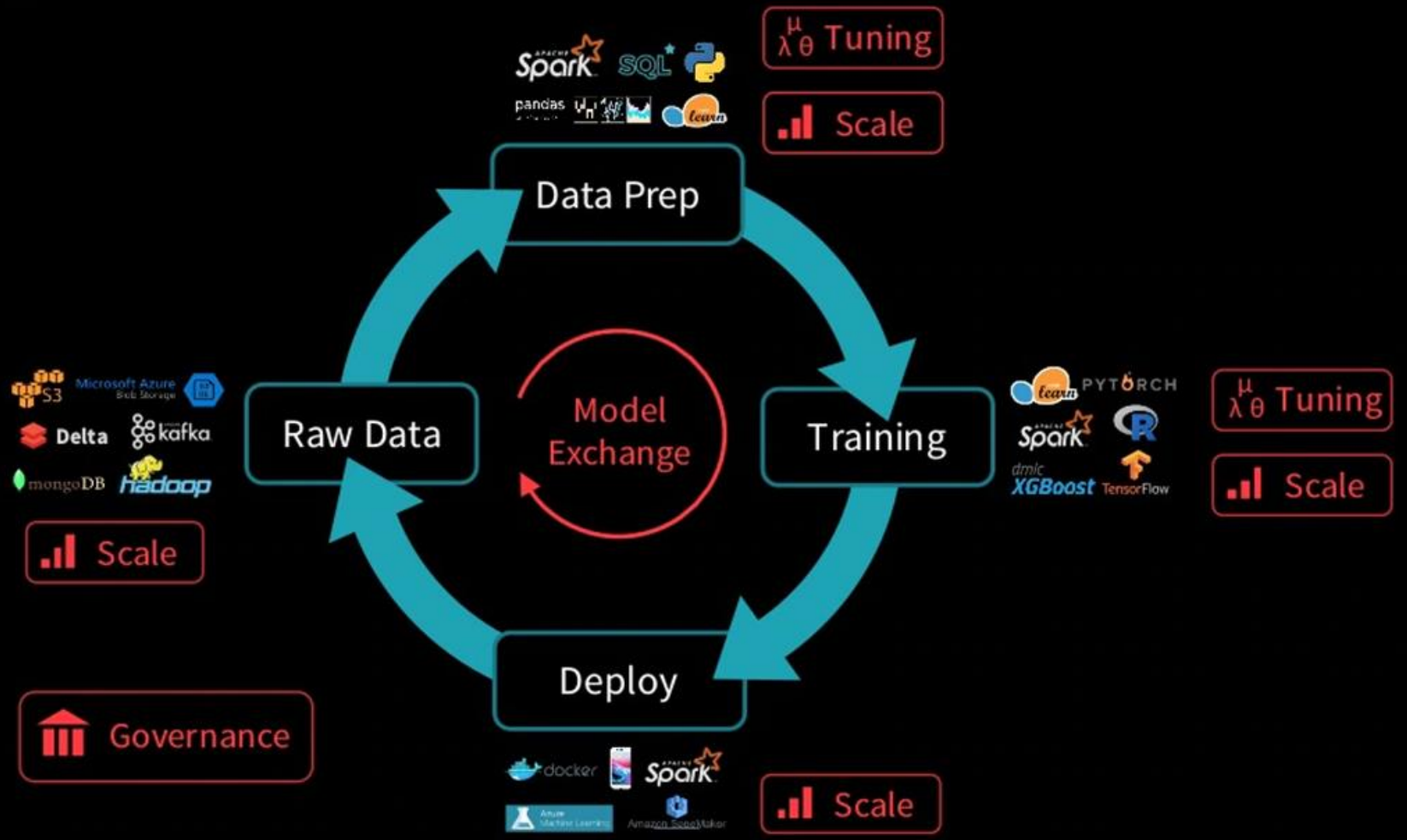
MLflow

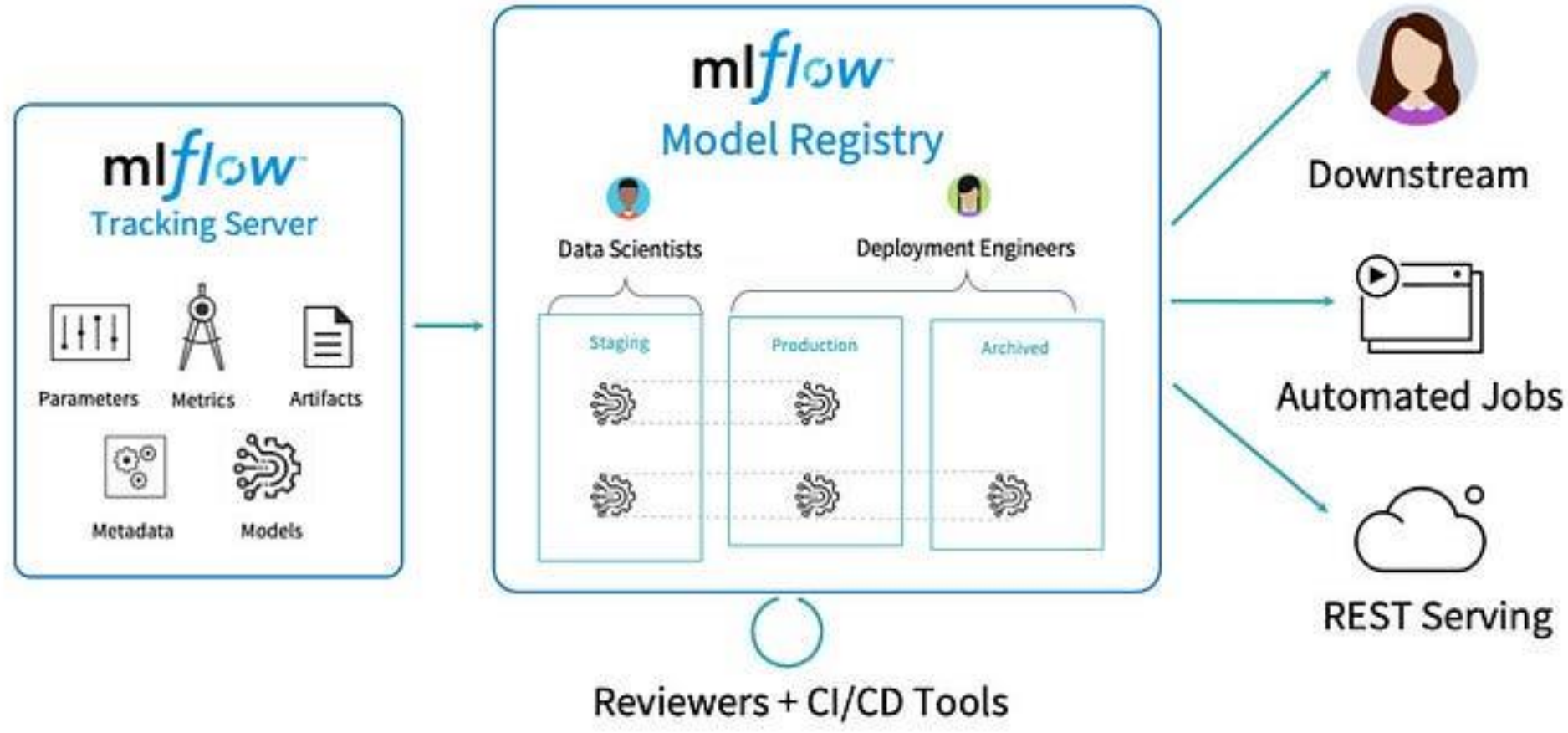

Tuning
Scale
Data Prep
Raw Data
Model Exchange
Training
Deploy
Governance
Scale
Tuning
Scale
Scale

박경규

MLflow

https://mlflow.org/

MLflow는 머신러닝 프로세스의 복잡성을 처리하는 데 도움을 주는 오픈 소스 플랫폼입니다.
MLflow는 머신러닝 프로젝트의 전체 수명 주기에 중점을 두어 각 단계를 관리, 추적 및 재현할 수 있도록 합니다.



