

konduit-serving と Spring Cloud Data Flow を用いたMLアプリケーション開発

スカイマインド株式会社

本橋 和貴

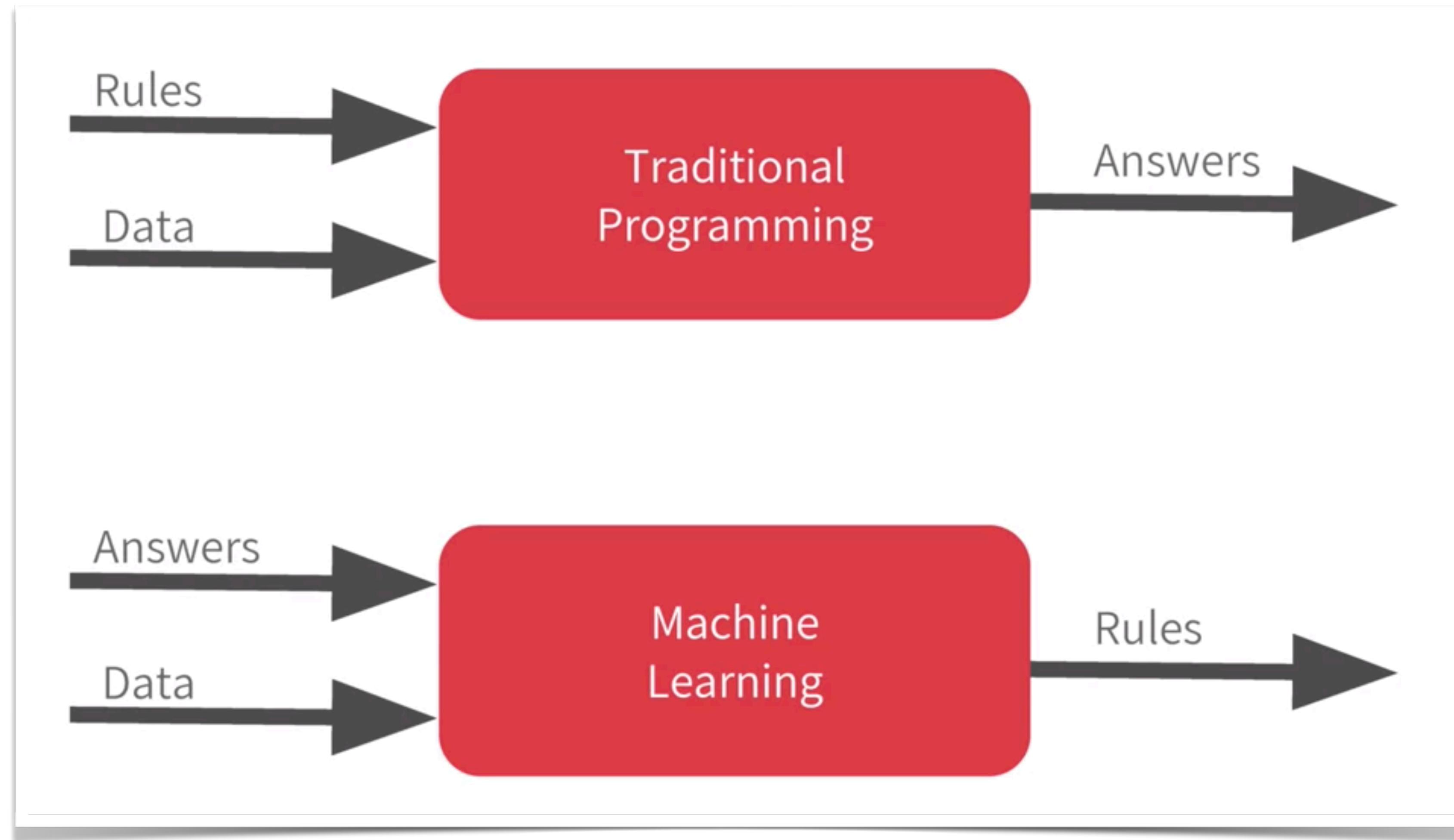
自己紹介

▶ 本橋 和貴 @kmotohas

- スカイマインド株式会社
 - Deep Learning Engineer (前職ではDL+ROS)
- 素粒子物理学実験 (LHC-ATLAS実験) 出身
 - 博士 (理学)
- 好きな本 : 詳説 Deep Learning – 実務者のためのアプローチ



