삼성 오픈소스 컨퍼런스

Use Case로 살펴보는 Production Deep Learning 서비스 인프라 Eco System

Production AI Pain Point 및 그 해결 방법 사례 공유

SK텔레콤 | AI센터 | 김훈동, 박찬엽

We are …

- Hoon Dong Kim.

- SK Telecom , AI Center , AI Product DevOps Team Leader.
- Microsoft BigData MVP(Most Valuable Professional) : 2016년 ~ 2017년
- Microsoft AI MVP(Most Valuable Professional) : 2018년 ~ 2019년
- Korea Spark User Group (스파크 사용자 모임) 운영진

http://hoondongkim.blogspot.kr [블로그]
https://www.slideshare.net/ssusere94328/ [슬라이드 쉐어]
https://www.facebook.com/kim.hoondong [SNS]

- Chan Yup Park

- SK Telecom , AI Center , AI Product DevOps Team, AI DevOps Engineer
- 툴링에 관심이 많아 docker, kubernetes의 운영/관리/배포등의 과정을 개선하는 일을 맡고 있습니다.
- R로 개발 경험을 시작해서 관련 생태계에 기여하기 위해 노력하고 있습니다.

https://mrchypark.github.io/[블로그]
https://github.com/mrchypark [github]

Agenda

1. Production AI Pain Point & Solution Use Case
2. Production AI Open Source Eco System
3. Production AI DevOps
4. Our Open Source Contribution

1. Production AI Pain Point & Solution Use Case

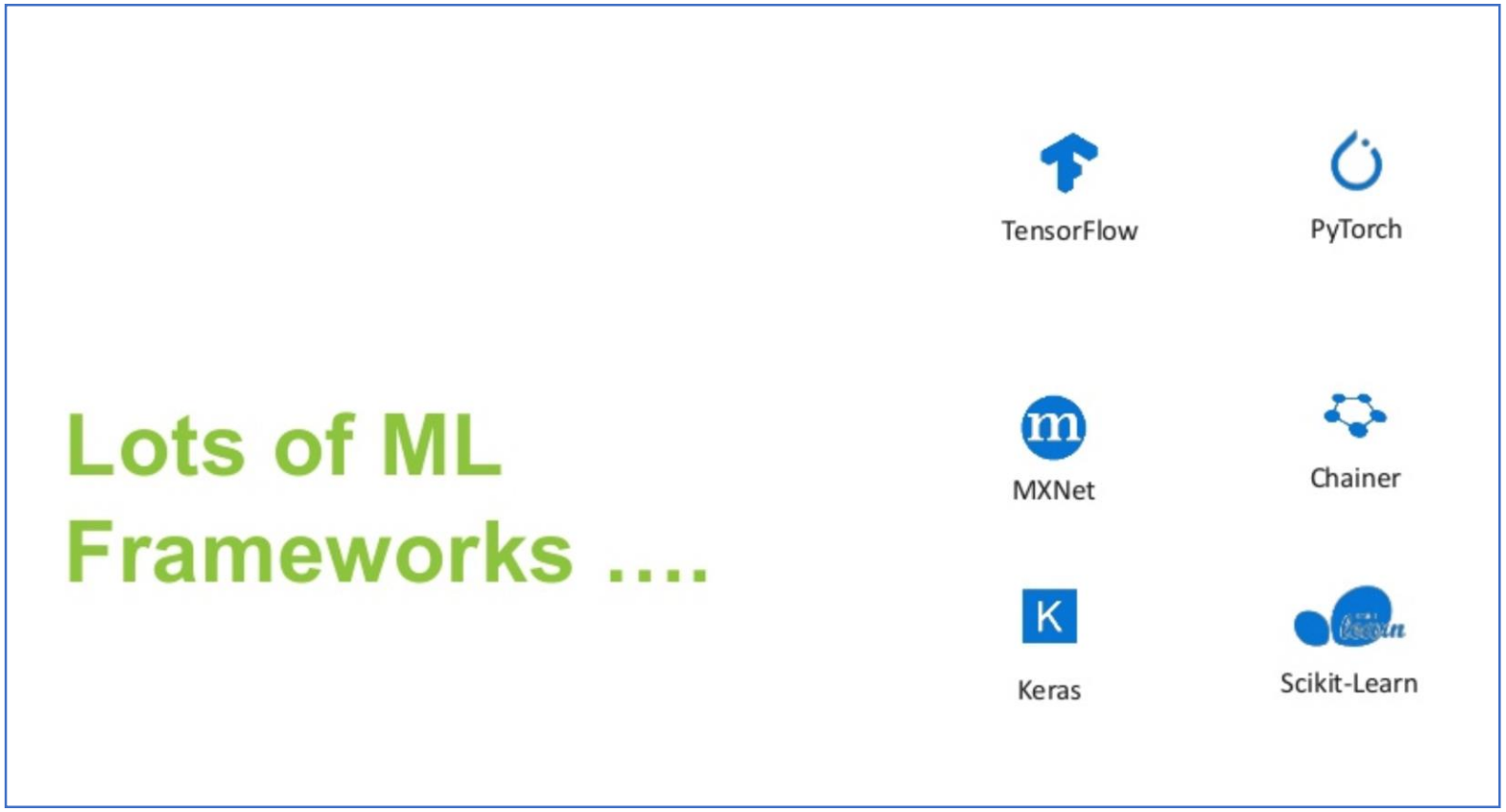
Our Pain Point 1 – ML Serving Dilemma

- [방법1] Tensorflow Serving
  - bazel 빌드 , C++ code, gRPC.
  - python serving performance 좋지 않음.
  - 요즘은 pytorch, mxnet 하시는 분들도 급격히 늘고 있음.
  - Scikit learn 전처리 모델은?
- [방법2] Cloud PaaS
  - Azure Machine learning Service , AWS SageMaker , GCP CloudML
  - 3사가 다 10%가 부족한 부분이 있음.
  - 매우 비쌈.
- [방법3] Flask 등을 이용한 범용 서빙 아키텍처 구성
  - 빠르고 쉽게 prototype 할 수 있으나, production을 하기 위해선 험난한 Engineering Art 가 필요함.
  - Python 은 너무 너무 * 10 느린 언어 임.
- [방법4] Full 사양 GPU 서버 혹은 VM 으로 Serving?
  - 월 4만원 짜리 CPU Docker vs 월 1000만원 짜리 최신 GPU VM

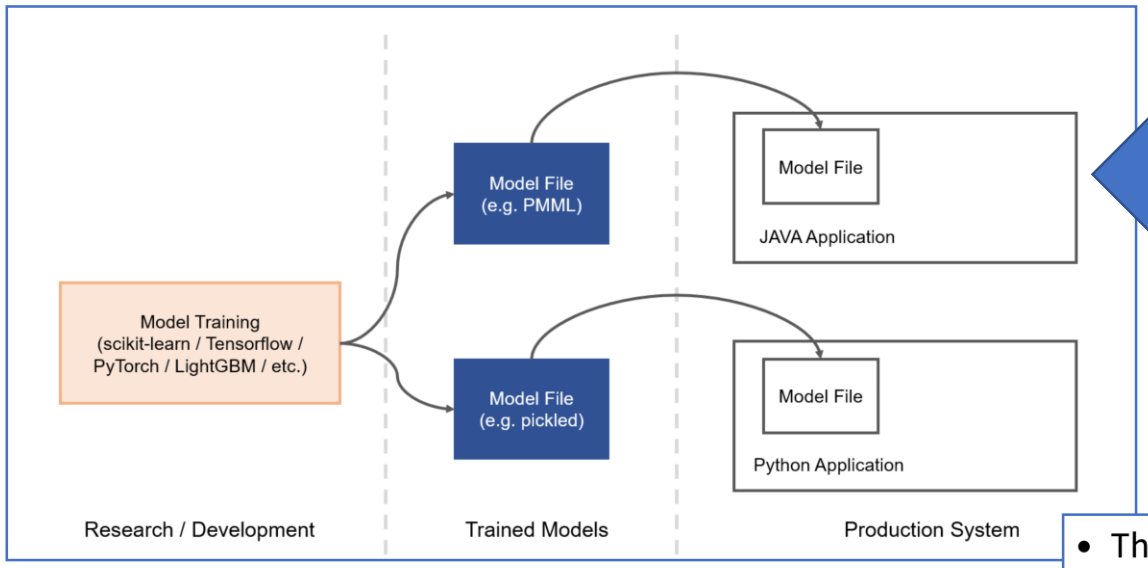
Our Pain Point 1 – ML Serving Dilemma

- ML Serving for Lots of Frameworks

Tensorflow Serving 만 가지고는…

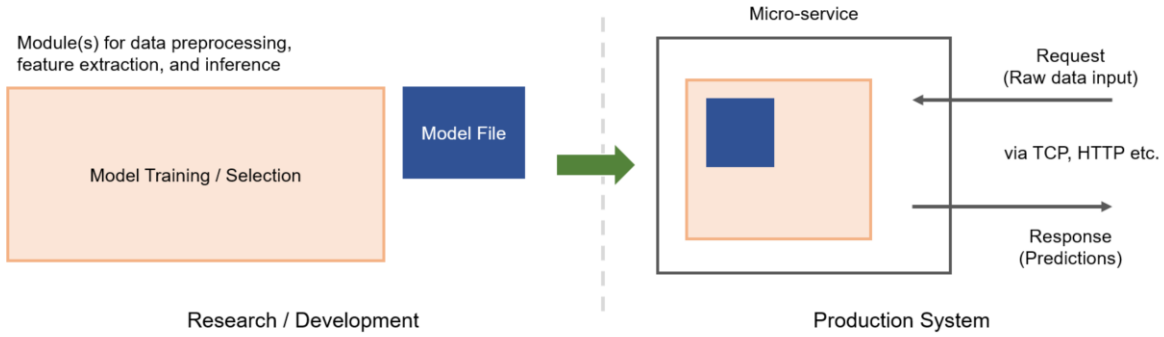


Our Pain Point 1 – 해결 팁



- **Predictive Model Markup Language (PMML)**
  - XML-based file format for model interchange
  - scikit-learn to PMML: sklearn2pmml
  - LightGBM to PMML: jpmml-lightgbm
- References: Converting Scikit-Learn to PMML

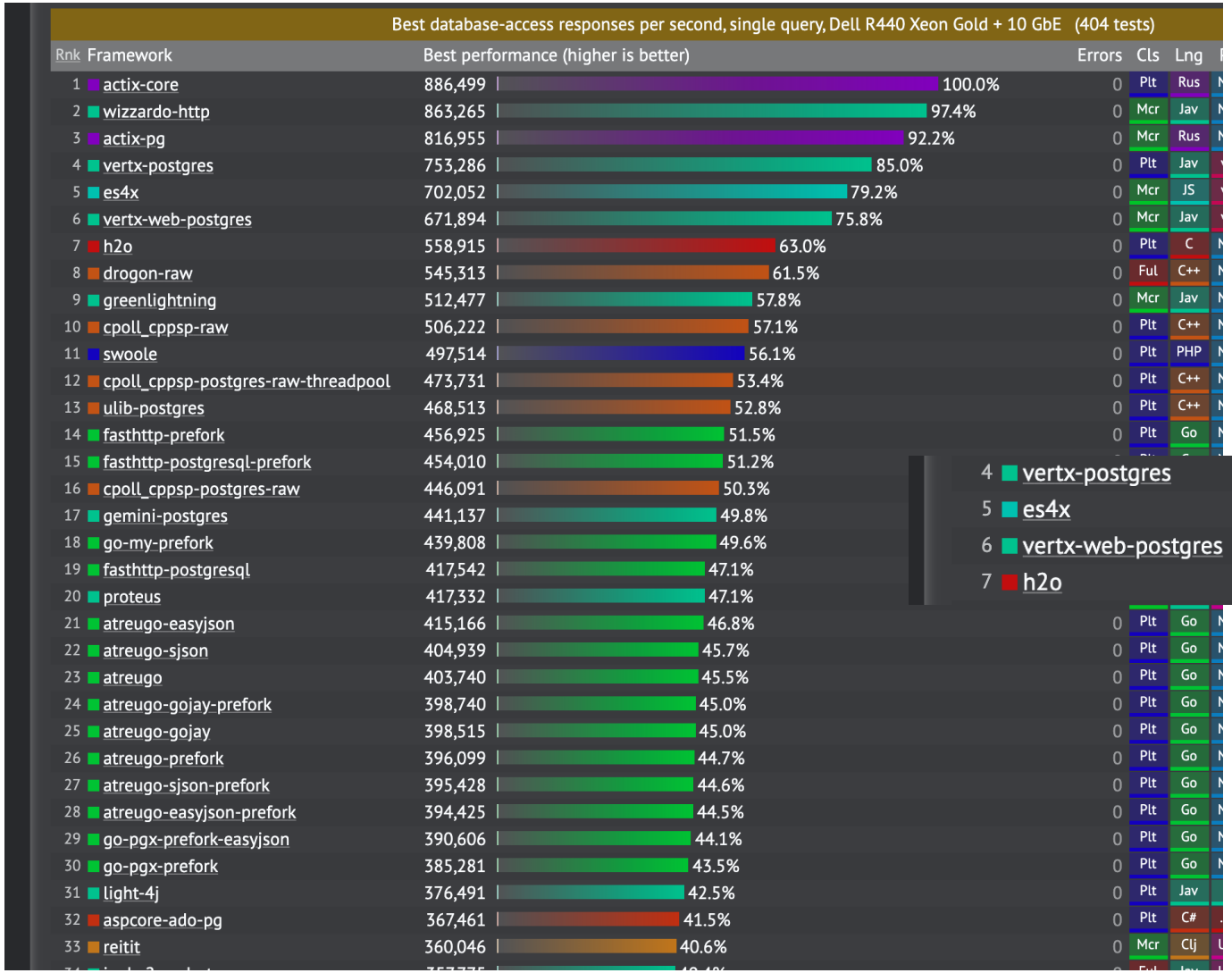
- The model (or even the module to load, pre-process and generate predictions) is **wrapped** in a **micro-service** that expose **endpoints** to receive requests from **clients**