MLflow 특징

Language Agnostic

- 모듈러 API-first approach
- ML 라이브러리 및 모든 프로그램 언어에서 사용 가능

호환성

- Tensorflow, PyTorch, Keras, Apache Spark,

  Sciikit Learn 등과 같은 수많은 라이브러리와 호환

통합Integration

- Docker Containers, Kubernetes cluster,

  Apache Spark 등의 형태로 모델을 프로덕션에 적용

개발

- Databricks 개발
- 2018년 6월 첫번째 버전 출시

MLflow 컴포넌트

Tracking

머신러닝 실험을 추적하여
결과와 매개변수를
기록, 분석 및 비교

Projects

재사용성 및 재현성을
보장하는 패키징

Models

패키징 모델에 대한
표준 단위 제공

Registry

모델 버전 관리, 스테이지 전환 및
주석을 포함하는
중앙 집중식 모델 저장소

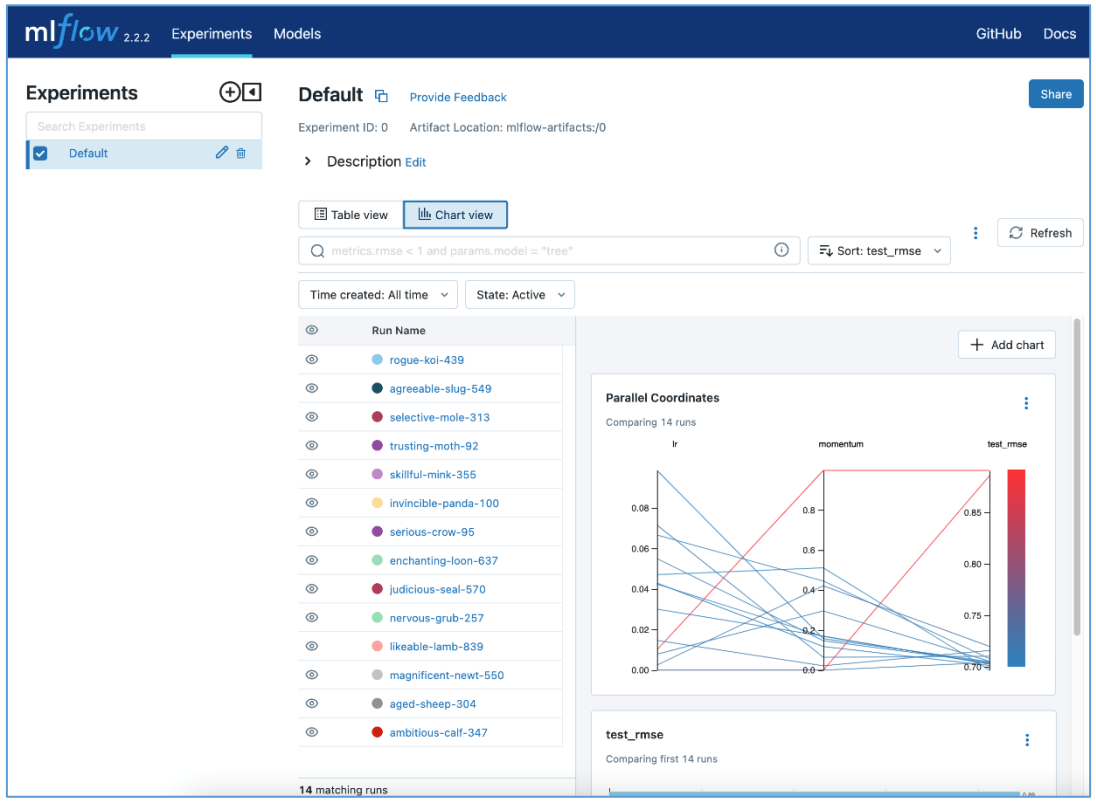
MLflow 컴포넌트 - Tracking

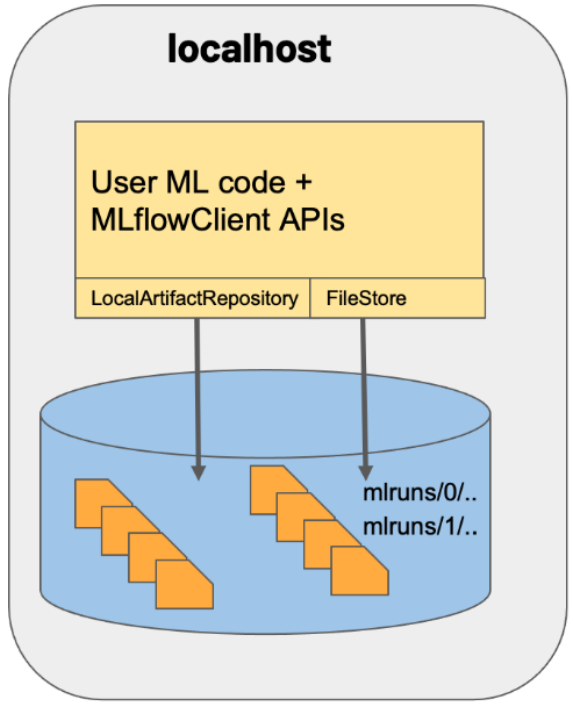
매개변수, 코드 버전, 메트릭 및 출력 파일을 로깅하기 위한 API와 그래픽 사용자 인터페이스(UI)를 제공합니다.

Python, REST, R API 및 Java API를 사용하여 실험을 로깅하고 쿼리할 수 있습니다.

실행 탐색, 메트릭 및 매개변수 시각화, 실험 비교를 위한 웹 기반 사용자 인터페이스를 제공합니다.