従来のプログラミングと機械学習



<https://www.coursera.org/learn/introduction-tensorflow>

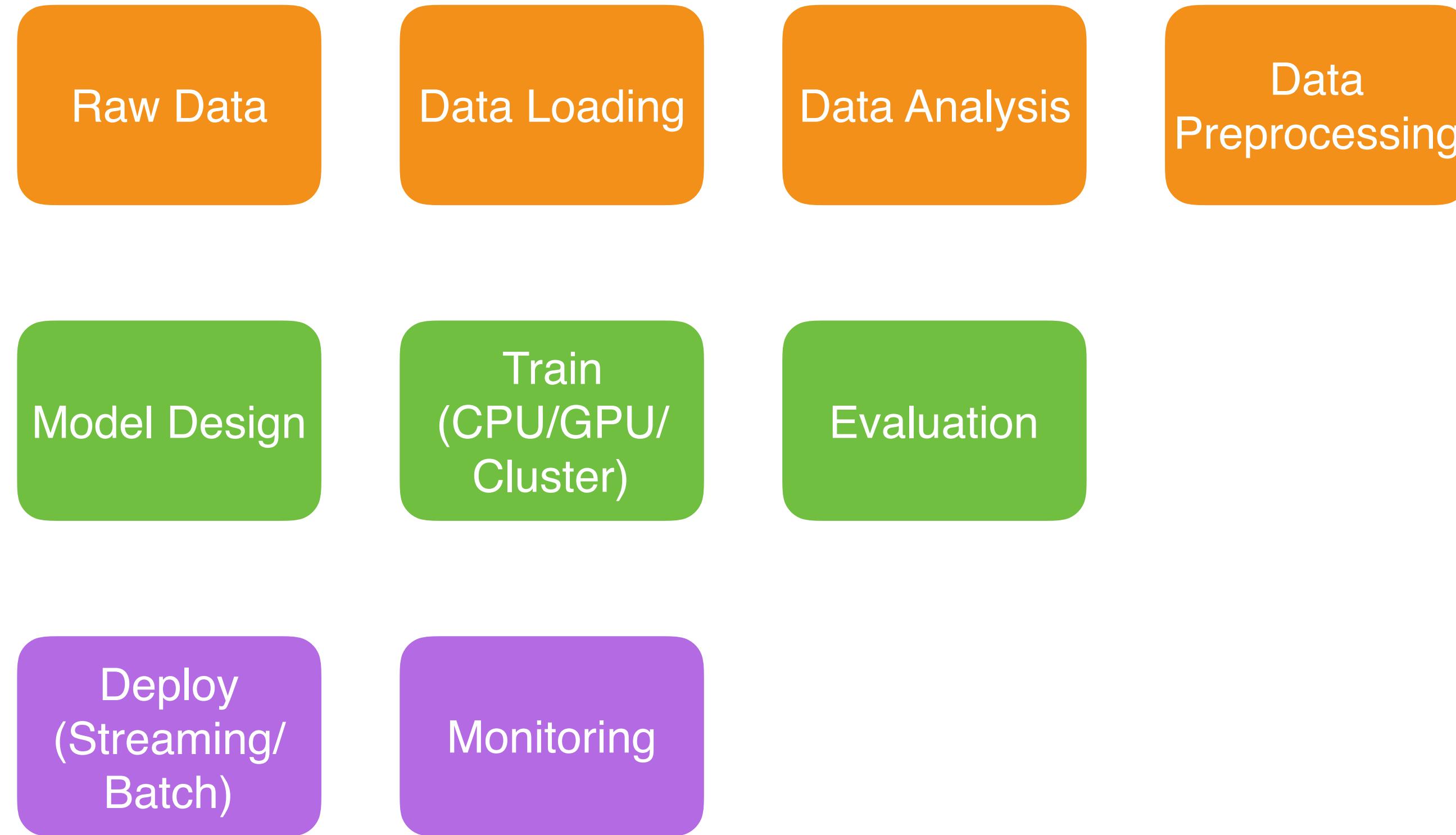
機械学習の時代

- ▶ ルールの記述が難しかった事象も自動化できる可能性がてきた
 - 觀測
 - 実験
 - 解析

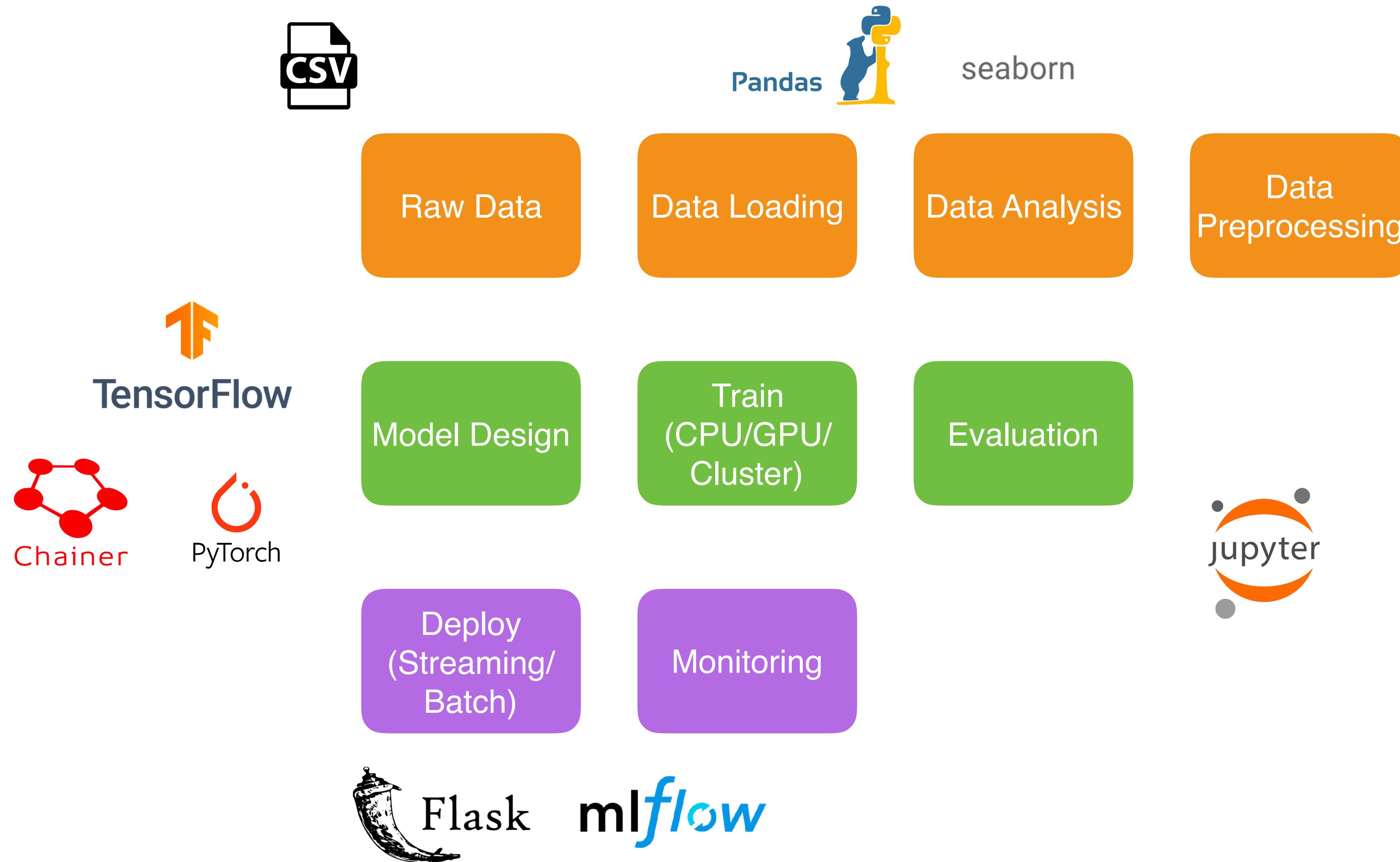
機械学習のライフサイクル

- ▶ Phase 1: 収集したデータセットで機械学習モデルを訓練
- ▶ Phase 2: 訓練したモデルをシステムに統合
- ▶ (Phase 3: 新たに収集したデータセットで再学習)

