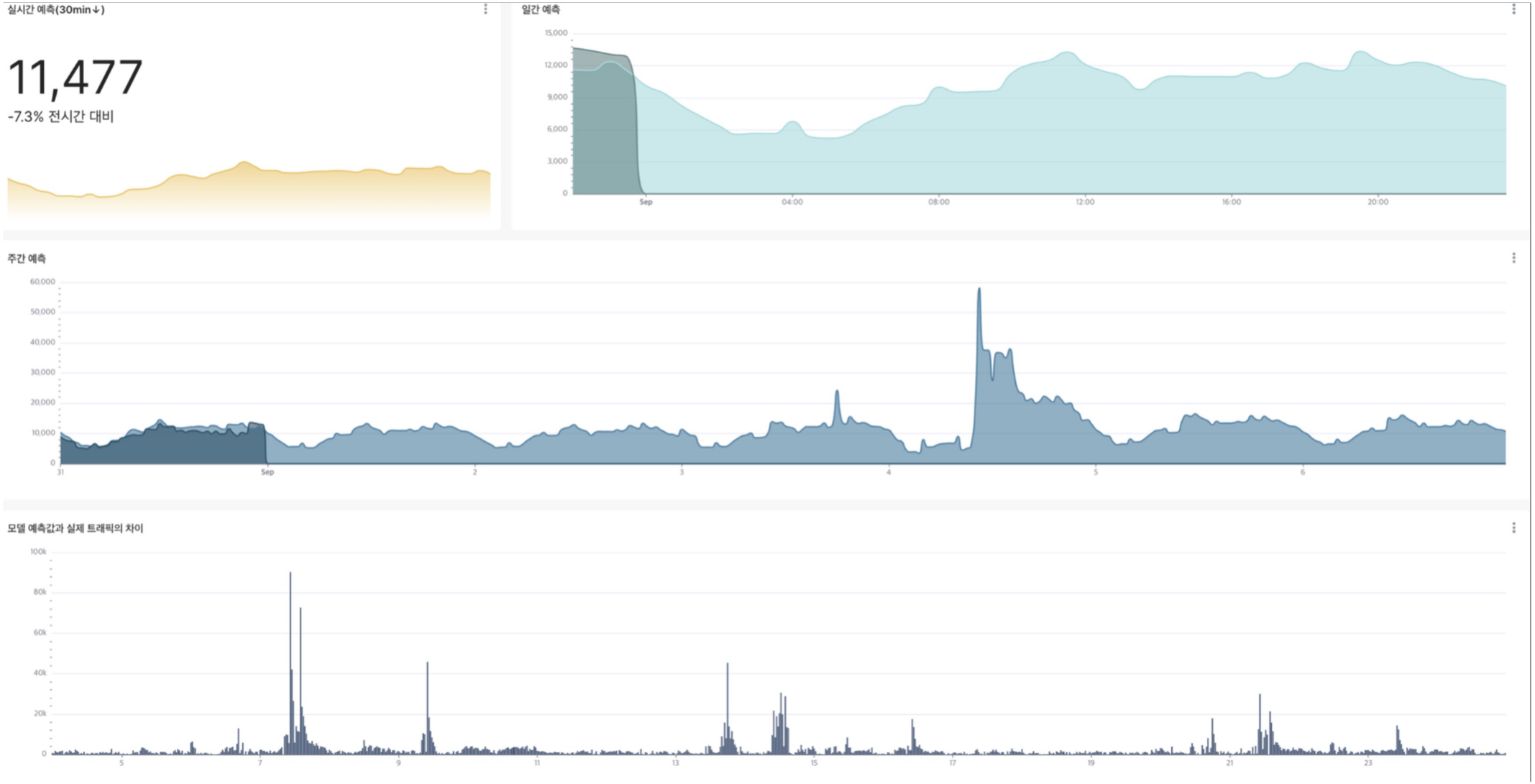
lb-pipeline.yaml

```
deploymentSpec:
  executors:
    exec-evaluate-model:
      container:
        args:
        - --executor_input
        - '{{$}}'
        - --function_to_execute
        - evaluate_model
        command:
        - sh
        - -c
        - "\nif ! [ -x \"$(command -v pip)\" ]; then\n    python3 -m ensurepip ||\
          \ python3 -m ensurepip --user || apt-get install python3-pip\nfi\n\nPIP_DISABLE_PIP_VERSION_CHECK=1\
          \ python3 -m pip install --quiet --no-warn-script-location 'kfp==2.11.0'\
          \ '--no-deps' 'typing-extensions>=3.7.4,<5; python_version<\"3.9\"'  &&\
          \  python3 -m pip install --quiet --no-warn-script-location 'scikit-learn==1.6.1'\
          \ && \"$0\" \"$@\"\n"
        - sh
        - -ec
        - 'program_path=$(mktemp -d)


          printf "%s" "$0" > "$program_path/ephemeral_component.py"

          _KFP_RUNTIME=true python3 -m kfp.dsl.executor_main                         --component_module_path
    "$program_path/ephemeral_component.py"                         "$@"

          '
        - "\nimport kfp\nfrom kfp import dsl\nfrom kfp.dsl import *\nfrom typing import\
          \ *\n\ndef evaluate_model(\n    data_mnt_path: str,\n    model_mnt_path:\
          \ str,\n    artifact_mnt_path: str,\n    model_version: str,\n    data_version:\
          \ str\n):\n    import os\n    import joblib\n    import json\n\n    import\
          \ pandas as pd\n    from sklearn.metrics import mean_absolute_error\n  \