로컬 및 원격 서버를 포함한 다양한 Tracking 서버를 지원하므로, 실험을 쉽게 확장하고 공유할 수 있습니다.





MLflow 컴포넌트 – Projects

머신러닝 코드의 패키징, 재현성 및 공유 프로세스를 간소화 하도록 설계 되었습니다.

코드와 종속성을 재현 가능한 형식으로 패키징 하여 다양한 환경에서 실험을 쉽게 공유하고 재현할 수 있습니다.

MLproject 파일은 프로젝트 환경, 매개 변수, 진입점 및 머신 러닝 코드와 데이터가 포함된 파일 세트를 설명하는 파일입니다.

CLI를 제공합니다. 다양한 프로젝트 템플릿을 지원하므로 머신러닝 프로젝트를 빠르게 시작할 수 있습니다.

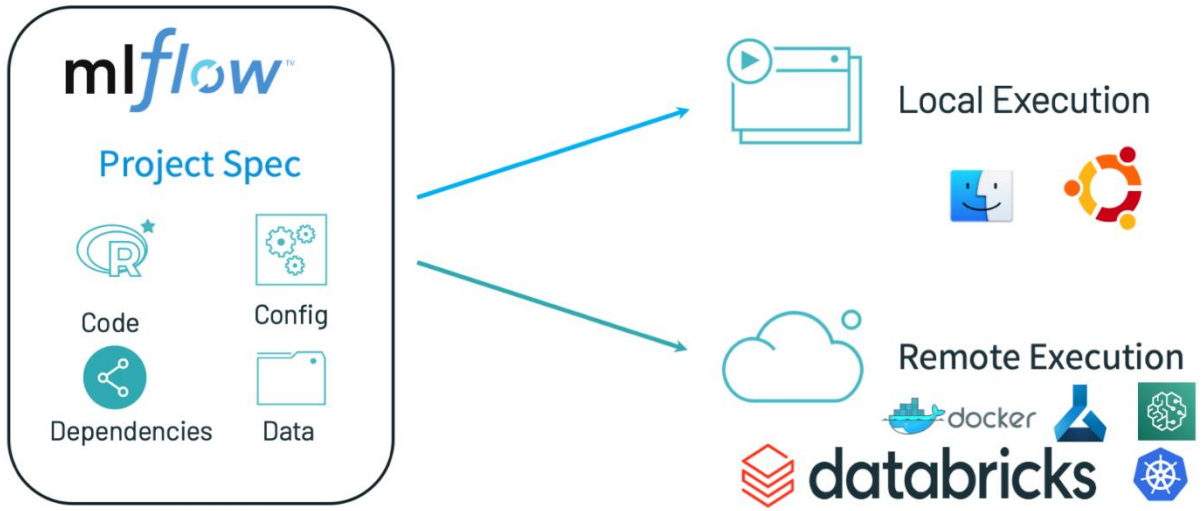
MLproject 파일

```yaml
name: My Project

conda_env: my_env.yaml
# Can have a docker_env instead of a conda_env, e.g.
# docker_env:
#    image:  mlflow-docker-example

entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default: 0.1}
    command: "python train.py -r {regularization} {data_file}"
  validate:
    parameters:
      data_file: path
    command: "python validate.py {data_file}"
```

MLflow Projects



MLflow 컴포넌트 – Models

다양한 환경에 머신 러닝 모델을 배포하는 프로세스를 간소화 하도록 설계되었습니다.

모델을 여러 가지 flavors ' 로 저장합니다.
Scikit-learn, TensorFlow, PyTorch, keras, Spark Mllib 등 많은 라이브러리(as model flavors)를 지원합니다.

사용자는 훈련된 모델을 파이프라인의 많은 다운스트림 도구에서 지원되는 표준 형식으로 패키징 할 수 있습니다.

REST API, Docker 컨테이너, 서버리스 기능 등 다양한 옵션을 사용하여 배포할 수 있습니다.

MLflow 모델 디렉터리 예시

```
# Directory written by mlflow.sklearn.save_model(model, "my_model")
my_model/
├── MLmodel
├── model.pkl
├── conda.yaml
├── python_env.yaml
└── requirements.txt
```

지원 라이브러리

Scikit-learn	CatBoost
TensorFlow	Spacy
PyTorch	Fastai
Keras	Statsmodels
Spark MLlib	Prophet
XGBoost	Python function (generic)
LightGBM	R function (generic)