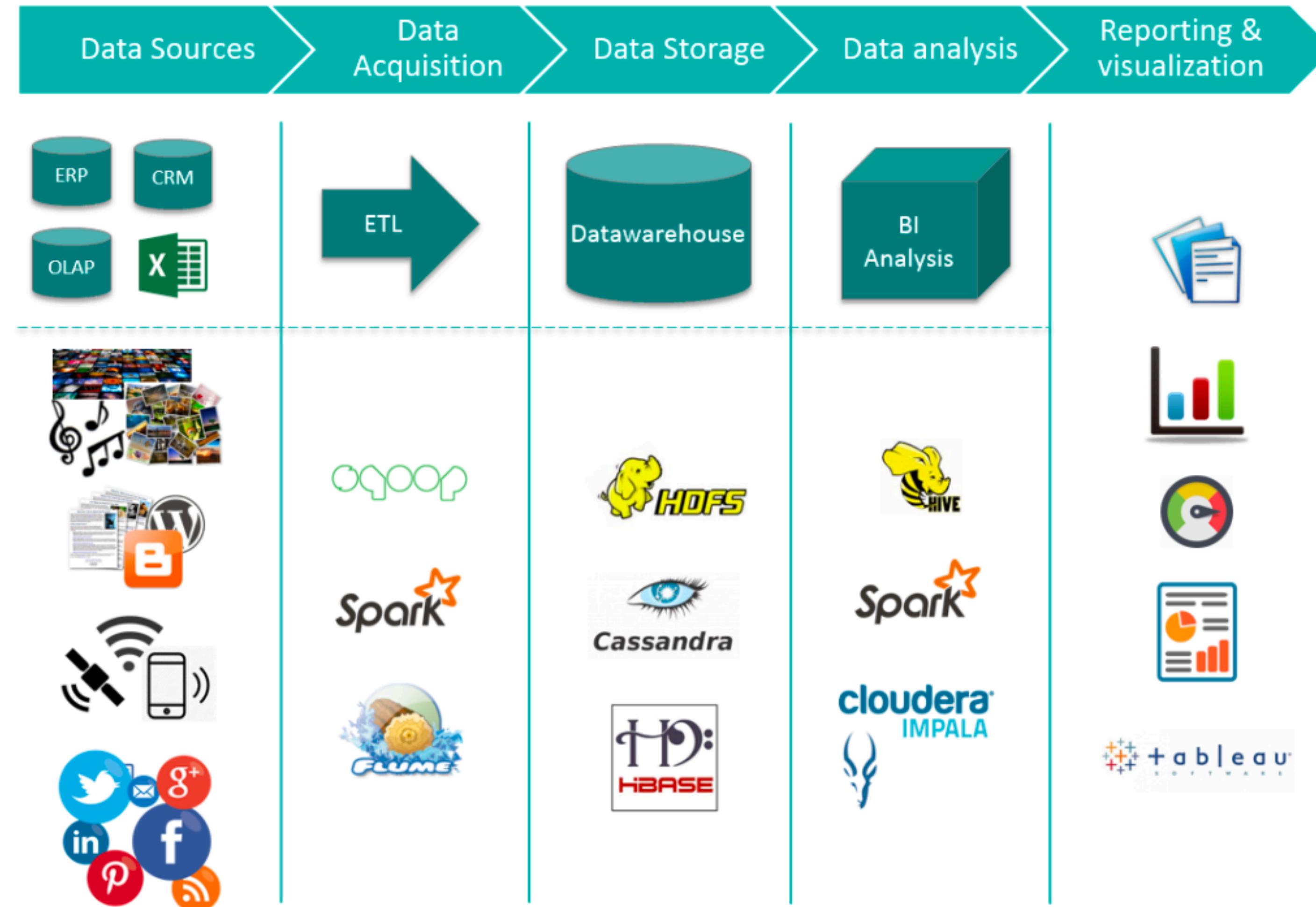
機械学習のライフサイクル



Python ML Ecosystem

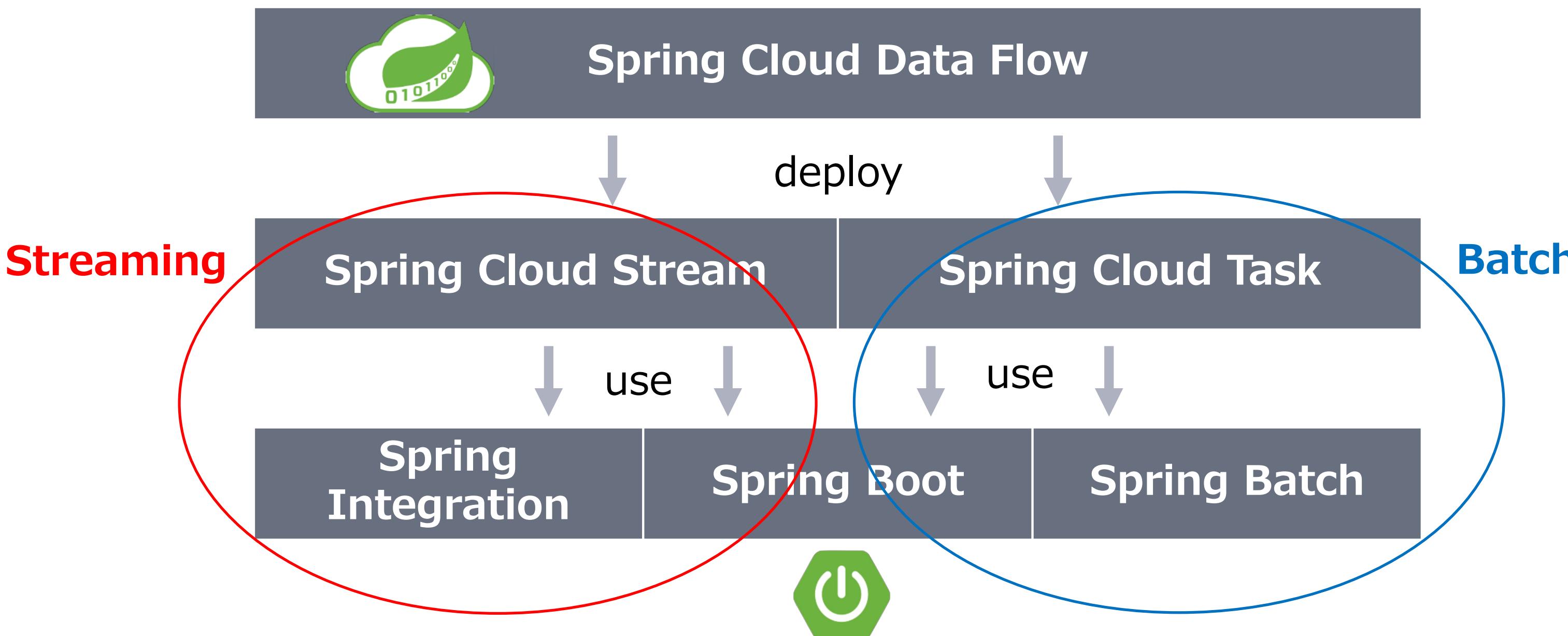


Big Data Ecosystem



<https://www.clearpeaks.com/big-data-ecosystem-spark-and-tableau/>

- ストリーム処理とバッチ処理に関する
Springアプリケーションの統合デプロイメント

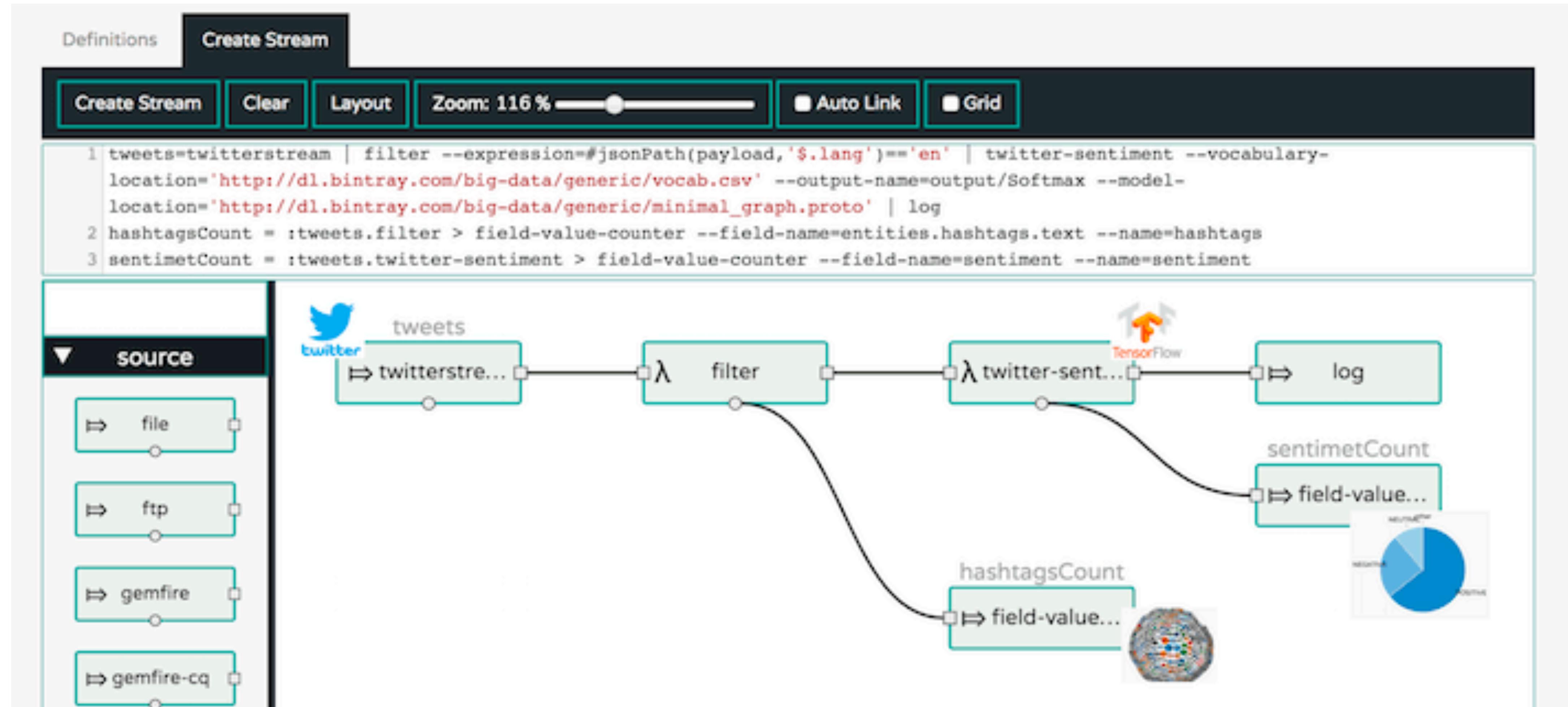


Copyright (C) 2016 Yahoo Japan Corporation. All Rights Reserved. 無断引用・転載禁止

5

<https://www.slideshare.net/techblogyahoo/spring-cloud-data-flow-streamctjp>

GUIベースでワークフローを定義可能



<https://github.com/spring-cloud-stream-app-starters/tensorflow/tree/master/spring-cloud-starter-stream-processor-twitter-sentiment>