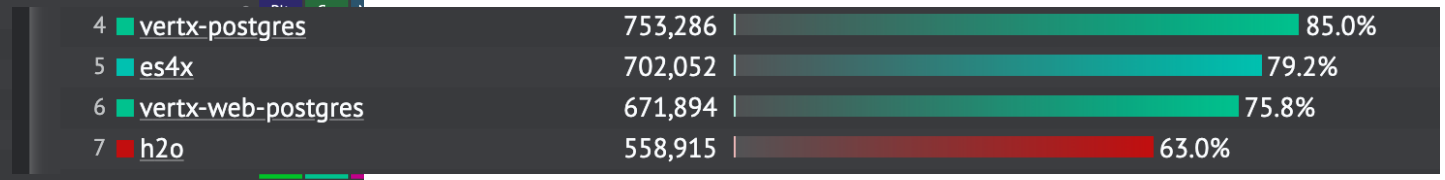


Our Pain Point 1 – ML Serving Dilemma 해결팁

- [방법1] Tensorflow Serving
  - bazel 빌드 , C++ code, gRPC.
  - python serving performance 좋지 않음.
  - 요즘은 pytorch, mxnet 하시는 분들도 급격히 늘고 있음.
  - Scikit learn 전처리 모델은?
- [방법2] Cloud PaaS
  - Azure Machine learning Service , AWS SageMaker , GCP CloudML
  - 3사가 다 10%가 부족한 부분이 있음.
  - 매우 비쌈.
- [방법3] Flask 등을 이용한 범용 서빙 아키텍처 구성
  - 빠르고 쉽게 prototype 할 수 있으나, production을 하기 위해선 **험난한 Engineering Art** 가 필요함.
  - Python 은 너무 너무 * 10 느린 언어 임.
- [방법4] Full 사양 GPU 서버 혹은 VM 으로 Serving?
  - 월 4만원 짜리 CPU Docker vs 월 1000만원 짜리 최신 GPU VM

Our Pain Point 2- Poor Python Performance

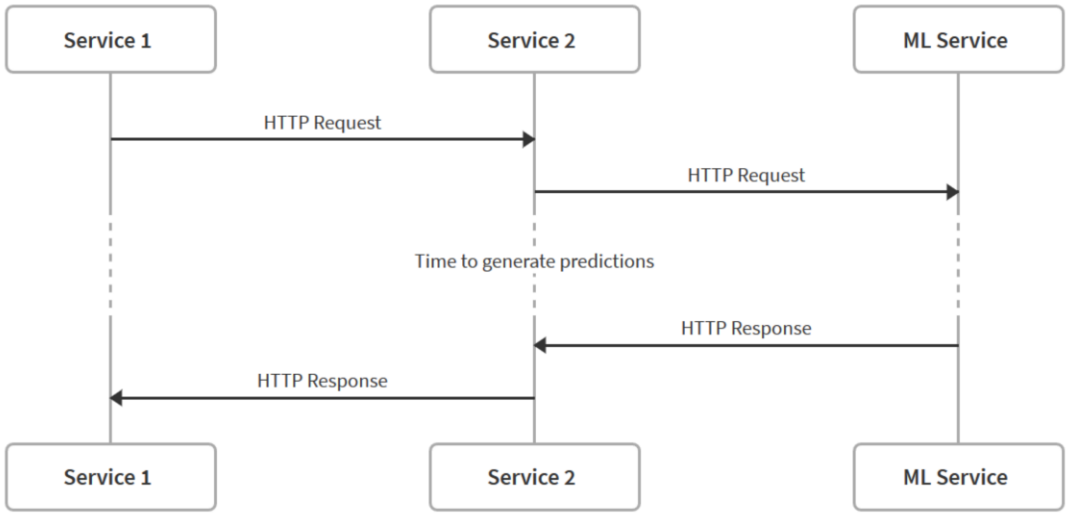




Our Pain Point 2 – Poor Python Performance

- Disadvantage of Using HTTP APIs

  - HTTP REST APIs are simple to implement, however the process is **synchronous**
  - The client must **wait** until the ML service has finished the process of generating predictions
  - In a complex system involving a lot of components, this may not be efficient



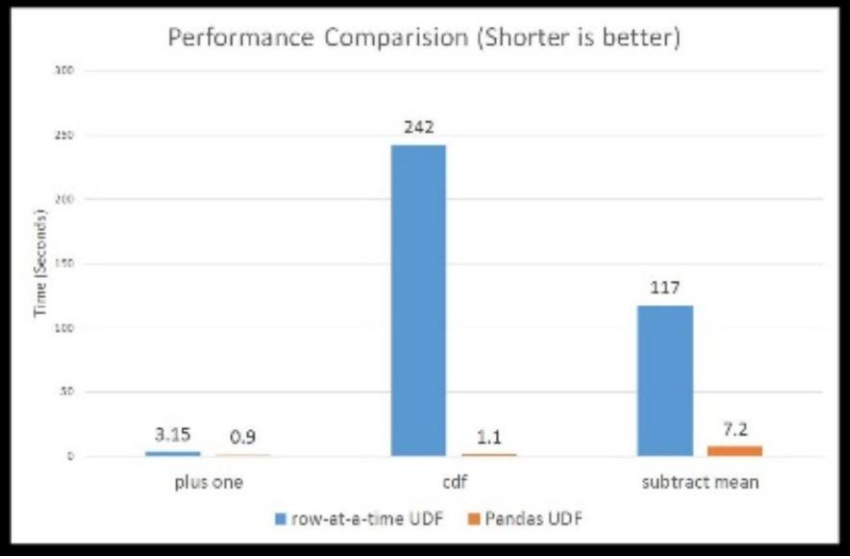
Our Pain Point 2 – 해결 팁

- Poor Python Performance
  - Pandas Data 함수 전달
    -> Data Copy 과정에서의 병목.
    -> Thread , Multi Process 에서 Thread Safe 하지 않음.
  - 함수형 언어 처럼 가상함수를 사용하고 싶지만…
    -> Python 에서 Map, Lambda를 : functionstools.partial
  - Python Thread 는 GIL 때문에 엄청나게 느림.
    -> 차라리 muti-process 가 낫다 : Go 의 Goroutine 쓰듯이 쓰고자 한다면… cotyledon
- MomoryView 활용
  - Data 핸들링 시, pointer 접근 하듯…
  - 대용량 Data Memory 복사 방지
- Microservice 로 잘게 쪼겐다.
- 모든 것은 비동기로…
- PySpark, ScalaSpark…
- PMML + Java 컨버전…

Our Pain Point 2 – 해결 팁

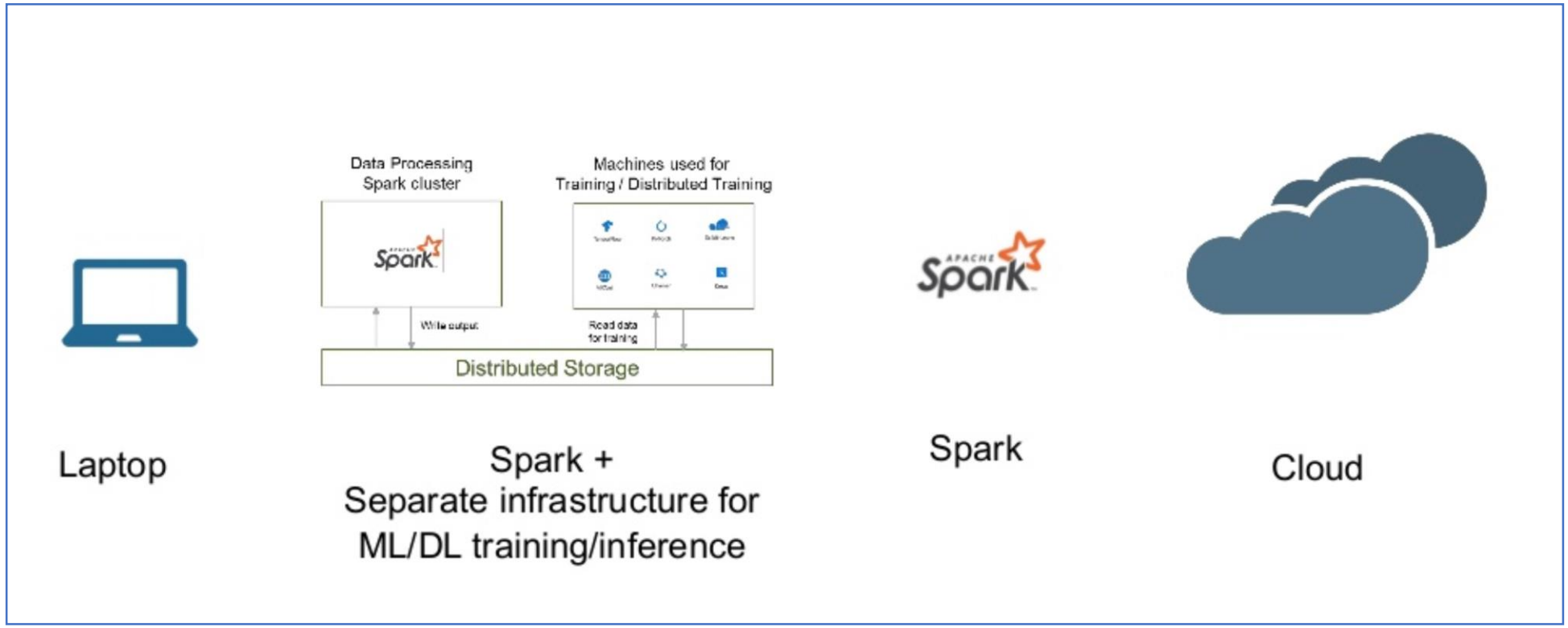
- Pandas : Poor Python Performance -> Pandas UDF

Pandas UDF was introduced in Spark 2.3, which uses Arrow for data exchange and utilizes Pandas for vectorized computation.



Our Pain Point 3

- Multiple System, Multiple Dev-env. , Multiple Production-env.



Our Pain Point 3 – 해결 팁

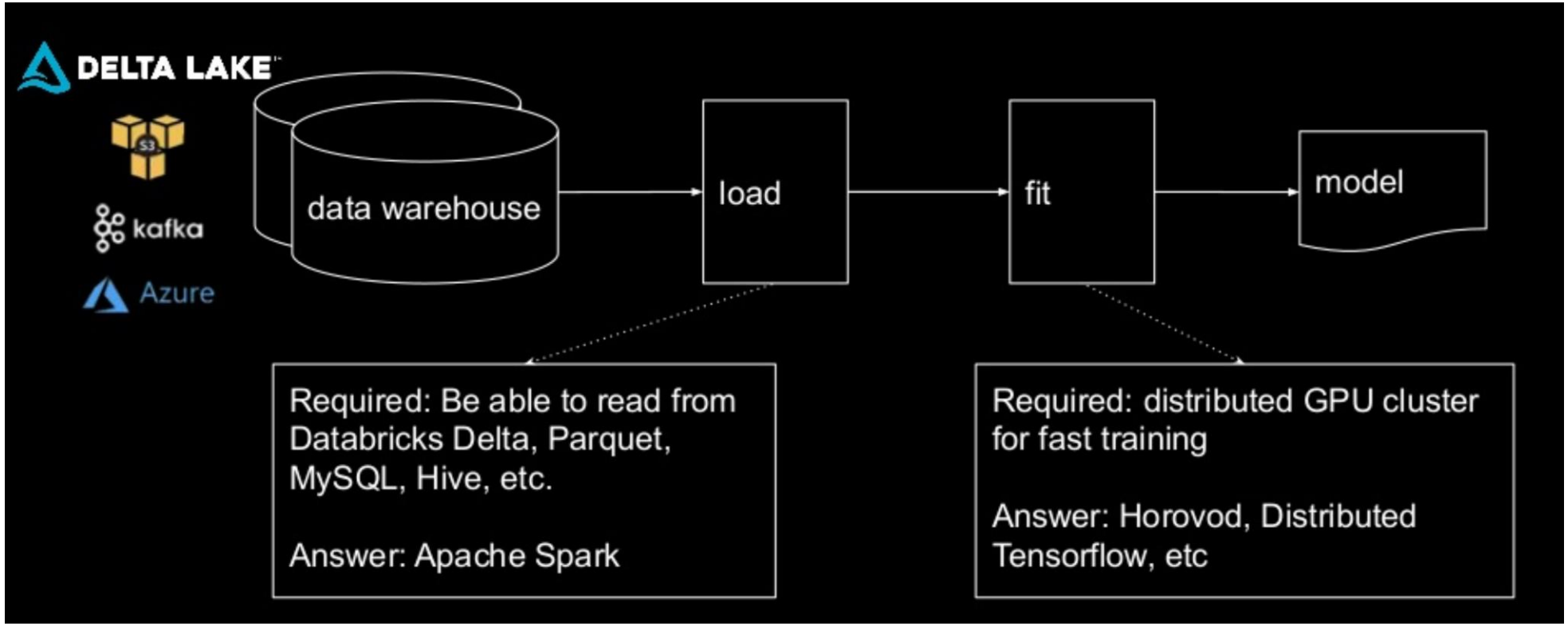
- Auto Scale Out, Auto Scale Down & Env. Abstraction.

Kubernetes is All You Needs.

But, 현실이 녹록치 만은 않습니다.
뒤에서 AI DevOps 부분으로 따로 언급~

Our Pain Point 4

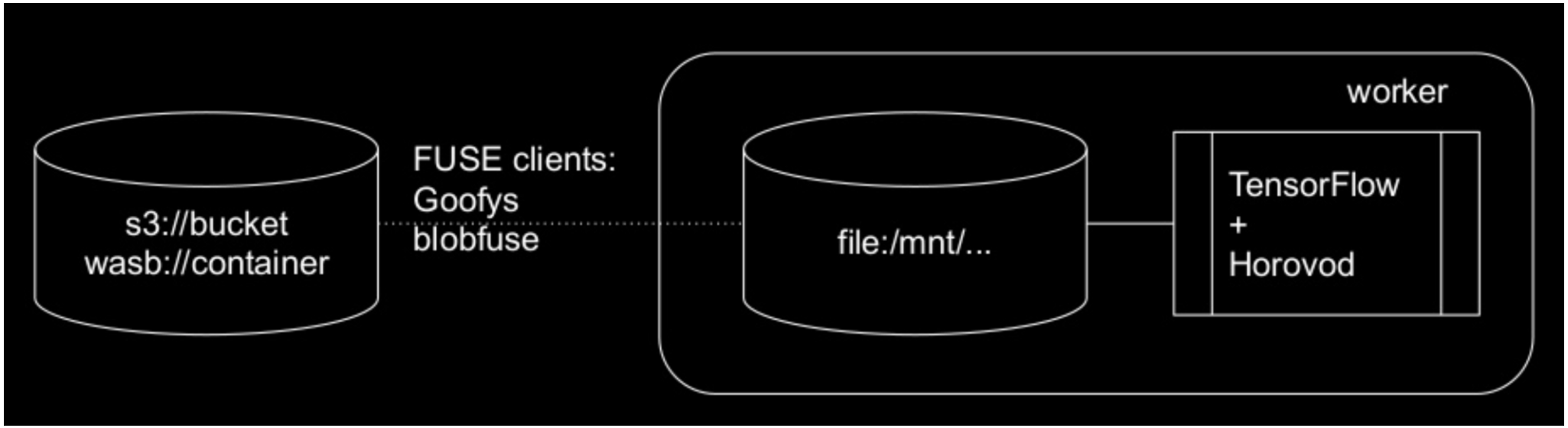
- BigData Scale Data Pre-processing & Distributed Training & Training Performance



자료 출처 : Databricks Web Sites

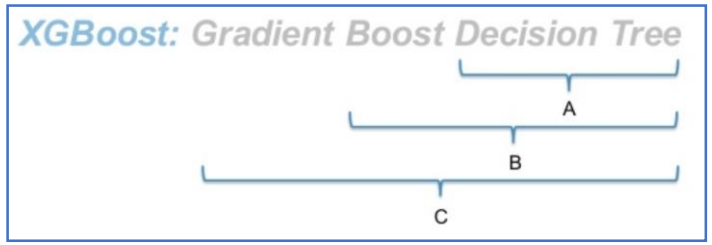
Our Pint Point 4 – 해결 팁

- BigData Scale Data Pre-processing & Distributed Training & Training Performance
- **Recommendation** : High-performance FUSE clients to mount remote storage as local files so DL frameworks can load/save data directly.