MLflow 컴포넌트 – Registry

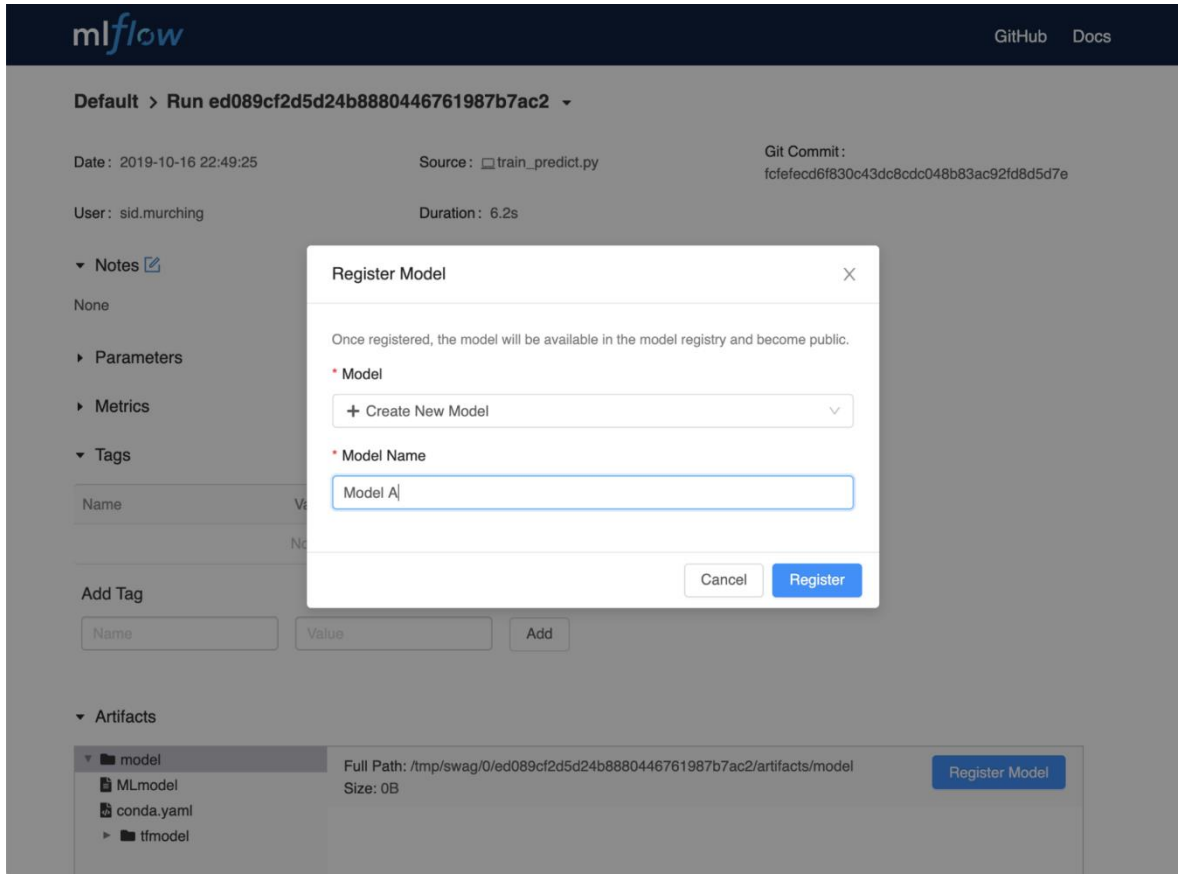
머신러닝 모델, 해당 모델의 버전, 메타데이터를 관리하기 위한 중앙 집중식 리포지토리입니다.

사용자에게 모델의 전체 라이프사이클을 공동으로 관리할 수 있는 일련의 API와 사용자 인터페이스를 제공합니다.

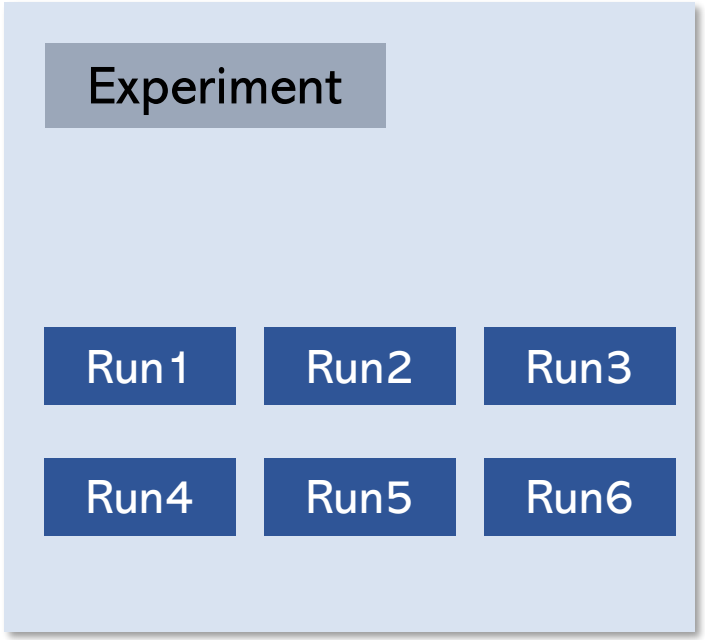
검색 인터페이스를 통해 모델 이름, 메타데이터, 기타 기준으로 모델을 검색할 수 있어, 적합한 모델을 빠르게 찾을 수 있습니다.