Source -> Processor -> Sink

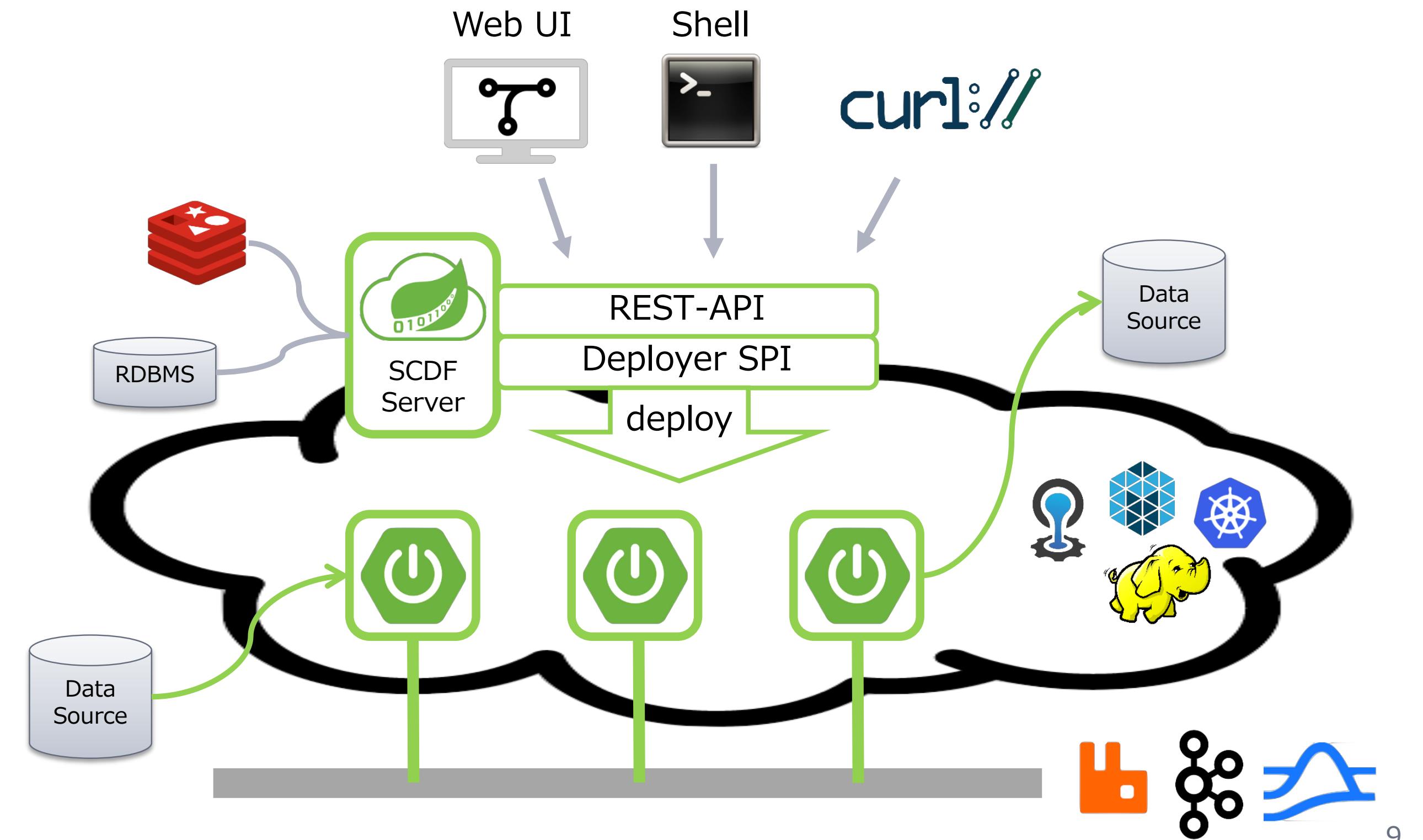


各コンポーネントはマイクロサービス

- ▶ 様々な環境下で動作
 - Cloud Foundry
 - Kubernetes
 - Hadoop YARN
 - Apache Mesos
 - ローカルコンピューター