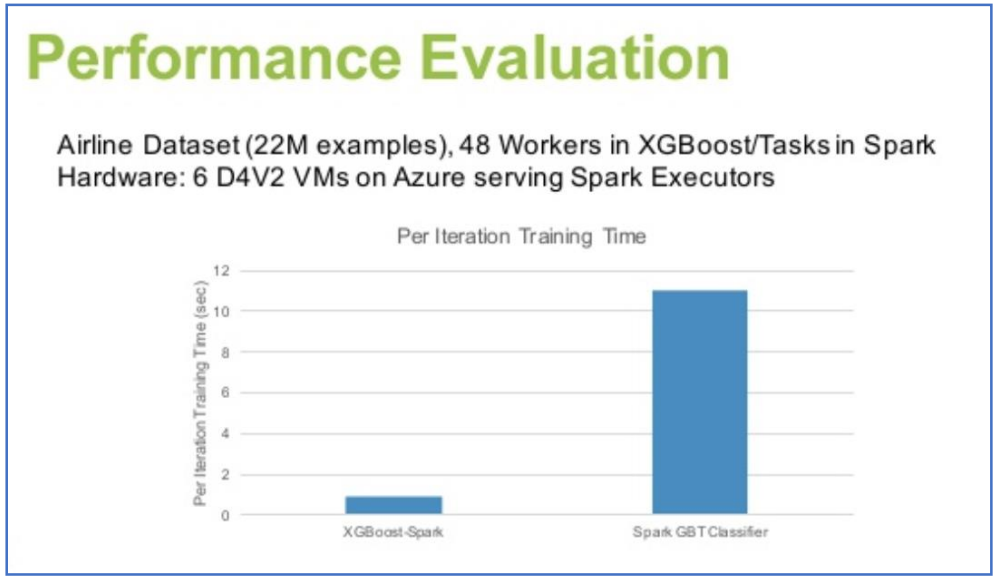


Our Pint Point 4 – 해결 팁

XGBoost in BigData Scale Production



- **XGBoost4J: Distributed XGBoost for Scala/Java**
- XGBoost4J-Spark : XGBoost on Apache Spark
  - **USE Case : Airbnb, Alibaba, eBay, Microsoft, Snapshots, Tencent, Uber, etc…**
- https://github.com/dmlc/xgboost/tree/master/jvm-packages



XGBoost + Spark + GPU

기타,
이런 류의
좀더 많은
Open Source
는 뒤에서
소개...

- XGBoost + Distributed mode via Spark
- XGBoost + GPU support via CUDA
- XGBoost + Multi-GPU via NCCL2
- XGBoost + Spark + GPU : Multi-GPU + Multi Node

Spark Cluster

- Standalone cluster on GCP
- 5 virtual machines, each has:
  - 64 vCPUs (32 physical cores)
  - 416 GB memory
  - 4 x NVIDIA Tesla T4
  - 400 GB SSD persistent disk
  - Default networking
- 4 Spark workers per VM

Sample Code

```
import ml.dmlc.xgboost4j.scala.spark.XGBoostClassifier
val xgbParam = Map("eta" -> 0.1f,
      "max_depth" -> 20,
      "max_leaves" -> 256,
      "grow_policy" -> "lossguide",
      "num_round" -> 100,
      "num_workers" -> 20,
      "nthread" -> 16,
      "tree_method" -> "gpu_hist")
val xgbClassifier = new XGBoostClassifier(xgbParam).
      setFeaturesCol("features").
      setLabelCol("labels")
```

Preliminary Results

		Accuracy (AUC)	Training Loop (Seconds)
Max Tree Depth = 8	CPU	0.832	1071.002
	GPU	0.832	139.641
	Speedup		766.97%
Max Tree Depth = 20	CPU	0.833	1088.662
	GPU	0.833	165.868
	Speedup		656.34%

자료출처 : Spark + AI Summit 2019

Our Pain Point 5 - Serving Cost & 해결 팁

- Low Cost & Agile 접근 방법에 대하여

딥러닝 모델러는 Graph DB, Microservice 를 모르고…,

NoSQL 개발자는 딥러닝을 모르고…

Deep Learning Framework 자체는 여기 까지만 구동 가능

이 단계로 포팅하려면, Deep Learning Output 그래프 노드 값을 Graph DB 화 하거나, Hash DB 등 으로 inverted-Index 화 해야 하며, 고 난이도의 Engineering 작업이 수반 됨.

이 단계로 되면, Allibaba Scale의 Front AI 서비스가 매우 빠르고 매우 저렴하게 적용 가능.

GPU VM > CPU VM > GPU Docker > CPU Docker > K8S PaaS > Docker PaaS > **Docker BaaS** > ML BaaS > **Serverless Microservice**

GPU on-premise

1 GPU VM ,
1 달에 얼마?

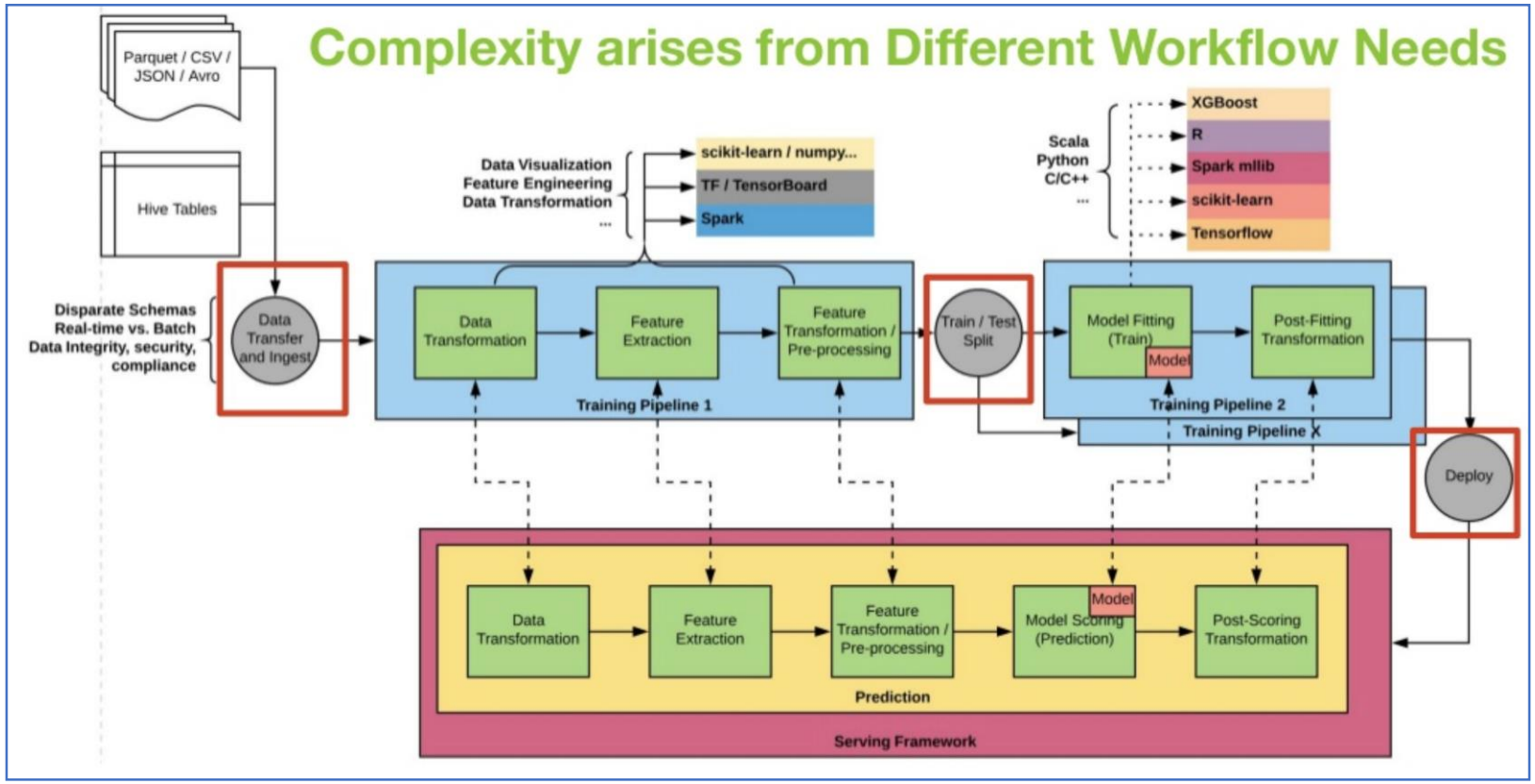
1 Docker ,
1시간에 얼마?
1달에 얼마?

1,000,000 Transaction 에 얼마???

Production AI 는 모델링 보다 Engineering 이 더 어렵다.
Production AI 는 Engineering Art 에 가깝다.
Cloud 를 잘 활용한다면, 많은 도움이 된다!

Our Pain Point 6

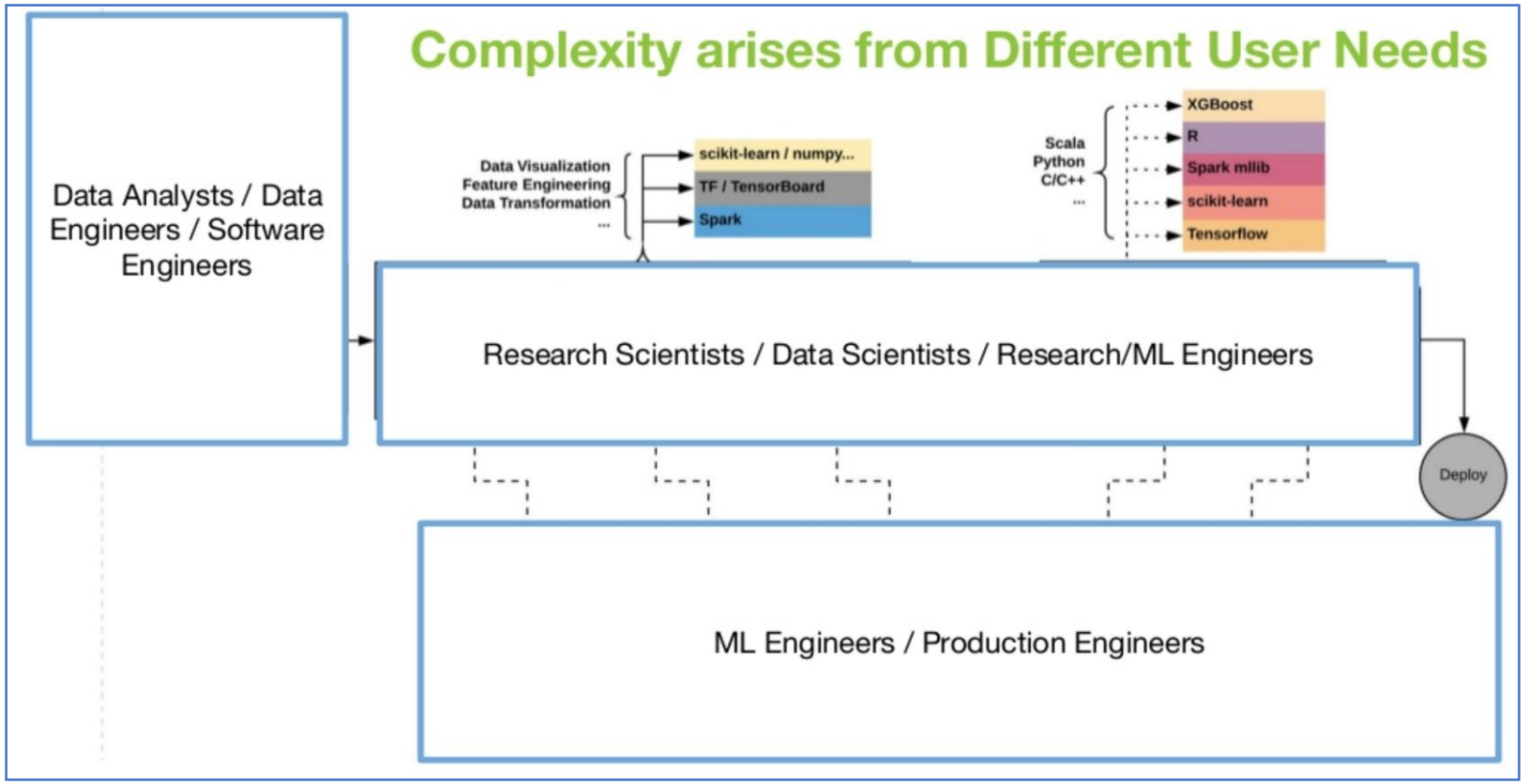
- None-Stop CI/CD deploy & DevOps & MLOps



자료출처 : Spark + AI Summit 2019

Our Pain Point 6 – 해결 팁 (AI DevOps)