          \  df = pd.read_csv(os.path.join(data_mnt_path,f'nlb-{data_version}-test.csv'))\n\
          \    X = df[['x1', 'x2', 'x3', 'x4']]\n    real = df['y']\n\n    model_dir\
          \ = os.path.join(model_mnt_path, model_version)\n    model = joblib.load(f'/{model_dir}/model.joblib')\n\
          \n    pred = model.predict(X)\n\n\n    mae = mean_absolute_error(real,pred)\n\
          \    # you can add more metrics (e.g. mse, r2, etc.)\n    metrics = {'mae':\
          \ mae}\n    print(f\"MAE: {mae}\")\n\n    artifacts_path = os.path.join(artifact_mnt_path,\
          \ model_version)\n    with open(os.path.join(artifacts_path,'score.json'),'w')\
          \ as f:\n        json.dump(metrics,f)\n\n"
        image: mlops.kr-central-2.kcr.dev/kc-kubeflow-registry/jupyter-scipy:v1.10.0.py311.1a
    exec-featurization:
      container:
        args:
        - --executor_input
        - '{{$}}'
        - --function_to_execute
        - featurization
        command:
        - sh
        - -c
        - "\nif ! [ -x \"$(command -v pip)\" ]; then\n    python3 -m ensurepip ||\
          \ python3 -m ensurepip --user || apt-get install python3-pip\nfi\n\nPIP_DISABLE_PIP_VERSION_CHECK=1\
          \ python3 -m pip install --quiet --no-warn-script-location 'kfp==2.11.0'\
          \ '--no-deps' 'typing-extensions>=3.7.4,<5; python_version<\"3.9\"' && \"\
          $0\" \"$@\"\n"
        - sh
        - -ec
        - 'program_path=$(mktemp -d)
```