각 모델 버전에 사람 이름, 생성 데이터, 설명 등과 같은 메타 데이터를 추가할 수 있고, 빠르게 모델을 찾는 데 도움이 됩니다.

MLflow UI를 사용하여 모델 등록하기



Experiment(실험)



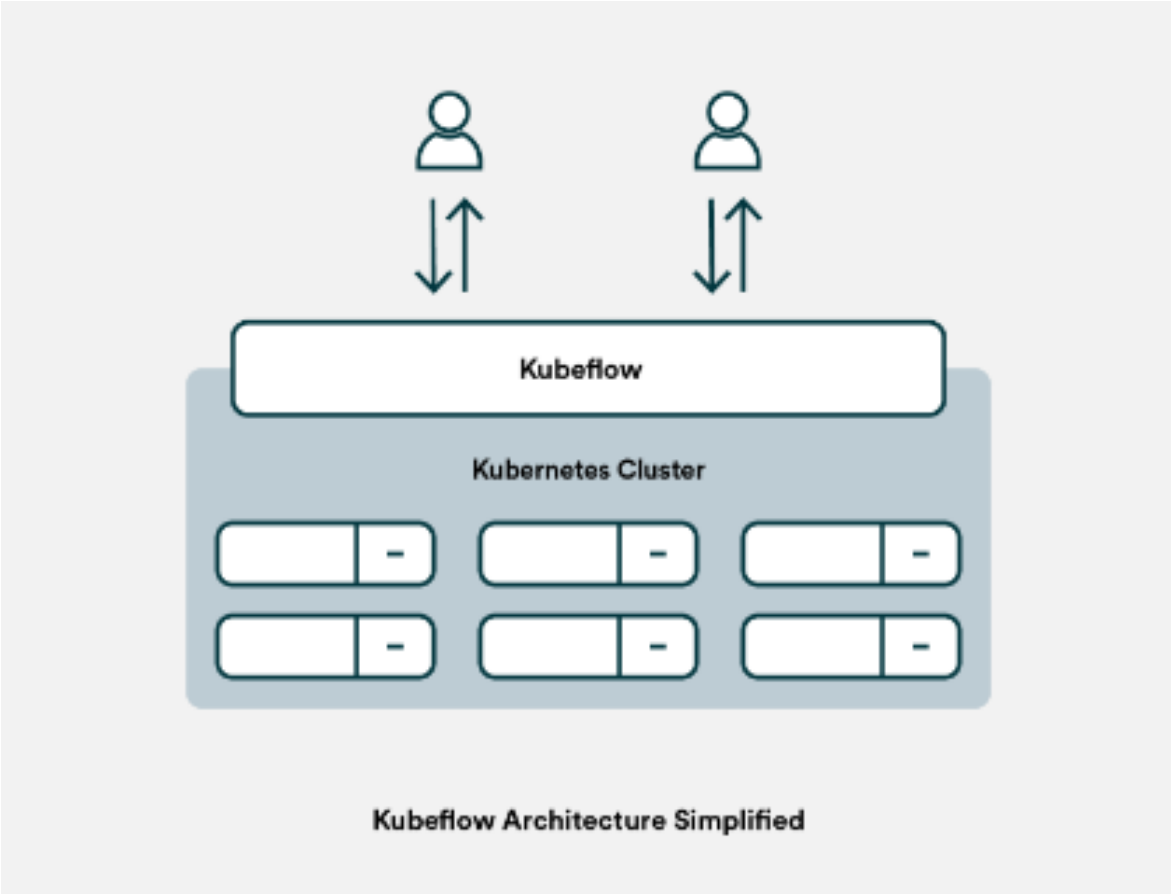
Experiment는 논리적인 실행(run)의 그룹입니다.
실행 그룹을 구성하고 비교할 수 있습니다.

실행(Run)은 코드의 단일 실행을 의미합니다.
각 실행은 코드 버전, 하이퍼 파라미터, 메트릭, 태그 등을 기록할 수 있습니다.