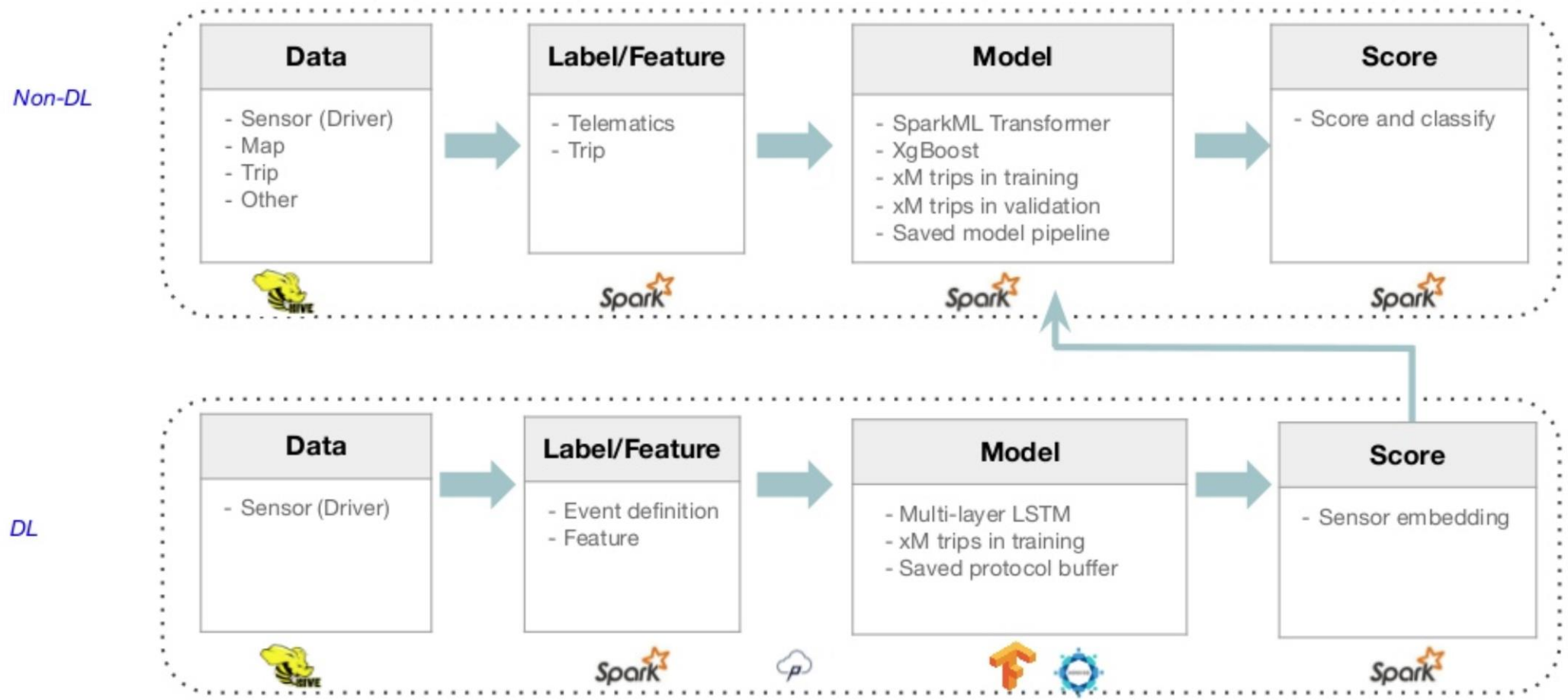
- None-Stop CI/CD deploy & DevOps & MLOps



자료출처 : Spark + AI Summit 2019

2. Production AI Open Source Eco System

BigData Scale {Deep Learning + None Deep Learning} Production Model Pipe Line



Horovod (Distributed Deep Learning at Scale)

- Open source library developed at Uber
- **Distributed** training for TensorFlow, Keras & PyTorch
- Uses bandwidth-optimal communication protocols & makes use of advanced networking
- Seamlessly installs via `pip install horovod`

Horovod (Distributed Deep Learning at Scale)

horovod.spark

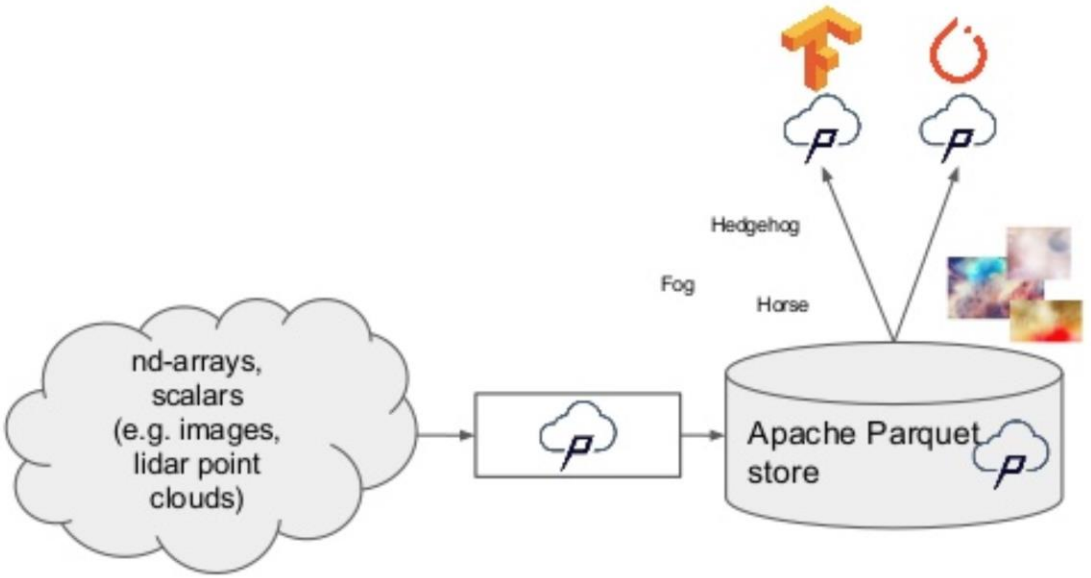
horovod.spark is a new feature in Horovod 0.16 release. Similar to HorovodRunner, it runs Horovod as a Spark job and takes python train functions. Its assumption is more general:

- no dependency on SSH,
- system-independent process termination,
- multiple Spark versions,
- and more … also check out horovodrun:)

Petastorm (BigData Scale Data Deep Learning)

- Open source library developed at Uber ATG
- Enables deep learning directly from Parquet
- Supports Tensorflow, PyTorch, and PySpark

Apache Parquet as a dataframe with tensors

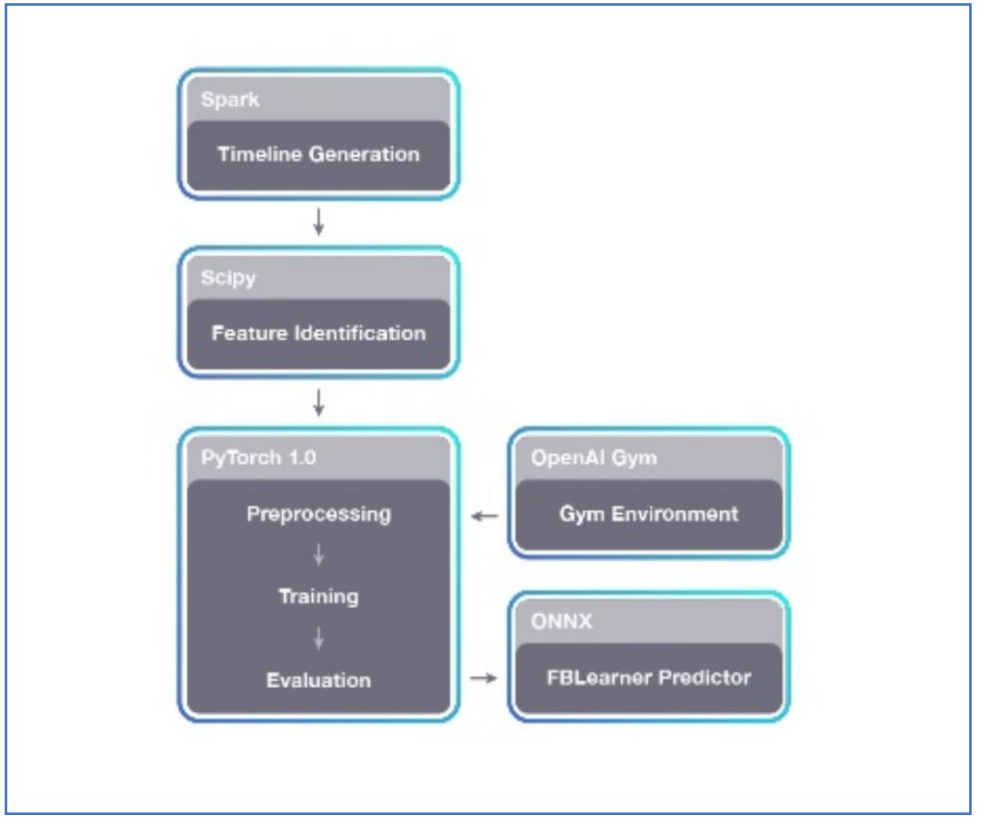


Horizon (Deep Reinforcement Learning at Scale)

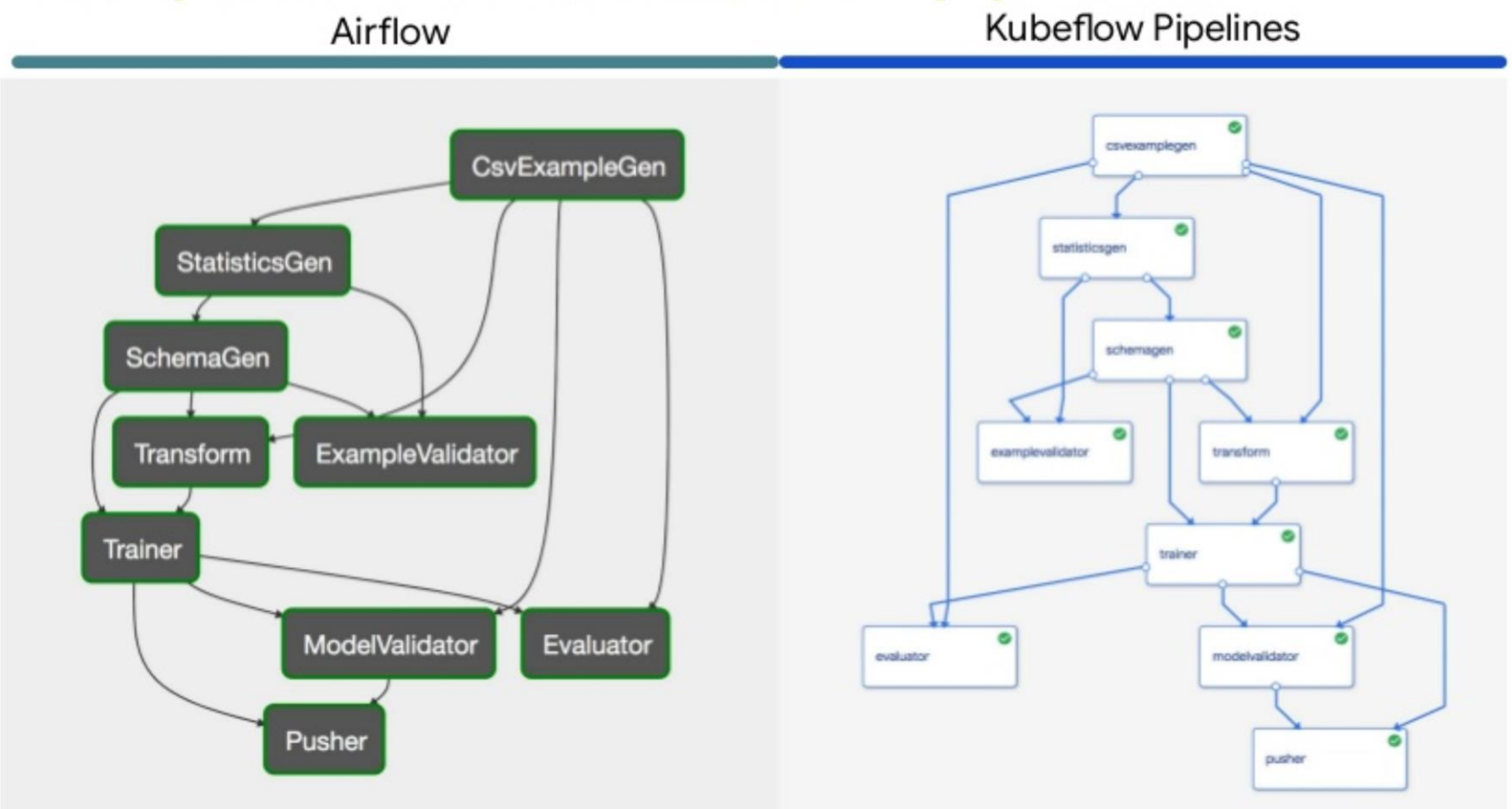
- Open Source End-To-End Large-scale RL framework By Facebook
  - distributed popular deep RL algorithms training.
  - workflow management.
  - data preprocessing.
  - feature transformation.
  - counterfactual policy evaluation.
  - optimized serving.
- Reinforcement Learning & Contextual Bandits.
- PyTorch for Modeling and Training.
- Caffe2 for Model Serving.
- https://reagent.ai/
- https://github.com/facebookresearch/ReAgent

Horizon (Deep Reinforcement Learning at Scale)

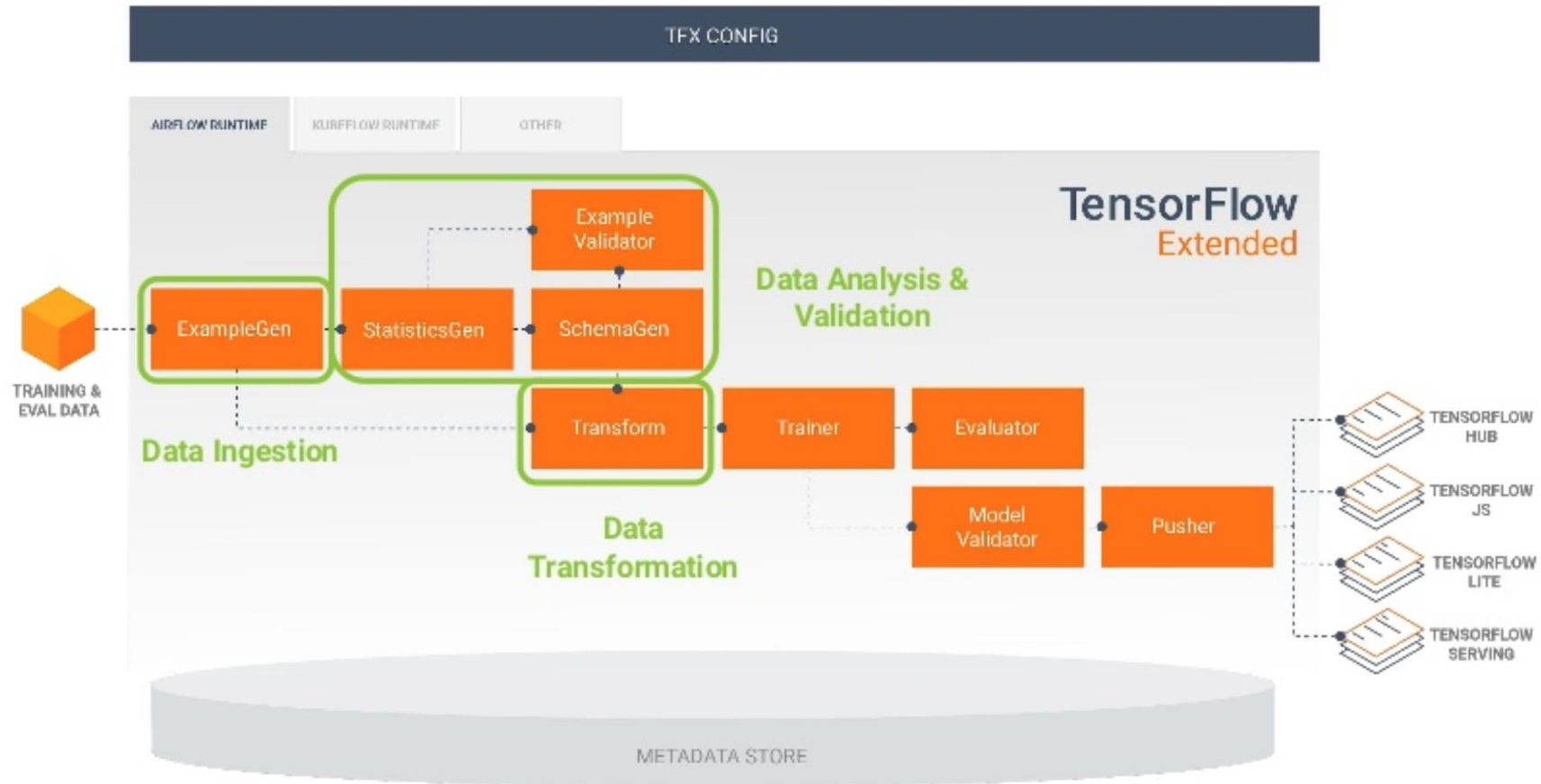
- Robust
- Massively Parallel
- Open Source
- Built on high-performance platforms
  - Spark
  - PyTorch
  - ONNX
- OpenAI Gym & Gridworld Integration tests