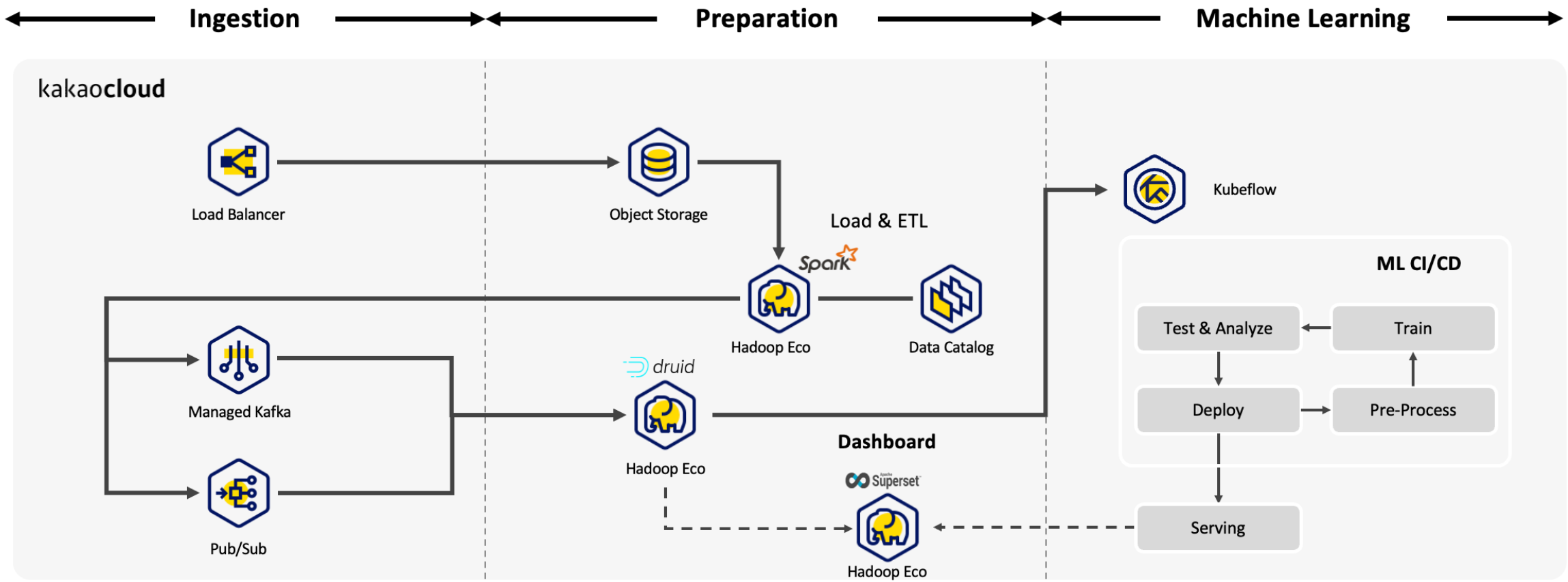
실행 및 모니터링


Kubeflow
kbm-u-owen (Admin)
Home
Notebooks
Tensorboards
Volumes
Katib Experiments
Model Registry
KServe Endpoints
Training Jobs
Pipelines
Pipelines
Experiments
Runs
Recurring Runs
Artifacts
Executions
build version - v1.10.0
Experiments > lb-exp-a
Run of lb-pipeline (86967)
Retry
Clone run
Terminate
Archive
Graph
Detail
Pipeline Spec
Layers | root
preprocessing
raw_data
featurization
tuning-with-katib
evaluate-model
push-to-registry
update-kserve
preprocessing
Input/Output
Task Details
Logs
"artifacts": [
"type": {
"schemaTitle": "system.D
"schemaVersion": "0.0.1"
},
"uri": "s3://kubeflow-9bc36d
"outputFile": "/tmp/kfp_outputs/output_metad
[KFP Executor 2025-08-13 14:33:31,971 INFO]: Note: N
[KFP Executor 2025-08-13 14:33:31,971 INFO]: NumExpr
[KFP Executor 2025-08-13 14:33:33,762 INFO]: Wrote e
I0813 14:33:33.886635 23 launcher_v2.go:692] Ex
"artifacts": {
"raw_data": {
"artifacts": [
"name": "",
"uri": "s3://kubeflow-9bc36d8d-a681-4472-a
"metadata": {}
I0813 14:33:33.887291 23 object_store.go:274] b
I0813 14:33:34.040105 23 object_store.go:283] u
I0813 14:33:34.199957 23 launcher_v2.go:151] pu
time="2025-08-13T14:33:34.538Z" level=info msg="sub-

트래픽 예측 (Superset 대시보드)