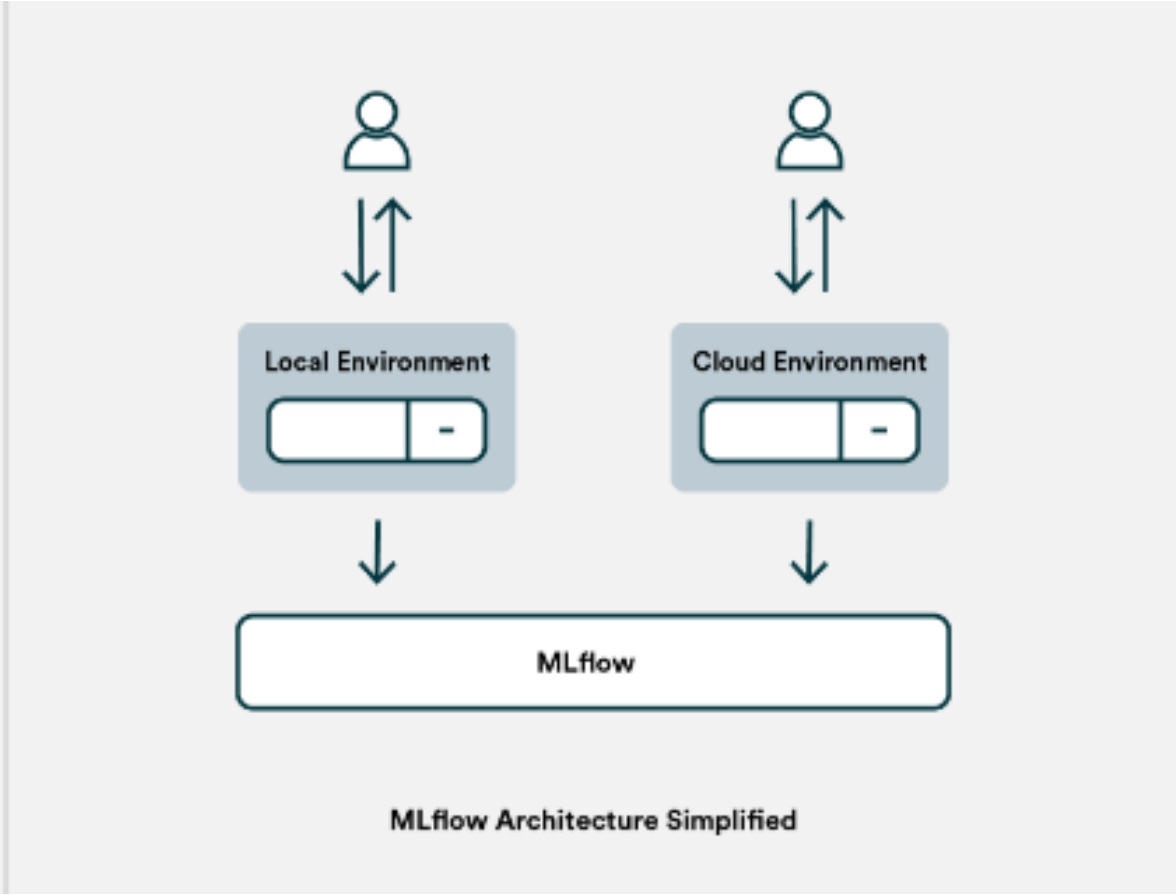
```
exp = mlflow.set_experiment(experiment_name="experment_1")
```

MLflow와 Kubeflow 비교

Kubeflow는 Kubernetes에서 머신러닝 애플리케이션을 개발, 배포, 관리하기 위한 컨테이너 오케스트레터입니다.
MLflow는 오케스트레이션 환경과 관계없이 실험 추적 및 모델 버전 관리/배포를 위한 Python 라이브러리입니다.



Kubeflow Architecture Simplified



MLflow Architecture Simplified

MLflow 설치 및 셋업

■ Windows

- **Python 가상환경 생성 : python -m venv py311**
- **가상환경 실행 : py311\Scripts\activate.bat**

Mlflow 설치 : pip install mlflow

■ macOS

- **Python 가상환경 생성 : python3 -m venv py311**
- **가상환경 실행 : source py311/bin/activate**
- **Mlflow 설치 : pip3 install mlflow**

■ 리눅스

- **Pyenv 설치**

```
curl https://pyenv.run | bash
```

- **pipenv 설치**

```
sudo apt install python3-pip
sudo pip3 install pipenv virtualenv
```

- **Mlflow 설치**

```
pipenv shell
pipenv --python 3.11
pipenv shell
pipenv install mlflow
```