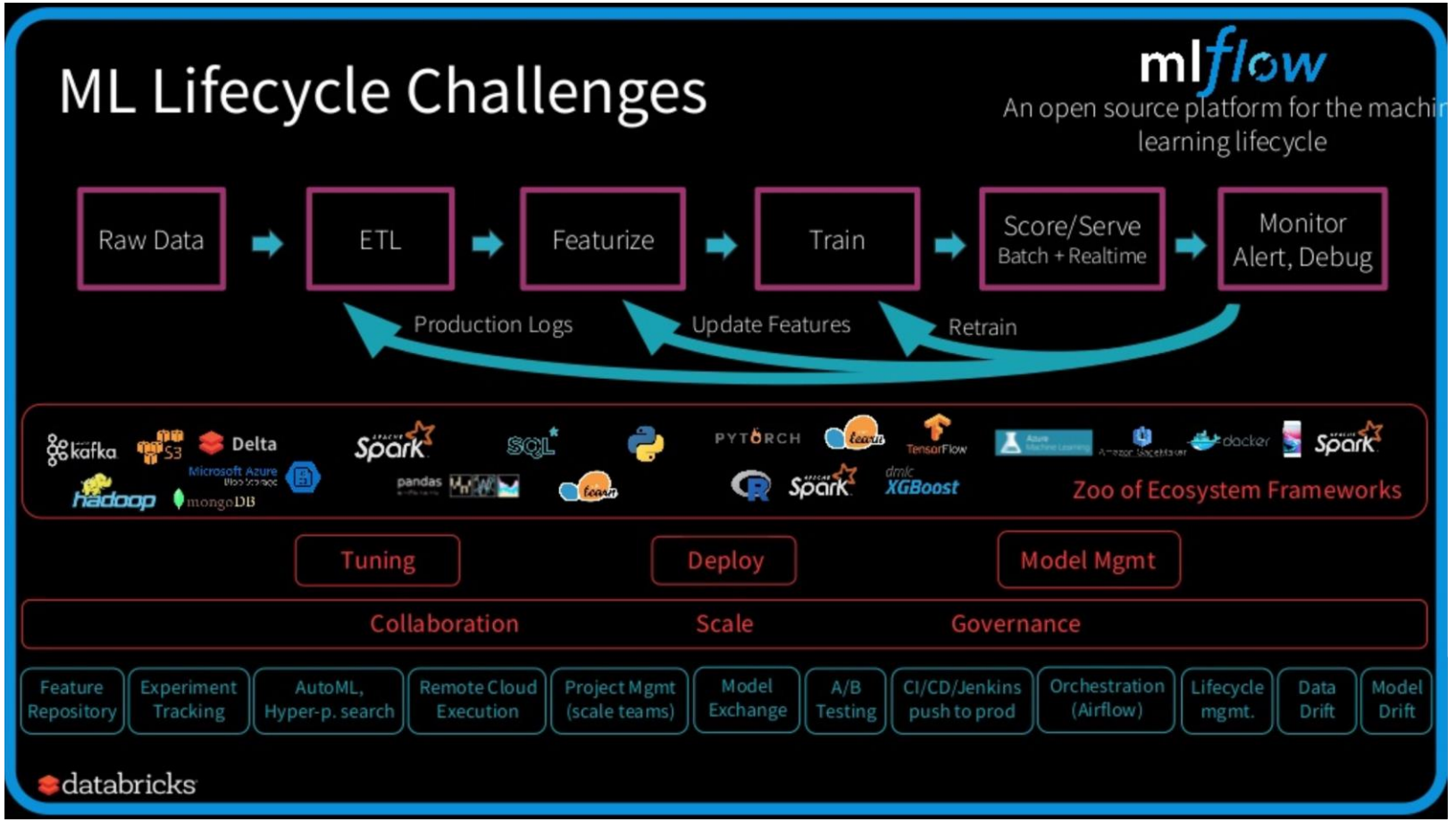
ML WorkFlow


Airflow
Kubeflow Pipelines
CsvExampleGen
StatisticsGen
SchemaGen
Transform
ExampleValidator
Trainer
ModelValidator
Evaluator
Pusher
csvexamplegen
statisticsgen
schemagen
examplevalidator
transform
trainer
evaluator
modelvalidator
pusher

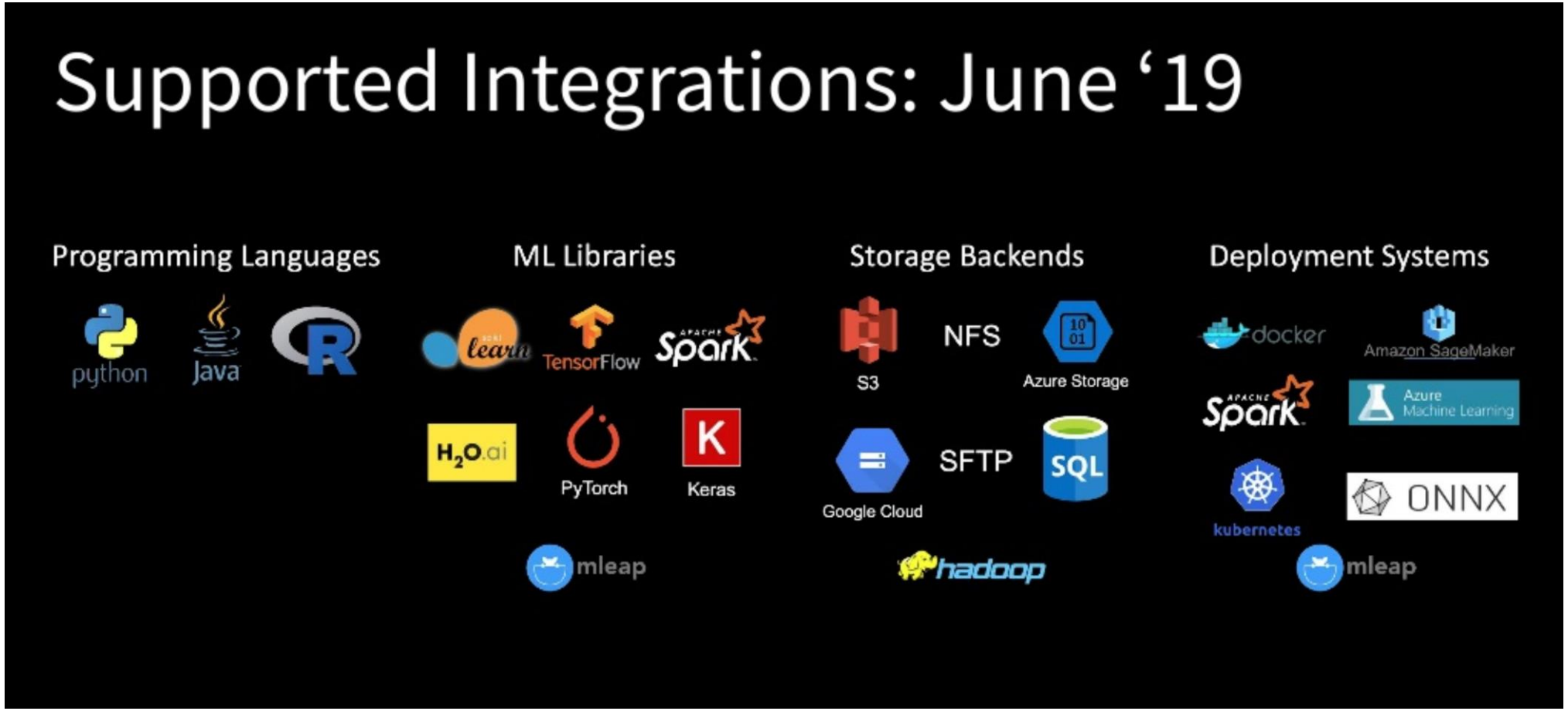
TFX


TFX CONFIG
AIRFLOW RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
Example Validator
ExampleGen
StatisticsGen
SchemaGen
Data Analysis & Validation
TRAINING & EVAL DATA
Transform
Trainer
Evaluator
Data Ingestion
Model Validator
Pusher
Data Transformation
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
METADATA STORE

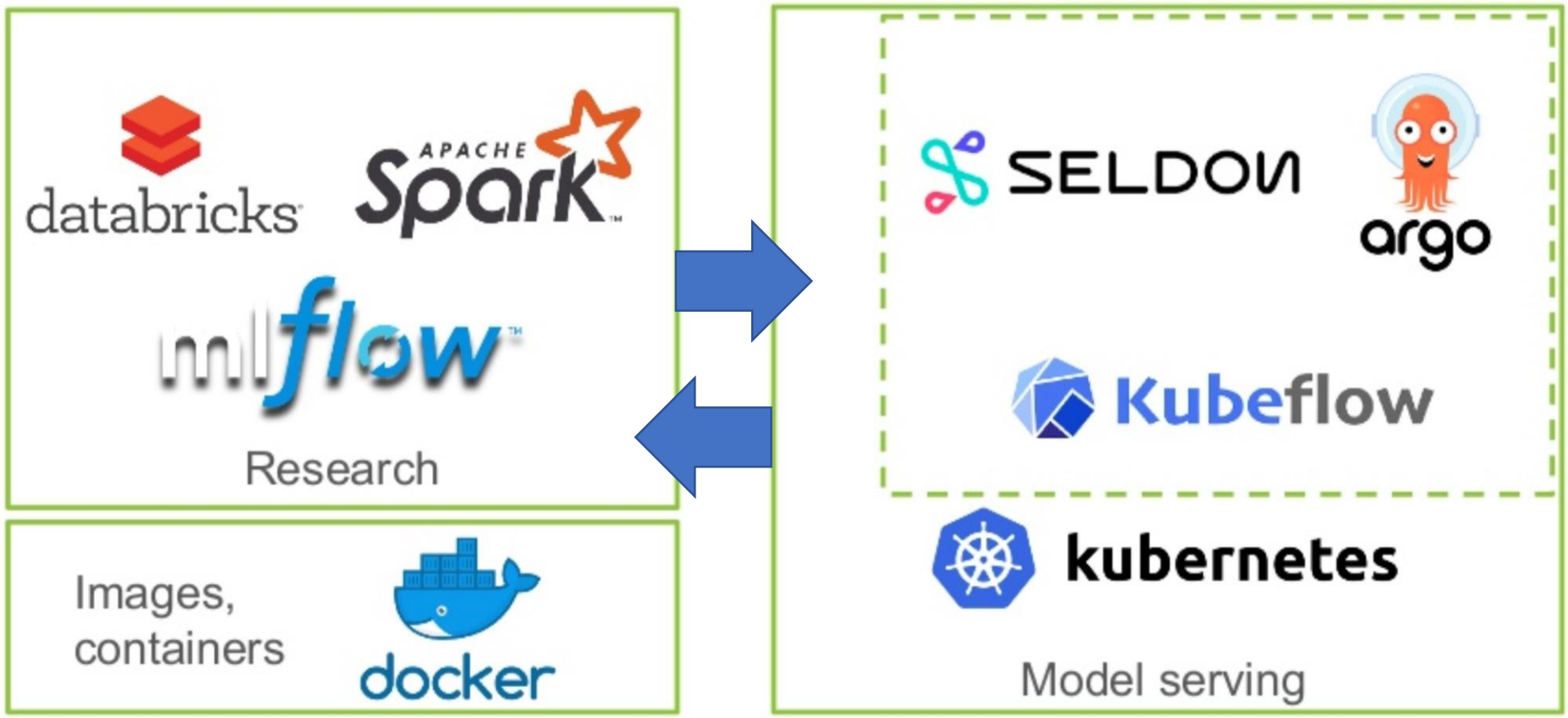
mlflow


ML Lifecycle Challenges
mlflow
An open source platform for the machin
learning lifecycle
Raw Data
ETL
Featurize
Train
Score/Serve
Batch + Realtime
Monitor
Alert, Debug
Production Logs
Update Features
Retrain
kafka
S3
Delta
Microsoft Azure
Spark
SQL
PYTORCH
TensorFlow
docker
hadoop
mongoDB
pandas
XGBoost
Zoo of Ecosystem Frameworks
Tuning
Deploy
Model Mgmt
Collaboration
Scale
Governance
Feature Repository
Experiment Tracking
AutoML, Hyper-p. search
Remote Cloud Execution
Project Mgmt (scale teams)
Model Exchange
A/B Testing
CI/CD/Jenkins push to prod
Orchestration (Airflow)
Lifecycle mgmt.
Data Drift
Model Drift
databricks

mlflow

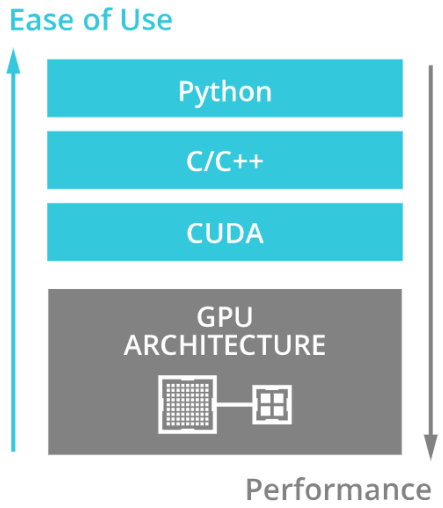

Supported Integrations: June '19
Programming Languages
python
Java
R
ML Libraries
learn
TensorFlow
Spark
H2O.ai
PyTorch
Keras
mleap
Storage Backends
S3
NFS
Azure Storage
Google Cloud
SFTP
SQL
hadoop
Deployment Systems
docker
Amazon SageMaker
Spark
Azure Machine Learning
kubernetes
ONNX
mleap

Serving at Scale



Rapids.ai

- End-to-End Data Analytics Pipeline with GPUs
- Apache Arrow , cuDF(on Spark) , cuML(like Pandas)



GPU DATA SCIENCE

ACCELERATED DATA SCIENCE
The RAPIDS suite of open source software libraries gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs.
Learn more »

SCALE OUT ON GPUS
Seamlessly scale from GPU workstations to multi-GPU servers and multi-node clusters with Dask.
Learn more about Dask »

PYTHON INTEGRATION
Accelerate your Python data science toolchain with minimal code changes and no new tools to learn.