Appendix: MLOps 구성



kakaocloud

3. 카카오클라우드 서비스

KC Kubeflow의 특징

편의성 카카오클라우드의 여러 서비스와 연계, 손쉽게 AI 플랫폼 구축, 운영

Network
- VPC
- Load Balancing

Node
- Virtual Machine
- Baremetal
- GPU

Auth
- IAM

Storage
- File Storage
- Object Storage

Data Store
- MySQL

KC Kubeflow의 특징

비용-성능 최적화

워크로드 특성에 맞춘 노드풀 증감 가능

→ 비용 & 성능 최적화

고가용성

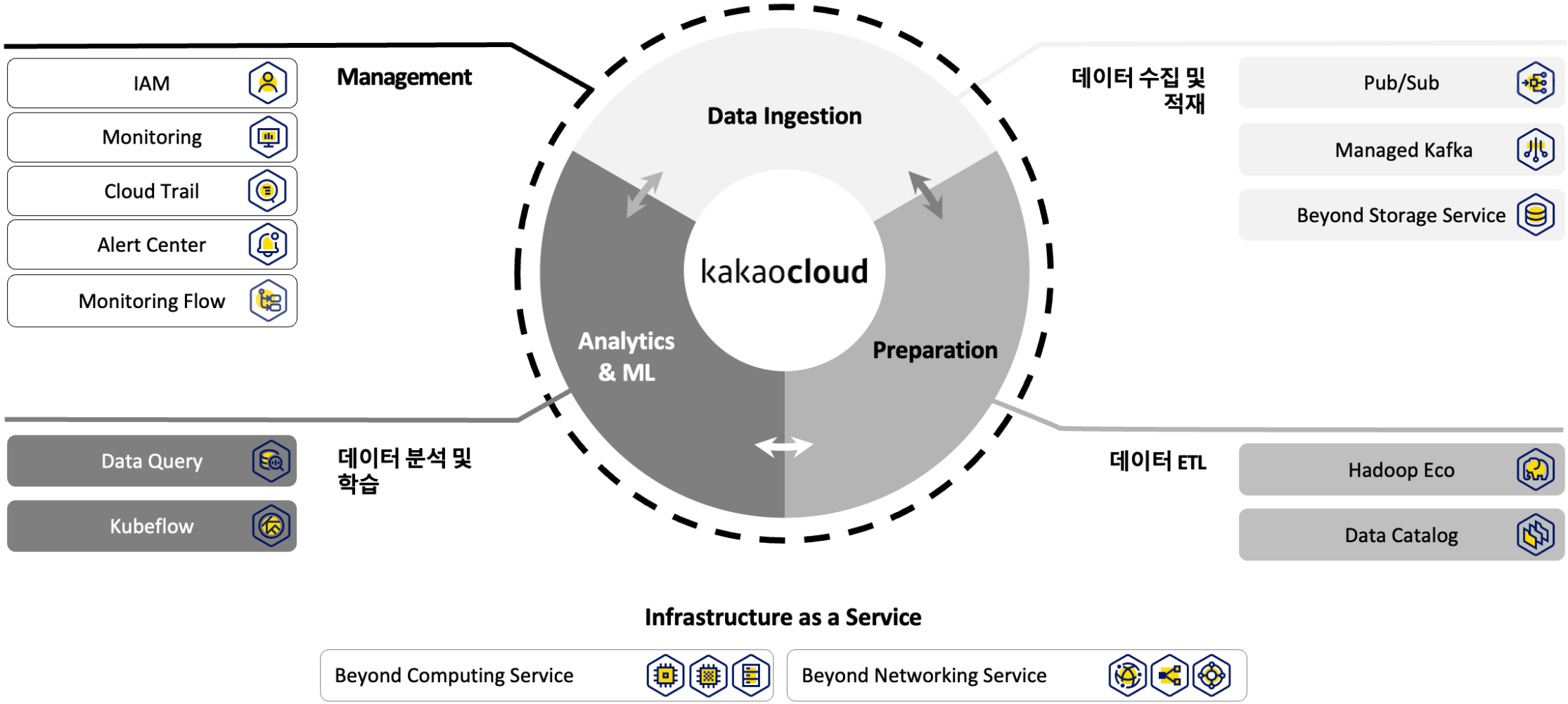
클러스트와 노드가 여러 AZ(가용 영역)에서 실행

→ 서비스 중단없는 고가용성 제공

KC Kubeflow의 특징

분리 다중 테넌트 환경	사용자/그룹 네임스페이스별 독립된 Quota 제한 및 모델 레지스트리, 오브젝트 스토리지 제공
다양한 실험 환경	다양한 ML 프레임워크 기반 노트북/파이프라인 이미지 제공
실시간 모니터링	콘솔내 GPU/CPU/Memory 사용량 모니터링 대시보드 제공
최신버전 지원 & 버그 픽스	타 글로벌 CSP (Azure, GCP) 보다 앞선 최신 버전 제공 (1.10 기준) 및 오픈소스 프로젝트 보다 빠른 버그 픽스

MLOps on Kakaocloud



kakaocloud

카카오엔터프라이즈에서
클라우드 부문 개발자/엔지니어를 영입합니다.

인재영입 사이트 접속

IaaS 개발

CI/CD 서비스 개발자
보안서비스 개발자
Infrastructure Platform 개발자
클라우드 네트워킹 서비스 개발자
클라우드 컴퓨팅 서비스 개발자

서비스 개발

클라우드 기술지원 엔지니어
클라우드 보안 기술지원 엔지니어
클라우드 웹서비스 개발자
클라우드 플랫폼 서버 개발자
Technical Account Manager

인프라

클라우드 Managed 서비스 엔지니어
클라우드 네트워크 엔지니어
Database 엔지니어(MySQL)
인프라 개발자
클라우드 인프라 서비스 엔지니어

PaaS 개발

Kubernetes Engine 개발자
빅데이터 플랫폼 엔지니어
빌링플랫폼 개발자
자동 관리형 DB 서비스 개발자
Monitoring Service 개발자
Managed Search Service 개발자
MLOps 엔지니어

정보보안

개인정보보호 담당자
정보보호 관리체계 통합 운영 담당자
보안 플랫폼 개발자