회귀 모델 BasicMLCode.py

BasicMLCode.py > ...

```python
import warnings
import argparse
import logging
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet

logging.basicConfig(level=logging.WARN)
logger = logging.getLogger(__name__)

# get arguments from command
parser = argparse.ArgumentParser()
parser.add_argument("--alpha", type=float, required=False, default=0.5)
parser.add_argument("--l1_ratio", type=float, required=False, default=0.5)
args = parser.parse_args()


# evaluation function
def eval_metrics(actual, pred):
    rmse = np.sqrt(mean_squared_error(actual, pred))
    mae = mean_absolute_error(actual, pred)
    r2 = r2_score(actual, pred)
    return rmse, mae, r2

```

회귀 모델

```python
# evaluation function
def eval_metrics(actual, pred):
    rmse = np.sqrt(mean_squared_error(actual, pred))
    mae = mean_absolute_error(actual, pred)
    r2 = r2_score(actual, pred)
    return rmse, mae, r2


if __name__ == "__main__":
    warnings.filterwarnings("ignore")
    np.random.seed(40)

    # Read the wine-quality csv file from local
    data = pd.read_csv("red-wine-quality.csv")
    data.to_csv("data/red-wine-quality.csv", index=False)

    # Split the data into training and test sets. (0.75, 0.25) split.
    train, test = train_test_split(data)

    # The predicted column is "quality" which is a scalar from [3, 9]
    train_x = train.drop(["quality"], axis=1)
    test_x = test.drop(["quality"], axis=1)
    train_y = train[["quality"]]
    test_y = test[["quality"]]
```

회귀 모델

```
45      alpha = args.alpha
46      l1_ratio = args.l1_ratio
47      print("args", args)
48      print(alpha, l1_ratio)
49
50      lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
51      lr.fit(train_x, train_y)
52
53      predicted_qualities = lr.predict(test_x)
54
55      (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
56
57      print("Elasticnet model (alpha={:f}, l1_ratio={:f}):".format(alpha, l1_ratio))
58      print("  RMSE: %s" % rmse)
59      print("  MAE: %s" % mae)
60      print("  R2: %s" % r2)
```

회귀모델 - MLflow 사용 BasicMLCode_MLflow.py

BasicMLCode_MLflow.py

BasicMLCode_MLflow.py > eval_metrics

```python
import warnings
import argparse
import logging
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet
import mlflow
import mlflow.sklearn

logging.basicConfig(level=logging.WARN)
logger = logging.getLogger(__name__)

# get arguments from command
parser = argparse.ArgumentParser()
parser.add_argument("--alpha", type=float, required=False, default=0.7)
parser.add_argument("--l1_ratio", type=float, required=False, default=0.7)
args = parser.parse_args()

```

회귀모델 - MLflow 사용

```
21
22  # evaluation function
23  def eval_metrics(actual, pred):
24      rmse = np.sqrt(mean_squared_error(actual, pred))
25      mae = mean_absolute_error(actual, pred)
26      r2 = r2_score(actual, pred)
27      return rmse, mae, r2
28
29
30  if __name__ == "__main__":
31      warnings.filterwarnings("ignore")
32      np.random.seed(40)
33
34      # Read the wine-quality csv file from local
35      data = pd.read_csv("red_wine_quality.csv")
36
37      # Split the data into training and test sets. (0.75, 0.25) split.
38      train, test = train_test_split(data)
39
40      # The predicted column is "quality" which is a scalar from [3, 9]
41      train_x = train.drop(["quality"], axis=1)
42      test_x = test.drop(["quality"], axis=1)
43      train_y = train[["quality"]]
44      test_y = test[["quality"]]
45
```

회귀모델 - MLflow 사용

```
46      alpha = args.alpha
47      l1_ratio = args.l1_ratio
48      exp = mlflow.set_experiment(experiment_name="experment_1")
49
50      with mlflow.start_run(experiment_id=exp.experiment_id):
51          lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
52          lr.fit(train_x, train_y)
53
54          predicted_qualities = lr.predict(test_x)
55
56          (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
57
58          print("Elasticnet model (alpha={:f}, l1_ratio={:f}):".format(alpha, l1_ratio))
59          print("  RMSE: %s" % rmse)
60          print("  MAE: %s" % mae)
61          print("  R2: %s" % r2)
62
63          mlflow.log_param("alpha", alpha)
64          mlflow.log_param("l1_ratio", l1_ratio)
65
66          mlflow.log_metric("rmse", rmse)
67          mlflow.log_metric("r2", r2)
68          mlflow.log_metric("mae", mae)
69
70          mlflow.sklearn.log_model(lr, "mymodel")
```

Thank you