Spring Cloud Data Flow とは

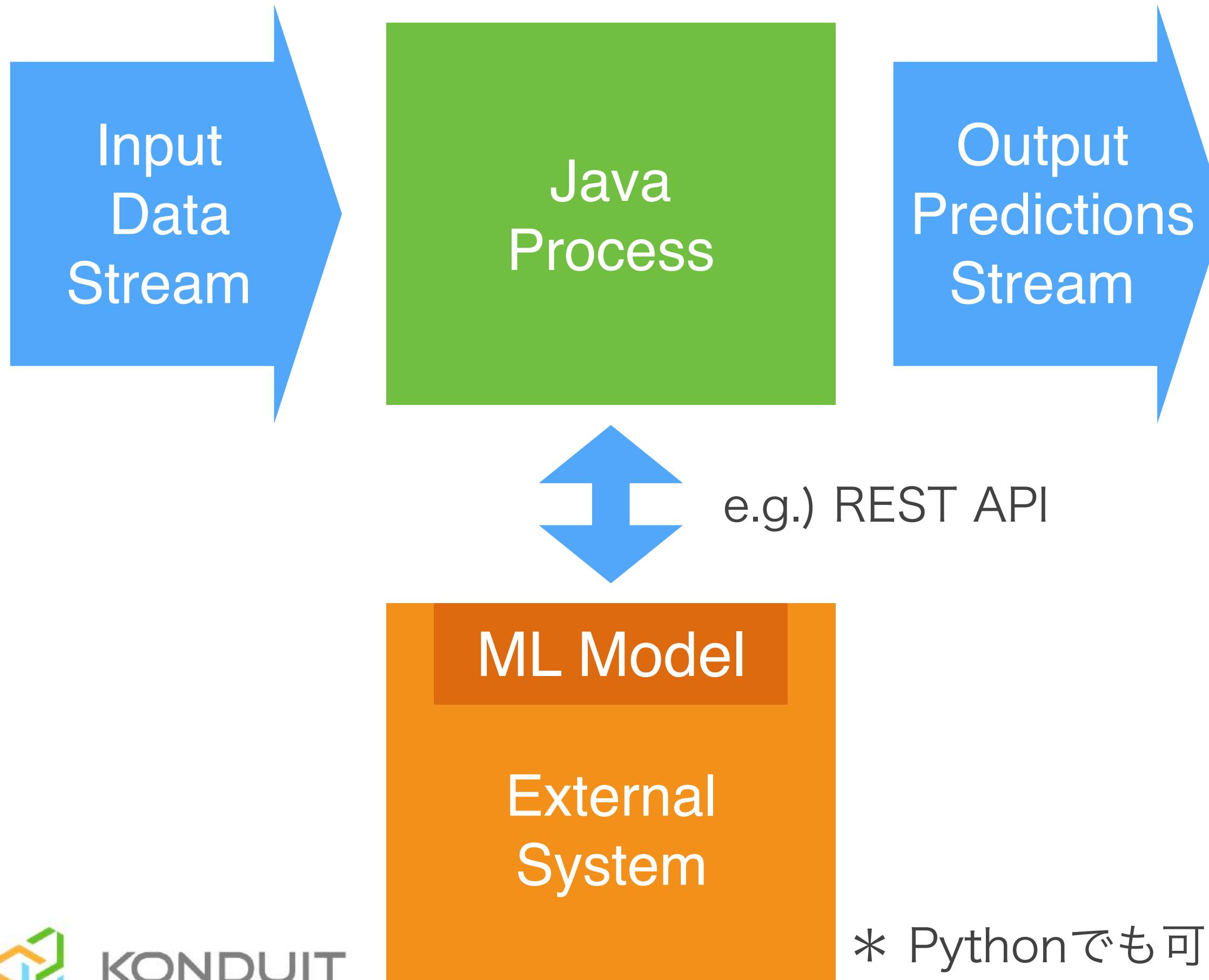
YAHOO!
JAPAN



<https://www.slideshare.net/techblogyahoo/spring-cloud-data-flow-streamctjp>

モデル推論の構成

Model Serving



Model Embedding



e.g.) TensorFlow frozen model / PMML / ONNX

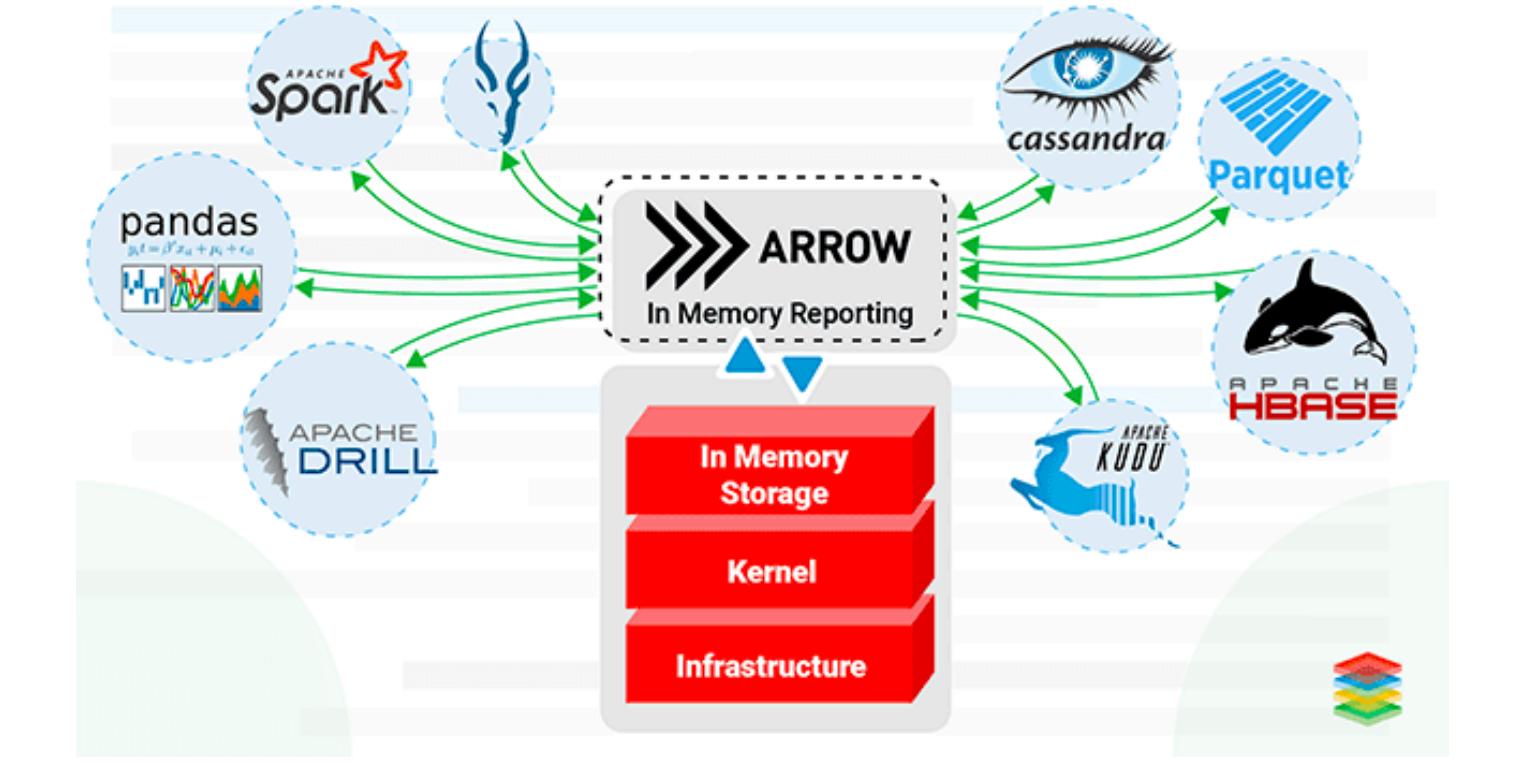
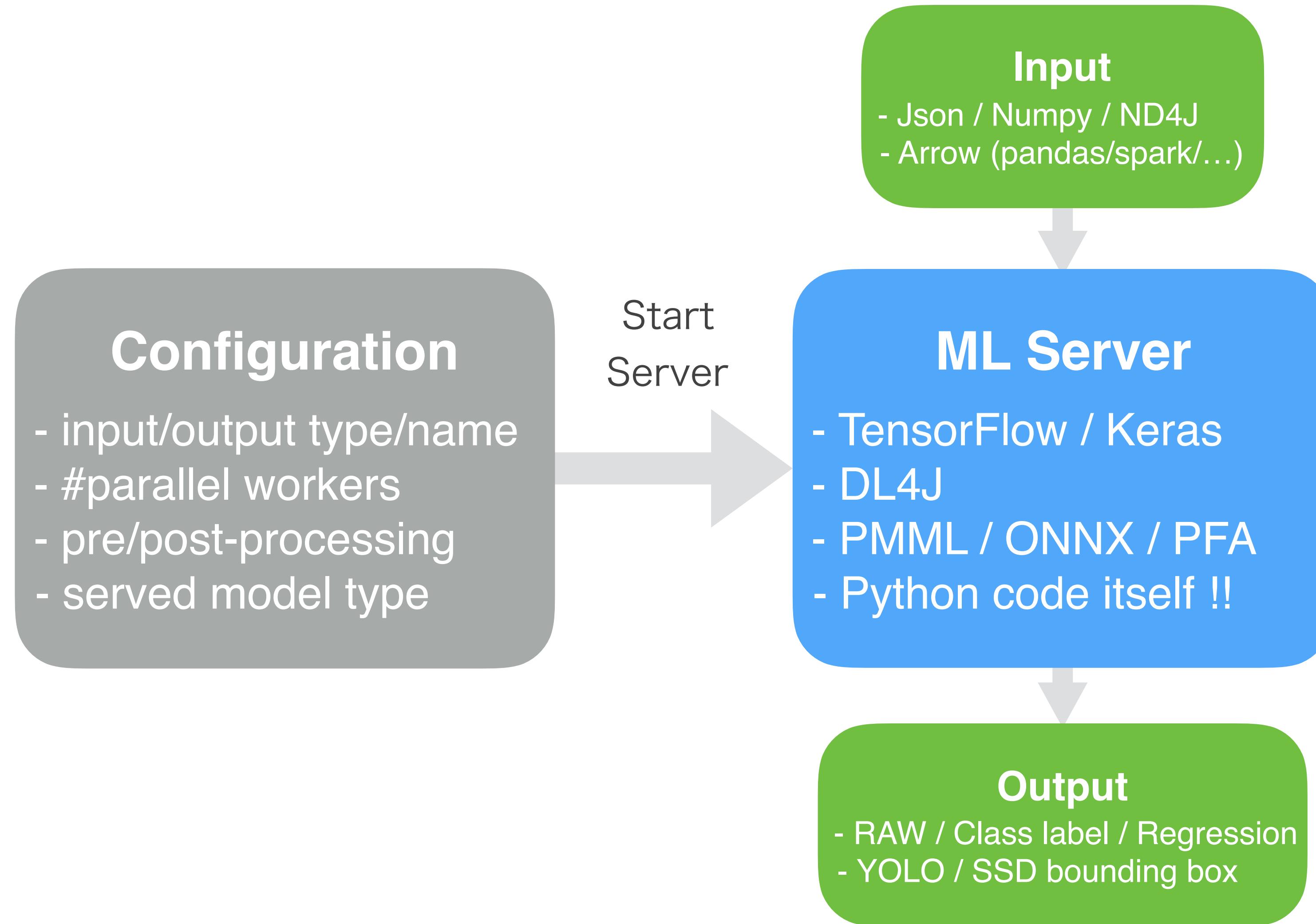
* Pythonで書いていた前処理・後処理をJavaで再記述

* Pythonでも可

Flask / Seldon / **konduit-serving** / etc

- ▶ Deeplearning4j 開発者 Adam Gibson を中心に開発されているJavaベースのオープンソースのMLモデルデプロイツール
 - JavaCPP: C++で書かれたコード (CUDA/TensorFlow/NumPy/etc) のJavaラッパーを作成
 - Vert.x: イベントドリブンアプリ開発フレームワーク
 - Deeplearning4j: JVM用DLフレームワーク、Keras model import
- ▶ Python SDK、Kubernetes統合、Grafana/Prometheus連携モニタリング