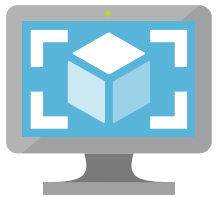
TOP MODEL ACCURACY
Increase machine learning model accuracy by iterating on models faster and deploying them more frequently.

REDUCED TRAINING TIME
Drastically improve your productivity with more interactive data science.
Learn more about XGBoost »

OPEN SOURCE
RAPIDS is an open source project. Supported by NVIDIA, it also relies on numba, apache arrow, and many more open source projects.
Learn more »

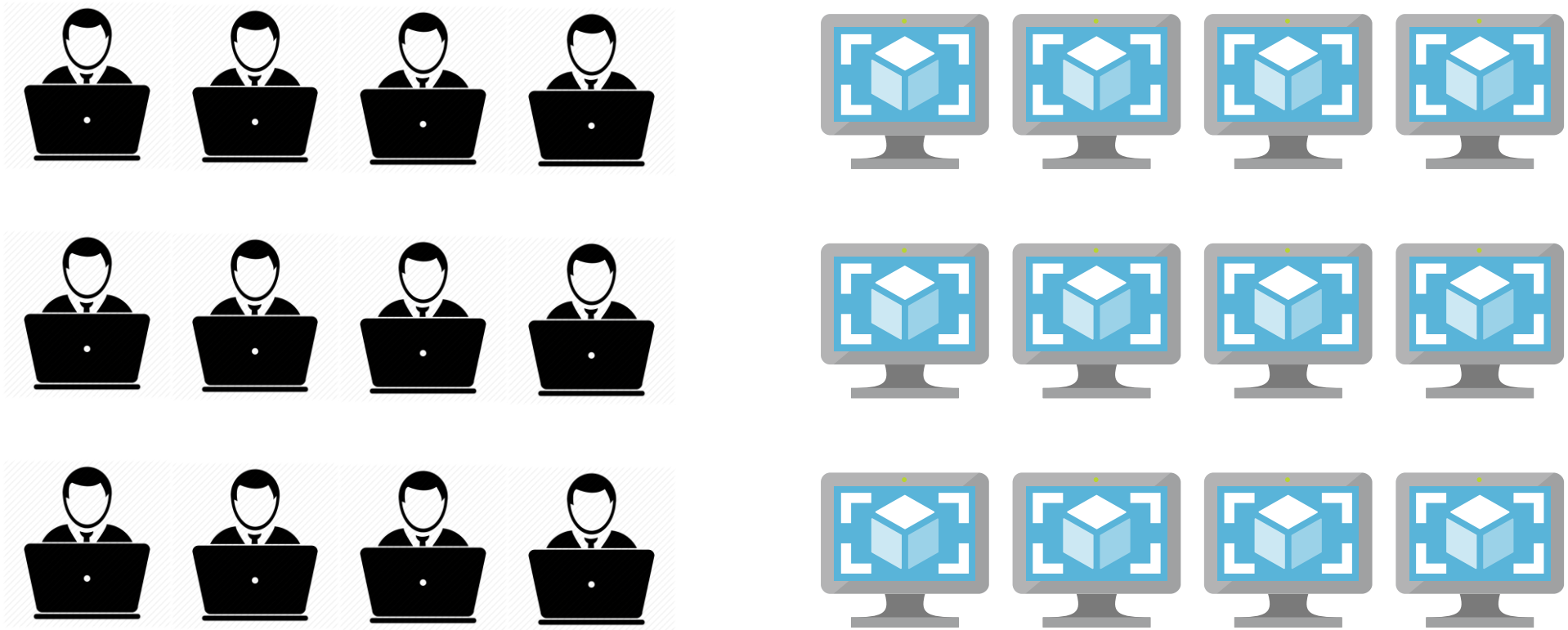
ONNX - Multiple ML/DL Framework Collaboration

3. Production AI DevOps



현재 직면한 문제

- 여러 개발자의 컴퓨터, 여러 배포 환경에서 실행시켜야 함.



재현성

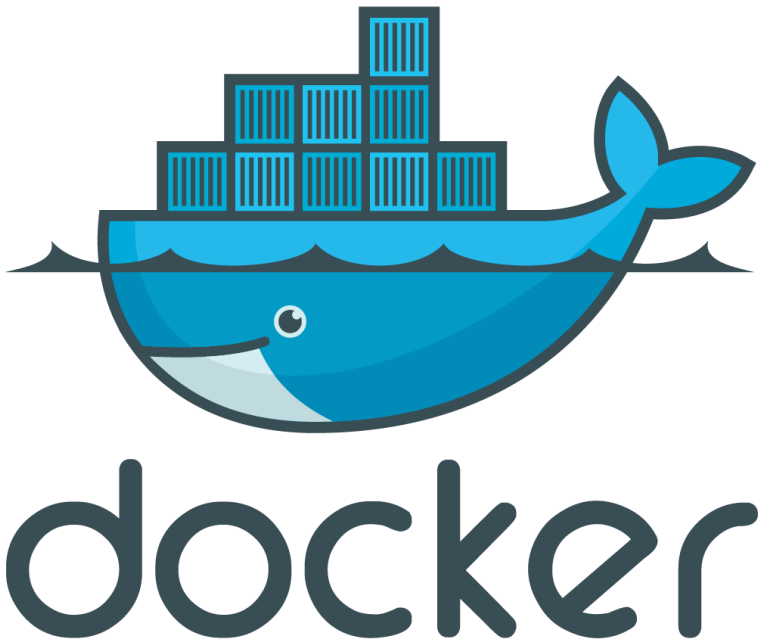
"한 연구로부터 도출된 결과와 결론을 서로 독립적인 다른 연구를 통해 확증하는 것"
(The confirmation of results and conclusions from one study obtained independently in another)

-출처 : https://www.sciencemag.org/

재현성

"한 **코드**로부터 도출된 **결과와 동작**을 서로 독립적인 다른 **머신**를 통해 **같은 동작**을 하게 하는 것"

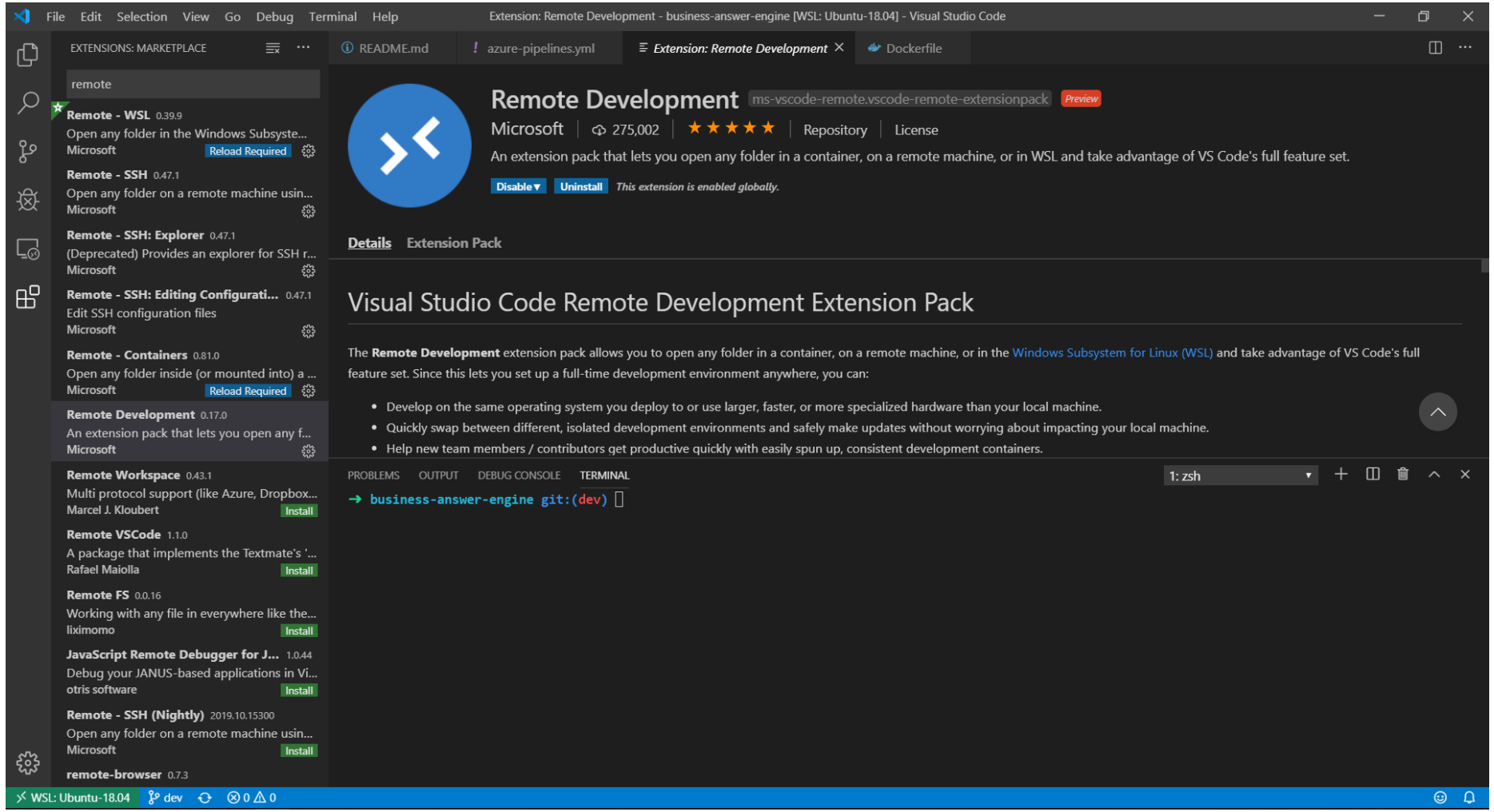
재현성을 위한 강력한 도구 container



재현성으로 해결 가능한 문제

- 개발 환경 공유
- 운영 환경 동작 보장

개발 환경 재현성을 위한 도구

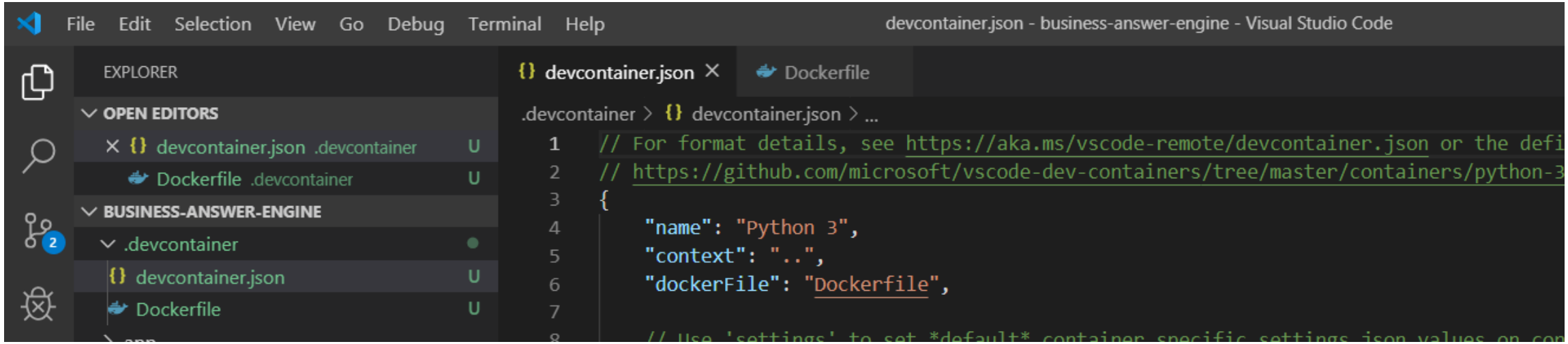


Remote target

- SSH
- Container
- WSL

Remote-container

Devcontainer 설정과 Dockerfile을 통해 개발 환경을 코드로 관리



DevOps 란

- 지속적 통합과 배포
- 마이크로서비스
- 코드형 인프라스트럭쳐
- 마이크로 서비스를 위한 모니터링
- 커뮤니케이션

-> 코드형 인프라스트럭쳐까지 방법론이 성숙되면서 GitOps라는 용어도 발생

출처 : https://aws.amazon.com/ko/devops/what-is-devops/

도커가 해주는 것

- 지속적 통합과 배포
- 마이크로서비스
- 코드형 인프라스트럭쳐
- 마이크로 서비스를 위한 모니터링
- 커뮤니케이션

-> 코드형 인프라스트럭쳐까지 방법론이 성숙되면서 GitOps라는 용어도 발생

출처 : https://aws.amazon.com/ko/devops/what-is-devops/

개발자의 영역

- 지속적 통합과 배포
- 마이크로서비스
- 코드형 인프라스트럭쳐
- 마이크로 서비스를 위한 모니터링
- 커뮤니케이션

-> 코드형 인프라스트럭쳐까지 방법론이 성숙되면서 GitOps라는 용어도 발생

출처 : https://aws.amazon.com/ko/devops/what-is-devops/

슬랙?