konduit-serving Python SDK



<https://www.xenonstack.com/insights/what-is-apache-arrow/>

pip install konduit

```
from konduit import *
serving_config = ServingConfig(..)
model_pipeline_step = ModelPipelineStep(..)
inference_config = InferenceConfiguration(..)

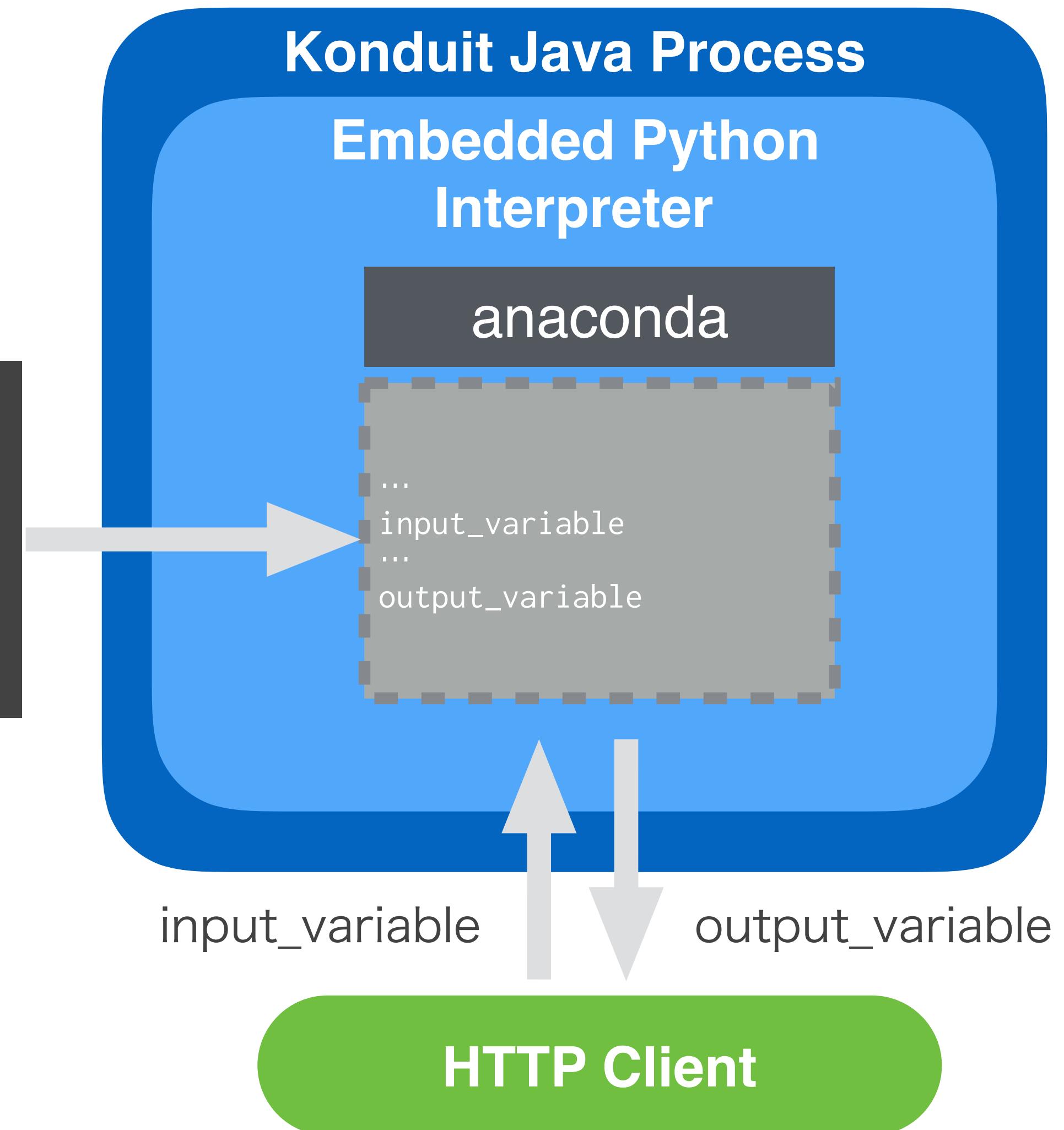
server = Server(config=inference_config)
server.start()

client = Client(..)
result = client.predict(input_data)
```

Embedded Python in Java

Inference Python Code

```
import torch
input_variable = # some tensor
# pre process here
model = load_model()
# post process here
output_variable = model.predict(input_variable)
```



1. configure **PythonPipelineStep**
2. client sends input data to server
3. store input data in memory
4. overwrite it to the pointer of “**input_variable**”
5. do inference
6. client retrieves output variable

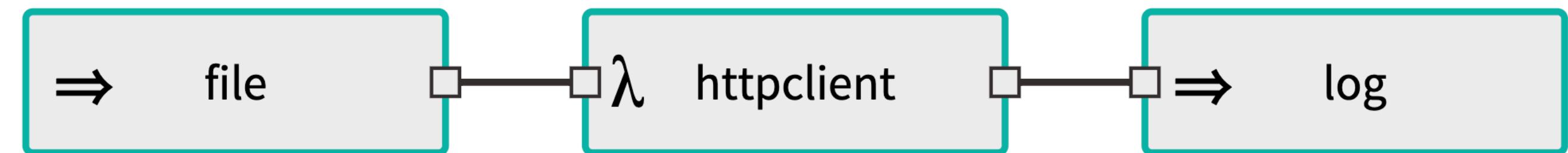
demo

- ▶ PyTorchで記述された顔認識コードを konduit-serving でデプロイ

- <https://github.com/Linzaer/Ultra-Light-Fast-Generic-Face-Detector-1MB>

- ▶ Spring Cloud Data Flow

- “file” source でディレクトリを監視
- 画像ファイルがディレクトリに追加されたら “httpclient” processer が base64 encode された画像をPOST
- Pythonコード内で画像を受け取り、顔認識を行い、画像に写っている顔の数を返す
- “log” sink でログ出力



- ▶ Spring Cloud Data Flow で簡単にデータ処理のフローを構築できる
 - 各コンポーネントは Spring Boot のマイクロサービスとして振る舞う
- ▶ conduit-serving は Java ベースのMLデプロイツール
 - 様々な形式のモデルをサポート
 - Pythonコードそのもの自体デプロイできる
 - “httpClient” を通じて Spring Cloud Data Flow と疎通