- 지속적 통합과 배포
- 마이크로서비스
- 코드형 인프라스트럭쳐
- 마이크로 서비스를 위한 모니터링
- 커뮤니케이션

-> 코드형 인프라스트럭쳐까지 방법론이 성숙되면서 GitOps라는 용어도 발생

출처 : https://aws.amazon.com/ko/devops/what-is-devops/

이것을 위한 도구가 필요

- 지속적 통합과 배포
- 마이크로서비스
- 코드형 인프라스트럭쳐
- 마이크로 서비스를 위한 모니터링
- 커뮤니케이션

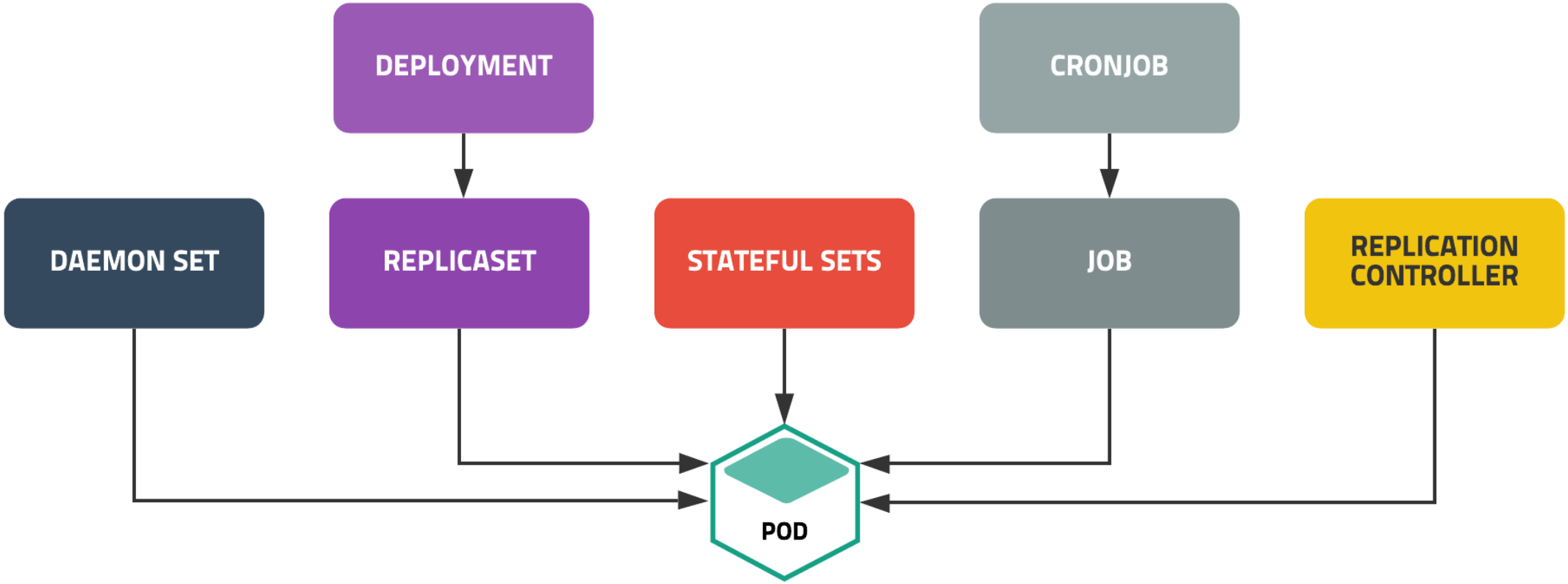
-> 코드형 인프라스트럭쳐까지 방법론이 성숙되면서 GitOps라는 용어도 발생

출처 : https://aws.amazon.com/ko/devops/what-is-devops/

자동 운영환경을 위한 kubernetes

kubernetes

Pod을 관리하는 상위 개념들



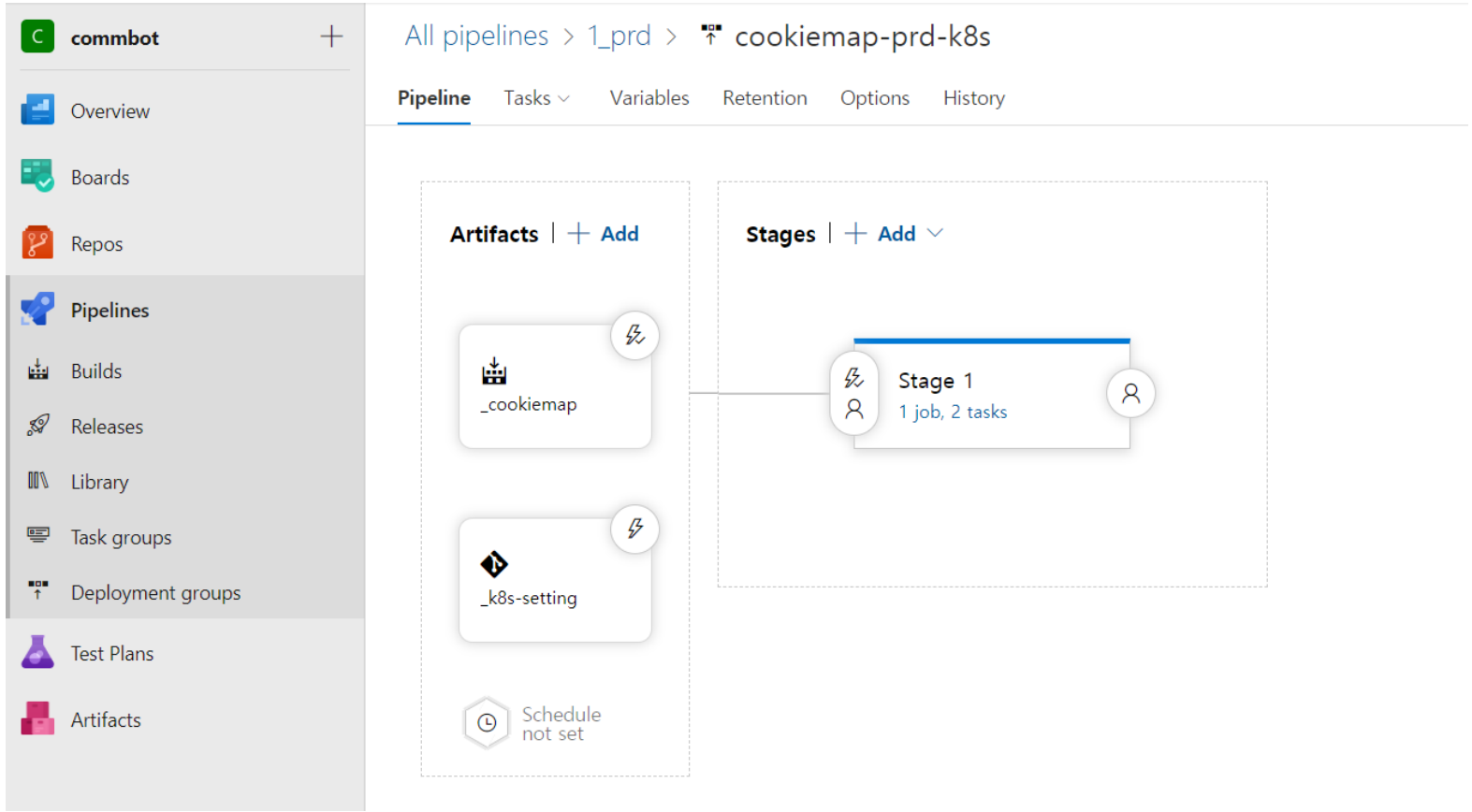
선언적 설정으로 인프라 관리

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: cookiemap
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: cookiemap
    spec:
      hostAliases:
      - ip: "52.231.34.29"
        hostnames:
        - "internal.commbot.nginx"
      containers:
      - name: cookiemap
        image: acrcommbotprdinternal.azurecr.io/cookiemap:latest
        imagePullPolicy: Always
        imagePullSecrets:
        - name: commbotprdsecret
        ports:
        - containerPort: 8080
        envFrom:
        - configMapRef:
            name: configmap-cookiemap
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: cookiemap
spec:
  sessionAffinity: ClientIP
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: cookiemap
---
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: cookiemap
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: cookiemap
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 70
```

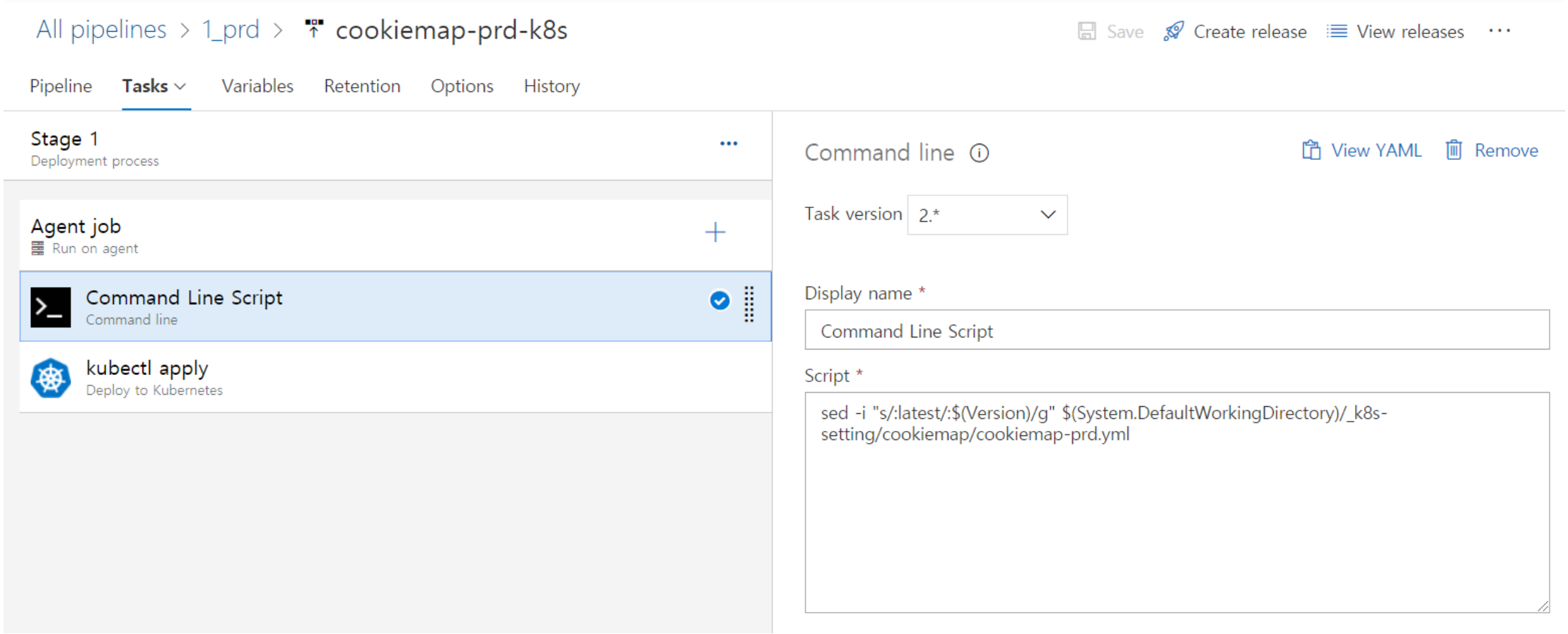
배포

- Azure Devops의 Pipelines의 기능을 활용



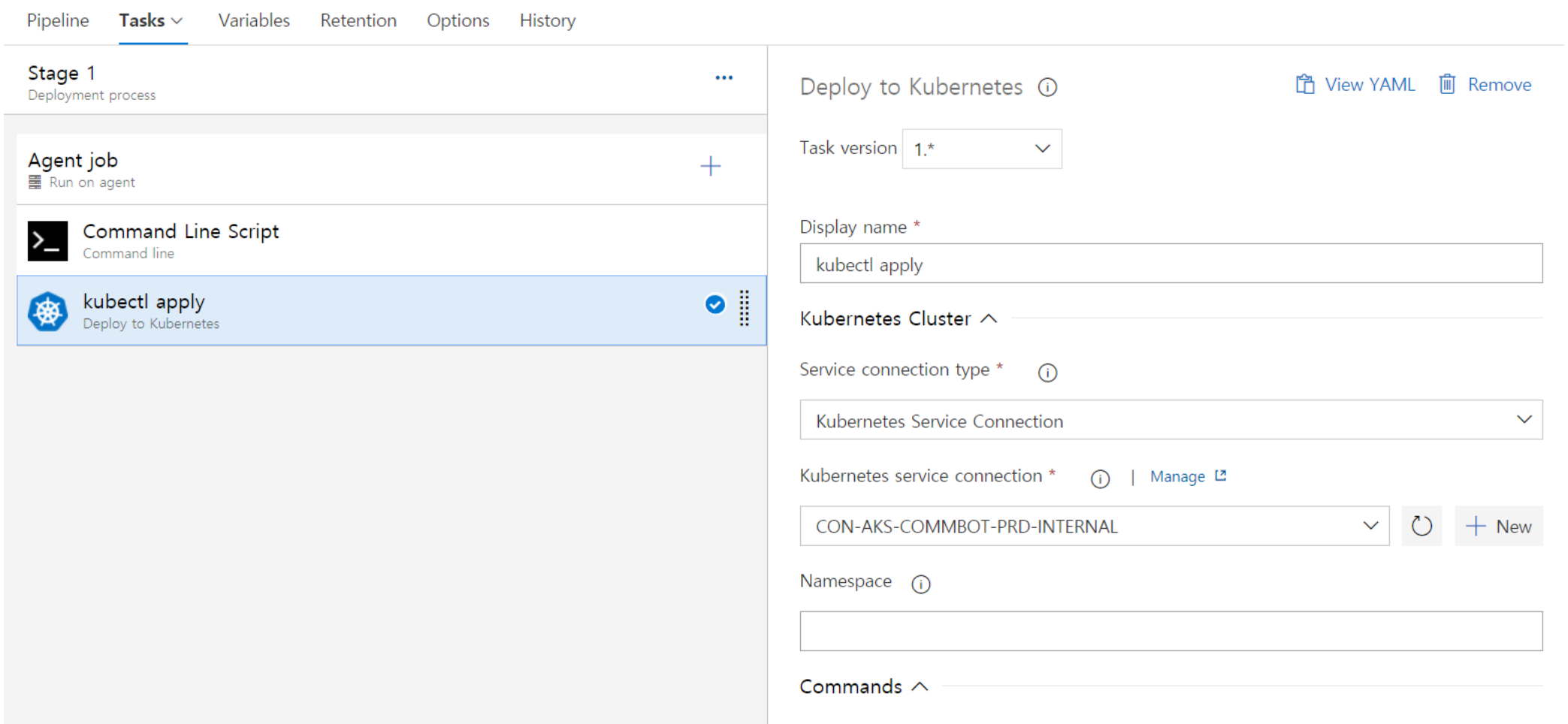
배포

- 새로운 이미지 배포를 위해 latest 태그 사용 지양



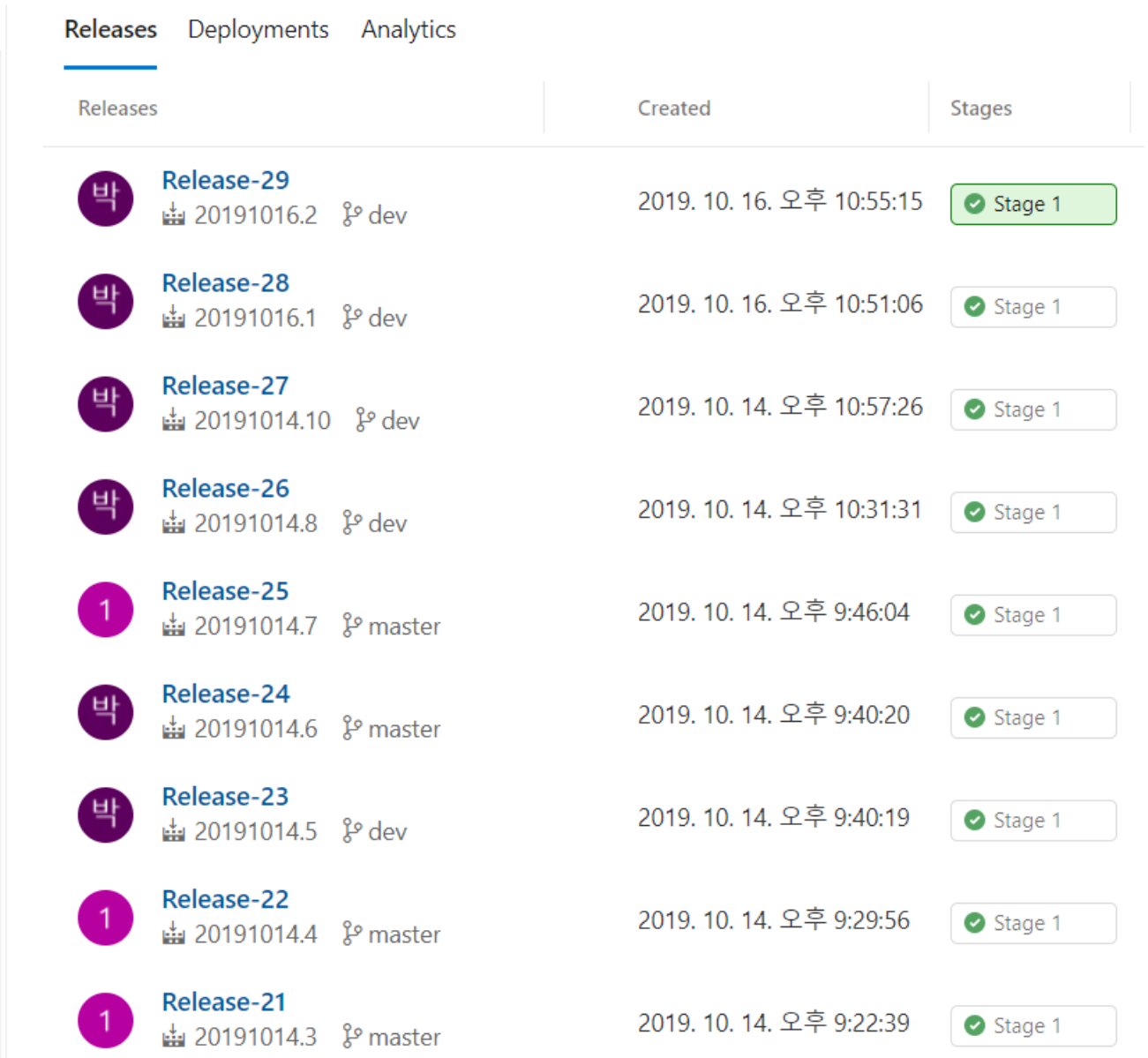
배포

- Kubectl 명령을 devops 내에서만 실행



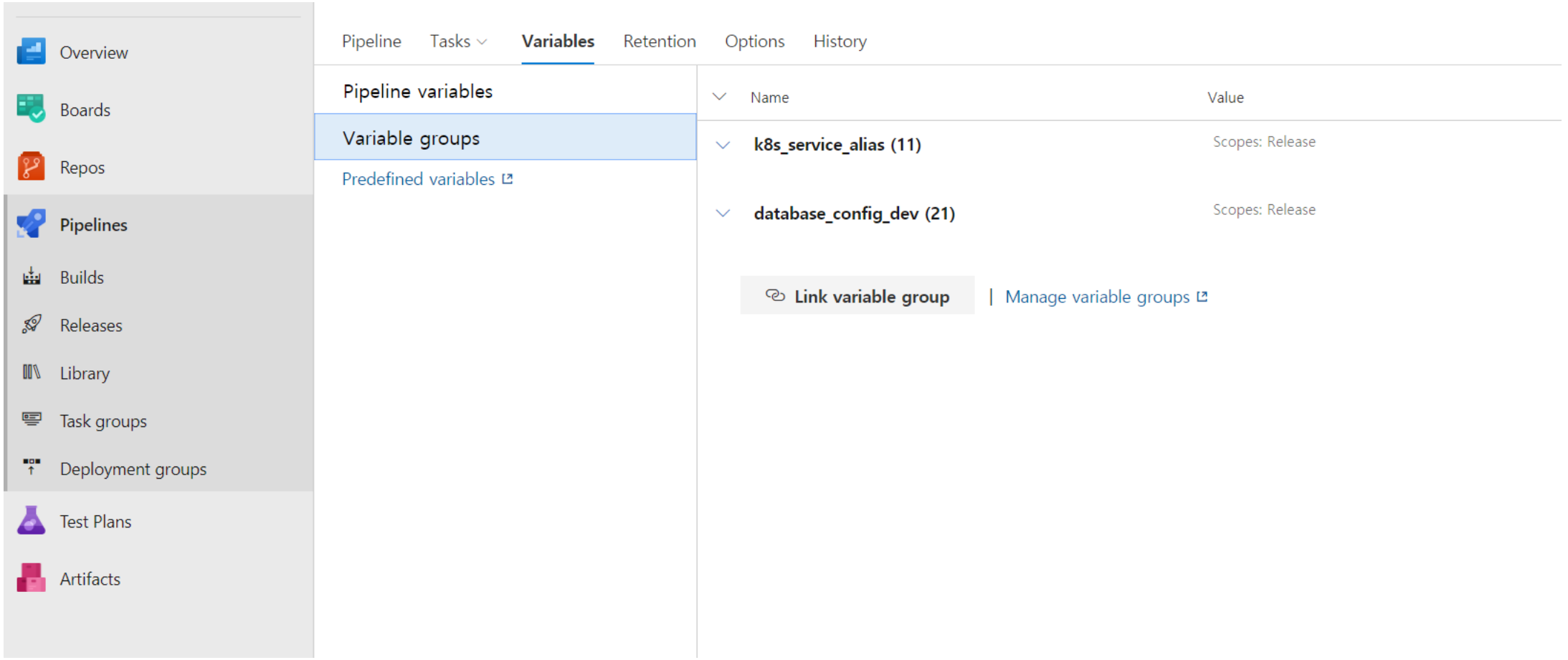
배포

- 배포 기록 관리
- 기록 기반 롤백



환경별 변수 관리

- Library에 글로벌 변수 작성, 파이프라인 별 적용



환경별 변수 관리

- 코드 내에서는 env 호출로 사용

config / **config.py**

Contents | History | Compare | Blame | Edit | Rename | Delete | Download

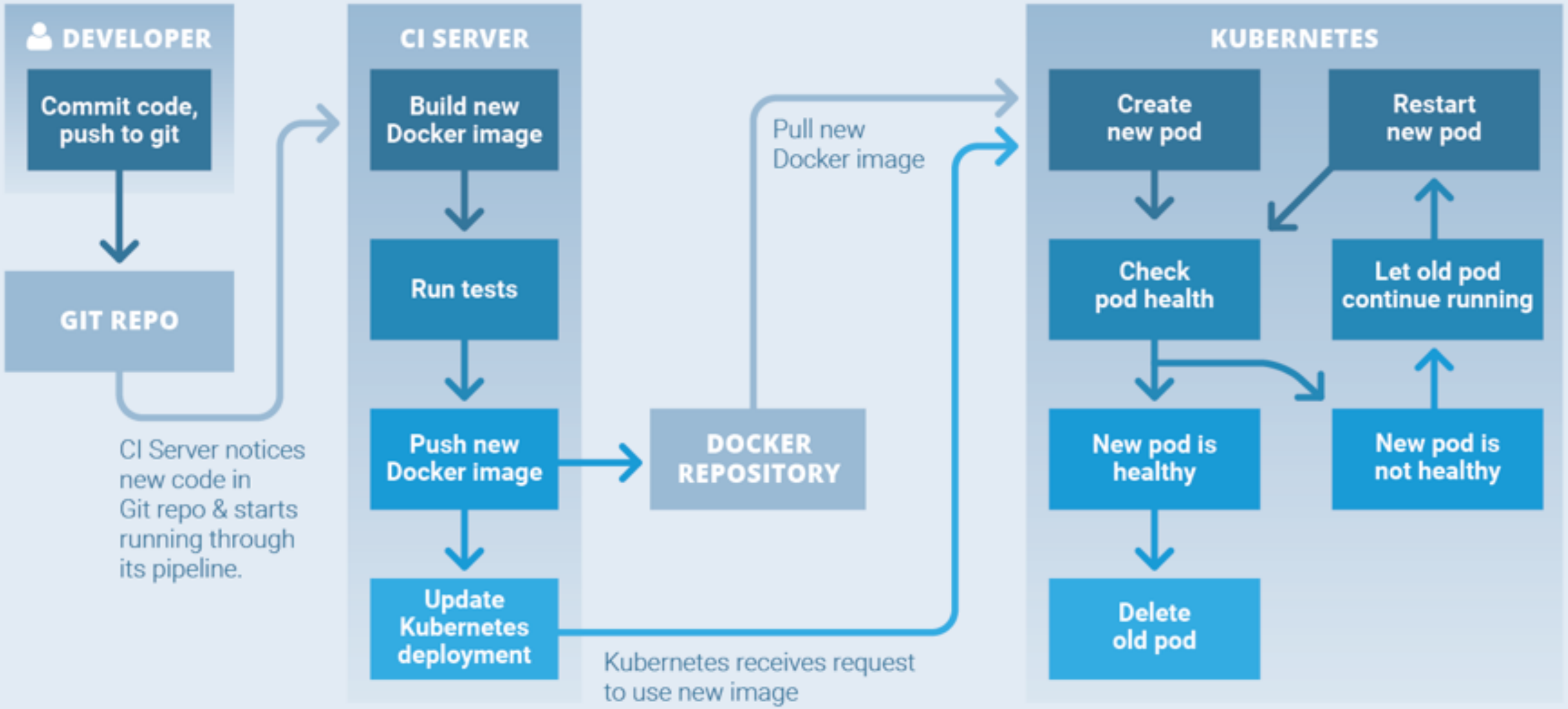
```python
import os

class log_connection_info:
    port = os.environ['ES_PORT']
    host = os.environ['ES_URL']
    elastic_user = os.environ['ES_USER']
    elastic_pw = os.environ['ES_PASSWORD']
    multi_turn_log_index = os.environ['ES_INDEX_MULTITURN']
    user_info_log_index = os.environ['ES_USERINFO']
    doc_type='_doc'


class redis_connection_info:
    host = os.environ['REDIS_CACHE_URL']
    port = os.environ['REDIS_CACHE_PORT']
    db = os.environ['REDIS_CACHE_DB']
    password = os.environ['REDIS_CACHE_KEY_PRIMARY']
    if os.environ['REDIS_CACHE_SSL'] == "TRUE":
        ssl = True
    else:
        ssl = False


```

CI/CD Pipeline Workflow with Kubernetes



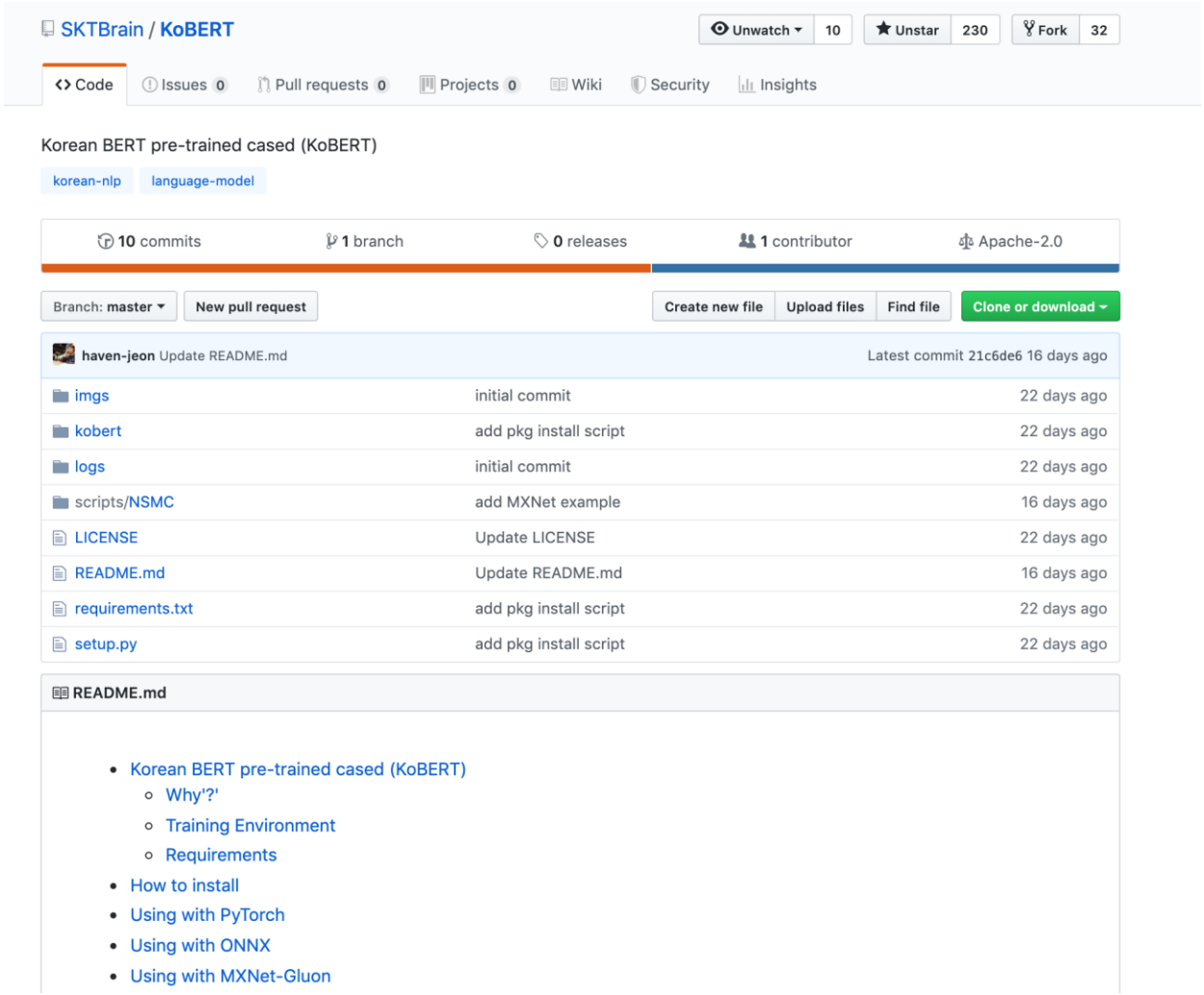
Function as a Service Flatform

- Kubernetes 위에서 동작하는 opensource serverless 도구들
- 개발자가 더욱 로직에 집중할 수 있게 자동화 가능

출처 : https://landscape.cncf.io/format=serverless&fullscreen=yes&zoom=150

Our Open Source

Korean BERT pre-trained cased (KoBERT)



https://github.com/SKTBrain/KoBERT

USE Case 소개(상세 내용은 …)

- 4 CPU * 10 Docker on Cloud
- 1.5GB LSTM Tensorlfow NLP 모델 실시간 Serving (batchsize = 1)
- 랜덤 쿼리 (캐쉬 의미 없음)
- Deep Learning Inference 1만 TPS 구현해 보기.
  - NginX + ASGI + FastAPI + asyncio + aiohttp + tornado
  - Pandas 걷어내기, MemoryView 활용하기.
  - Distributed Shared Session으로 Redis 사용.

Q&A

김훈동 : https://www.facebook.com/kim.hoondong
박찬엽 : https://www.facebook.com/mrchypark